aws

User Guide

# Amazon Simple Storage Service

**API Version 2006-03-01**

# Amazon Simple Storage Service: User Guide

# Table of Contents

# What is Amazon S3?

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



# Features of Amazon S3

## Storage classes

Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard or S3 Express One Zone for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud object storage class available today, with data access speeds up to 10x faster and with request costs 50 percent lower than S3 Standard. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. Additionally, to further increase access speed and support hundreds of thousands of requests per second, data is stored in a new bucket type: an Amazon S3 directory bucket. For more information, see What is S3 Express One Zone? and Directory buckets.

You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data.

For more information, see Using Amazon S3 storage classes.

## Storage management

Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.

- S3 Lifecycle – Configure a lifecycle configuration to manage your objects and store them cost effectively throughout their lifecycle. You can transition objects to other S3 storage classes or expire objects that reach the end of their lifetimes.
- S3 Object Lock – Prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require *write-once-read-many (WORM)* storage or to simply add another layer of protection against object changes and deletions.
- S3 Replication – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.
- S3 Batch Operations – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as **Copy**, **Invoke AWS Lambda function**, and **Restore** on millions or billions of objects.

# Access management and security

Amazon S3 provides features for auditing and managing access to your buckets and objects. By default, S3 buckets and the objects in them are private. You have access only to the S3 resources that you create. To grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources, you can use the following features.

- S3 Block Public Access – Block public access to S3 buckets and objects. By default, Block Public Access settings are turned on at the bucket level. We recommend that you keep all Block Public Access settings enabled unless you know that you need to turn off one or more of them for your specific use case. For more information, see Configuring block public access settings for your S3 buckets.
- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources, including your Amazon S3 resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- Amazon S3 access points – Configure named network endpoints with dedicated access policies to manage data access at scale for shared datasets in Amazon S3.
- Access control lists (ACLs) – Grant read and write permissions for individual buckets and objects to authorized users. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM user policies for access control instead of ACLs. Policies are a simplified and more flexible access control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources. For more information about the specific cases when you'd use ACLs instead of resource-based policies or IAM user policies, see Managing access with ACLs.
- S3 Object Ownership – Take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable or enable ACLs. By default, ACLs are disabled. With ACLs disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively by using access-management policies.
- IAM Access Analyzer for S3 – Evaluate and monitor your S3 bucket access policies, ensuring that the policies provide only the intended access to your S3 resources.

## Data processing

To transform data and trigger workflows to automate a variety of other processing activities at scale, you can use the following features.

- S3 Object Lambda – Add your own code to S3 GET, HEAD, and LIST requests to modify and process data as it is returned to an application. Filter rows, dynamically resize images, redact confidential data, and much more.
- Event notifications – Trigger workflows that use Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and AWS Lambda when a change is made to your S3 resources.

## Storage logging and monitoring

Amazon S3 provides logging and monitoring tools that you can use to monitor and control how your Amazon S3 resources are being used. For more information, see Monitoring tools.

### Automated monitoring tools

- Amazon CloudWatch metrics for Amazon S3 – Track the operational health of your S3 resources and configure billing alerts when estimated charges reach a user-defined threshold.
- AWS CloudTrail – Record actions taken by a user, a role, or an AWS service in Amazon S3. CloudTrail logs provide you with detailed API tracking for S3 bucket-level and object-level operations.

### Manual monitoring tools

- Server access logging – Get detailed records for the requests that are made to a bucket. You can use server access logs for many use cases, such as conducting security and access audits, learning about your customer base, and understanding your Amazon S3 bill.
- AWS Trusted Advisor – Evaluate your account by using AWS best practice checks to identify ways to optimize your AWS infrastructure, improve security and performance, reduce costs, and monitor service quotas. You can then follow the recommendations to optimize your services and resources.

## Analytics and insights

Amazon S3 offers features to help you gain visibility into your storage usage, which empowers you to better understand, analyze, and optimize your storage at scale.

- Amazon S3 Storage Lens – Understand, analyze, and optimize your storage. S3 Storage Lens provides 60+ usage and activity metrics and interactive dashboards to aggregate data for your entire organization, specific accounts, AWS Regions, buckets, or prefixes.
- Storage Class Analysis – Analyze storage access patterns to decide when it's time to move data to a more cost-effective storage class.
- S3 Inventory with Inventory reports – Audit and report on objects and their corresponding metadata and configure other Amazon S3 features to take action in Inventory reports. For example, you can report on the replication and encryption status of your objects. For a list of all the metadata available for each object in Inventory reports, see Amazon S3 Inventory list.

## Strong consistency

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. This behavior applies to both writes of new objects as well as PUT requests that overwrite existing objects and DELETE requests. In addition, read operations on Amazon S3 Select, Amazon S3 access control lists (ACLs), Amazon S3 Object Tags, and object metadata (for example, the HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model.

# How Amazon S3 works

Amazon S3 is an object storage service that stores data as objects within buckets. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects.

To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a *key* (or *key name*), which is the unique identifier for the object within the bucket.

S3 provides features that you can configure to support your specific use case. For example, you can use S3 Versioning to keep multiple versions of an object in the same bucket, which allows you to restore objects that are accidentally deleted or overwritten.

Buckets and the objects in them are private and can be accessed only if you explicitly grant access permissions. You can use bucket policies, AWS Identity and Access Management (IAM) policies, access control lists (ACLs), and S3 Access Points to manage access.

**Topics**



## Buckets

A bucket is a container for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account. To request an increase, visit the Service Quotas console.

Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `amzn-s3-demo-bucket` bucket in the US West (Oregon) Region, then it is addressable by using the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`. For more information, see Accessing a Bucket.

When you create a bucket, you enter a bucket name and choose the AWS Region where the bucket will reside. After you create a bucket, you cannot change the name of the bucket or its Region. Bucket names must follow the bucket naming rules. You can also configure a bucket to use S3 Versioning or other storage management features.

Buckets also:

- Organize the Amazon S3 namespace at the highest level.
- Identify the account responsible for storage and data transfer charges.

- Provide access control options, such as bucket policies, access control lists (ACLs), and S3 Access Points, that you can use to manage access to your Amazon S3 resources.
- Serve as the unit of aggregation for usage reporting.

For more information about buckets, see Buckets overview.

## Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as `Content-Type`. You can also specify custom metadata at the time that the object is stored.

An object is uniquely identified within a bucket by a key (name) and a version ID (if S3 Versioning is enabled on the bucket). For more information about objects, see Amazon S3 objects overview.

### Keys

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object. So you can think of Amazon S3 as a basic data map between "bucket + key + version" and the object itself.

Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version. For example, in the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`, *amzn-s3-demo-bucket* is the name of the bucket and `photos/puppy.jpg` is the key.

For more information about object keys, see Creating object key names.

### S3 Versioning

You can use S3 Versioning to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets. You can easily recover from both unintended user actions and application failures.

For more information, see Using versioning in S3 buckets.

## Version ID

When you enable S3 Versioning in a bucket, Amazon S3 generates a unique version ID for each object added to the bucket. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as CopyObject and PutObject, the new objects get a unique version ID.

For more information, see Using versioning in S3 buckets.

## Bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size.

Bucket policies use JSON-based access policy language that is standard across AWS. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies allow or deny requests based on the elements in the policy, including the requester, S3 actions, resources, and aspects or conditions of the request (for example, the IP address used to make the request). For example, you can create a bucket policy that grants cross-account permissions to upload objects to an S3 bucket while ensuring that the bucket owner has full control of the uploaded objects. For more information, see Examples of Amazon S3 bucket policies.

In your bucket policy, you can use wildcard characters on Amazon Resource Names (ARNs) and other values to grant permissions to a subset of objects. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

## S3 Access Points

Amazon S3 Access Points are named network endpoints with dedicated access policies that describe how data can be accessed using that endpoint. Access Points are attached to buckets that you can use to perform S3 object operations, such as GetObject and PutObject. Access Points simplify managing data access at scale for shared datasets in Amazon S3.

Each access point has its own access point policy. You can configure Block Public Access settings for each access point. To restrict Amazon S3 data access to a private network, you can also configure any access point to accept requests only from a virtual private cloud (VPC).

For more information, see Managing data access with Amazon S3 access points.

## Access control lists (ACLs)

You can use ACLs to grant read and write permissions to authorized users for individual buckets and objects. Each bucket and object has an ACL attached to it as a subresource. The ACL defines which AWS accounts or groups are granted access and the type of access. ACLs are an access control mechanism that predates IAM. For more information about ACLs, see Access control list (ACL) overview.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

## Regions

You can choose the geographical AWS Region where Amazon S3 stores the buckets that you create. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Objects stored in an AWS Region never leave the Region unless you explicitly transfer or replicate them to another Region. For example, objects stored in the Europe (Ireland) Region never leave it.

> **Note**
>
> You can access Amazon S3 and its features only in the AWS Regions that are enabled for your account. For more information about enabling a Region to create and manage AWS resources, see Managing AWS Regions in the *AWS General Reference*.

For a list of Amazon S3 Regions and endpoints, see Regions and endpoints in the *AWS General Reference*.

# Amazon S3 data consistency model

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. This behavior applies to both writes to new objects as well as PUT requests that overwrite existing objects and DELETE requests. In addition, read operations on Amazon S3 Select, Amazon S3 access controls lists (ACLs), Amazon S3 Object Tags, and object metadata (for example, the HEAD object) are strongly consistent.

Updates to a single key are atomic. For example, if you make a PUT request to an existing key from one thread and perform a GET request on the same key from a second thread concurrently, you will get either the old data or the new data, but never partial or corrupt data.

Amazon S3 achieves high availability by replicating data across multiple servers within AWS data centers. If a PUT request is successful, your data is safely stored. Any read (GET or LIST request) that is initiated following the receipt of a successful PUT response will return the data written by the PUT request. Here are examples of this behavior:

- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. The new object appears in the list.
- A process replaces an existing object and immediately tries to read it. Amazon S3 returns the new data.
- A process deletes an existing object and immediately tries to read it. Amazon S3 does not return any data because the object has been deleted.
- A process deletes an existing object and immediately lists keys within its bucket. The object does not appear in the listing.

**Note**

- Amazon S3 does not support object locking for concurrent writers. If two PUT requests are simultaneously made to the same key, the request with the latest timestamp wins. If this is an issue, you must build an object-locking mechanism into your application.
- Updates are key-based. There is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

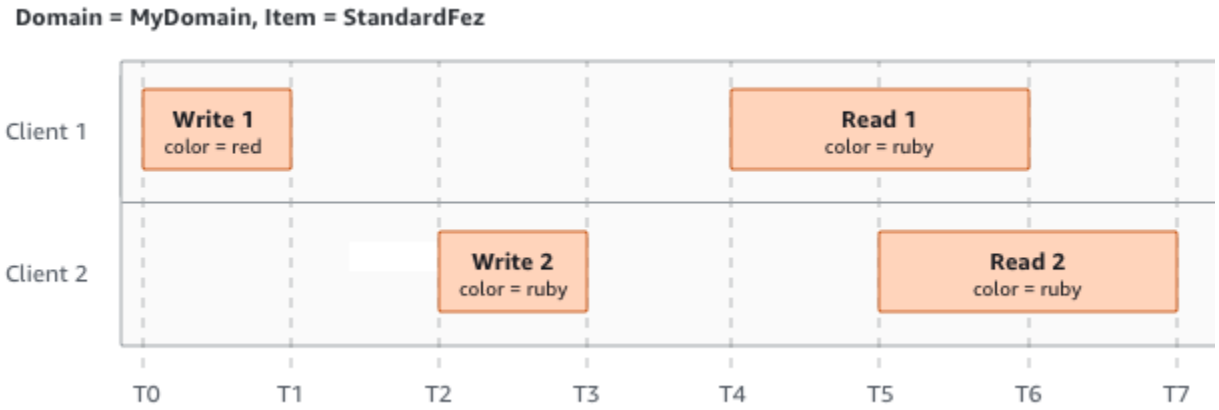
Bucket configurations have an eventual consistency model. Specifically, this means that:

- If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list.
- If you enable versioning on a bucket for the first time, it might take a short amount of time for the change to be fully propagated. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (PUT or DELETE requests) on objects in the bucket.

## Concurrent applications

This section provides examples of behavior to be expected from Amazon S3 when multiple clients are writing to the same items.

In this example, both W1 (write 1) and W2 (write 2) finish before the start of R1 (read 1) and R2 (read 2). Because S3 is strongly consistent, R1 and R2 both return `color = ruby`.




In the next example, W2 does not finish before the start of R1. Therefore, R1 might return `color = ruby` or `color = garnet`. However, because W1 and W2 finish before the start of R2, R2 returns `color = garnet`.


In the last example, W2 begins before W1 has received an acknowledgment. Therefore, these writes are considered concurrent. Amazon S3 internally uses last-writer-wins semantics to determine which write takes precedence. However, the order in which Amazon S3 receives the requests and the order in which applications receive acknowledgments cannot be predicted because of various factors, such as network latency. For example, W2 might be initiated by an Amazon EC2 instance in the same Region, while W1 might be initiated by a host that is farther away. The best way to determine the final value is to perform a read after both writes have been acknowledged.


## Related services

After you load your data into Amazon S3, you can use it with other AWS services. The following are the services that you might use most frequently:

- **Amazon Elastic Compute Cloud (Amazon EC2)** – Provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 eliminates your need to invest in hardware

upfront, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

- **Amazon EMR** – Helps businesses, researchers, data analysts, and developers easily and cost-effectively process vast amounts of data. Amazon EMR uses a hosted Hadoop framework running on the web-scale infrastructure of Amazon EC2 and Amazon S3.
- **AWS Snow Family** – Helps customers that need to run operations in austere, non-data center environments, and in locations where there's a lack of consistent network connectivity. You can use AWS Snow Family devices to locally and cost-effectively access the storage and compute power of the AWS Cloud in places where an internet connection might not be an option.
- **AWS Transfer Family** – Provides fully managed support for file transfers directly into and out of Amazon S3 or Amazon Elastic File System (Amazon EFS) using Secure Shell (SSH) File Transfer Protocol (SFTP), File Transfer Protocol over SSL (FTPS), and File Transfer Protocol (FTP).

# Accessing Amazon S3

You can work with Amazon S3 in any of the following ways:

## AWS Management Console

The console is a web-based user interface for managing Amazon S3 and AWS resources. If you've signed up for an AWS account, you can access the Amazon S3 console by signing into the AWS Management Console and choosing **S3** from the AWS Management Console home page.

## AWS Command Line Interface

You can use the AWS command line tools to issue commands or build scripts at your system's command line to perform AWS (including S3) tasks.

The AWS Command Line Interface (AWS CLI) provides commands for a broad set of AWS services. The AWS CLI is supported on Windows, macOS, and Linux. To get started, see the *AWS Command Line Interface User Guide*. For more information about the commands for Amazon S3, see s3api and s3control in the *AWS CLI Command Reference*.

## AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and so on).

The AWS SDKs provide a convenient way to create programmatic access to S3 and AWS. Amazon S3 is a REST service. You can send requests to Amazon S3 using the AWS SDK libraries, which wrap the underlying Amazon S3 REST API and simplify your programming tasks. For example, the SDKs take care of tasks such as calculating signatures, cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools for AWS.

Every interaction with Amazon S3 is either authenticated or anonymous. If you are using the AWS SDKs, the libraries compute the signature for authentication from the keys that you provide. For more information about how to make requests to Amazon S3, see Making requests.

## Amazon S3 REST API

The architecture of Amazon S3 is designed to be programming language-neutral, using AWS-supported interfaces to store and retrieve objects. You can access S3 and AWS programmatically by using the Amazon S3 REST API. The REST API is an HTTP interface to Amazon S3. With the REST API, you use standard HTTP requests to create, fetch, and delete buckets and objects.

To use the REST API, you can use any toolkit that supports HTTP. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matches the style of standard HTTP usage.

If you make direct REST API calls in your application, you must write the code to compute the signature and add it to the request. For more information about how to make requests to Amazon S3, see Making requests.

> **Note**
>
> SOAP API support over HTTP is deprecated, but it is still available over HTTPS. Newer Amazon S3 features are not supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

# Paying for Amazon S3

Pricing for Amazon S3 is designed so that you don't have to plan for the storage requirements of your application. Most storage providers require you to purchase a predetermined amount of storage and network transfer capacity. In this scenario, if you exceed that capacity, your service is shut off or you are charged high overage fees. If you do not exceed that capacity, you pay as though you used it all.

Amazon S3 charges you only for what you actually use, with no hidden fees and no overage charges. This model gives you a variable-cost service that can grow with your business while giving you the cost advantages of the AWS infrastructure. For more information, see Amazon S3 Pricing.

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. However, you are charged only for the services that you use. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS free tier.

To see your bill, go to the Billing and Cost Management Dashboard in the AWS Billing and Cost Management console. To learn more about AWS account billing, see the *AWS Billing User Guide*. If you have questions concerning AWS billing and AWS accounts, contact AWS Support.

# PCI DSS compliance

Amazon S3 supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

# Getting started with Amazon S3

You can get started with Amazon S3 by working with buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to the bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Video: Getting started with Amazon S3**

**Prerequisites**

Before you begin, confirm that you've completed the steps in Prerequisite: Setting up Amazon S3.

**Topics**

# Prerequisite: Setting up Amazon S3

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. You are charged only for the services that you use.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

To set up Amazon S3, use the steps in the following sections.

When you sign up for AWS and set up Amazon S3, you can optionally change the display language in the AWS Management Console. For more information, see Changing the language of the AWS Management Console in the *AWS Management Console Getting Started Guide*.

**Topics**



## Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

**To sign up for an AWS account**

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

## Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

### Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

### Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

### Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

**Assign access to additional users**

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

# Step 1: Create your first S3 bucket

After you sign up for AWS, you're ready to create a bucket in Amazon S3 using the AWS Management Console. Every object in Amazon S3 is stored in a *bucket*. Before you can store data in Amazon S3, you must create a bucket.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

> **Note**
>
> You are not charged for creating a bucket. You are charged only for storing objects in the bucket and for transferring objects in and out of the bucket. The charges that you incur through following the examples in this guide are minimal (less than $1). For more information about storage charges, see Amazon S3 pricing.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   > **Note**
   >
   > To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **General purpose**.
7. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, dots (.), and hyphens (-). For best compatibility, we recommend that you avoid using dots (.) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.

   After you create the bucket, you cannot change its name. For more information about naming buckets, see Bucket naming rules.

   > **Important**
   >
   > Avoid including sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

8. AWS Management Console allows you to copy an existing bucket's settings to your new bucket. If you do not want to copy the settings of an existing bucket, skip to the next step.

   > **Note**
   >
   > This option:
   >
   > - Is not available in the AWS CLI and is only available in console
   > - Is not available for directory buckets
   > - Does not copy the bucket policy from the existing bucket to the new bucket

   To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you would like to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window re-opens.

   Under **Copy settings from existing bucket**, you will now see the name of the bucket you selected. You will also see a **Restore defaults** option that you can use to remove the copied bucket settings. Review the remaining bucket settings, on the **Create bucket** page. You will see that they now match the settings of the bucket that you selected. You can skip to the final step.

9. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   **ACLs disabled**

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

     By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

  If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

10. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

    By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

    > **Note**
    >
    > To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

11. (Optional) Under **Bucket Versioning**, you can choose if you wish to keep variants of objects in your bucket. For more information about versioning, see Using versioning in S3 buckets.

    To disable or enable versioning on your bucket, choose either **Disable** or **Enable**.

12. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs used to categorize storage.

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

13. Under **Default encryption**, choose **Edit**.

14. To configure default encryption, under **Encryption type**, choose one of the following:

    - **Amazon S3 managed key (SSE-S3)**
    - **AWS Key Management Service key (SSE-KMS)**

    > **Important**
    >
    > If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

    Buckets and new objects are encrypted with server-side encryption with an **Amazon S3 managed key** as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

    For more information about using Amazon S3 server-side encryption to encrypt your data, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

15. If you chose **AWS Key Management Service key (SSE-KMS)**, do the following:

    a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

      Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

    - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.

- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN. For more information on cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information on SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS).
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS).

b. When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

   To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

16. (Optional) If you want to enable S3 Object Lock, do the following:

    a. Choose **Advanced settings**.

> **Important**
>
> Enabling Object Lock also enables versioning for the bucket. After enabling you must configure the Object Lock default retention and legal hold settings to protect new objects from being deleted or overwritten.

b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

For more information, see Using S3 Object Lock.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning` and `s3:PutBucketObjectLockConfiguration`.

17. Choose **Create bucket**.

You've created a bucket in Amazon S3.

**Next step**

To add an object to your bucket, see Step 2: Upload an object to your bucket.

# Step 2: Upload an object to your bucket

After creating a bucket in Amazon S3, you're ready to upload an object to the bucket. An object can be any kind of file: a text file, a photo, a video, and so on.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**To upload an object to a bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to upload your object to.
3. On the **Objects** tab for your bucket, choose **Upload**.
4. Under **Files and folders**, choose **Add files**.
5. Choose a file to upload, and then choose **Open.**
6. Choose **Upload**.

You've successfully uploaded an object to your bucket.

**Next step**

To view your object, see Step 3: Download an object.

# Step 3: Download an object

After you upload an object to a bucket, you can view information about your object and download the object to your local computer.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

## Using the S3 console

This section explains how to use the Amazon S3 console to download an object from an S3 bucket.

> **Note**
>
> - You can download only one object at a time.
> - If you use the Amazon S3 console to download an object whose key name ends with a period (`.`), the period is removed from the key name of the downloaded object. To retain

the period at the end of the name of the downloaded object, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

**To download an object from an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to download an object from.
3. You can download an object from an S3 bucket in any of the following ways:
   - Select the check box next to the object, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.
   - If you want to download a specific version of the object, turn on **Show versions** (located next to the search box). Select the check box next to the version of the object that you want, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.

You've successfully downloaded your object.

**Next step**

To copy and paste your object within Amazon S3, see Step 4: Copy your object to a folder.

# Step 4: Copy your object to a folder

You've already added an object to a bucket and downloaded the object. Now, you create a folder and copy the object and paste it into the folder.

**Note**

For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**To copy an object to a folder**

1. In the **Buckets** list, choose your bucket name.

2. Choose **Create folder** and configure a new folder:

   a. Enter a folder name (for example, `favorite-pics`).

   b. For the folder encryption setting, choose **Disable**.

   c. Choose **Save**.

3. Navigate to the Amazon S3 bucket or folder that contains the objects that you want to copy.
4. Select the check box to the left of the names of the objects that you want to copy.
5. Choose **Actions** and choose **Copy** from the list of options that appears.

   Alternatively, choose **Copy** from the options in the upper right.

6. Choose the destination folder:

   a. Choose **Browse S3**.

   b. Choose the option button to the left of the folder name.

      To navigate into a folder and choose a subfolder as your destination, choose the folder name.

   c. Choose **Choose destination**.

   The path to your destination folder appears in the **Destination** box. In **Destination**, you can alternately enter your destination path, for example, s3://*bucket-name*/*folder-name*/.

7. In the bottom right, choose **Copy**.

   Amazon S3 copies your objects to the destination folder.

**Next step**

To delete an object and a bucket in Amazon S3, see Step 5: Delete your objects and bucket.

# Step 5: Delete your objects and bucket

When you no longer need an object or a bucket, we recommend that you delete them to prevent further charges. If you completed this getting started walkthrough as a learning exercise, and you don't plan to use your bucket or objects, we recommend that you delete your bucket and objects so that charges no longer accrue.

Before you delete your bucket, empty the bucket or delete the objects in the bucket. After you delete your objects and bucket, they are no longer available.

If you want to continue to use the same bucket name, we recommend that you delete the objects or empty the bucket, but don't delete the bucket. After you delete a bucket, the name becomes available to reuse. However, another AWS account might create a bucket with the same name before you have a chance to reuse it.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



## Deleting an object

If you want to choose which objects you delete without emptying all the objects from your bucket, you can delete an object.

1. In the **Buckets** list, choose the name of the bucket that you want to delete an object from.
2. Select the object that you want to delete.
3. Choose **Delete** from the options in the upper right.
4. On the **Delete objects** page, type **`delete`** to confirm deletion of your objects.
5. Choose **Delete objects**.

## Emptying your bucket

If you plan to delete your bucket, you must first empty your bucket, which deletes all the objects in the bucket.

**To empty a bucket**

1. In the **Buckets** list, select the bucket that you want to empty, and then choose **Empty**.
2. To confirm that you want to empty the bucket and delete all the objects in it, in **Empty bucket**, type **`permanently delete`**.

   > **Important**
   >
   > Emptying the bucket cannot be undone. Objects added to the bucket while the empty bucket action is in progress will be deleted.

3. To empty the bucket and delete all the objects in it, and choose **Empty**.

   An **Empty bucket: Status** page opens that you can use to review a summary of failed and successful object deletions.
4. To return to your bucket list, choose **Exit**.

## Deleting your bucket

After you empty your bucket or delete all the objects from your bucket, you can delete your bucket.

1. To delete a bucket, in the **Buckets** list, select the bucket.
2. Choose **Delete**.
3. To confirm deletion, in **Delete bucket**, type the name of the bucket.

   > **Important**
   >
   > Deleting a bucket cannot be undone. Bucket names are unique. If you delete your bucket, another AWS user can use the name. If you want to continue to use the same bucket name, don't delete your bucket. Instead, empty and keep the bucket.

4. To delete your bucket, choose **Delete bucket**.

# Next steps

In the preceding examples, you learned how to perform some basic Amazon S3 tasks.

The following topics explain the learning paths that you can use to gain a deeper understanding of Amazon S3 so that you can implement it in your applications.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



## Understand common use cases

You can use Amazon S3 to support your specific use case. The AWS Solutions Library and AWS Blog provide use-case specific information and tutorials. The following are some common use cases for Amazon S3:

- **Backup and storage** – Use Amazon S3 storage management features to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.
- **Application hosting** – Deploy, install, and manage web applications that are reliable, highly scalable, and low-cost. For example, you can configure your Amazon S3 bucket to host a static website. For more information, see Hosting a static website using Amazon S3.
- **Media hosting** – Build a highly available infrastructure that hosts video, photo, or music uploads and downloads.
- **Software delivery** – Host your software applications for customers to download.

## Control access to your buckets and objects

Amazon S3 provides a variety of security features and tools. For an overview, see Access Management.

By default, S3 buckets and the objects in them are private. You have access only to the S3 resources that you create. You can use the following features to grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources.

- S3 Block Public Access – Block public access to S3 buckets and objects. By default, Block Public Access settings are turned on at the bucket level.
- AWS Identity and Access Management (IAM) identities – Use IAM or AWS IAM Identity Center to create IAM identities in your AWS account to manage access to your Amazon S3 resources. For example, you can use IAM with Amazon S3 to control the type of access that a user or group of users has to an Amazon S3 bucket that your AWS account owns. For more information about IAM identities and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- Access control lists (ACLs) – Grant read and write permissions for individual buckets and objects to authorized users. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM user policies for access control instead of ACLs. Policies are a simplified and more flexible access-control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources. For more information about the specific cases when you'd use ACLs instead of resource-based policies or IAM user policies, see Identity and Access Management for Amazon S3.
- S3 Object Ownership – Take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable or enable ACLs. By default, ACLs are disabled. With ACLs disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively by using access-management policies.
- IAM Access Analyzer for S3 – Evaluate and monitor your S3 bucket access policies, ensuring that the policies provide only the intended access to your S3 resources.

## Manage and monitor your storage

- Managing your storage – After you create buckets and upload objects in Amazon S3, you can manage your object storage. For example, you can use S3 Versioning and S3 Replication for disaster recovery, S3 Lifecycle to manage storage costs, and S3 Object Lock to meet compliance requirements.

- Monitoring your storage – Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. You can monitor storage activity and costs. Also, we recommend that you collect monitoring data from all the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs.
- Analytics and insights – You can also use analytics and insights in Amazon S3 to understand, analyze, and optimize your storage usage. For example, use Amazon S3 Storage Lens to understand, analyze, and optimize your storage. S3 Storage Lens provides 29+ usage and activity metrics and interactive dashboards to aggregate data for your entire organization, specific accounts, Regions, buckets, or prefixes. Use Storage Class Analysis to analyze storage access patterns to decide when it's time to move your data to a more cost-effective storage class.

## Develop with Amazon S3

Amazon S3 is a REST service. You can send requests to Amazon S3 using the REST API or the AWS SDK libraries, which wrap the underlying Amazon S3 REST API, simplifying your programming tasks. You can also use the AWS Command Line Interface (AWS CLI) to make Amazon S3 API calls. For more information, see Making requests.

The Amazon S3 REST API is an HTTP interface to Amazon S3. With the REST API, you use standard HTTP requests to create, fetch, and delete buckets and objects. To use the REST API, you can use any toolkit that supports HTTP. You can even use a browser to fetch objects, as long as they are anonymously readable. For more information, see Developing with Amazon S3 using the REST API.

To help you build applications using the language of your choice, we provide the following resources.

**AWS CLI**

You can access the features of Amazon S3 using the AWS CLI. To download and configure the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

The AWS CLI provides two tiers of commands for accessing Amazon S3: High-level (s3) commands and API-level (s3api and s3control) commands. The high-level S3 commands simplify performing common tasks, such as creating, manipulating, and deleting objects and buckets. The s3api and s3control commands expose direct access to all Amazon S3 API operations, which you can use to carry out advanced operations that might not be possible with the high-level commands alone.

For a list of Amazon S3 AWS CLI commands, see s3, s3api, and s3control.

**AWS SDKs and Explorers**

You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying REST API. The AWS Mobile SDKs and the Amplify JavaScript library are also available for building connected mobile and web applications using AWS.

In addition to the AWS SDKs, AWS Explorers are available for Visual Studio and Eclipse for Java IDE. In this case, the SDKs and the explorers are bundled together as AWS Toolkits.

For more information, see Developing with Amazon S3 using the AWS SDKs.

**Sample Code and Libraries**

The AWS Developer Center and AWS Code Sample Catalog have sample code and libraries written especially for Amazon S3. You can use these code samples to understand how to implement the Amazon S3 API. You can also view the *Amazon Simple Storage Service API Reference* to understand the Amazon S3 API operations in detail.

## Learn from tutorials

You can get started with step-by-step tutorials to learn more about Amazon S3. These tutorials are intended for a lab-type environment, and they use fictitious company names, user names, and so on. Their purpose is to provide general guidance. They are not intended for direct use in a production environment without careful review and adaptation to meet the unique needs of your organization's environment.

### Getting started

- Tutorial: Storing and retrieving a file with Amazon S3
- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the Amazon S3 Glacier storage classes

### Optimizing storage costs

- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the Amazon S3 Glacier storage classes
- Tutorial: Optimizing costs and gaining visibility into usage with S3 Storage Lens

### Managing storage

- Tutorial: Getting started with Amazon S3 Multi-Region Access Points
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

### Hosting videos and websites

- Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53
- Tutorial: Configuring a static website on Amazon S3
- Tutorial: Configuring a static website using a custom domain registered with Route 53

### Processing data

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved
- Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

### Protecting data

- Tutorial: Checking the integrity of data in Amazon S3 with additional checksums
- Tutorial: Replicating data within and between AWS Regions using S3 Replication
- Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

## Explore training and support

You can learn from AWS experts to advance your skills and get expert assistance achieving your objectives.

- **Training** – Training resources provide a hands-on approach to learning Amazon S3. For more information, see AWS training and certification and AWS online tech talks.
- **Discussion Forums** – On the forum, you can review posts to understand what you can and can't do with Amazon S3. You can also post your questions. For more information, see Discussion Forums.
- **Technical Support** – If you have further questions, you can contact Technical Support.

# Tutorials

The following tutorials present complete end-to-end procedures for common Amazon S3 tasks. These tutorials are intended for a lab-type environment, and they use fictitious company names, user names, and so on. Their purpose is to provide general guidance. They are not intended for direct use in a production environment without careful review and adaptation to meet the unique needs of your organization's environment.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

## Getting started

- Tutorial: Storing and retrieving a file with Amazon S3
- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the Amazon S3 Glacier storage classes

## Optimizing storage costs

- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the Amazon S3 Glacier storage classes
- Tutorial: Optimizing costs and gaining visibility into usage with S3 Storage Lens

## Managing storage

- Tutorial: Getting started with Amazon S3 Multi-Region Access Points
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

## Hosting videos and websites

- Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53
- Tutorial: Configuring a static website on Amazon S3
- Tutorial: Configuring a static website using a custom domain registered with Route 53

## Processing data

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved
- Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

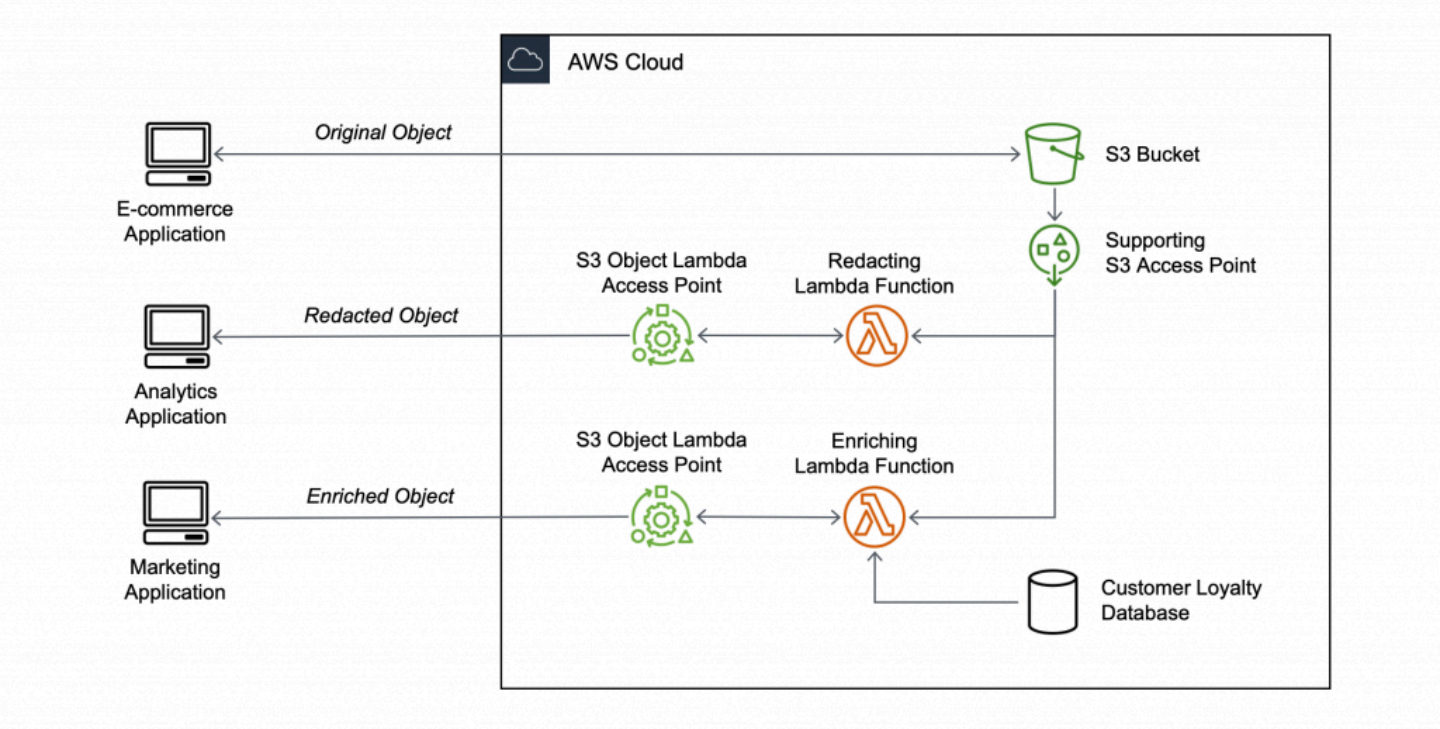
## Protecting data

- Tutorial: Checking the integrity of data in Amazon S3 with additional checksums
- Tutorial: Replicating data within and between AWS Regions using S3 Replication
- Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

# Tutorial: Transforming data for your application with S3 Object Lambda

When you store data in Amazon S3, you can easily share it for use by multiple applications. However, each application might have unique data format requirements, and might need modification or processing of your data for a specific use case. For example, a dataset created by an ecommerce application might include personally identifiable information (PII). When the same data is processed for analytics, this PII is not needed and should be redacted. However, if the same dataset is used for a marketing campaign, you might need to enrich the data with additional details, such as information from the customer loyalty database.

With S3 Object Lambda, you can add your own code to process data retrieved from S3 before returning it to an application. Specifically, you can configure an AWS Lambda function and attach it to an S3 Object Lambda Access Point. When an application sends standard S3 GET requests through the S3 Object Lambda Access Point, the specified Lambda function is invoked to process any data retrieved from an S3 bucket through the supporting S3 access point. Then, the S3 Object Lambda Access Point returns the transformed result back to the application. You can author and execute your own custom Lambda functions, tailoring the S3 Object Lambda data transformation to your specific use case, all with no changes required to your applications.




## Objective

In this tutorial, you learn how to add custom code to standard S3 GET requests to modify the requested object retrieved from S3 so that the object suit the needs of the requesting client or application. Specifically, you learn how to transform all the text in the original object stored in S3 to uppercase through S3 Object Lambda.

> **Note**
>
> This tutorial uses Python code to transform the data, for examples using other AWS SDKs see Transform data for your application with S3 Object Lambda in the AWS SDK Code Examples Library.

**Topics**



## Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions. You also must install Python version 3.8 or later.

**Substeps**

- Create an IAM user with permissions in your AWS account (console)
- Install Python 3.8 or later on your local machine

### Create an IAM user with permissions in your AWS account (console)

You can create an IAM user for the tutorial. To complete this tutorial, your IAM user must attach the following IAM policies to access relevant AWS resources and perform specific actions. For more information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*.

Your IAM user requires the following policies:

- AmazonS3FullAccess – Grants permissions to all Amazon S3 actions, including permissions to create and use an Object Lambda Access Point.
- AWSLambda_FullAccess – Grants permissions to all Lambda actions.
- IAMFullAccess – Grants permissions to all IAM actions.

- IAMAccessAnalyzerReadOnlyAccess – Grants permissions to read all access information provided by IAM Access Analyzer.
- CloudWatchLogsFullAccess – Grants full access to CloudWatch Logs.

> **Note**
>
> For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*. This tutorial also uses full-access AWS managed policies. For production use, we recommend that you instead grant only the minimum permissions necessary for your use case, in accordance with security best practices.

## Install Python 3.8 or later on your local machine

Use the following procedure to install Python 3.8 or later on your local machine. For more installation instructions, see the Downloading Python page in the *Python Beginners Guide*.

1. Open your local terminal or shell and run the following command to determine whether Python is already installed, and if so, which version is installed.

   ```
   python --version
   ```

2. If you don't have Python 3.8 or later, download the official installer of Python 3.8 or later that's suitable for your local machine.
3. Run the installer by double-clicking the downloaded file, and follow the steps to complete the installation.

   For **Windows users**, choose **Add Python 3.X to PATH** in the installation wizard before choosing **Install Now**.
4. Restart your terminal by closing and reopening it.
5. Run the following command to verify that Python 3.8 or later is installed correctly.

For **macOS users**, run this command:

```
python3 --version
```

For **Windows users**, run this command:

```
python --version
```

6. Run the following command to verify that the pip3 package manager is installed. If you see a pip version number and python 3.8 or later in the command response, that means the pip3 package manager is installed successfully.

```
pip --version
```

## Step 1: Create an S3 bucket

Create a bucket to store the original data that you plan to transform.

**To create a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name (for example, **`tutorial-bucket`**) for your bucket.

   For more information about naming buckets in Amazon S3, see Bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.

   For more information about the bucket Region, see Buckets overview.
6. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

We recommend that you keep all Block Public Access settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

7. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a bucket.

8. Choose **Create bucket**.

## Step 2: Upload a file to the S3 bucket

Upload a text file to the S3 bucket. This text file contains the original data that you will transform to uppercase later in this tutorial.

For example, you can upload a `tutorial.txt` file that contains the following text:

```
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

**To upload a file to a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**. For example, you can upload the `tutorial.txt` file example mentioned earlier.
7. Choose **Upload**.

# Step 3: Create an S3 access point

To use an S3 Object Lambda Access Point to access and transform the original data, you must create an S3 access point and associate it with the S3 bucket that you created in Step 1. The access point must be in the same AWS Region as the objects that you want to transform.

Later in this tutorial, you'll use this access point as a supporting access point for your Object Lambda Access Point.

**To create an access point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. On the **Access Points** page, choose **Create access point**.
4. In the **Access point name** field, enter the name (for example, **`tutorial-access-point`**) for the access point.

   For more information about naming access points, see Rules for naming Amazon S3 access points.
5. In the **Bucket name** field, enter the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**). S3 attaches the access point to this bucket.

   (Optional) You can choose **Browse S3** to browse and search the buckets in your account. If you choose **Browse S3**, choose the desired bucket, and then choose **Choose path** to populate the **Bucket name** field with that bucket's name.
6. For **Network origin**, choose **Internet**.

   For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.
7. By default, all Block Public Access settings are turned on for your access point. We recommend that you keep **Block *all* public access** enabled.

   For more information, see Managing public access to access points.
8. For all other access point settings, keep the default settings.

   (Optional) You can modify the access point settings to support your use case. For this tutorial, we recommend keeping the default settings.

(Optional) If you need to manage access to your access point, you can specify an access point policy. For more information, see Access point policy examples.

9. Choose **Create access point**.

## Step 4: Create a Lambda function

To transform original data, create a Lambda function for use with your S3 Object Lambda Access Point.

**Substeps**

- Write Lambda function code and create a deployment package with a virtual environment
- Create a Lambda function with an execution role (console)
- Deploy your Lambda function code with .zip file archives and configure the Lambda function (console)

### Write Lambda function code and create a deployment package with a virtual environment

1. On your local machine, create a folder with the folder name `object-lambda` for the virtual environment to use later in this tutorial.
2. In the `object-lambda` folder, create a file with a Lambda function that changes all text in the original object to uppercase. For example, you can use the following function written in Python. Save this function in a file named `transform.py`.

```
import boto3
import requests
from botocore.config import Config

# This function capitalizes all text in the original object
def lambda_handler(event, context):
    object_context = event["getObjectContext"]
    # Get the presigned URL to fetch the requested original object
    # from S3
    s3_url = object_context["inputS3Url"]
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]
```

```
    # Get the original S3 object using the presigned URL
    response = requests.get(s3_url)
    original_object = response.content.decode("utf-8")

    # Transform all text in the original object to uppercase
    # You can replace it with your custom code based on your use case
    transformed_object = original_object.upper()

    # Write object back to S3 Object Lambda
    s3 = boto3.client('s3', config=Config(signature_version='s3v4'))
    # The WriteGetObjectResponse API sends the transformed data
    # back to S3 Object Lambda and then to the user
    s3.write_get_object_response(
        Body=transformed_object,
        RequestRoute=request_route,
        RequestToken=request_token)

    # Exit the Lambda function: return the status code
    return {'status_code': 200}
```

**Note**

The preceding example Lambda function loads the entire requested object into memory before transforming it and returning it to the client. Alternatively, you can stream the object from S3 to avoid loading the entire object into memory. This approach can be useful when working with large objects. For more information about streaming responses with Object Lambda Access Points, see the streaming examples in Working with `GetObject` requests in Lambda.

When you're writing a Lambda function for use with an S3 Object Lambda Access Point, the function is based on the input event context that S3 Object Lambda provides to the Lambda function. The event context provides information about the request being made in the event passed from S3 Object Lambda to Lambda. It contains the parameters that you use to create the Lambda function.

The fields used to create the preceding Lambda function are as follows:

The field of `getObjectContext` means the input and output details for connections to Amazon S3 and S3 Object Lambda. It has the following fields:

- `inputS3Url` – A presigned URL that the Lambda function can use to download the original object from the supporting access point. By using a presigned URL, the Lambda function doesn't need to have Amazon S3 read permissions to retrieve the original object and can only access the object processed by each invocation.
- `outputRoute` – A routing token that is added to the S3 Object Lambda URL when the Lambda function calls `WriteGetObjectResponse` to send back the transformed object.
- `outputToken` – A token used by S3 Object Lambda to match the `WriteGetObjectResponse` call with the original caller when sending back the transformed object.

For more information about all the fields in the event context, see Event context format and usage and Writing Lambda functions for S3 Object Lambda Access Points.

3. In your local terminal, enter the following command to install the `virtualenv` package:

```
python -m pip install virtualenv
```

4. In your local terminal, open the `object-lambda` folder that you created earlier, and then enter the following command to create and initialize a virtual environment called `venv`.

```
python -m virtualenv venv
```

5. To activate the virtual environment, enter the following command to execute the `activate` file from the environment's folder:

   For **macOS users**, run this command:

```
source venv/bin/activate
```

   For **Windows users**, run this command:

```
.\venv\Scripts\activate
```

Now, your command prompt changes to show **(venv)**, indicating that the virtual environment is active.

6. To install the required libraries, run the following commands line by line in the `venv` virtual environment.

   These commands install updated versions of the dependencies of your `lambda_handler` Lambda function. These dependencies are the AWS SDK for Python (Boto3) and the requests module.

   ```
   pip3 install boto3
   ```

   ```
   pip3 install requests
   ```

7. To deactivate the virtual environment, run the following command:

   ```
   deactivate
   ```

8. To create a deployment package with the installed libraries as a `.zip` file named `lambda.zip` at the root of the `object-lambda` directory, run the following commands line by line in your local terminal.

   > **Tip**
   >
   > The following commands might need to be adjusted to work in your particular environment. For example, a library might appear in `site-packages` or `dist-packages`, and the first folder might be `lib` or `lib64`. Also, the `python` folder might be named with a different Python version. To locate a specific package, use the `pip show` command.

   For **macOS users**, run these commands:

   ```
   cd venv/lib/python3.8/site-packages
   ```

   ```
   zip -r ../../../../lambda.zip .
   ```

   For **Windows users**, run these commands:

```
cd .\venv\Lib\site-packages\
```

```
powershell Compress-Archive * ../../../lambda.zip
```

The last command saves the deployment package to the root of the `object-lambda` directory.

9. Add the function code file `transform.py` to the root of your deployment package.

   For **macOS users**, run these commands:

   ```
   cd ../../../../
   ```

   ```
   zip -g lambda.zip transform.py
   ```

   For **Windows users**, run these commands:

   ```
   cd ..\..\..\
   ```

   ```
   powershell Compress-Archive -update transform.py lambda.zip
   ```

   After you complete this step, you should have the following directory structure:

   ```
   lambda.zip$
     # transform.py
     # __pycache__
     | boto3/
     # certifi/
     # pip/
     # requests/
     ...
   ```

## Create a Lambda function with an execution role (console)

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.

2. In the left navigation pane, choose **Functions**.
3. Choose **Create function**.
4. Choose **Author from scratch**.
5. Under **Basic information**, do the following:

   a. For **Function name**, enter **`tutorial-object-lambda-function`**.

   b. For **Runtime**, choose **Python 3.8** or a later version.
6. Expand the **Change default execution role** section. Under **Execution role**, choose **Create a new role with basic Lambda permissions**.

   In Step 5 later in this tutorial, you attach the **AmazonS3ObjectLambdaExecutionRolePolicy** to this Lambda function's execution role.
7. Keep the remaining settings set to the defaults.
8. Choose **Create function**.

## Deploy your Lambda function code with .zip file archives and configure the Lambda function (console)

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the Lambda function that you created earlier (for example, **`tutorial-object-lambda-function`**).
3. On the Lambda function's details page, choose the **Code** tab. In the **Code Source** section, choose **Upload from** and then **.zip file**.
4. Choose **Upload** to select your local `.zip` file.
5. Choose the `lambda.zip` file that you created earlier, and then choose **Open**.
6. Choose **Save**.
7. In the **Runtime settings** section, choose **Edit**.
8. On the **Edit runtime settings** page, confirm that **Runtime** is set to **Python 3.8** or a later version.
9. To tell the Lambda runtime which handler method in your Lambda function code to invoke, enter **`transform.lambda_handler`** for **Handler**.

When you configure a function in Python, the value of the handler setting is the file name and the name of the handler module, separated by a dot. For example, `transform.lambda_handler` calls the `lambda_handler` method defined in the `transform.py` file.

10. Choose **Save**.
11. (Optional) On your Lambda function's details page, choose the **Configuration** tab. In the left navigation pane, choose **General configuration**, then choose **Edit**. In the **Timeout** field, enter **1** min **0** sec. Keep the remaining settings set to the defaults, and choose **Save**.

    **Timeout** is the amount of time that Lambda allows a function to run for an invocation before stopping it. The default is 3 seconds. The maximum duration for a Lambda function used by S3 Object Lambda is 60 seconds. Pricing is based on the amount of memory configured and the amount of time that your code runs.

## Step 5: Configure an IAM policy for your Lambda function's execution role

To enable your Lambda function to provide customized data and response headers to the `GetObject` caller, your Lambda function's execution role must have IAM permissions to call the `WriteGetObjectResponse` API.

**To attach an IAM policy to your Lambda function role**

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
3. On your Lambda function's details page, choose the **Configuration** tab, and then choose **Permissions** in the left navigation pane.
4. Under **Execution role**, choose the link of the **Role name**. The IAM console opens.
5. On the IAM console's **Summary** page for your Lambda function's execution role, choose the **Permissions** tab. Then, from the **Add Permissions** menu, choose **Attach policies**.

6. On the **Attach Permissions** page, enter **`AmazonS3ObjectLambdaExecutionRolePolicy`** in the search box to filter the list of policies. Select the check box next to the name of the **AmazonS3ObjectLambdaExecutionRolePolicy** policy.
7. Choose **Attach policies**.

## Step 6: Create an S3 Object Lambda Access Point

An S3 Object Lambda Access Point provides the flexibility to invoke a Lambda function directly from an S3 GET request so that the function can process data retrieved from an S3 access point. When creating and configuring an S3 Object Lambda Access Point, you must specify the Lambda function to invoke and provide the event context in JSON format as custom parameters for Lambda to use.

**To create an S3 Object Lambda Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
4. For **Object Lambda Access Point name**, enter the name that you want to use for the Object Lambda Access Point (for example, **`tutorial-object-lambda-accesspoint`**).
5. For **Supporting Access Point**, enter or browse to the standard access point that you created in Step 3 (for example, **`tutorial-access-point`**), and then choose **Choose supporting Access Point**.
6. For **S3 APIs**, to retrieve objects from the S3 bucket for Lambda function to process, select **GetObject**.
7. For **Invoke Lambda function**, you can choose either of the following two options for this tutorial.
   - Choose **Choose from functions in your account**, and then choose the Lambda function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**) from the **Lambda function** dropdown list.
   - Choose **Enter ARN**, and then enter the Amazon Resource Name (ARN) of the Lambda function that you created in Step 4.
8. For **Lambda function version**, choose **$LATEST** (the latest version of the Lambda function that you created in Step 4).

9. (Optional) If you need your Lambda function to recognize and process GET requests with range and part number headers, select **Lambda function supports requests using range** and **Lambda function supports requests using part numbers**. Otherwise, clear these two check boxes.

   For more information about how to use range or part numbers with S3 Object Lambda, see Working with Range and partNumber headers.

10. (Optional) Under **Payload - *optional***, add JSON text to provide your Lambda function with additional information.

    A payload is optional JSON text that you can provide to your Lambda function as input for all invocations coming from a specific S3 Object Lambda Access Point. To customize the behaviors for multiple Object Lambda Access Points that invoke the same Lambda function, you can configure payloads with different parameters, thereby extending the flexibility of your Lambda function.

    For more information about payload, see Event context format and usage.

11. (Optional) For **Request metrics - *optional***, choose **Disable** or **Enable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate. For more information, see CloudWatch pricing.

12. Under **Object Lambda Access Point policy - *optional***, keep the default setting.

    (Optional) You can set a resource policy. This resource policy grants the `GetObject` API permission to use the specified Object Lambda Access Point.

13. Keep the remaining settings set to the defaults, and choose **Create Object Lambda Access Point**.

## Step 7: View the transformed data

Now, S3 Object Lambda is ready to transform your data for your use case. In this tutorial, S3 Object Lambda transforms all the text in your object to uppercase.

### Substeps

- View the transformed data in your S3 Object Lambda Access Point
- Run a Python script to print the original and transformed data

## View the transformed data in your S3 Object Lambda Access Point

When you request to retrieve a file through your S3 Object Lambda Access Point, you make a `GetObject` API call to S3 Object Lambda. S3 Object Lambda invokes the Lambda function to transform your data, and then returns the transformed data as the response to the standard S3 `GetObject` API call.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. On the **Objects** tab of your S3 Object Lambda Access Point, select the file that has the same name (for example, `tutorial.txt`) as the one that you uploaded to the S3 bucket in Step 2.

   This file should contain all the transformed data.
5. To view the transformed data, choose **Open** or **Download**.

## Run a Python script to print the original and transformed data

You can use S3 Object Lambda with your existing applications. To do so, update your application configuration to use the new S3 Object Lambda Access Point ARN that you created in Step 6 to retrieve data from S3.

The following example Python script prints both the original data from the S3 bucket and the transformed data from the S3 Object Lambda Access Point.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the radio button to the left of the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. Choose **Copy ARN**.
5. Save the ARN for use later.

6. Write a Python script on your local machine to print both the original data (for example, `tutorial.txt`) from your S3 Bucket and the transformed data (for example, `tutorial.txt`) from your S3 Object Lambda Access Point). You can use the following example script.

```
import boto3
from botocore.config import Config

s3 = boto3.client('s3', config=Config(signature_version='s3v4'))

def getObject(bucket, key):
    objectBody = s3.get_object(Bucket = bucket, Key = key)
    print(objectBody["Body"].read().decode("utf-8"))
    print("\n")

print('Original object from the S3 bucket:')
# Replace the two input parameters of getObject() below with
# the S3 bucket name that you created in Step 1 and
# the name of the file that you uploaded to the S3 bucket in Step 2
getObject("tutorial-bucket",
          "tutorial.txt")

print('Object transformed by S3 Object Lambda:')
# Replace the two input parameters of getObject() below with
# the ARN of your S3 Object Lambda Access Point that you saved earlier and
# the name of the file with the transformed data (which in this case is
# the same as the name of the file that you uploaded to the S3 bucket
# in Step 2)
getObject("arn:aws:s3-object-lambda:us-west-2:111122223333:accesspoint/tutorial-
object-lambda-accesspoint",
          "tutorial.txt")
```

7. Save your Python script with a custom name (for example, `tutorial_print.py` ) in the folder (for example, `object-lambda`) that you created in Step 4 on your local machine.

8. In your local terminal, run the following command from the root of the directory (for example, `object-lambda`) that you created in Step 4.

```
python3 tutorial_print.py
```

You should see both the original data and the transformed data (all text as uppercase) through the terminal. For example, you should see something like the following text.

```
Original object from the S3 bucket:
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.

Object transformed by S3 Object Lambda:
AMAZON S3 OBJECT LAMBDA TUTORIAL:
YOU CAN ADD YOUR OWN CODE TO PROCESS DATA RETRIEVED FROM S3 BEFORE
RETURNING IT TO AN APPLICATION.
```

## Step 8: Clean up

If you transformed your data through S3 Object Lambda only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

**Substeps**

- Delete the Object Lambda Access Point
- Delete the S3 access point
- Delete the execution role for your Lambda function
- Delete the Lambda function
- Delete the CloudWatch log group
- Delete the original file in the S3 source bucket
- Delete the S3 source bucket
- Delete the IAM user

### Delete the Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the radio button to the left of the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. Choose **Delete**.

5. Confirm that you want to delete your Object Lambda Access Point by entering its name in the text field that appears, and then choose **Delete**.

## Delete the S3 access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. Navigate to the access point that you created in Step 3 (for example, **`tutorial-access-point`**), and choose the radio button next to the name of the access point.
4. Choose **Delete**.
5. Confirm that you want to delete your access point by entering its name in the text field that appears, and then choose **Delete**.

## Delete the execution role for your Lambda function

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Choose the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
4. On your Lambda function's details page, choose the **Configuration** tab, and then choose **Permissions** in the left navigation pane.
5. Under **Execution role**, choose the link of the **Role name**. The IAM console opens.
6. On the IAM console's **Summary** page of your Lambda function's execution role, choose **Delete role**.
7. In the **Delete role** dialog box, choose **Yes, delete**.

## Delete the Lambda function

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Select the check box to the left of the name of the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).

3. Choose **Actions**, and then choose **Delete**.
4. In the **Delete function** dialog box, choose **Delete**.

## Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Log groups**.
3. Find the log group whose name ends with the Lambda function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
4. Select the check box to the left of the name of the log group.
5. Choose **Actions**, and then choose **Delete log group(s)**.
6. In the **Delete log group(s)** dialog box, choose **Delete**.

## Delete the original file in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the original file to in Step 2 (for example, **`tutorial-bucket`**).
4. Select the check box to the left of the name of the object that you want to delete (for example, `tutorial.txt`).
5. Choose **Delete**.
6. On the **Delete objects** page, in the **Permanently delete objects?** section, confirm that you want to delete this object by entering **`permanently delete`** in the text box.
7. Choose **Delete objects**.

## Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the radio button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

### Delete the IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Users**, and then select the check box next to the user name that you want to delete.
3. At the top of the page, choose **Delete**.
4. In the **Delete *`user name`*?** dialog box, enter the user name in the text input field to confirm the deletion of the user. Choose **Delete**.

## Next steps

After completing this tutorial, you can customize the Lambda function for your use case to modify the data returned by standard S3 GET requests.

The following is a list of common use cases for S3 Object Lambda:

- Masking sensitive data for security and compliance.

  For more information, see Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.
- Filtering certain rows of data to deliver specific information.
- Augmenting data with information from other services or databases.
- Converting across data formats, such as converting XML to JSON for application compatibility.
- Compressing or decompressing files as they are being downloaded.
- Resizing and watermarking images.

  For more information, see Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved.

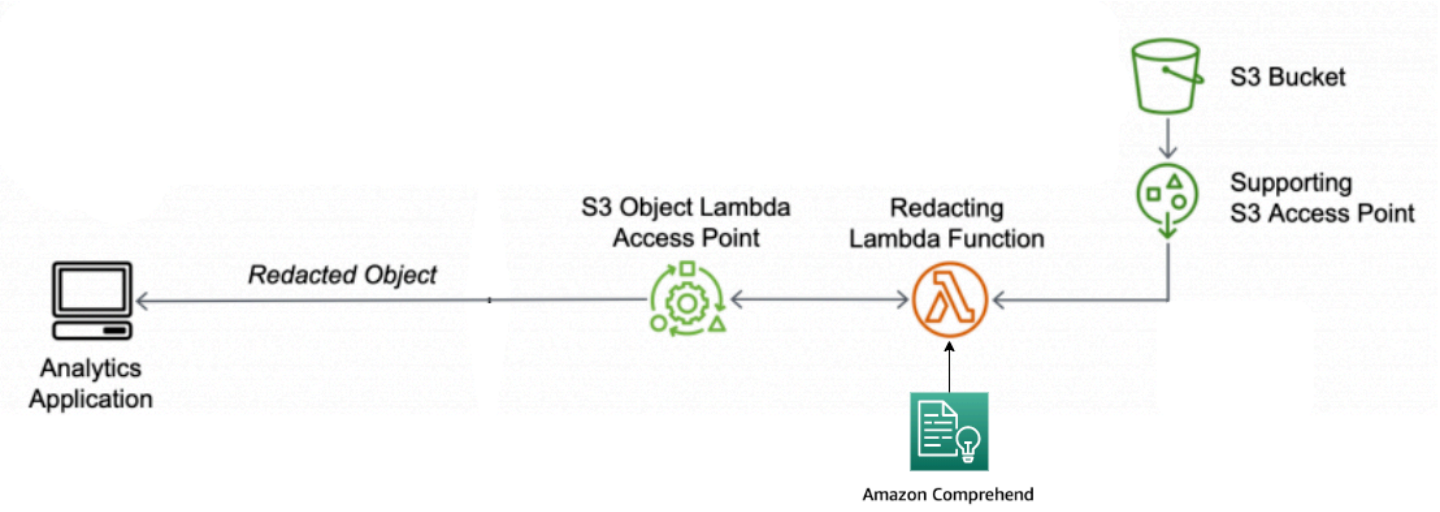
- Implementing custom authorization rules to access data.

For more information about S3 Object Lambda, see Transforming objects with S3 Object Lambda.

# Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend

When you're using Amazon S3 for shared datasets for multiple applications and users to access, it's important to restrict privileged information, such as personally identifiable information (PII), to only authorized entities. For example, when a marketing application uses some data containing PII, it might need to first mask PII data to meet data privacy requirements. Also, when an analytics application uses a production order inventory dataset, it might need to first redact customer credit card information to prevent unintended data leakage.

With S3 Object Lambda and a prebuilt AWS Lambda function powered by Amazon Comprehend, you can protect PII data retrieved from S3 before returning it to an application. Specifically, you can use the prebuilt Lambda function as a redacting function and attach it to an S3 Object Lambda Access Point. When an application (for example, an analytics application) sends standard S3 GET requests, these requests made through the S3 Object Lambda Access Point invoke the prebuilt redacting Lambda function to detect and redact PII data retrieved from an S3 bucket through a supporting S3 access point. Then, the S3 Object Lambda Access Point returns the redacted result back to the application.




In the process, the prebuilt Lambda function uses Amazon Comprehend, a natural language processing (NLP) service, to capture variations in how PII is represented, regardless of how PII exists

in text (such as numerically or as a combination of words and numbers). Amazon Comprehend can even use context in the text to understand if a 4-digit number is a PIN, the last four numbers of a Social Security number (SSN), or a year. Amazon Comprehend processes any text file in UTF-8 format and can protect PII at scale without affecting accuracy. For more information, see What is Amazon Comprehend? in the *Amazon Comprehend Developer Guide*.

**Objective**

In this tutorial, you learn how to use S3 Object Lambda with the prebuilt Lambda function `ComprehendPiiRedactionS3ObjectLambda`. This function uses Amazon Comprehend to detect PII entities. It then redacts these entities by replacing them with asterisks. By redacting PII, you conceal sensitive data, which can help with security and compliance.

You also learn how to use and configure a prebuilt AWS Lambda function in the AWS Serverless Application Repository to work together with S3 Object Lambda for easy deployment.

**Topics**



## Prerequisites: Create an IAM user with permissions

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management user (IAM user) with correct permissions.

You can create an IAM user for the tutorial. To complete this tutorial, your IAM user must attach the following IAM policies to access relevant AWS resources and perform specific actions.

> **Note**
>
> For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*. This tutorial also uses full-access policies. For production use, we recommend that you instead grant only the minimum permissions necessary for your use case, in accordance with security best practices.

Your IAM user requires the following AWS managed policies:

- AmazonS3FullAccess – Grants permissions to all Amazon S3 actions, including permissions to create and use an Object Lambda Access Point.
- AWSLambda_FullAccess – Grants permissions to all Lambda actions.
- AWSCloudFormationFullAccess – Grants permissions to all AWS CloudFormation actions.
- IAMFullAccess – Grants permissions to all IAM actions.
- IAMAccessAnalyzerReadOnlyAccess – Grants permissions to read all access information provided by IAM Access Analyzer.

You can directly attach these existing policies when creating an IAM user. For more information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*.

In addition, your IAM user requires a customer managed policy. To grant the IAM user permissions to all AWS Serverless Application Repository resources and actions, you must create an IAM policy and attach the policy to the IAM user.

**To create and attach an IAM policy to your IAM user**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create policy**.

4. On the **Visual editor** tab, for **Service**, choose **Choose a service**. Then, choose **Serverless Application Repository**.
5. For **Actions**, under **Manual actions**, select **All Serverless Application Repository actions (serverlessrepo:*)** for this tutorial.

   As a security best practice, you should allow permissions to only those actions and resources that a user needs, based on your use case. For more information, see Security best practices in IAM in the *IAM User Guide*.
6. For **Resources**, choose **All resources** for this tutorial.

   As a best practice, you should define permissions for only specific resources in specific accounts. Alternatively, you can grant least privilege using condition keys. For more information, see Grant least privilege in the *IAM User Guide*.
7. Choose **Next: Tags**.
8. Choose **Next: Review**.
9. On the **Review policy** page, enter a **Name** (for example, **`tutorial-serverless-application-repository`**) and a **Description** (optional) for the policy that you are creating. Review the policy summary to make sure that you have granted the intended permissions, and then choose **Create policy** to save your new policy.
10. In the left navigation pane, choose **Users**. Then, choose the IAM user for this tutorial.
11. On the **Summary** page of the chosen user, choose the **Permissions** tab, and then choose **Add permissions**.
12. Under **Grant permissions**, choose **Attach existing policies directly**.
13. Select the check box next to the policy that you just created (for example, **`tutorial-serverless-application-repository`**) and then choose **Next: Review**.
14. Under **Permissions summary**, review the summary to make sure that you attached the intended policy. Then, choose **Add permissions**.

## Step 1: Create an S3 bucket

Create a bucket to store the original data that you plan to transform.

**To create a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name (for example, **`tutorial-bucket`**) for your bucket.

   For more information about naming buckets in Amazon S3, see Bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.

   For more information about the bucket Region, see Buckets overview.
6. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

   We recommend that you keep all Block Public Access settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
7. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a bucket.
8. Choose **Create bucket**.

## Step 2: Upload a file to the S3 bucket

Upload a text file containing known PII data of various types, such as names, banking information, phone numbers, and SSNs, to the S3 bucket as the original data that you will redact PII from later in this tutorial.

For example, you can upload following the `tutorial.txt` file. This is an example input file from Amazon Comprehend.

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services,
LLC credit card account 1111-0000-1111-0008 has a minimum payment
of $24.53 that is due by July 31st. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account number XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Any City,
```

```
WA 98121.
After your payment is received, you will receive a confirmation
text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at 206-555-0199 or
email at support@anycompany.com.
```

**To upload a file to a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**. For example, you can upload the `tutorial.txt` file example mentioned earlier.
7. Choose **Upload**.

## Step 3: Create an S3 access point

To use an S3 Object Lambda Access Point to access and transform the original data, you must create an S3 access point and associate it with the S3 bucket that you created in Step 1. The access point must be in the same AWS Region as the objects you want to transform.

Later in this tutorial, you'll use this access point as a supporting access point for your Object Lambda Access Point.

**To create an access point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. On the **Access Points** page, choose **Create access point**.
4. In the **Access point name** field, enter the name (for example, **`tutorial-pii-access-point`**) for the access point.

For more information about naming access points, see Rules for naming Amazon S3 access points.

5. In the **Bucket name** field, enter the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**). S3 attaches the access point to this bucket.

   (Optional) You can choose **Browse S3** to browse and search the buckets in your account. If you choose **Browse S3**, choose the desired bucket, and then choose **Choose path** to populate the **Bucket name** field with that bucket's name.

6. For **Network origin**, choose **Internet**.

   For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.

7. By default, all block public access settings are turned on for your access point. We recommend that you keep **Block *all* public access** enabled. For more information, see Managing public access to access points.

8. For all other access point settings, keep the default settings.

   (Optional) You can modify the access point settings to support your use case. For this tutorial, we recommend keeping the default settings.

   (Optional) If you need to manage access to your access point, you can specify an access point policy. For more information, see Access point policy examples.

9. Choose **Create access point**.

## Step 4: Configure and deploy a prebuilt Lambda function

To redact PII data, configure and deploy the prebuilt AWS Lambda function `ComprehendPiiRedactionS3ObjectLambda` for use with your S3 Object Lambda Access Point.

**To configure and deploy the Lambda function**

1. Sign in to the AWS Management Console and view the `ComprehendPiiRedactionS3ObjectLambda` function in the AWS Serverless Application Repository.
2. For **Application settings**, under **Application name**, keep the default value (`ComprehendPiiRedactionS3ObjectLambda`) for this tutorial.

(Optional) You can enter the name that you want to give to this application. You might want to do this if you plan to configure multiple Lambda functions for different access needs for the same shared dataset.

3. For **MaskCharacter**, keep the default value (*****). The mask character replaces each character in the redacted PII entity.
4. For **MaskMode**, keep the default value (**MASK**). The **MaskMode** value specifies whether the PII entity is redacted with the `MASK` character or the `PII_ENTITY_TYPE` value.
5. To redact the specified types of data, for **PiiEntityTypes**, keep the default value **ALL**. The **PiiEntityTypes** value specifies the PII entity types to be considered for redaction.

   For more information about the list of supported PII entity types, see Detect Personally Identifiable Information (PII) in the *Amazon Comprehend Developer Guide*.
6. Keep the remaining settings set to the defaults.

   (Optional) If you want to configure additional settings for your specific use case, see the **Readme file** section on the left side of the page.
7. Select the check box next to **I acknowledge that this app creates custom IAM roles**.
8. Choose **Deploy**.
9. On the new application's page, under **Resources**, choose the **Logical ID** of the Lambda function that you deployed to review the function on the Lambda function page.

## Step 5: Create an S3 Object Lambda Access Point

An S3 Object Lambda Access Point provides the flexibility to invoke a Lambda function directly from an S3 GET request so that the function can redact PII data retrieved from an S3 access point. When creating and configuring an S3 Object Lambda Access Point, you must specify the redacting Lambda function to invoke and provide the event context in JSON format as custom parameters for Lambda to use.

The event context provides information about the request being made in the event passed from S3 Object Lambda to Lambda. For more information about all the fields in the event context, see Event context format and usage.

**To create an S3 Object Lambda Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
4. For **Object Lambda Access Point name**, enter the name that you want to use for the Object Lambda Access Point (for example, **`tutorial-pii-object-lambda-accesspoint`**).
5. For **Supporting Access Point**, enter or browse to the standard access point that you created in Step 3 (for example, **`tutorial-pii-access-point`**), and then choose **Choose supporting Access Point**.
6. For **S3 APIs**, to retrieve objects from the S3 bucket for Lambda function to process, select **GetObject**.
7. For **Invoke Lambda function**, you can choose either of the following two options for this tutorial.
   - Choose **Choose from functions in your account** and choose the Lambda function that you deployed in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**) from the **Lambda function** dropdown list.
   - Choose **Enter ARN**, and then enter the Amazon Resource Name (ARN) of the Lambda function that you created in Step 4.
8. For **Lambda function version**, choose **$LATEST** (the latest version of the Lambda function that you deployed in Step 4).
9. (Optional) If you need your Lambda function to recognize and process GET requests with range and part number headers, select **Lambda function supports requests using range** and **Lambda function supports requests using part numbers**. Otherwise, clear these two check boxes.

   For more information about how to use range or part numbers with S3 Object Lambda, see Working with Range and partNumber headers.
10. (Optional) Under **Payload - *optional***, add JSON text to provide your Lambda function with additional information.

    A payload is optional JSON text that you can provide to your Lambda function as input for all invocations coming from a specific S3 Object Lambda Access Point. To customize the behaviors

for multiple Object Lambda Access Points that invoke the same Lambda function, you can configure payloads with different parameters, thereby extending the flexibility of your Lambda function.

For more information about payload, see Event context format and usage.

11. (Optional) For **Request metrics - *optional***, choose **Disable** or **Enable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate. For more information, see CloudWatch pricing.
12. Under **Object Lambda Access Point policy - *optional***, keep the default setting.

    (Optional) You can set a resource policy. This resource policy grants the `GetObject` API permission to use the specified Object Lambda Access Point.
13. Keep the remaining settings set to the defaults, and choose **Create Object Lambda Access Point**.

## Step 6: Use the S3 Object Lambda Access Point to retrieve the redacted file

Now, S3 Object Lambda is ready to redact PII data from your original file.

**To use the S3 Object Lambda Access Point to retrieve the redacted file**

When you request to retrieve a file through your S3 Object Lambda Access Point, you make a `GetObject` API call to S3 Object Lambda. S3 Object Lambda invokes the Lambda function to redact your PII data and returns the transformed data as the response to the standard S3 `GetObject` API call.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the S3 Object Lambda Access Point that you created in Step 5 (for example, **`tutorial-pii-object-lambda-accesspoint`**).
4. On the **Objects** tab of your S3 Object Lambda Access Point, select the file that has the same name (for example, `tutorial.txt`) as the one that you uploaded to the S3 bucket in Step 2.

   This file should contain all the transformed data.
5. To view the transformed data, choose **Open** or **Download**.

You should be able to see the redacted file, as shown in the following example.

```
Hello *********. Your AnyCompany Financial Services,
LLC credit card account ******************* has a minimum payment
of $24.53 that is due by *********. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account ********** with the routing number *********.

Your latest statement was mailed to ********************************.
After your payment is received, you will receive a confirmation
text message at ************.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at ************ or
email at **********************.
```

## Step 7: Clean up

If you redacted your data through S3 Object Lambda only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

**Substeps**

- Delete the Object Lambda Access Point
- Delete the S3 access point
- Delete the Lambda function
- Delete the CloudWatch log group
- Delete the original file in the S3 source bucket
- Delete the S3 source bucket
- Delete the IAM role for your Lambda function
- Delete the customer managed policy for your IAM user
- Delete the IAM user

### Delete the Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the option button to the left of the S3 Object Lambda Access Point that you created in Step 5 (for example, **`tutorial-pii-object-lambda-accesspoint`**).
4. Choose **Delete**.
5. Confirm that you want to delete your Object Lambda Access Point by entering its name in the text field that appears, and then choose **Delete**.

## Delete the S3 access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. Navigate to the access point that you created in Step 3 (for example, **`tutorial-pii-access-point`**), and choose the option button next to the name of the access point.
4. Choose **Delete**.
5. Confirm that you want to delete your access point by entering its name in the text field that appears, and then choose **Delete**.

## Delete the Lambda function

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the function that you created in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
3. Choose **Actions**, and then choose **Delete**.
4. In the **Delete function** dialog box, choose **Delete**.

## Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Log groups**.

3. Find the log group whose name ends with the Lambda function that you created in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
4. Choose **Actions**, and then choose **Delete log group(s)**.
5. In the **Delete log group(s)** dialog box, choose **Delete**.

## Delete the original file in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the original file to in Step 2 (for example, **`tutorial-bucket`**).
4. Select the check box to the left of the name of the object that you want to delete (for example, `tutorial.txt`).
5. Choose **Delete**.
6. On the **Delete objects** page, in the **Permanently delete objects?** section, confirm that you want to delete this object by entering **`permanently delete`** in the text box.
7. Choose **Delete objects**.

## Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the option button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

## Delete the IAM role for your Lambda function

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the left navigation pane, choose **Roles**, and then select the check box next to the role name that you want to delete. The role name starts with the name of the Lambda function that you deployed in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
3. Choose **Delete**.
4. In the **Delete** dialog box, enter the role name in the text input field to confirm deletion. Then, choose **Delete**.

### Delete the customer managed policy for your IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. On the **Policies** page, enter the name of the customer managed policy that you created in the Prerequisites (for example, **`tutorial-serverless-application-repository`**) in the search box to filter the list of policies. Select the option button next to the name of the policy that you want to delete.
4. Choose **Actions**, and then choose **Delete**.
5. Confirm that you want to delete this policy by entering its name in the text field that appears, and then choose **Delete**.

### Delete the IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Users**, and then select the check box next to the user name that you want to delete.
3. At the top of the page, choose **Delete**.
4. In the **Delete *`user name`*?** dialog box, enter the user name in the text input field to confirm the deletion of the user. Choose **Delete**.

## Next steps

After completing this tutorial, you can further explore the following related use cases:

- You can create multiple S3 Object Lambda Access Points and enable them with prebuilt Lambda functions that are configured differently to redact specific types of PII depending on the data accessors' business needs.

  Each type of user assumes an IAM role and only has access to one S3 Object Lambda Access Point (managed through IAM policies). Then, you attach each `ComprehendPiiRedactionS3ObjectLambda` Lambda function configured for a different redaction use case to a different S3 Object Lambda Access Point. For each S3 Object Lambda Access Point, you can have a supporting S3 access point to read data from an S3 bucket that stores the shared dataset.

  For more information about how to create an S3 bucket policy that allows users to read from the bucket only through S3 access points, see Configuring IAM policies for using access points.

  For more information about how to grant a user permission to access the Lambda function, the S3 access point, and the S3 Object Lambda Access Point, see Configuring IAM policies for Object Lambda Access Points.
- You can build your own Lambda function and use S3 Object Lambda with your customized Lambda function to meet your specific data needs.

  For example, to explore various data values, you can use S3 Object Lambda and your own Lambda function that uses additional Amazon Comprehend features, such as entity recognition, key phrase recognition, sentiment analysis, and document classification, to process data. You can also use S3 Object Lambda together with Amazon Comprehend Medical, a HIPAA-eligible NLP service, to analyze and extract data in a context-aware manner.

  For more information about how to transform data with S3 Object Lambda and your own Lambda function, see Tutorial: Transforming data for your application with S3 Object Lambda.

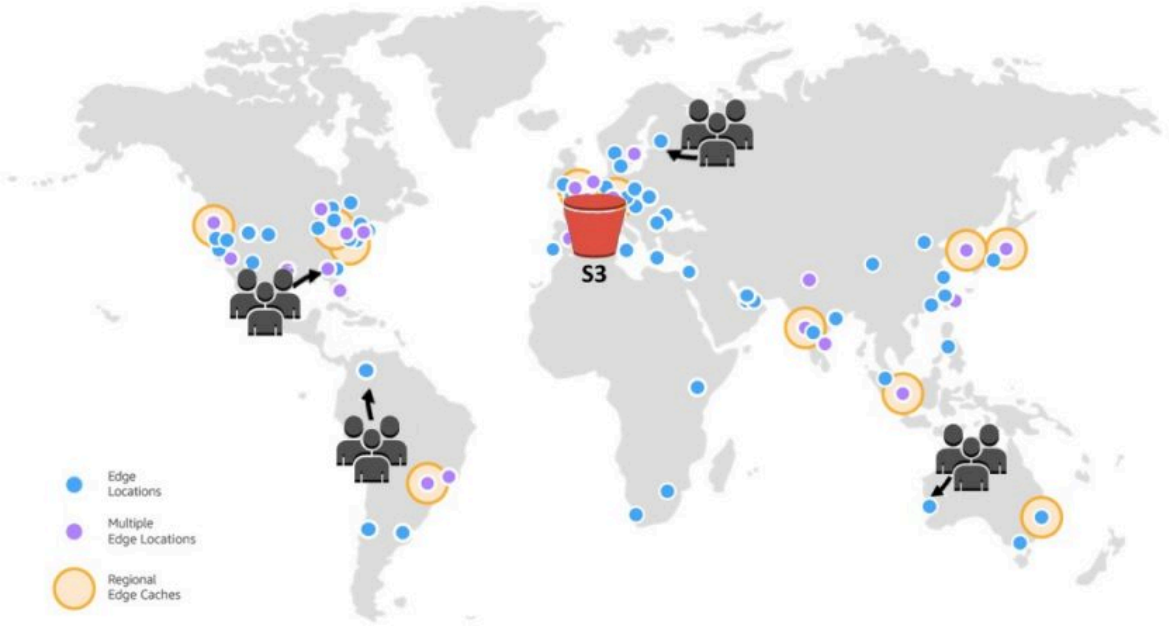
# Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53

You can use Amazon S3 with Amazon CloudFront to host videos for on-demand viewing in a secure and scalable way. Video on demand (VOD) streaming means that your video content is stored on a server and viewers can watch it at any time.

CloudFront is a fast, highly secure, and programmable content delivery network (CDN) service. CloudFront can deliver your content securely over HTTPS from all of the CloudFront edge locations

around the globe. For more information about CloudFront, see What is Amazon CloudFront? in the *Amazon CloudFront Developer Guide*.

CloudFront caching reduces the number of requests that your origin server must respond to directly. When a viewer (end user) requests a video that you serve with CloudFront, the request is routed to a nearby edge location closer to where the viewer is located. CloudFront serves the video from its cache, retrieving it from the S3 bucket only if it is not already cached. This caching management feature accelerates the delivery of your video to viewers globally with low latency, high throughput, and high transfer speeds. For more information about CloudFront caching management, see Optimizing caching and availability in the *Amazon CloudFront Developer Guide*.




### Objective

In this tutorial, you configure an S3 bucket to host on-demand video streaming using CloudFront for delivery and Amazon Route 53 for Domain Name System (DNS) and custom domain management.

### Topics

- Prerequisites: Register and configure a custom domain with Route 53
- Step 1: Create an S3 bucket

## Prerequisites: Register and configure a custom domain with Route 53

Before you start this tutorial, you must register and configure a custom domain (for example, **`example.com`**) with Route 53 so that you can configure your CloudFront distribution to use a custom domain name later.

Without a custom domain name, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

```
https://CloudFront distribution domain name/Path to an S3 video
```

For example, **`https://d111111abcdef8.cloudfront.net/sample.mp4`**.

After you configure your CloudFront distribution to use a custom domain name configured with Route 53, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

```
https://CloudFront distribution alternate domain name/Path to an S3 video
```

For example, **`https://www.example.com/sample.mp4`**. A custom domain name is simpler and more intuitive for your viewers to use.

To register a custom domain, see Registering a new domain using Route 53 in the *Amazon Route 53 Developer Guide*.

When you register a domain name with Route 53, Route 53 creates the hosted zone for you, which you will use later in this tutorial. This hosted zone is where you store information about how to route traffic for your domain, for example, to an Amazon EC2 instance or a CloudFront distribution.

There are fees associated with domain registration, your hosted zone, and DNS queries received by your domain. For more information, see Amazon Route 53 Pricing.

> **Note**
>
> When you register a domain, it costs money immediately and it's irreversible. You can choose not to auto-renew the domain, but you pay up front and own it for the year. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*.

## Step 1: Create an S3 bucket

Create a bucket to store the original video that you plan to stream.

**To create a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name for your bucket (for example, **`tutorial-bucket`**).

   For more information about naming buckets in Amazon S3, see Bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.

   If possible, you should pick the Region that is closest to the majority of your viewers. For more information about the bucket Region, see Buckets overview.
6. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

   Even with **Block *all* public access** enabled, viewers can still access the uploaded video through CloudFront. This feature is a major advantage of using CloudFront to host a video stored in S3.

We recommend that you keep all settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

7. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a bucket.

8. Choose **Create bucket**.

## Step 2: Upload a video to the S3 bucket

The following procedure describes how to upload a video file to an S3 bucket by using the console. If you're uploading many large video files to S3, you might want to use Amazon S3 Transfer Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

**To upload a file to the bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**.

   For example, you can upload a video file named `sample.mp4`.

7. Choose **Upload**.

## Step 3: Create a CloudFront origin access identity

To restrict direct access to the video from your S3 bucket, create a special CloudFront user called an origin access identity (OAI). You will associate the OAI with your distribution later in this tutorial. By

using an OAI, you make sure that viewers can't bypass CloudFront and get the video directly from the S3 bucket. Only the CloudFront OAI can access the file in the S3 bucket. For more information, see Restricting access to Amazon S3 content by using an OAI in the *Amazon CloudFront Developer Guide*.

**To create a CloudFront OAI**

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, under the **Security** section, choose **Origin access**.
3. Under the **Identities** tab, choose **Create origin access identity**.
4. Enter a name (for example, **`S3-OAI`**) for the new origin access identity.
5. Choose **Create**.

## Step 4: Create a CloudFront distribution

To use CloudFront to serve and distribute the video in your S3 bucket, you must create a CloudFront distribution.

**Substeps**

- Create a CloudFront distribution
- Review the bucket policy

### Create a CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. Choose **Create distribution**.
4. In the **Origin** section, for **Origin domain**, choose the domain name of your S3 origin, which starts with the name of the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
5. For **Origin access**, choose **Legacy access identities**.
6. Under **Origin access identity**, choose the origin access identity that you created in Step 3 (for example, **`S3-OAI`**).

7. Under **Bucket policy**, choose **Yes, update the bucket policy**.
8. In the **Default cache behavior** section, under **Viewer protocol policy**, choose **Redirect HTTP to HTTPS**.

   When you choose this feature, HTTP requests are automatically redirected to HTTPS to secure your website and protect your viewers' data.
9. For the other settings in the **Default cache behaviors** section, keep the default values.

   (Optional) You can control how long your file stays in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means that your viewers get better performance because your files are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin. For more information, see Managing how long content stays in the cache (expiration) in the *Amazon CloudFront Developer Guide*.
10. For the other sections, keep the remaining settings set to the defaults.

    For more information about the different settings options, see Values That You Specify When You Create or Update a Distribution in the *Amazon CloudFront Developer Guide*.
11. At the bottom of the page, choose **Create distribution**.
12. On the **General** tab for your CloudFront distribution, under **Details**, the value of the **Last modified** column for your distribution changes from **Deploying** to the timestamp when the distribution was last modified. This process typically takes a few minutes.

## Review the bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you used earlier as the origin of your CloudFront distribution (for example, **`tutorial-bucket`**).
4. Choose the **Permissions** tab.
5. In the **Bucket policy** section, confirm that you see a statement similar to the following in the bucket policy text:

```
{
    "Version": "2008-10-17",
```

```
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tutorial-bucket/*"
        }
    ]
}
```

This is the statement that your CloudFront distribution added to your bucket policy when you chose **Yes, update the bucket policy** earlier.

This bucket policy update indicates that you successfully configured the CloudFront distribution to restrict access to the S3 bucket. Because of this restriction, objects in the bucket can be accessed only through your CloudFront distribution.

## Step 5: Access the video through the CloudFront distribution

Now, CloudFront can serve the video stored in your S3 bucket. To access your video through CloudFront, you must combine your CloudFront distribution domain name with the path to the video in the S3 bucket.

**To create a URL to the S3 video using the CloudFront distribution domain name**

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. To get the distribution domain name, do the following:

    a. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).

   b. After finding the distribution in the list, widen the **Domain name** column to copy the domain name value for your CloudFront distribution.

4. In a new browser tab, paste the distribution domain name that you copied.
5. Return to the previous browser tab, and open the S3 console at https://console.aws.amazon.com/s3/.
6. In the left navigation pane, choose **Buckets**.
7. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
8. In the **Objects** list, choose the name of the video that you uploaded in Step 2 (for example, `sample.mp4`).
9. On the object detail page, in the **Object overview** section, copy the value of the **Key**. This value is the path to the uploaded video object in the S3 bucket.
10. Return to the browser tab where you previously pasted the distribution domain name, enter a forward slash (**/**) after the distribution domain name, and then paste the path to the video that you copied earlier (for example, `sample.mp4`).

    Now, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

    ```
    https://CloudFront distribution domain name/Path to the S3 video
    ```

    Replace *`CloudFront distribution domain name`* and *`Path to the S3 video`* with the appropriate values. An example URL is **`https://d111111abcdef8.cloudfront.net/sample.mp4`**.

## Step 6: Configure your CloudFront distribution to use your custom domain name

To use your own domain name instead of the CloudFront domain name in the URL to access the S3 video, add an alternate domain name to your CloudFront distribution.

### Substeps

- Request an SSL certificate
- Add the alternate domain name to your CloudFront distribution

- Create a DNS record to route traffic from your alternate domain name to your CloudFront distribution's domain name
- Check whether IPv6 is enabled for your distribution and create another DNS record if needed

### Request an SSL certificate

To allow your viewers to use HTTPS and your custom domain name in the URL for your video streaming, use AWS Certificate Manager (ACM) to request a Secure Sockets Layer (SSL) certificate. The SSL certificate establishes an encrypted network connection to the website.

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. If the introductory page appears, under **Provision certificates**, choose **Get Started**.
3. On the **Request a certificate** page, choose **Request a public certificate**, and then choose **Request a certificate**.
4. On the **Add domain names** page, enter the fully qualified domain name (FQDN) of the site that you want to secure with an SSL/TLS certificate. You can use an asterisk (*) to request a wildcard certificate to protect several site names in the same domain. For this tutorial, enter **`*`** and the custom domain name that you configured in Prerequisites. For example, enter **`*.example.com`**, and then choose **Next**.

   For more information, see To request an ACM public certificate (console) in the *AWS Certificate Manager User Guide*.
5. On the **Select validation method** page, choose **DNS validation**. Then, choose **Next**.

   If you are able to edit your DNS configuration, we recommend that you use DNS domain validation rather than email validation. DNS validation has multiple benefits over email validation. For more information, see Option 1: DNS validation in the *AWS Certificate Manager User Guide*.
6. (Optional) On the **Add tags** page, tag your certificate with metadata.
7. Choose **Review**.
8. On the **Review** page, verify that the information under **Domain name** and **Validation method** are correct. Then, choose **Confirm and request**.

The **Validation** page shows that your request is being processed and that the certificate domain is being validated. The certificate awaiting validation is in the **Pending validation** status.

9. On the **Validation** page, choose the down arrow to the left of your custom domain name, and then choose **Create record in Route 53** to validate your domain ownership through DNS.

   Doing this adds a CNAME record provided by AWS Certificate Manager to your DNS configuration.

10. In the **Create record in Route 53** dialog box, choose **Create**.

    The **Validation** page should display a status notification of **Success** at the bottom.

11. Choose **Continue** to view the **Certificates** list page.

    The **Status** for your new certificate changes from **Pending validation** to **Issued** within 30 minutes.

## Add the alternate domain name to your CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. Choose the ID for the distribution that you created in Step 4.
4. On the **General** tab, go to the **Settings** section, and choose **Edit**.
5. On the **Edit settings** page, for **Alternate domain name (CNAME) - *optional***, choose **Add item** to add the custom domain names that you want to use in the URL for the S3 video served by this CloudFront distribution.

   In this tutorial, for example, if you want to route traffic for a subdomain, such as `www.example.com`, enter the subdomain name (`www`) with the domain name (`example.com`). Specifically, enter **`www.example.com`**.

   > **Note**
   >
   > The alternate domain name (CNAME) that you add must be covered by the SSL certificate that you previously attached to your CloudFront distribution.

6. For **Custom SSL certificate - *optional***, choose the SSL certificate that you requested earlier (for example, **`*.example.com`**).

   > **Note**
   >
   > If you don't see the SSL certificate immediately after you request it, wait 30 minutes, and then refresh the list until the SSL certificate is available for you to select.

7. Keep the remaining settings set to the defaults. Choose **Save changes**.
8. On the **General** tab for the distribution, wait for the value of **Last modified** to change from **Deploying** to the timestamp when the distribution was last modified.

## Create a DNS record to route traffic from your alternate domain name to your CloudFront distribution's domain name

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).
4. Choose **Create record**, and then use the **Quick create record** method.
5. For **Record name**, keep the value for the record name the same as the alternate domain name of the CloudFront distribution that you added earlier.

   In this tutorial, to route traffic to a subdomain, such as `www.example.com`, enter the subdomain name without the domain name. For example, enter only **`www`** in the text field before your custom domain name.
6. For **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
7. For **Value**, choose the **Alias** toggle to enable the alias resource.
8. Under **Route traffic to**, choose **Alias to CloudFront distribution** from the dropdown list.
9. In the search box that says **Choose distribution**, choose the domain name of the CloudFront distribution that you created in Step 4.

   To find the domain name of your CloudFront distribution, do the following:

a. In a new browser tab, sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v3/home.

b. In the left navigation pane, choose **Distributions**.

c. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).

d. After finding the distribution in the list, widen the **Domain name** column to see the domain name value for your CloudFront distribution.

10. On the **Create record** page in the Route 53 console, for the remaining settings, keep the defaults.

11. Choose **Create records**.

## Check whether IPv6 is enabled for your distribution and create another DNS record if needed

If IPv6 is enabled for your distribution, you must create another DNS record.

1. To check whether IPv6 is enabled for your distribution, do the following:

   a. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.

   b. In the left navigation pane, choose **Distributions**.

   c. Choose the ID of the CloudFront distribution that you created in Step 4.

   d. On the **General** tab, under **Settings**, check whether **IPv6** is set to **Enabled**.

      If IPv6 is enabled for your distribution, you must create another DNS record.

2. If IPv6 is enabled for your distribution, do the following to create a DNS record:

   a. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.

   b. In the left navigation pane, choose **Hosted zones**.

   c. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).

   d. Choose **Create record**, and then use the **Quick create record** method.

e. For **Record name**, in the text field before your custom domain name, type the same value that you typed when you created the IPv4 DNS record earlier. For example, in this tutorial, to route traffic for the subdomain `www.example.com`, enter only **`www`**.

f. For **Record type**, choose **AAAA - Routes traffic to an IPv6 address and some AWS resources**.

g. For **Value**, choose the **Alias** toggle to enable the alias resource.

h. Under **Route traffic to**, choose **Alias to CloudFront distribution** from the dropdown list.

i. In the search box that says **Choose distribution**, choose the domain name of the CloudFront distribution that you created in Step 4.

j. For the remaining settings, keep the defaults.

k. Choose **Create records**.

## Step 7: Access the S3 video through the CloudFront distribution with the custom domain name

To access the S3 video using the custom URL, you must combine your alternate domain name with the path to the video in the S3 bucket.

**To create a custom URL to access the S3 video through the CloudFront distribution**

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. To get the alternate domain name of your CloudFront distribution, do the following:

   a. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket name for the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).

   b. After finding the distribution in the list, widen the **Alternate domain names** column to copy the value of the alternate domain name of your CloudFront distribution.
4. In a new browser tab, paste the alternate domain name of the CloudFront distribution.
5. Return to the previous browser tab, and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
6. Find the path to your S3 video, as explained in Step 5.

7. Return to the browser tab where you previously pasted the alternate domain name, enter a forward slash (**/**), and then paste the path to your S3 video (for example, `sample.mp4`).

   Now, your S3 video is publicly accessible and hosted through CloudFront at a custom URL that looks similar to the following:

   ```
   https://CloudFront distribution alternate domain name/Path to the S3 video
   ```

   Replace *CloudFront distribution alternate domain name* and *Path to the S3 video* with the appropriate values. An example URL is **https://www.example.com/sample.mp4**.

## (Optional) Step 8: View data about requests received by your CloudFront distribution

**To view data about requests received by your CloudFront distribution**

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, under **Reports & analytics**, choose the reports from the console, ranging from **Cache statistics**, **Popular Objects**, **Top Referrers**, **Usage**, and **Viewers**.

   You can filter each report dashboard. For more information, see CloudFront Reports in the Console in the *Amazon CloudFront Developer Guide*.
3. To filter data, choose the ID of the CloudFront distribution that you created in Step 4.

## Step 9: Clean up

If you hosted an S3 streaming video using CloudFront and Route 53 only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

> **Note**
>
> When you register a domain, it costs money immediately and it's irreversible. You can choose not to auto-renew the domain, but you pay up front and own it for the year. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*.

**Substeps**

- Delete the CloudFront distribution
- Delete the DNS record
- Delete the public hosted zone for your custom domain
- Delete the custom domain name from Route 53
- Delete the original video in the S3 source bucket
- Delete the S3 source bucket

## Delete the CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket name for the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. To delete the CloudFront distribution, you must disable it first.
   - If the value of the **Status** column is **Enabled** and the value of **Last modified** is the timestamp when the distribution was last modified, continue to disable the distribution before deleting it.
   - If the value of **Status** is **Enabled** and the value of **Last modified** is **Deploying**, wait until the value of **Status** changes to the timestamp when the distribution was last modified. Then continue to disable the distribution before deleting it.
5. To disable the CloudFront distribution, do the following:
   a. In the **Distributions** list, select the check box next to the ID for the distribution that you want to delete.
   b. To disable the distribution, choose **Disable**, and then choose **Disable** to confirm.

      If you disable a distribution that has an alternate domain name associated with it, CloudFront stops accepting traffic for that domain name (such as `www.example.com`), even if another distribution has an alternate domain name with a wildcard (`*`) that matches the same domain (such as `*.example.com`).

c. The value of **Status** immediately changes to **Disabled**. Wait until the value of **Last modified** changes from **Deploying** to the timestamp when the distribution was last modified.

   Because CloudFront must propagate this change to all edge locations, it might take a few minutes before the update is complete and the **Delete** option is available for you to delete the distribution.

6. To delete the disabled distribution, do the following:

   a. Choose the check box next to the ID for the distribution that you want to delete.

   b. Choose **Delete**, and then choose **Delete** to confirm.

## Delete the DNS record

If you want to delete the public hosted zone for the domain (including the DNS record), see Delete the public hosted zone for your custom domain in the *Amazon Route 53 Developer Guide*. If you only want to delete the DNS record created in Step 6, do the following:

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).
4. In the list of records, select the check box next to the records that you want to delete (the records that you created in Step 6).

   > **Note**
   >
   > You can't delete records that have a **Type** value of **NS** or **SOA**.

5. Choose **Delete records**.
6. To confirm the deletion, choose **Delete**.

   Changes to records take time to propagate to the Route 53 DNS servers. Currently, the only way to verify that your changes have propagated is to use the GetChange API action. Changes usually propagate to all Route 53 name servers within 60 seconds.

## Delete the public hosted zone for your custom domain

**Warning**

If you want to keep your domain registration but stop routing internet traffic to your website or web application, we recommend that you delete records in the hosted zone (as described in the prior section) instead of deleting the hosted zone.
If you delete a hosted zone, someone else can use the domain and route traffic to their own resources using your domain name.
In addition, if you delete a hosted zone, you can't undelete it. You must create a new hosted zone and update the name servers for your domain registration, which can take up to 48 hours to take effect.
If you want to make the domain unavailable on the internet, you can first transfer your DNS service to a free DNS service and then delete the Route 53 hosted zone. This prevents future DNS queries from possibly being misrouted.

1. If the domain is registered with Route 53, see Adding or changing name servers and glue records for a domain in the *Amazon Route 53 Developer Guide* for information about how to replace Route 53 name servers with name servers for the new DNS service.
2. If the domain is registered with another registrar, use the method provided by the registrar to change name servers for the domain.

**Note**

If you're deleting a hosted zone for a subdomain (`www.example.com`), you don't need to change name servers for the domain (`example.com`).

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that you want to delete.
4. On the **Records** tab for your hosted zone, confirm that the hosted zone that you want to delete contains only an **NS** and an **SOA** record.

   If it contains additional records, delete them first.

If you created any NS records for subdomains in the hosted zone, delete those records too.

5. On the **DNSSEC signing** tab for your hosted zone, disable DNNSSEC signing if it was enabled. For more information, see Disabling DNSSEC signing in the *Amazon Route 53 Developer Guide*.
6. At the top of the details page of the hosted zone, choose **Delete zone**.
7. To confirm the deletion, enter **`delete`**, and then choose **Delete**.

## Delete the custom domain name from Route 53

For most top-level domains (TLDs), you can delete the registration if you no longer want it. If you delete a domain name registration from Route 53 before the registration is scheduled to expire, AWS does not refund the registration fee. For more information, see Deleting a domain name registration in the *Amazon Route 53 Developer Guide*.

> **Important**
>
> If you want to transfer the domain between AWS accounts or transfer the domain to another registrar, don't delete the domain and expect to immediately reregister it. Instead, see the applicable documentation in the *Amazon Route 53 Developer Guide*:
>
> - Transferring a domain to a different AWS account
> - Transferring a domain from Amazon Route 53 to another registrar

## Delete the original video in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the video to in Step 2 (for example, **`tutorial-bucket`**).
4. On the **Objects** tab, select the check box next to the name of the object that you want to delete (for example, `sample.mp4`).
5. Choose **Delete**.
6. Under **Permanently delete objects?**, enter **`permanently delete`** to confirm that you want to delete this object.

7. Choose **Delete objects**.

### Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, select the option button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

## Next steps

After you complete this tutorial, you can further explore the following related use cases:

- Transcode S3 videos into streaming formats needed by a particular television or connected device before hosting these videos with a CloudFront distribution.

  To use Amazon S3 Batch Operations, AWS Lambda and AWS Elemental MediaConvert to batch-transcode a collection of videos to a variety of output media formats, see Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert.
- Host other objects stored in S3, such as images, audio, motion graphics, style sheets, HTML, JavaScript, React apps, and so on, using CloudFront and Route 53.

  For example, see Tutorial: Configuring a static website using a custom domain registered with Route 53 and Speeding up your website with Amazon CloudFront.
- Use Amazon S3 Transfer Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. Transfer Acceleration improves transfer performance by routing traffic through the CloudFront globally distributed edge locations and over the AWS backbone networks. It also uses network protocol optimizations. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

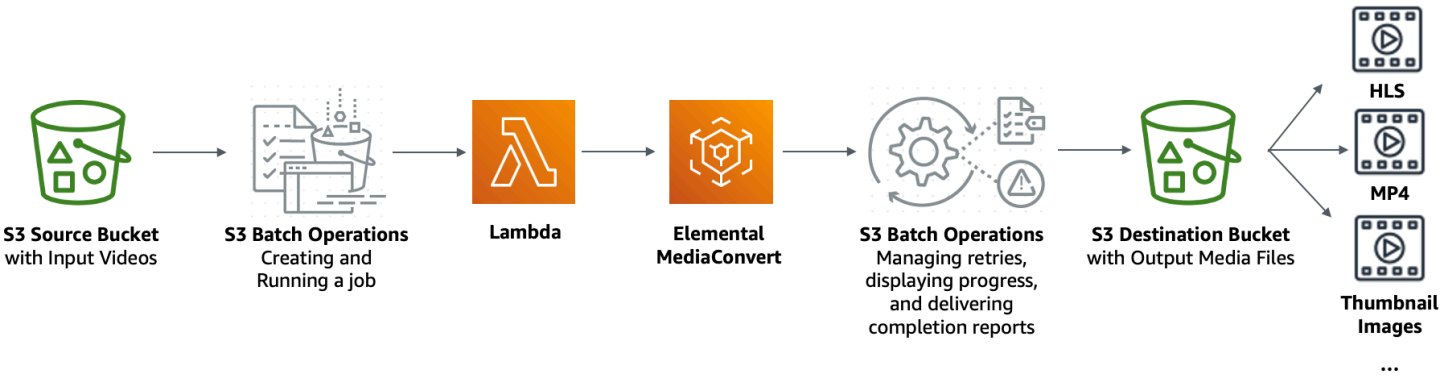
# Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

Video consumers use devices of all shapes, sizes, and vintages to enjoy media content. This wide array of devices presents a challenge for content creators and distributors. Instead of being in a one-size-fits-all format, videos must be converted so that they can span a broad range of sizes, formats, and bitrates. This conversion task is even more challenging when you have a large number of videos that must be converted.

AWS offers you a method to build a scalable, distributed architecture that does the following:

- Ingests input videos
- Processes the videos for playback on a wide range of devices
- Stores the transcoded media files
- Delivers the output media files to meet demand

When you have extensive video repositories stored in Amazon S3, you can transcode these videos from their source formats into multiple file types in the size, resolution, and format needed by a particular video player or device. Specifically, S3 Batch Operations provides you with a solution to invoke AWS Lambda functions for existing input videos in an S3 source bucket. Then, the Lambda functions call AWS Elemental MediaConvert to perform large-scale video transcoding tasks. The converted output media files are stored in an S3 destination bucket.




## Objective

In this tutorial, you learn how to set up S3 Batch Operations to invoke a Lambda function for batch-transcoding of videos stored in an S3 source bucket. The Lambda function calls

MediaConvert to transcode the videos. The outputs for each video in the S3 source bucket are as follows:

- An HTTP Live Streaming (HLS) adaptive bitrate stream for playback on devices of multiple sizes and varying bandwidths
- An MP4 video file
- Thumbnail images collected at intervals

**Topics**



## Prerequisites

Before you start this tutorial, you must have an Amazon S3 source bucket (for example, **`tutorial-bucket-1`**) with videos to be transcoded already stored in it.

You can give the bucket another name if you want. For more information about bucket names in Amazon S3, see Bucket naming rules.

For the S3 source bucket, keep the settings related to **Block Public Access settings for this bucket** set to the defaults (**Block *all* public access** is enabled). For more information, see Creating a bucket.

For more information about uploading videos to the S3 source bucket, see Uploading objects. If you're uploading many large video files to S3, you might want to use Amazon S3 Transfer

Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

## Step 1: Create an S3 bucket for the output media files

In this step, you create an S3 destination bucket to store the converted output media files. You also create a Cross Origin Resource Sharing (CORS) configuration to allow cross-origin access to the transcoded media files stored in your S3 destination bucket.

**Substeps**

- Create a bucket for the output media files
- Add a CORS configuration to the S3 output bucket

### Create a bucket for the output media files

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. For **Bucket name**, enter a name for your bucket (for example, **`tutorial-bucket-2`**).
5. For **Region**, choose the AWS Region where you want the bucket to reside.
6. To ensure public access to your output media files, in **Block Public Access settings for this bucket**, clear **Block *all* public access**.

> **Warning**
>
> Before you complete this step, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off Block Public Access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.
> If you don't want to clear the Block Public Access settings, you can use Amazon CloudFront to deliver the transcoded media files to viewers (end users). For more

information, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53.

7. Select the check box next to **I acknowledge that the current settings might result in this bucket and the objects within becoming public.**
8. Keep the remaining settings set to the defaults.
9. Choose **Create bucket**.

## Add a CORS configuration to the S3 output bucket

A JSON CORS configuration defines a way for client web applications (video players in this context) that are loaded in one domain to play transcoded output media files in a different domain.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created earlier (for example, **`tutorial-bucket-2`**).
4. Choose the **Permissions** tab.
5. In the **Cross-origin resource sharing (CORS)** section, choose **Edit**.
6. In the CORS configuration text box, copy and paste the following CORS configuration.

   The CORS configuration must be in JSON format. In this example, the `AllowedOrigins` attribute uses the wildcard character (*) to specify all origins. If you know your specific origin, you can restrict the `AllowedOrigins` attribute to your specific player URL. For more information about configuring this and other attributes, see Elements of a CORS configuration.

```
[
    {
        "AllowedOrigins": [
            "*"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedHeaders": [
            "*"
```

```
        ],
        "ExposeHeaders": []

    }
]
```

7. Choose **Save changes**.

## Step 2: Create an IAM role for MediaConvert

To use AWS Elemental MediaConvert to transcode input videos stored in your S3 bucket, you must have an AWS Identity and Access Management (IAM) service role to grant MediaConvert permissions to read and write video files from and to your S3 source and destination buckets. When you run transcoding jobs, the MediaConvert console uses this role.

**To create an IAM role for MediaConvert**

1. Create an IAM role with a role name that you choose (for example, **`tutorial-mediaconvert-role`**). To create this role, follow the steps in Create your MediaConvert role in IAM (console) in the *AWS Elemental MediaConvert User Guide*.
2. After you create the IAM role for MediaConvert, in the list of **Roles**, choose the name of the role for MediaConvert that you created (for example, **`tutorial-mediaconvert-role`**).
3. On the **Summary** page, copy the **Role ARN** (which starts with `arn:aws:iam::`), and save the ARN for use later.

   For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

## Step 3: Create an IAM role for your Lambda function

To batch-transcode videos with MediaConvert and S3 Batch Operations, you use a Lambda function to connect these two services to convert videos. This Lambda function must have an IAM role that grants the Lambda function permissions to access MediaConvert and S3 Batch Operations.

**Substeps**

- Create an IAM role for your Lambda function
- Embed an inline policy for the IAM role of your Lambda function

## Create an IAM role for your Lambda function

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then under **Common use cases**, choose **Lambda**.
4. Choose **Next: Permissions**.
5. On the **Attach permissions policies** page, enter **`AWSLambdaBasicExecutionRole`** in the **Filter policies** box. To attach the managed policy **AWSLambdaBasicExecutionRole** to this role to grant write permissions to Amazon CloudWatch Logs, select the check box next to **AWSLambdaBasicExecutionRole**.
6. Choose **Next: Tags**.
7. (Optional) Add tags to the managed policy.
8. Choose **Next: Review**.
9. For **Role name**, enter **`tutorial-lambda-transcode-role`**.
10. Choose **Create role**.

## Embed an inline policy for the IAM role of your Lambda function

To grant permissions to the MediaConvert resource that's needed for the Lambda function to execute, you must use an inline policy.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.
3. In the **Roles** list, choose the name of the IAM role that you created earlier for your Lambda function (for example, **`tutorial-lambda-transcode-role`**).
4. Choose the **Permissions** tab.
5. Choose **Add inline policy**.
6. Choose the **JSON** tab, and then copy and paste the following JSON policy.

   In the JSON policy, replace the example ARN value of `Resource` with the role ARN of the IAM role for MediaConvert that you created in Step 2 (for example, **`tutorial-mediaconvert-role`**).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
```

```
    ]
}
```

7. Choose **Review Policy**.
8. For **Name**, enter **`tutorial-lambda-policy`**.
9. Choose **Create Policy**.

   After you create an inline policy, it is automatically embedded in the IAM role of your Lambda function.

## Step 4: Create a Lambda function for video transcoding

In this section of the tutorial, you build a Lambda function using the SDK for Python to integrate with S3 Batch Operations and MediaConvert. To start transcoding the videos already stored in your S3 source bucket, you run an S3 Batch Operations job that directly invokes the Lambda function for each video in the S3 source bucket. Then, the Lambda function submits a transcoding job for each video to MediaConvert.

**Substeps**

- Write Lambda function code and create a deployment package
- Create a Lambda function with an execution role (console)
- Deploy your Lambda function with .zip file archives and configure the Lambda function (console)

### Write Lambda function code and create a deployment package

1. On your local machine, create a folder named `batch-transcode`.
2. In the `batch-transcode` folder, create a file with JSON job settings. For example, you can use the settings provided in this section, and name the file `job.json`.

   A `job.json` file specifies the following:

   - Which files to transcode
   - How you want to transcode your input videos
   - What output media files you want to create
   - What to name the transcoded files
   - Where to save the transcoded files

- Which advanced features to apply, and so on

In this tutorial, we use the following `job.json` file to create the following outputs for each video in the S3 source bucket:

- An HTTP Live Streaming (HLS) adaptive bitrate stream for playback on multiple devices of differing sizes and varying bandwidths
- An MP4 video file
- Thumbnail images collected at intervals

This example `job.json` file uses Quality-Defined Variable Bitrate (QVBR) to optimize video quality. The HLS output is Apple-compliant (audio unmixed from video, segment duration of 6 seconds, and optimized video quality through auto QVBR).

If you don't want to use the example settings provided here, you can generate a `job.json` specification based on your use case. To ensure consistency across your outputs, make sure that your input files have similar video and audio configurations. For any input files with different video and audio configurations, create separate automations (unique `job.json` settings). For more information, see Example AWS Elemental MediaConvert job settings in JSON in the *AWS Elemental MediaConvert User Guide*.

```
{
  "OutputGroups": [
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
```

```
            "VideoPid": 481,
            "AudioPids": [
              482,
              483,
              484,
              485,
              486,
              487,
              488,
              489,
              490,
              491,
              492
            ]
          }
        },
        "VideoDescription": {
          "Width": 640,
          "ScalingBehavior": "DEFAULT",
          "Height": 360,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 50,
          "CodecSettings": {
            "Codec": "H_264",
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "GopSize": 2,
              "Slices": 1,
              "GopBReference": "DISABLED",
              "MaxBitrate": 1200000,
              "SlowPal": "DISABLED",
              "SpatialAdaptiveQuantization": "ENABLED",
              "TemporalAdaptiveQuantization": "ENABLED",
              "FlickerAdaptiveQuantization": "DISABLED",
              "EntropyEncoding": "CABAC",
              "FramerateControl": "INITIALIZE_FROM_SOURCE",
              "RateControlMode": "QVBR",
              "CodecProfile": "MAIN",
              "Telecine": "NONE",
```

```
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
```

```
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 960,
            "ScalingBehavior": "DEFAULT",
            "Height": 540,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 3500000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
```

```
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
```

```
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 5000000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
```

```
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_720"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {}
          },
          "AudioDescriptions": [
            {
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "Bitrate": 96000,
                  "CodingMode": "CODING_MODE_2_0",
                  "SampleRate": 48000
                }
              }
            }
          ],
          "OutputSettings": {
            "HlsSettings": {
```

```
            "AudioGroupId": "program_audio",
            "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
          }
        },
        "NameModifier": "_audio"
      }
    ],
    "OutputGroupSettings": {
      "Type": "HLS_GROUP_SETTINGS",
      "HlsGroupSettings": {
        "ManifestDurationFormat": "INTEGER",
        "SegmentLength": 6,
        "TimedMetadataId3Period": 10,
        "CaptionLanguageSetting": "OMIT",
        "Destination": "s3://EXAMPLE-BUCKET/HLS/",
        "DestinationSettings": {
          "S3Settings": {
            "AccessControl": {
              "CannedAcl": "PUBLIC_READ"
            }
          }
        },
        "TimedMetadataId3Frame": "PRIV",
        "CodecSpecification": "RFC_4281",
        "OutputSelection": "MANIFESTS_AND_SEGMENTS",
        "ProgramDateTimePeriod": 600,
        "MinSegmentLength": 0,
        "DirectoryStructure": "SINGLE_DIRECTORY",
        "ProgramDateTime": "EXCLUDE",
        "SegmentControl": "SEGMENTED_FILES",
        "ManifestCompression": "NONE",
        "ClientCache": "ENABLED",
        "StreamInfResolution": "INCLUDE"
      }
    }
  },
  {
    "CustomName": "MP4",
    "Name": "File Group",
    "Outputs": [
      {
        "ContainerSettings": {
          "Container": "MP4",
          "Mp4Settings": {
```

```
            "CslgAtom": "INCLUDE",
            "FreeSpaceBox": "EXCLUDE",
            "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
          }
        },
        "VideoDescription": {
          "Width": 1280,
          "ScalingBehavior": "DEFAULT",
          "Height": 720,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 100,
          "CodecSettings": {
            "Codec": "H_264",
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "ParNumerator": 1,
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "HrdBufferInitialFillPercentage": 90,
              "GopSize": 2,
              "Slices": 2,
              "GopBReference": "ENABLED",
              "HrdBufferSize": 10000000,
              "MaxBitrate": 5000000,
              "ParDenominator": 1,
              "EntropyEncoding": "CABAC",
              "RateControlMode": "QVBR",
              "CodecProfile": "HIGH",
              "MinIInterval": 0,
              "AdaptiveQuantization": "AUTO",
              "CodecLevel": "AUTO",
              "FieldEncoding": "PAFF",
              "SceneChangeDetect": "ENABLED",
              "QualityTuningLevel": "SINGLE_PASS_HQ",
              "UnregisteredSeiTimecode": "DISABLED",
              "GopSizeUnits": "SECONDS",
              "ParControl": "SPECIFIED",
              "NumberBFramesBetweenReferenceFrames": 3,
              "RepeatPps": "DISABLED",
              "DynamicSubGop": "ADAPTIVE"
            }
```

```
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "AudioDescriptions": [
            {
              "AudioTypeControl": "FOLLOW_INPUT",
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "AudioDescriptionBroadcasterMix": "NORMAL",
                  "Bitrate": 160000,
                  "RateControlMode": "CBR",
                  "CodecProfile": "LC",
                  "CodingMode": "CODING_MODE_2_0",
                  "RawFormat": "NONE",
                  "SampleRate": 48000,
                  "Specification": "MPEG4"
                }
              },
              "LanguageCodeControl": "FOLLOW_INPUT",
              "AudioType": 0
            }
          ]
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/MP4/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
    {
```

```
      "CustomName": "Thumbnails",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "RAW"
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    }
```

```
  ],
  "AdAvailOffset": 0,
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. In the `batch-transcode` folder, create a file with a Lambda function. You can use the following Python example and name the file `convert.py`.

   S3 Batch Operations sends specific task data to a Lambda function and requires result data back. For request and response examples for the Lambda function, information about response and result codes, and example Lambda functions for S3 Batch Operations, see Invoke AWS Lambda function.

```
import json
import os
from urllib.parse import urlparse
import uuid
import boto3

"""
When you run an S3 Batch Operations job, your job
invokes this Lambda function. Specifically, the Lambda function is
invoked on each video object listed in the manifest that you specify
```

```
for the S3 Batch Operations job in Step 5.

Input parameter "event": The S3 Batch Operations event as a request
                         for the Lambda function.

Input parameter "context": Context about the event.

Output: A result structure that Amazon S3 uses to interpret the result
        of the operation. It is a job response returned back to S3 Batch
 Operations.
"""
def handler(event, context):

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
    task_id = event['tasks'][0]['taskId']

    source_s3_key = event['tasks'][0]['s3Key']
    source_s3_bucket = event['tasks'][0]['s3BucketArn'].split(':::')[-1]
    source_s3 = 's3://' + source_s3_bucket + '/' + source_s3_key

    result_list = []
    result_code = 'Succeeded'
    result_string = 'The input video object was converted successfully.'

    # The type of output group determines which media players can play
    # the files transcoded by MediaConvert.
    # For more information, see Creating outputs with AWS Elemental MediaConvert.
    output_group_type_dict = {
        'HLS_GROUP_SETTINGS': 'HlsGroupSettings',
        'FILE_GROUP_SETTINGS': 'FileGroupSettings',
        'CMAF_GROUP_SETTINGS': 'CmafGroupSettings',
        'DASH_ISO_GROUP_SETTINGS': 'DashIsoGroupSettings',
        'MS_SMOOTH_GROUP_SETTINGS': 'MsSmoothGroupSettings'
    }

    try:
        job_name = 'Default'
        with open('job.json') as file:
            job_settings = json.load(file)

        job_settings['Inputs'][0]['FileInput'] = source_s3

        # The path of each output video is constructed based on the values of
```

```
        # the attributes in each object of OutputGroups in the job.json file.
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['DestinationBucket'],
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

        job_metadata_dict = {
            'assetID': str(uuid.uuid4()),
            'application': os.environ['Application'],
            'input': source_s3,
            'settings': job_name
        }

        region = os.environ['AWS_DEFAULT_REGION']
        endpoints = boto3.client('mediaconvert', region_name=region) \
            .describe_endpoints()
        client = boto3.client('mediaconvert', region_name=region,
                              endpoint_url=endpoints['Endpoints'][0]['Url'],
                              verify=False)

        try:
            client.create_job(Role=os.environ['MediaConvertRole'],
                              UserMetadata=job_metadata_dict,
                              Settings=job_settings)
        # You can customize error handling based on different error codes that
        # MediaConvert can return.
        # For more information, see MediaConvert error codes.
        # When the result_code is TemporaryFailure, S3 Batch Operations retries
        # the task before the job is completed. If this is the final retry,
        # the error message is included in the final report.
        except Exception as error:
```

```
            result_code = 'TemporaryFailure'
            raise

    except Exception as error:
        if result_code != 'TemporaryFailure':
            result_code = 'PermanentFailure'
        result_string = str(error)

    finally:
        result_list.append({
            'taskId': task_id,
            'resultCode': result_code,
            'resultString': result_string,
        })

    return {
        'invocationSchemaVersion': invocation_schema_version,
        'treatMissingKeyAs': 'PermanentFailure',
        'invocationId': invocation_id,
        'results': result_list
    }
```

4. To create a deployment package with `convert.py` and `job.json` as a `.zip` file named `lambda.zip`, in your local terminal, open the `batch-transcode` folder that you created earlier, and run the following command.

   For **macOS users**, run the following command:

   ```
   zip -r lambda.zip convert.py job.json
   ```

   For **Windows users**, run the following commands:

   ```
   powershell Compress-Archive convert.py lambda.zip
   ```

   ```
   powershell Compress-Archive -update job.json lambda.zip
   ```

## Create a Lambda function with an execution role (console)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.

2. In the left navigation pane, choose **Functions**.
3. Choose **Create function**.
4. Choose **Author from scratch**.
5. Under **Basic information**, do the following:

   a. For **Function name**, enter **`tutorial-lambda-convert`**.

   b. For **Runtime**, choose **Python 3.8** or a later version of Python.
6. Choose **Change default execution role**, and under **Execution role**, choose **Use an existing role**.
7. Under **Existing role**, choose the name of the IAM role that you created for your Lambda function in Step 3 (for example, **`tutorial-lambda-transcode-role`**).
8. For the remaining settings, keep the defaults.
9. Choose **Create function**.

## Deploy your Lambda function with .zip file archives and configure the Lambda function (console)

1. In the **Code Source** section of the page for the Lambda function that you created (for example, **`tutorial-lambda-convert`**), choose **Upload from** and then **.zip file**.
2. Choose **Upload** to select your local `.zip` file.
3. Choose the `lambda.zip` file that you created earlier, and choose **Open**.
4. Choose **Save**.
5. In the **Runtime settings** section, choose **Edit**.
6. To tell the Lambda runtime which handler method in your Lambda function code to invoke, enter **`convert.handler`** in the **Handler** field.

   When you configure a function in Python, the value of the handler setting is the file name and the name of the handler module, separated by a dot (`.`). For example, `convert.handler` calls the `handler` method defined in the `convert.py` file.
7. Choose **Save**.
8. On your Lambda function page, choose the **Configuration** tab. In the left navigation pane on the **Configuration** tab, choose **Environment variables**, and then choose **Edit**.
9. Choose **Add environment variable**. Then, enter the specified **Key** and **Value** for each of the following environment variables:

- **Key**: **`DestinationBucket`** **Value**: **`tutorial-bucket-2`**

  This value is the S3 bucket for output media files that you created in Step 1.

- **Key**: **`MediaConvertRole`** **Value**: **`arn:aws:iam::`*`111122223333`*`:role/tutorial-mediaconvert-role`**

  This value is the ARN of the IAM role for MediaConvert that you created in Step 2. Make sure to replace this ARN with the actual ARN of your IAM role.

- **Key**: **`Application`** **Value**: **`Batch-Transcoding`**

  This value is the name of the application.

10. Choose **Save**.
11. (Optional) On the **Configuration** tab, in the **General configuration** section of the left navigation pane, choose **Edit**. In the **Timeout** field, enter **2** min **0** sec. Then, choose **Save**.

    **Timeout** is the amount of time that Lambda allows a function to run for an invocation before stopping it. The default is 3 seconds. Pricing is based on the amount of memory configured and the amount of time that your code runs. For more information, see AWS Lambda pricing.

## Step 5: Configure Amazon S3 Inventory for your S3 source bucket

After setting up the transcoding Lambda function, create an S3 Batch Operations job to transcode a set of videos. First, you need a list of input video objects that you want S3 Batch Operations to run the specified transcoding action on. To get a list of input video objects, you can generate an S3 Inventory report for your S3 source bucket (for example, **`tutorial-bucket-1`**).

**Substeps**

- Create and configure a bucket for S3 Inventory reports for input videos
- Configure Amazon S3 Inventory for your S3 video source bucket
- Check the inventory report for your S3 video source bucket

### Create and configure a bucket for S3 Inventory reports for input videos

To store an S3 Inventory report that lists the objects of the S3 source bucket, create an S3 Inventory destination bucket, and then configure a bucket policy for the bucket to write inventory files to the S3 source bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. For **Bucket name**, enter a name for your bucket (for example, **`tutorial-bucket-3`**).
5. For **AWS Region**, choose the AWS Region where you want the bucket to reside.

   The inventory destination bucket must be in the same AWS Region as the source bucket where you are setting up S3 Inventory. The inventory destination bucket can be in a different AWS account.
6. In **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).
7. For the remaining settings, keep the defaults.
8. Choose **Create bucket**.
9. In the **Buckets** list, choose the name of the bucket that you just created (for example, **`tutorial-bucket-3`**).
10. To grant Amazon S3 permission to write data for the inventory reports to the S3 Inventory destination bucket, choose the **Permissions** tab.
11. Scroll down to the **Bucket policy** section, and choose **Edit**. The **Bucket policy** page opens.
12. To grant permissions for S3 Inventory, in the **Policy** field, paste the following bucket policy.

    Replace the three example values with the following values:

    - The name of the bucket that you created to store the inventory reports (for example, *`tutorial-bucket-3`*).
    - The name of the source bucket that stores the input videos (for example, *`tutorial-bucket-1`*).
    - The AWS account ID that you used to create the S3 video source bucket (for example, *`111122223333`*.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Sid":"InventoryAndAnalyticsExamplePolicy",
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":"s3:PutObject",
      "Resource":["arn:aws:s3:::tutorial-bucket-3/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::tutorial-bucket-1"
           },
         "StringEquals": {
             "aws:SourceAccount": "111122223333",
             "s3:x-amz-acl": "bucket-owner-full-control"
          }
       }
    }
  ]
}
```

13. Choose **Save changes**.

## Configure Amazon S3 Inventory for your S3 video source bucket

To generate a flat file list of video objects and metadata, you must configure S3 Inventory for your S3 video source bucket. These scheduled inventory reports can include all the objects in the bucket or objects grouped by a shared prefix. In this tutorial, the S3 Inventory report includes all the video objects in your S3 source bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. To configure an S3 Inventory report of the input videos in your S3 source bucket, in the **Buckets** list, choose the name of the S3 source bucket (for example, **`tutorial-bucket-1`**).
4. Choose the **Management** tab.
5. Scroll down to the **Inventory configurations** section, and choose **Create inventory configuration**.
6. For **Inventory configuration name**, enter a name (for example, **`tutorial-inventory-config`**).
7. Under **Inventory scope**, choose **Current version only** for **Object versions** and keep the other **Inventory scope** settings set to the defaults for this tutorial.

8. In the **Report details** section, for **Destination bucket**, choose **This account**.
9. For **Destination**, choose **Browse S3**, and choose the destination bucket that you created earlier to save the inventory reports to (for example, **`tutorial-bucket-3`**). Then choose **Choose path**.

   The inventory destination bucket must be in the same AWS Region as the source bucket where you are setting up S3 Inventory. The inventory destination bucket can be in a different AWS account.

   Under the **Destination** bucket field, the **Destination bucket permission** is added to the inventory destination bucket policy, allowing Amazon S3 to place data in the inventory destination bucket. For more information, see Creating a destination bucket policy.
10. For **Frequency**, choose **Daily**.
11. For **Output format**, choose **CSV**.
12. For **Status**, choose **Enable**.
13. In the **Server-side encryption** section, choose **Disable** for this tutorial.

    For more information, see Configuring inventory by using the S3 console and Granting Amazon S3 permission to use your customer managed key for encryption.
14. In the **Additional fields - *optional*** section, select **Size**, **Last modified**, and **Storage class**.
15. Choose **Create**.

For more information, see Configuring inventory by using the S3 console.

## Check the inventory report for your S3 video source bucket

When an inventory report is published, the manifest files are sent to the S3 Inventory destination bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the video source bucket (for example, **`tutorial-bucket-1`**).
4. Choose **Management**.

5. To see if your S3 Inventory report is ready so that you can create an S3 Batch Operations job in Step 7, under **Inventory configurations**, check whether the **Create job from manifest** button is enabled.

   > **Note**
   >
   > It can take up to 48 hours to deliver the first inventory report. If the **Create job from manifest** button is disabled, the first inventory report has not been delivered. Wait until the first inventory report is delivered and the **Create job from manifest** button is enabled before you create an S3 Batch Operations job in Step 7.

6. To check an S3 Inventory report (`manifest.json`), in the **Destination** column, choose the name of the inventory destination bucket that you created earlier for storing inventory reports (for example, **`tutorial-bucket-3`**).
7. On the **Objects** tab, choose the existing folder with the name of your S3 source bucket (for example, **`tutorial-bucket-1`**). Then choose the name that you entered in **Inventory configuration name** when you created the inventory configuration earlier (for example, **`tutorial-inventory-config`**).

   You can see a list of folders with the generation dates of the reports as their names.
8. To check the daily S3 Inventory report for a particular date, choose the folder with the corresponding generation date name, and then choose `manifest.json`.
9. To check the details of the inventory report on a specific date, on the **manifest.json** page, choose **Download** or **Open**.

## Step 6: Create an IAM role for S3 Batch Operations

To use S3 Batch Operations to do batch-transcoding, you must first create an IAM role to give Amazon S3 permissions to perform S3 Batch Operations.

### Substeps

- Create an IAM policy for S3 Batch Operations
- Create an S3 Batch Operations IAM role and attach permissions policies

## Create an IAM policy for S3 Batch Operations

You must create an IAM policy that gives S3 Batch Operations permission to read the input manifest, invoke the Lambda function, and write the S3 Batch Operations job completion report.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. Choose the **JSON** tab.
5. In the **JSON** text field, paste the following JSON policy.

   In the JSON policy, replace the four example values with the following values:

   - The name of the source bucket that stores your input videos (for example, *tutorial-bucket-1*).
   - The name of the inventory destination bucket that you created in Step 5 to store `manifest.json` files (for example, *tutorial-bucket-3*).
   - The name of the bucket that you created in Step 1 to store output media files (for example, *tutorial-bucket-2*). In this tutorial, we put job completion reports in the destination bucket for output media files.
   - The role ARN of the Lambda function that you created in Step 4. To find and copy the role ARN of the Lambda function, do the following:
     - In a new browser tab, open the **Functions** page on the Lambda console at https://console.aws.amazon.com/lambda/home#/functions.
     - In **Functions** list, choose the name of the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
     - Choose **Copy ARN**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::tutorial-bucket-1/*",
                "arn:aws:s3:::tutorial-bucket-3/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::tutorial-bucket-2/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-lambda-
convert"
            ]
        }
    ]
}
```

6. Choose **Next: Tags**.
7. Choose **Next: Review**.
8. In the **Name** field, enter **`tutorial-s3batch-policy`**.
9. Choose **Create policy**.

## Create an S3 Batch Operations IAM role and attach permissions policies

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then choose the **S3** service.
4. Under **Select your use case**, choose **S3 Batch Operations**.

5. Choose **Next: Permissions**.
6. Under **Attach permissions policies**, enter the name of the IAM policy that you created earlier (for example, **`tutorial-s3batch-policy`**) in the search box to filter the list of policies. Select the check box next to the name of the policy (for example, **`tutorial-s3batch-policy`**).
7. Choose **Next: Tags**.
8. Choose **Next: Review**.
9. For **Role name**, enter **`tutorial-s3batch-role`**.
10. Choose **Create role**.

    After you create the IAM role for S3 Batch Operations, the following trust policy is automatically attached to the role. This trust policy allows the S3 Batch Operations service principal to assume the IAM role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

## Step 7: Create and run an S3 Batch Operations job

To create an S3 Batch Operations job to process the input videos in your S3 source bucket, you must specify parameters for this particular job.

> **Note**
>
> Before you start creating an S3 Batch Operations job, make sure that the **Create job from manifest** button is enabled. For more information, see Check the inventory report for your S3 video source bucket. If the **Create job from manifest** button is disabled, the first inventory report has not been delivered and you must wait until the button is enabled.

After you configure Amazon S3 Inventory for your S3 source bucket in Step 5, it can take up to 48 hours to deliver the first inventory report.

**Substeps**

- Create an S3 Batch Operations job
- Run the S3 Batch Operations job to invoke your Lambda function
- (Optional) Check your completion report
- (Optional) Monitor each Lambda invocation in the Lambda console
- (Optional) Monitor each MediaConvert video-transcoding job in the MediaConvert console

## Create an S3 Batch Operations job

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose **Create job**.
4. For **AWS Region**, choose the Region where you want to create your job.

   In this tutorial, to use the S3 Batch Operations job to invoke a Lambda function, you must create the job in the same Region as the S3 video source bucket where the objects referenced in the manifest are located.
5. In the **Manifest** section, do the following:

   a. For **Manifest format**, choose **S3 Inventory report (manifest.json)**.

   b. For **Manifest object**, choose **Browse S3** to find the bucket that you created in Step 5 for storing inventory reports (for example, **`tutorial-bucket-3`**). On the **Manifest object** page, navigate through the object names until you find a `manifest.json` file for a specific date. This file lists the information about all the videos that you want to batch-transcode. When you've found the `manifest.json` file that you want to use, choose the option button next to it. Then choose **Choose path**.

   c. (Optional) For **Manifest object version ID - *optional***, enter the version ID for the manifest object if you want to use a version other than the most recent.
6. Choose **Next**.

7. To use the Lambda function to transcode all the objects listed in the selected `manifest.json` file, under **Operation type**, choose **Invoke AWS Lambda function**.
8. In the **Invoke Lambda function** section, do the following:
   a. Choose **Choose from functions in your account**.
   b. For **Lambda function**, choose the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
   c. For **Lambda function version**, keep the default value **$LATEST**.
9. Choose **Next**. The **Configure additional options** page opens.
10. In the **Additional options** section, keep the default settings.

    For more information about these options, see Batch Operations job request elements.
11. In the **Completion report** section, for **Path to completion report destination**, choose **Browse S3**. Find the bucket that you created for output media files in Step 1 (for example, **`tutorial-bucket-2`**). Choose the option button next to that bucket's name. Then choose **Choose path**.

    For the remaining **Completion report** settings, keep the defaults. For more information about completion report settings, see Batch Operations job request elements. A completion report maintains a record of the job's details and the operations performed.
12. In the **Permissions** section, choose **Choose from existing IAM roles**. For **IAM role**, choose the IAM role for your S3 Batch Operations job that you created in Step 6 (for example, **`tutorial-s3batch-role`**).
13. Choose **Next**.
14. On the **Review** page, review the settings. Then choose **Create job**.

    After S3 finishes reading your S3 Batch Operations job's manifest, it sets the **Status** of the job to **Awaiting your confirmation to run**. To see updates to the job's status, refresh the page. You can't run your job until its status is **Awaiting your confirmation to run**.

## Run the S3 Batch Operations job to invoke your Lambda function

Run your Batch Operations job to invoke your Lambda function for video transcoding. If your job fails, you can check your completion report to identify the cause.

**To run the S3 Batch Operations job**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. In the **Jobs** list, choose the **Job ID** of the job on the first row, which is the S3 Batch Operations job that you created earlier.
4. Choose **Run job**.
5. Review your job parameters again, and confirm that the value for **Total objects listed in manifest** is the same as the number of objects in the manifest. Then choose **Run job**.

   Your S3 Batch Operations job page opens.
6. After the job starts running, on your job page, under **Status**, check the progress of your S3 Batch Operations job, such as **Status**, **% Complete**, **Total succeeded (rate)**, **Total failed (rate)**, **Date terminated**, and **Reason for termination**.

   When the S3 Batch Operations job completes, view the data on your job page to confirm that the job finished as expected.

   If more than 50 percent of an S3 Batch Operations job's object operations fail after more than 1,000 operations have been attempted, the job automatically fails. To check your completion report to identify the cause of the failures, use the following optional procedure.

## (Optional) Check your completion report

You can use your completion report to determine which objects failed and the cause of the failures.

**To check your completion report for details about failed objects**

1. On the page of your S3 Batch Operations job, scroll down to the **Completion report** section, and choose the link under **Completion report destination**.

   The S3 output destination bucket's page opens.
2. On the **Objects** tab, choose the folder that has a name ending with the job ID of the S3 Batch Operations job that you created earlier.
3. Choose **results/**.
4. Select the check box next to the `.csv` file.

5. To view the job report, choose **Open** or **Download**.

## (Optional) Monitor each Lambda invocation in the Lambda console

After the S3 Batch Operations job starts running, the job invokes the Lambda function for each input video object. S3 writes logs of each Lambda invocation to CloudWatch Logs. You can use the Lambda console's monitoring dashboard to monitor your Lambda function.

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. In the **Functions** list, choose the name of the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
4. Choose the **Monitor** tab.
5. Under **Metrics**, see the runtime metrics for your Lambda function.
6. Under **Logs**, view log data for each Lambda invocation through CloudWatch Logs Insights.

> **Note**
>
> When you use S3 Batch Operations with a Lambda function, the Lambda function is invoked on each object. If your S3 Batch Operations job is large, it can invoke multiple Lambda functions at the same time, causing a spike in Lambda concurrency.
> Each AWS account has a Lambda concurrency quota per Region. For more information, see AWS Lambda Function Scaling in the *AWS Lambda Developer Guide*. A best practice for using Lambda functions with S3 Batch Operations is to set a concurrency limit on the Lambda function itself. Setting a concurrency limit keeps your job from consuming most of your Lambda concurrency and potentially throttling other functions in your account. For more information, see Managing Lambda reserved concurrency in the *AWS Lambda Developer Guide*.

## (Optional) Monitor each MediaConvert video-transcoding job in the MediaConvert console

A MediaConvert job does the work of transcoding a media file. When your S3 Batch Operations job invokes your Lambda function for each video, each Lambda function invocation creates a MediaConvert transcoding job for each input video.

1. Sign in to the AWS Management Console and open the MediaConvert console at https://console.aws.amazon.com/mediaconvert/.
2. If the MediaConvert introductory page appears, choose **Get started**.
3. From the list of **Jobs**, view each row to monitor the transcoding task for each input video.
4. Identify the row of a job that you want to check, and choose the **Job ID** link to open the job details page.
5. On the **Job summary** page, under **Outputs**, choose the link for the HLS, MP4, or Thumbnails output, depending on what is supported by your browser, to go to the S3 destination bucket for the output media files.
6. In the corresponding folder (HLS, MP4, or Thumbnails) of your S3 output destination bucket, choose the name of the output media file object.

   The object's detail page opens.
7. On the object's detail page, under **Object overview**, choose the link under **Object URL** to watch the transcoded output media file.

## Step 8: Check the output media files from your S3 destination bucket

**To check the output media files from your S3 destination bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the S3 destination bucket for output media files that you created in Step 1 (for example, **`tutorial-bucket-2`**).
4. On the **Objects** tab, each input video has a folder that has the name of the input video. Each folder contains the transcoded output media files for an input video.

   To check the output media files for an input video, do the following:

   a. Choose the folder with the name of the input video that you want to check.

   b. Choose the **Default/** folder.

   c. Choose the folder for a transcoded format (HLS, MP4, or thumbnails in this tutorial).

   d. Choose the name of the output media file.

e. To watch the transcoded file, on the object's details page, choose the link under **Object URL**.

   Output media files in the HLS format are split into short segments. To play these videos, embed the object URL of the `.m3u8` file in a compatible player.

## Step 9: Clean up

If you transcoded videos using S3 Batch Operations, Lambda, and MediaConvert only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

**Substeps**

- Delete the S3 Inventory configuration for your S3 source bucket
- Delete the Lambda function
- Delete the CloudWatch log group
- Delete the IAM roles together with the inline policies for the IAM roles
- Delete the customer-managed IAM policy
- Empty the S3 buckets
- Delete the S3 buckets

### Delete the S3 Inventory configuration for your S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of your source bucket (for example, **`tutorial-bucket-1`**).
4. Choose the **Management** tab.
5. In the **Inventory configurations** section, choose the option button next to the inventory configuration that you created in Step 5 (for example, **`tutorial-inventory-config`**).
6. Choose **Delete**, and then choose **Confirm**.

## Delete the Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Select the check box next to the function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
4. Choose **Actions**, and then choose **Delete**.
5. In the **Delete function** dialog box, choose **Delete**.

## Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Logs**, and then choose **Log groups**.
3. Select the check box next to the log group that has a name ending with the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
4. Choose **Actions**, and then choose **Delete log group(s)**.
5. In the **Delete log group(s)** dialog box, choose **Delete**.

## Delete the IAM roles together with the inline policies for the IAM roles

To delete the IAM roles that you created in Step 2, Step 3, and Step 6, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then select the check boxes next to the role names that you want to delete.
3. At the top of the page, choose **Delete**.
4. In the confirmation dialog box, enter the required response in the text input field based on the prompt, and choose **Delete**.

## Delete the customer-managed IAM policy

To delete the customer-managed IAM policy that you created in Step 6, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose the option button next to the policy that you created in Step 6 (for example, **`tutorial-s3batch-policy`**). You can use the search box to filter the list of policies.
4. Choose **Actions**, and then choose **Delete**.
5. Confirm that you want to delete this policy by entering its name in the text field, and then choose **Delete**.

## Empty the S3 buckets

To empty the S3 buckets that you created in Prerequisites, Step 1, and Step 5, do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.

## Delete the S3 buckets

To delete the S3 buckets that you created in Prerequisites, Step 1, and Step 5, do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the option button next to the name of the bucket that you want to delete.
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

## Next steps

After completing this tutorial, you can further explore other relevant use cases:

- You can use Amazon CloudFront to stream the transcoded media files to viewers across the globe. For more information, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53.
- You can transcode videos at the moment when you upload them to the S3 source bucket. To do so, you can configure an Amazon S3 event trigger that automatically invokes the Lambda function to transcode new objects in S3 with MediaConvert. For more information, see Tutorial: Using an Amazon S3 trigger to invoke a Lambda function in the *AWS Lambda Developer Guide*.

# Tutorial: Configuring a static website on Amazon S3

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

You can configure an Amazon S3 bucket to function like a website. This example walks you through the steps of hosting a website on Amazon S3.

**Important**

The following tutorial requires disabling Block Public Access. We recommend keeping Block Public Access enabled. If you want to keep all four Block Public Access settings enabled and host a static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Amazon S3 static websites support only HTTP endpoints. Amazon CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS.

HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks. For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

**Topics**



## Step 1: Create a bucket

The following instructions provide an overview of how to create your buckets for website hosting. For detailed, step-by-step instructions on creating a bucket, see Creating a bucket.

**To create a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.
3. Enter the **Bucket name** (for example, **`example.com`**).
4. Choose the Region where you want to create the bucket.

   Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.
5. To accept the default settings and create the bucket, choose **Create**.

# Step 2: Enable static website hosting

After you create a bucket, you can enable static website hosting for your bucket. You can create a new bucket or use an existing bucket.

**To enable static website hosting**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable static website hosting for.
3. Choose **Properties**.
4. Under **Static website hosting**, choose **Edit**.
5. Choose **Use this bucket to host a website**.
6. Under **Static website hosting**, choose **Enable**.
7. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.
8. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.
9. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

   For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.
10. Choose **Save changes**.

Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.

11. Under **Static website hosting**, note the **Endpoint**.

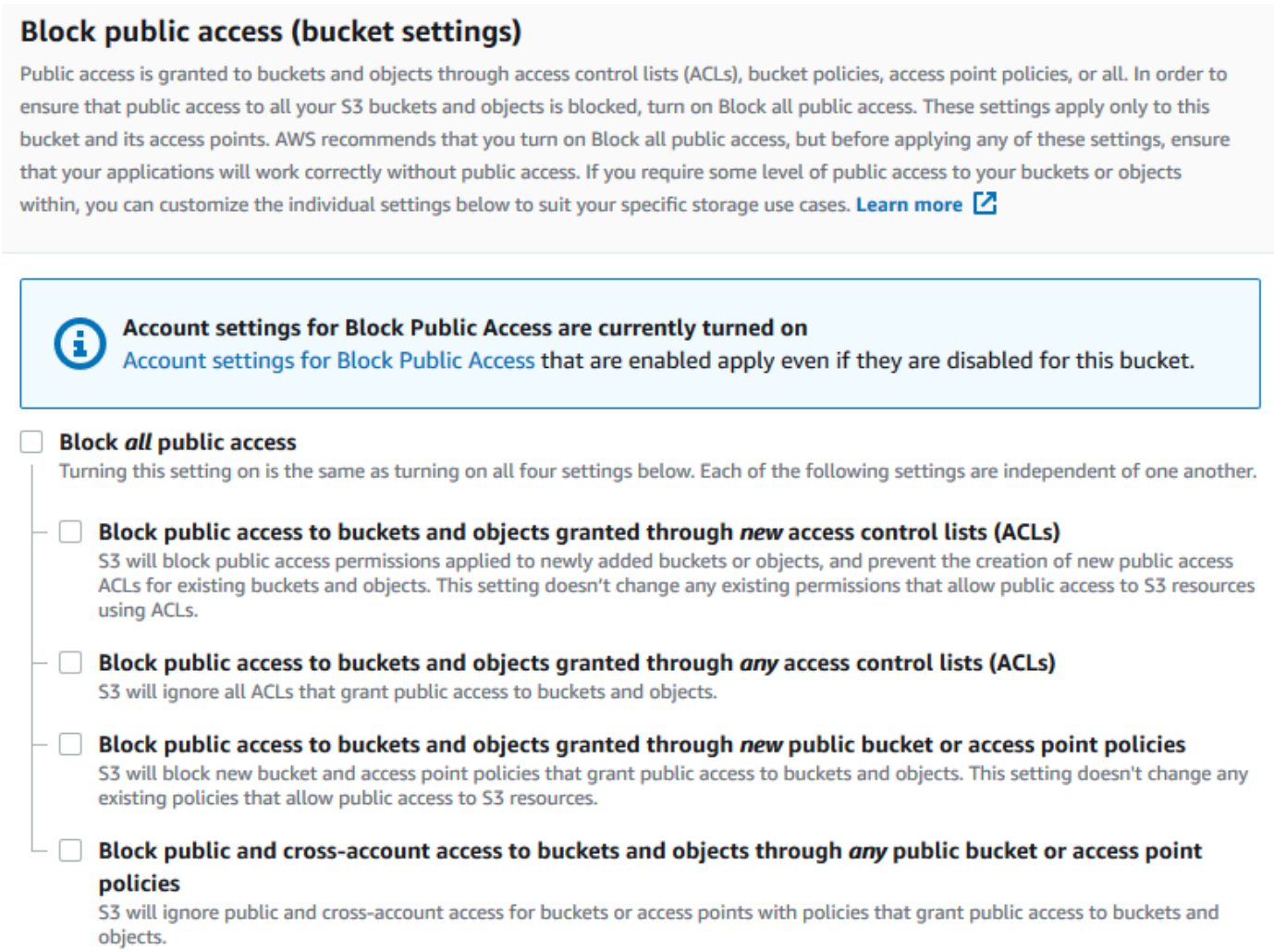
    The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

## Step 3: Edit Block Public Access settings

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.




Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

## Step 4: Add a bucket policy that makes your bucket content publicly available

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

**Important**

The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *Bucket-Name* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

## Step 5: Configure an index document

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

**To configure the index document**

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.
5. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

   After enabling static website hosting, proceed to step 6.
6. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.
7. (Optional) Upload other website content to your bucket.

## Step 6: Configure an error document

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

**To configure an error document**

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error document** name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.
3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.
5. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

After enabling static website hosting, proceed to step 6.

6. To upload the error document to your bucket, do one of the following:

   - Drag and drop the error document file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

## Step 7: Test your website endpoint

After you configure static website hosting for your bucket, you can test your website endpoint.

> **Note**
>
> Amazon S3 does not support HTTPS access to the website. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3.
> For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Properties**.
3. At the bottom of the page, under **Static website hosting**, choose your **Bucket website endpoint**.

   Your index document opens in a separate browser window.

You now have a website hosted on Amazon S3. This website is available at the Amazon S3 website endpoint. However, you might have a domain, such as `example.com`, that you want to use to serve the content from the website you created. You might also want to use Amazon S3 root domain support to serve requests for both `http://www.example.com` and `http://example.com`. This requires additional steps. For an example, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

## Step 8: Clean up

If you created your static website only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges. After you delete your AWS resources, your website is no longer available. For more information, see Deleting a bucket.

# Tutorial: Configuring a static website using a custom domain registered with Route 53

Suppose that you want to host a static website on Amazon S3. You've registered a domain with Amazon Route 53 (for example, `example.com`), and you want requests for `http://www.example.com` and `http://example.com` to be served from your Amazon S3 content. You can use this walkthrough to learn how to host a static website and create redirects on Amazon S3 for a website with a custom domain name that is registered with Route 53. You can work with an existing website that you want to host on Amazon S3, or use this walkthrough to start from scratch.

After you complete this walkthrough, you can optionally use Amazon CloudFront to improve the performance of your website. For more information, see Speeding up your website with Amazon CloudFront.

> **Note**
>
> Amazon S3 website endpoints do not support HTTPS or access points. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3. For a tutorial about how to host your content securely with CloudFront and Amazon S3, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53. For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

### Automating static website setup with an AWS CloudFormation template

You can use an AWS CloudFormation template to automate your static website setup. The AWS CloudFormation template sets up the components that you need to host a secure static website so that you can focus more on your website's content and less on configuring components.

The AWS CloudFormation template includes the following components:

- Amazon S3 – Creates an Amazon S3 bucket to host your static website.
- CloudFront – Creates a CloudFront distribution to speed up your static website.
- Lambda@Edge – Uses Lambda@Edge to add security headers to every server response. Security headers are a group of headers in the web server response that tell web browsers to take extra security precautions. For more information, see the blog post Adding HTTP security headers using Lambda@Edge and Amazon CloudFront.

This AWS CloudFormation template is available for you to download and use. For information and instructions, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

**Topics**



## Before you begin

As you follow the steps in this example, you work with the following services:

**Amazon Route 53** – You use Route 53 to register domains and to define where you want to route internet traffic for your domain. The example shows how to create Route 53 alias records that route traffic for your domain (`example.com`) and subdomain (`www.example.com`) to an Amazon S3 bucket that contains an HTML file.

**Amazon S3** – You use Amazon S3 to create buckets, upload a sample website page, configure permissions so that everyone can see the content, and then configure the buckets for website hosting.

## Step 1: Register a custom domain with Route 53

If you don't already have a registered domain name, such as `example.com`, register one with Route 53. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*. After you register your domain name, you can create and configure your Amazon S3 buckets for website hosting.

## Step 2: Create two buckets

To support requests from both the root domain and subdomain, you create two buckets.

- **Domain bucket** – `example.com`
- **Subdomain bucket** – `www.example.com`

These bucket names must match your domain name exactly. In this example, the domain name is `example.com`. You host your content out of the root domain bucket (`example.com`). You create a redirect request for the subdomain bucket (`www.example.com`). If someone enters `www.example.com` in their browser, they are redirected to `example.com` and see the content that is hosted in the Amazon S3 bucket with that name.

**To create your buckets for website hosting**

The following instructions provide an overview of how to create your buckets for website hosting. For detailed, step-by-step instructions on creating a bucket, see Creating a bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create your root domain bucket:

    a. Choose **Create bucket**.

b. Enter the **Bucket name** (for example, **`example.com`**).

c. Choose the Region where you want to create the bucket.

Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.

d. To accept the default settings and create the bucket, choose **Create**.

3. Create your subdomain bucket:

a. Choose **Create bucket**.

b. Enter the **Bucket name** (for example, **`www.example.com`**).

c. Choose the Region where you want to create the bucket.

Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.

d. To accept the default settings and create the bucket, choose **Create**.

In the next step, you configure `example.com` for website hosting.

## Step 3: Configure your root domain bucket for website hosting

In this step, you configure your root domain bucket (`example.com`) as a website. This bucket will contain your website content. When you configure a bucket for website hosting, you can access the website using the Website endpoints.

**To enable static website hosting**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable static website hosting for.
3. Choose **Properties**.
4. Under **Static website hosting**, choose **Edit**.
5. Choose **Use this bucket to host a website**.
6. Under **Static website hosting**, choose **Enable**.

7. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.

8. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.

9. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

   For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.

10. Choose **Save changes**.

    Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.

11. Under **Static website hosting**, note the **Endpoint**.

    The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

After you edit block public access settings and add a bucket policy that allows public read access, you can use the website endpoint to access your website.

In the next step, you configure your subdomain (`www.example.com`) to redirect requests to your domain (`example.com`).

## Step 4: Configure your subdomain bucket for website redirect

After you configure your root domain bucket for website hosting, you can configure your subdomain bucket to redirect all requests to the domain. In this example, all requests for `www.example.com` are redirected to `example.com`.

**To configure a redirect request**

1. On the Amazon S3 console, in the **Buckets** list, choose your subdomain bucket name (`www.example.com` in this example).
2. Choose **Properties**.
3. Under **Static website hosting**, choose **Edit**.
4. Choose **Redirect requests for an object**.
5. In the **Target bucket** box, enter your root domain, for example, **`example.com`**.
6. For **Protocol**, choose **http**.
7. Choose **Save changes**.

## Step 5: Configure logging for website traffic

If you want to track the number of visitors accessing your website, you can optionally enable logging for your root domain bucket. For more information, see Logging requests with server access logging. If you plan to use Amazon CloudFront to speed up your website, you can also use CloudFront logging.

**To enable server access logging for your root domain bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the same Region where you created the bucket that is configured as a static website, create a bucket for logging, for example `logs.example.com`.
3. Create a folder for the server access logging log files (for example, `logs`).
4. (Optional) If you want to use CloudFront to improve your website performance, create a folder for the CloudFront log files (for example, `cdn`).

> **Important**
>
> When you create or update a distribution and enable CloudFront logging, CloudFront updates the bucket access control list (ACL) to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the Bucket owner enforced setting for S3 Object Ownership to disable ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

5. In the **Buckets** list, choose your root domain bucket.
6. Choose **Properties**.
7. Under **Server access logging**, choose **Edit**.
8. Choose **Enable**.
9. Under the **Target bucket**, choose the bucket and folder destination for the server access logs:
   - Browse to the folder and bucket location:
     1. Choose **Browse S3**.
     2. Choose the bucket name, and then choose the logs folder.
     3. Choose **Choose path**.
   - Enter the S3 bucket path, for example, `s3://logs.example.com/logs/`.
10. Choose **Save changes**.

    In your log bucket, you can now access your logs. Amazon S3 writes website access logs to your log bucket every 2 hours.

## Step 6: Upload index and website content

In this step, you upload your index document and optional website content to your root domain bucket.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

**To configure the index document**

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.

5. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

   After enabling static website hosting, proceed to step 6.

6. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

7. (Optional) Upload other website content to your bucket.

## Step 7: Upload an error document

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

**To configure an error document**

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error document** name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.
3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.
5. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

   After enabling static website hosting, proceed to step 6.
6. To upload the error document to your bucket, do one of the following:

   - Drag and drop the error document file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

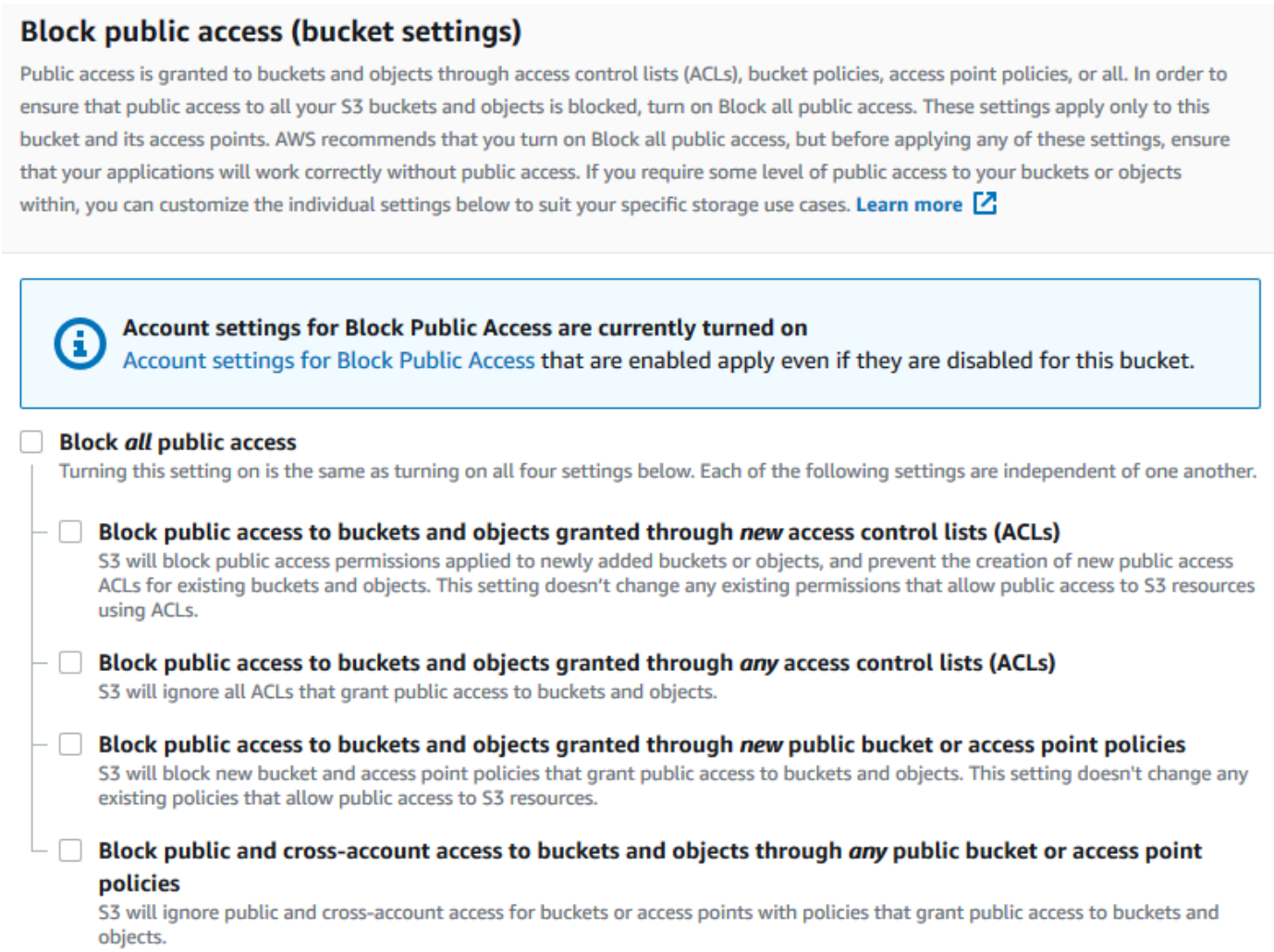
## Step 8: Edit S3 Block Public Access settings

In this example, you edit block public access settings for the domain bucket (`example.com`) to allow public access.

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.




Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

## Step 9: Attach a bucket policy

In this example, you attach a bucket policy to the domain bucket (`example.com`) to allow public read access. You replace the *`Bucket-Name`* in the example bucket policy with the name of your domain bucket, for example `example.com`.

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

**Important**

The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *Bucket-Name* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

In the next step, you can figure out your website endpoints and test your domain endpoint.

## Step 10: Test your domain endpoint

After you configure your domain bucket to host a public website, you can test your endpoint. For more information, see Website endpoints. You can only test the endpoint for your domain bucket because your subdomain bucket is set up for website redirect and not static website hosting.

> **Note**
>
> Amazon S3 does not support HTTPS access to the website. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3.
> For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Properties**.
3. At the bottom of the page, under **Static website hosting**, choose your **Bucket website endpoint**.

   Your index document opens in a separate browser window.

In the next step, you use Amazon Route 53 to enable customers to use both of your custom URLs to navigate to your site.

## Step 11: Add alias records for your domain and subdomain

In this step, you create the alias records that you add to the hosted zone for your domain maps `example.com` and `www.example.com`. Instead of using IP addresses, the alias records use the

Amazon S3 website endpoints. Amazon Route 53 maintains a mapping between the alias records and the IP addresses where the Amazon S3 buckets reside. You create two alias records, one for your root domain and one for your subdomain.

**Add an alias record for your root domain and subdomain**

**To add an alias record for your root domain (`example.com`)**

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.

   > **Note**
   >
   > If you don't already use Route 53, see Step 1: Register a domain in the *Amazon Route 53 Developer Guide*. After completing your setup, you can resume the instructions.

2. Choose **Hosted zones**.
3. In the list of hosted zones, choose the name of the hosted zone that matches your domain name.
4. Choose **Create record**.
5. Choose **Switch to wizard**.

   > **Note**
   >
   > If you want to use quick create to create your alias records, see Configuring Route 53 to route traffic to an S3 Bucket.

6. Choose **Simple routing**, and choose **Next**.
7. Choose **Define simple record**.
8. In **Record name**, accept the default value, which is the name of your hosted zone and your domain.
9. In **Value/Route traffic to**, choose **Alias to S3 website endpoint**.
10. Choose the Region.
11. Choose the S3 bucket.

    The bucket name should match the name that appears in the **Name** box. In the **Choose S3 bucket** list, the bucket name appears with the Amazon S3 website endpoint for the Region

where the bucket was created, for example, `s3-website-us-west-1.amazonaws.com (example.com)`.

**Choose S3 bucket** lists a bucket if:

- You configured the bucket as a static website.
- The bucket name is the same as the name of the record that you're creating.
- The current AWS account created the bucket.

If your bucket does not appear in the **Choose S3 bucket** list, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, **`s3-website-us-west-2.amazonaws.com`**. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.

12. In **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
13. For **Evaluate target health**, choose **No**.
14. Choose **Define simple record**.

**To add an alias record for your subdomain (`www.example.com`)**

1. Under **Configure records**, choose **Define simple record**.
2. In **Record name** for your subdomain, type `www`.
3. In **Value/Route traffic to**, choose **Alias to S3 website endpoint**.
4. Choose the Region.
5. Choose the S3 bucket, for example, `s3-website-us-west-2.amazonaws.com (www.example.com)`.

   If your bucket does not appear in the **Choose S3 bucket** list, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, **`s3-website-us-west-2.amazonaws.com`**. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.
6. In **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
7. For **Evaluate target health**, choose **No**.
8. Choose **Define simple record**.

9. On the **Configure records** page, choose **Create records**.

> **Note**
>
> Changes generally propagate to all Route 53 servers within 60 seconds. When propagation is done, you can route traffic to your Amazon S3 bucket by using the names of the alias records that you created in this procedure.

**Add an alias record for your root domain and subdomain (old Route 53 console)**

**To add an alias record for your root domain (`example.com`)**

The Route 53 console has been redesigned. In the Route 53 console you can temporarily use the old console. If you choose to work with the old Route 53 console, use the procedure below.

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.

   > **Note**
   >
   > If you don't already use Route 53, see Step 1: Register a domain in the *Amazon Route 53 Developer Guide*. After completing your setup, you can resume the instructions.

2. Choose **Hosted Zones**.
3. In the list of hosted zones, choose the name of the hosted zone that matches your domain name.
4. Choose **Create Record Set**.
5. Specify the following values:

   **Name**

   Accept the default value, which is the name of your hosted zone and your domain.

   For the root domain, you don't need to enter any additional information in the **Name** field.

   **Type**

   Choose **A – IPv4 address**.

**Alias**

Choose **Yes**.

**Alias Target**

In the **S3 website endpoints** section of the list, choose your bucket name.

The bucket name should match the name that appears in the **Name** box. In the **Alias Target** listing, the bucket name is followed by the Amazon S3 website endpoint for the Region where the bucket was created, for example `example.com (s3-website-us-west-2.amazonaws.com)`. **Alias Target** lists a bucket if:

- You configured the bucket as a static website.
- The bucket name is the same as the name of the record that you're creating.
- The current AWS account created the bucket.

If your bucket does not appear in the **Alias Target** listing, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, `s3-website-us-west-2`. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.

**Routing Policy**

Accept the default value of **Simple**.

**Evaluate Target Health**

Accept the default value of **No**.

6. Choose **Create**.

**To add an alias record for your subdomain (`www.example.com`)**

1. In the hosted zone for your root domain (`example.com`), choose **Create Record Set**.
2. Specify the following values:

**Name**

For the subdomain, enter `www` in the box.

**Type**

Choose **A – IPv4 address**.

**Alias**

Choose **Yes**.

**Alias Target**

In the **S3 website endpoints** section of the list, choose the same bucket name that appears in the **Name** field—for example, `www.example.com (s3-website-us-west-2.amazonaws.com)`.

**Routing Policy**

Accept the default value of **Simple**.

**Evaluate Target Health**

Accept the default value of **No**.

3. Choose **Create**.

> **Note**
>
> Changes generally propagate to all Route 53 servers within 60 seconds. When propagation is done, you can route traffic to your Amazon S3 bucket by using the names of the alias records that you created in this procedure.

## Step 12: Test the website

Verify that the website and the redirect work correctly. In your browser, enter your URLs. In this example, you can try the following URLs:

- **Domain** (`http://example.com`) – Displays the index document in the `example.com` bucket.
- **Subdomain** (`http://www.example.com`) – Redirects your request to `http://example.com`. You see the index document in the `example.com` bucket.

If your website or redirect links don't work, you can try the following:

- **Clear cache** – Clear the cache of your web browser.
- **Check name servers** – If your web page and redirect links don't work after you've cleared your cache, you can compare the name servers for your domain and the name servers for your hosted zone. If the name servers don't match, you might need to update your domain name servers to match those listed under your hosted zone. For more information, see Adding or changing name servers and glue records for a domain.

After you've successfully tested your root domain and subdomain, you can set up an Amazon CloudFront distribution to improve the performance of your website and provide logs that you can use to review website traffic. For more information, see Speeding up your website with Amazon CloudFront.

## Speeding up your website with Amazon CloudFront

You can use Amazon CloudFront to improve the performance of your Amazon S3 website. CloudFront makes your website files (such as HTML, images, and video) available from data centers around the world (known as *edge locations*). When a visitor requests a file from your website, CloudFront automatically redirects the request to a copy of the file at the nearest edge location. This results in faster download times than if the visitor had requested the content from a data center that is located farther away.

CloudFront caches content at edge locations for a period of time that you specify. If a visitor requests content that has been cached for longer than the expiration date, CloudFront checks the origin server to see if a newer version of the content is available. If a newer version is available, CloudFront copies the new version to the edge location. Changes that you make to the original content are replicated to edge locations as visitors request the content.

### Using CloudFront without Route 53

The tutorial on this page uses Route 53 to point to your CloudFront distribution. However, if you want to serve content hosted in an Amazon S3 bucket using CloudFront without using Route 53, see Amazon CloudFront Tutorials: Setting up a Dynamic Content Distribution for Amazon S3. When you serve content hosted in an Amazon S3 bucket using CloudFront, you can use any bucket name, and both HTTP and HTTPS are supported.

### Automating set up with an AWS CloudFormation template

For more information about using an AWS CloudFormation template to configure a secure static website that creates a CloudFront distribution to serve your website, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

**Topics**



## Step 1: Create a CloudFront distribution

First, you create a CloudFront distribution. This makes your website available from data centers around the world.

**To create a distribution with an Amazon S3 origin**

1. Open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. Choose **Create Distribution**.
3. On the **Create Distribution** page, in the **Origin Settings** section, for **Origin Domain Name**, enter the Amazon S3 website endpoint for your bucket—for example, **`example.com.s3-website.us-west-1.amazonaws.com`**.

   CloudFront fills in the **Origin ID** for you.
4. For **Default Cache Behavior Settings**, keep the values set to the defaults.

   With the default settings for **Viewer Protocol Policy**, you can use HTTPS for your static website. For more information these configuration options, see Values that You Specify When You Create or Update a Web Distribution in the *Amazon CloudFront Developer Guide*.
5. For **Distribution Settings**, do the following:

   a. Leave **Price Class** set to **Use All Edge Locations (Best Performance)**.

   b. Set **Alternate Domain Names (CNAMEs)** to the root domain and www subdomain. In this tutorial, these are `example.com` and `www.example.com`.

> **Important**
>
> Before you perform this step, note the requirements for using alternate domain names, in particular the need for a valid SSL/TLS certificate.

c. For **SSL Certificate**, choose **Custom SSL Certificate (example.com)**, and choose the custom certificate that covers the domain and subdomain names.

   For more information, see SSL Certificate in the *Amazon CloudFront Developer Guide*.

d. In **Default Root Object**, enter the name of your index document, for example, `index.html`.

   If the URL used to access the distribution doesn't contain a file name, the CloudFront distribution returns the index document. The **Default Root Object** should exactly match the name of the index document for your static website. For more information, see Configuring an index document.

e. Set **Logging** to **On**.

> **Important**
>
> When you create or update a distribution and enable CloudFront logging, CloudFront updates the bucket access control list (ACL) to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the Bucket owner enforced setting for S3 Object Ownership to disable ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

f. For **Bucket for Logs**, choose the logging bucket that you created.

   For more information about configuring a logging bucket, see (Optional) Logging web traffic.

g. If you want to store the logs that are generated by traffic to the CloudFront distribution in a folder, in **Log Prefix**, enter the folder name.

h. Keep all other settings at their default values.

6. Choose **Create Distribution**.
7. To see the status of the distribution, find the distribution in the console and check the **Status** column.

   A status of `InProgress` indicates that the distribution is not yet fully deployed.

   After your distribution is deployed, you can reference your content with the new CloudFront domain name.
8. Record the value of **Domain Name** shown in the CloudFront console, for example, `dj4p1rv6mvubz.cloudfront.net`.
9. To verify that your CloudFront distribution is working, enter the domain name of the distribution in a web browser.

   If your website is visible, the CloudFront distribution works. If your website has a custom domain registered with Amazon Route 53, you will need the CloudFront domain name to update the record set in the next step.

## Step 2: Update the record sets for your domain and subdomain

Now that you have successfully created a CloudFront distribution, update the alias record in Route 53 to point to the new CloudFront distribution.

**To update the alias record to point to a CloudFront distribution**

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation, choose **Hosted zones**.
3. On the **Hosted Zones** page, choose the hosted zone that you created for your subdomain, for example, `www.example.com`.
4. Under **Records**, select the *A* record that you created for your subdomain.
5. Under **Record details**, choose **Edit record**.
6. Under **Route traffic to**, choose **Alias to CloudFront distribution**.
7. Under **Choose distribution**, choose the CloudFront distribution.
8. Choose **Save**.
9. To redirect the *A* record for the root domain to the CloudFront distribution, repeat this procedure for the root domain, for example, `example.com`.

The update to the record sets takes effect within 2–48 hours.

10. To see whether the new *A* records have taken effect, in a web browser, enter your subdomain URL, for example, `http://www.example.com`.

    If the browser no longer redirects you to the root domain (for example, `http://example.com`), the new A records are in place. When the new *A* record has taken effect, traffic routed by the new *A* record to the CloudFront distribution is not redirected to the root domain. Any visitors who reference the site by using `http://example.com` or `http://www.example.com` are redirected to the nearest CloudFront edge location, where they benefit from faster download times.

    > **Tip**
    >
    > Browsers can cache redirect settings. If you think the new *A* record settings should have taken effect, but your browser still redirects `http://www.example.com` to `http://example.com`, try clearing your browser history and cache, closing and reopening your browser application, or using a different web browser.

## (Optional) Step 3: Check the log files

The access logs tell you how many people are visiting the website. They also contain valuable business data that you can analyze with other services, such as Amazon EMR.

CloudFront logs are stored in the bucket and folder that you choose when you create a CloudFront distribution and enable logging. CloudFront writes logs to your log bucket within 24 hours from when the corresponding requests are made.

**To see the log files for your website**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the logging bucket for your website.
3. Choose the CloudFront logs folder.
4. Download the `.gzip` files written by CloudFront before opening them.

If you created your website only as a learning exercise, you can delete the resources that you allocated so that you no longer accrue charges. To do so, see Cleaning up your example resources. After you delete your AWS resources, your website is no longer available.

## Cleaning up your example resources

If you created your static website as a learning exercise, you should delete the AWS resources that you allocated so that you no longer accrue charges. After you delete your AWS resources, your website is no longer available.

**Tasks**

- Step 1: Delete the Amazon CloudFront distribution
- Step 2: Delete the Route 53 hosted zone
- Step 3: Disable logging and delete your S3 bucket

### Step 1: Delete the Amazon CloudFront distribution

Before you delete an Amazon CloudFront distribution, you must disable it. A disabled distribution is no longer functional and does not accrue charges. You can enable a disabled distribution at any time. After you delete a disabled distribution, it is no longer available.

**To disable and delete a CloudFront distribution**

1. Open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. Select the distribution that you want to disable, and then choose **Disable**.
3. When prompted for confirmation, choose **Yes, Disable**.
4. Select the disabled distribution, and then choose **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

### Step 2: Delete the Route 53 hosted zone

Before you delete the hosted zone, you must delete the record sets that you created. You don't need to delete the NS and SOA records; these are automatically deleted when you delete the hosted zone.

**To delete the record sets**

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the list of domain names, select your domain name, and then choose **Go to Record Sets**.
3. In the list of record sets, select the *A* records that you created.

   The type of each record set is listed in the **Type** column.
4. Choose **Delete Record Set**.
5. When prompted for confirmation, choose **Confirm**.

**To delete a Route 53 hosted zone**

1. Continuing from the previous procedure, choose **Back to Hosted Zones**.
2. Select your domain name, and then choose **Delete Hosted Zone**.
3. When prompted for confirmation, choose **Confirm**.

## Step 3: Disable logging and delete your S3 bucket

Before you delete your S3 bucket, make sure that logging is disabled for the bucket. Otherwise, AWS continues to write logs to your bucket as you delete it.

**To disable logging for a bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Under **Buckets**, choose your bucket name, and then choose **Properties**.
3. From **Properties**, choose **Logging**.
4. Clear the **Enabled** check box.
5. Choose **Save**.

Now, you can delete your bucket. For more information, see Deleting a bucket.

# Creating, configuring, and working with Amazon S3 buckets

To store your data in Amazon S3, you work with resources known as buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to a bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

> **Note**
>
> With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

The topics in this section provide an overview of working with buckets in Amazon S3. They include information about naming, creating, accessing, and deleting buckets. For more information about viewing or listing objects in a bucket, see Organizing, listing, and working with your objects.

**Topics**

- Setting default server-side encryption behavior for Amazon S3 buckets
- Working with Mountpoint for Amazon S3
- Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration
- Using Requester Pays buckets for storage transfers and usage
- Bucket quotas, restrictions, and limitations

## Buckets overview

To upload your data (photos, videos, documents, etc.) to Amazon S3, you must first create an S3 bucket in one of the AWS Regions.

A bucket is a container for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account. To request an increase, visit the Service Quotas console.

Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `amzn-s3-demo-bucket` bucket in the US West (Oregon) Region, then it is addressable by using the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`. For more information, see Accessing a Bucket.

In terms of implementation, buckets and objects are AWS resources, and Amazon S3 provides APIs for you to manage them. For example, you can create a bucket and upload objects using the Amazon S3 API. You can also use the Amazon S3 console to perform these operations. The console uses the Amazon S3 APIs to send requests to Amazon S3.

This section describes how to work with buckets. For information about working with objects, see Amazon S3 objects overview.

Amazon S3 supports global buckets, which means that each bucket name must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US)).

After a bucket is created, the name of that bucket cannot be used by another AWS account in the same partition until the bucket is deleted. You should not depend on specific bucket naming conventions for availability or security verification purposes. For bucket naming guidelines, see Bucket naming rules.

Amazon S3 creates buckets in a Region that you specify. To reduce latency, minimize costs, or address regulatory requirements, choose any AWS Region that is geographically close to you. For example, if you reside in Europe, you might find it advantageous to create buckets in the Europe (Ireland) or Europe (Frankfurt) Regions. For a list of Amazon S3 Regions, see Regions and Endpoints in the *AWS General Reference*.

**Note**

For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Note**

Objects that belong to a bucket that you create in a specific AWS Region never leave that Region, unless you explicitly transfer them to another Region. For example, objects that are stored in the Europe (Ireland) Region never leave it.

**Topics**



## About permissions

You can use your AWS account root user credentials to create a bucket and perform any other Amazon S3 operation. However, we recommend that you do not use the root user credentials of your AWS account to make requests, such as to create a bucket. Instead, create an AWS Identity and Access Management (IAM) user, and grant that user full access (users by default have no permissions).

These users are referred to as *administrators*. You can use the administrator user credentials, instead of the root user credentials of your account, to interact with AWS and perform tasks, such as create a bucket, create users, and grant them permissions.

For more information, see AWS account root user credentials and IAM user credentials in the *AWS General Reference* and Security best practices in IAM in the *IAM User Guide*.

The AWS account that creates a resource owns that resource. For example, if you create an IAM user in your AWS account and grant the user permission to create a bucket, the user can create a bucket. But the user does not own the bucket; the AWS account that the user belongs to owns the bucket. The user needs additional permission from the resource owner to perform any other bucket operations. For more information about managing permissions for your Amazon S3 resources, see Identity and Access Management for Amazon S3.

## Managing public access to buckets

Public access is granted to buckets and objects through bucket policies, access control lists (ACLs), or both. To help you manage public access to Amazon S3 resources, Amazon S3 provides settings to block public access. Amazon S3 Block Public Access settings can override ACLs and bucket policies so that you can enforce uniform limits on public access to these resources. You can apply Block Public Access settings to individual buckets or to all buckets in your account.

To ensure that all of your Amazon S3 buckets and objects have their public access blocked, all four settings for Block Public Access are enabled by default when you create a new bucket. We recommend that you turn on all four settings for Block Public Access for your account too. These settings block all public access for all current and future buckets.

Before applying these settings, verify that your applications will work correctly without public access. If you require some level of public access to your buckets or objects—for example, to host a static website, as described at Hosting a static website using Amazon S3—you can customize the individual settings to suit your storage use cases. For more information, see Blocking public access to your Amazon S3 storage.

However, we highly recommend keeping Block Public Access enabled. If you want to keep all four Block Public Access settings enabled and host a static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Amazon S3 static websites support only HTTP endpoints. Amazon CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS. HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks.

For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

**Note**

If you see an `Error` when you list your buckets and their public access settings, you might not have the required permissions. Make sure that you have the following permissions added to your user or role policy:

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

In some rare cases, requests can also fail because of an AWS Region outage.

## Bucket configuration options

Amazon S3 supports various options for you to configure your bucket. For example, you can configure your bucket for website hosting, add a configuration to manage the lifecycle of objects in the bucket, and configure the bucket to log all access to the bucket. Amazon S3 supports subresources for you to store and manage the bucket configuration information. You can use the Amazon S3 API to create and manage these subresources. However, you can also use the console or the AWS SDKs.

**Note**

There are also object-level configurations. For example, you can configure object-level permissions by configuring an access control list (ACL) specific to that object.

These are referred to as subresources because they exist in the context of a specific bucket or object. The following table lists subresources that enable you to manage bucket-specific configurations.

| Subresource | Description |
| --- | --- |
| *cors* (cross-origin resource sharing) | You can configure your bucket to allow cross-origin requests.<br><br>For more information, see Using cross-origin resource sharing (CORS). |
| *event notification* | You can enable your bucket to send you notifications of specified bucket events.<br><br>For more information, see Amazon S3 Event Notifications. |
| *lifecycle* | You can define lifecycle rules for objects in your bucket that have a well-defined lifecycle. For example, you can define a rule to archive objects one year after creation, or delete an object 10 years after creation.<br><br>For more information, see Managing your storage lifecycle. |
| *location* | When you create a bucket, you specify the AWS Region where you want Amazon S3 to create the bucket. Amazon S3 stores this information in the location subresource and provides an API for you to retrieve this infor mation. |
| *logging* | Logging enables you to track requests for access to your bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. Access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill.<br><br>For more information, see Logging requests with server access logging. |
| *object locking* | To use S3 Object Lock, you must enable it for a bucket. You can also optionally configure a default retention mode and period that applies to new objects that are placed in the bucket.<br><br>For more information, see Using S3 Object Lock. |
| *policy* and *ACL* (access control list) | All your resources (such as buckets and objects) are private by default . Amazon S3 supports both bucket policy and access control list (ACL) |

| Subresource | Description |
| --- | --- |
| | options for you to grant and manage bucket-level permissions. Amazon S3 stores the permission information in the *policy* and *acl* subresources.<br><br>For more information, see Identity and Access Management for Amazon S3. |
| *replication* | Replication is the automatic, asynchronous copying of objects across buckets in different or the same AWS Regions. For more information, see Replicating objects overview. |
| *requestPayment* | By default, the AWS account that creates the bucket (the bucket owner) pays for downloads from the bucket. Using this subresource, the bucket owner can specify that the person requesting the download will be charged for the download. Amazon S3 provides an API for you to manage this subresource.<br><br>For more information, see Using Requester Pays buckets for storage transfers and usage. |
| *tagging* | You can add cost allocation tags to your bucket to categorize and track your AWS costs. Amazon S3 provides the *tagging* subresource to store and manage tags on a bucket. Using tags you apply to your bucket, AWS generates a cost allocation report with usage and costs aggregated by your tags.<br><br>For more information, see Billing and usage reporting for Amazon S3. |
| *transfer acceleration* | Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of the globally distributed edge locations of Amazon CloudFront.<br><br>For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration. |

| Subresource | Description |
| --- | --- |
| *versioning* | Versioning helps you recover accidental overwrites and deletes.<br><br>We recommend versioning as a best practice to recover objects from being deleted or overwritten by mistake.<br><br>For more information, see Using versioning in S3 buckets. |
| *website* | You can configure your bucket for static website hosting. Amazon S3 stores this configuration by creating a *website* subresource.<br><br>For more information, see Hosting a static website using Amazon S3. |

# Bucket naming rules

The following rules apply for naming general purpose buckets and directory buckets in Amazon S3:

**Topics**

- General purpose buckets naming rules
- Directory bucket naming rules

## General purpose buckets naming rules

The following naming rules apply for general purpose buckets.

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Bucket names must not contain two adjacent periods.
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `sthree-configurator`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.

- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.
- Bucket names must not end with the suffix `--x-s3`. This suffix is reserved for directory buckets. For more information, see Directory bucket naming rules.
- Bucket names must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US)).
- A bucket name cannot be used by another AWS account in the same partition until the bucket is deleted.
- Buckets used with Amazon S3 Transfer Acceleration can't have dots (.) in their names. For more information about Transfer Acceleration, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

For best compatibility, we recommend that you avoid using dots (.) in bucket names, except for buckets that are used only for static website hosting. If you include dots in a bucket's name, you can't use virtual-host-style addressing over HTTPS, unless you perform your own certificate validation. This is because the security certificates used for virtual hosting of buckets don't work for buckets with dots in their names.

This limitation doesn't affect buckets used for static website hosting, because static website hosting is only available over HTTP. For more information about virtual-host-style addressing, see Virtual hosting of buckets. For more information about static website hosting, see Hosting a static website using Amazon S3.

**Note**

Before March 1, 2018, buckets created in the US East (N. Virginia) Region could have names that were up to 255 characters long and included uppercase letters and underscores. Beginning March 1, 2018, new buckets in US East (N. Virginia) must conform to the same rules applied in all other Regions.

For information on object key names, see Creating object key names.

### Example general purpose bucket names

The following example bucket names are valid and follow the recommended naming guidelines for general purpose buckets:

- `docexamplebucket1`
- `log-delivery-march-2020`
- `my-hosted-content`

The following example bucket names are valid but not recommended for uses other than static website hosting:

- `docexamplewebsite.com`
- `www.docexamplewebsite.com`
- `my.example.s3.bucket`

The following example bucket names are *not* valid:

- `doc_example_bucket` (contains underscores)
- `DocExampleBucket` (contains uppercase letters)
- `doc-example-bucket-` (ends with a hyphen)

## Directory bucket naming rules

The following naming rules apply for directory buckets.

- Be unique within the chosen AWS Region and Availability Zone.
- Name must be between 3 (min) and 63 (max) characters long, including the suffix.
- Consists only of lowercase letters, numbers and hyphens (-).
- Begin and end with a letter or number.
- Must include the following suffix: `--`*`azid`*`--x-s3`.
- Bucket names must not start with the prefix `xn--`.

- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `sthree-configurator`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

> **Note**
>
> When you create a directory bucket by using the console a suffix is automatically added to the base name that you provide. This suffix includes the Availability Zone ID of the Availability Zone that you chose.
> When you create a directory bucket by using an API you must provide the full suffix, including the Availability Zone ID, in your request. For a list of Availability Zone IDs, see S3 Express One Zone Availability Zones and Regions.

## Accessing and listing an Amazon S3 bucket

To list and access your Amazon S3 buckets, you can use various tools. Review the following tools to determine which approach fits your use case:

- **Amazon S3 console:** With the Amazon S3 console, you can easily access a bucket and modify the bucket's properties. You can also perform most bucket operations by using the console UI, without having to write any code.
- **AWS CLI**: If you need to access multiple buckets, you can save time by using the AWS Command Line Interface (AWS CLI) to automate common and repetitive tasks. Scriptability and repeatability for common actions are frequent considerations as organizations scale. For more information, see Developing with Amazon S3 using the AWS CLI.
- **Amazon S3 REST API:** You can use the Amazon S3 REST API to write your own programs and access buckets programmatically. Amazon S3 supports an API architecture in which your buckets

and objects are resources, each with a resource URI that uniquely identifies the resource. For more information, see Developing with Amazon S3 using the REST API.

Depending on the use case for your Amazon S3 bucket, there are different recommended methods to access the underlying data in your buckets. The following list includes common use cases for accessing your data.

- **Static websites** – You can use Amazon S3 to host a static website. In this use case, you can configure your S3 bucket to function like a website. For an example that walks you through the steps of hosting a website on Amazon S3, see Tutorial: Configuring a static website on Amazon S3.

  To host a static website with security settings like Block Public Access enabled, we recommend using Amazon CloudFront with Origin Access Control (OAC) and implementing additional security headers, such as HTTPS. For more information, see Getting started with a secure static website.

  > **Note**
  >
  > Amazon S3 supports both virtual-hosted–style and path-style URLs for static website access. Because buckets can be accessed using path-style and virtual-hosted–style URLs, we recommend that you create buckets with DNS-compliant bucket names. For more information, see Bucket quotas, restrictions, and limitations.

- **Shared datasets** – As you scale on Amazon S3, it's common to adopt a multi-tenant model, where you assign different end customers or business units to unique prefixes within a shared bucket. By using Amazon S3 access points, you can divide one large bucket policy into separate, discrete access point policies for each application that needs to access the shared dataset. This approach makes it simpler to focus on building the right access policy for an application without disrupting what any other application is doing within the shared dataset. For more information, see Managing data access with Amazon S3 access points.
- **High-throughput workloads** – Mountpoint for Amazon S3 is a high-throughput open source file client for mounting an Amazon S3 bucket as a local file system. With Mountpoint, your applications can access objects stored in Amazon S3 through file-system operations, such as open and read. Mountpoint automatically translates these operations into S3 object API calls, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface. For more information, see Working with Mountpoint for Amazon S3.

- **Multi-Region applications** – Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the public internet, Multi-Region Access Points provide built-in network resilience with acceleration of internet-based requests to Amazon S3. For more information, see Multi-Region Access Points in Amazon S3.
- **Building new applications** – You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying Amazon S3 REST API. To build connected mobile and web applications, you can use the AWS Mobile SDKs and the AWS Amplify JavaScript library. For more information, see Developing with Amazon S3 using the AWS SDKs.
- **Secure Shell (SSH) File Transfer Protocol (SFTP)** – If you're trying to securely transfer sensitive data over the internet, you can use an SFTP-enabled server with your Amazon S3 bucket. AWS SFTP is a network protocol that supports the full security and authentication functionality of SSH. With this protocol, you have fine-grained control over user identity, permissions, and keys or you can use IAM policies to manage access. To associate an SFTP enabled server with your Amazon S3 bucket, make sure to create your SFTP-enabled server first. Then, you set up user accounts, and associate the server with an Amazon S3 bucket. For a walkthrough of this process, see AWS Transfer for SFTP – Fully Managed SFTP Service for Amazon S3 in *AWS Blogs*.

## Listing a bucket

To list all of your buckets, you must have the `s3:ListAllMyBuckets` permission. To access a bucket, make sure to also obtain the required AWS Identity and Access Management (IAM) permissions to list the contents of the specified bucket. For an example bucket policy that grants access to an S3 bucket, see Allowing an IAM user access to one of your buckets. If you're encountering an HTTP Access Denied (403 Forbidden) error, see Bucket policies and IAM policies.

You can list your bucket by using the Amazon S3 console, the AWS CLI, or the AWS SDKs.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **General purpose buckets** list, choose the bucket that you want to view.

> **Note**
>
> The **General purpose buckets** list includes buckets that are located in all AWS Regions.

#### Using the AWS CLI

To use the AWS CLI to access an S3 bucket or generate a listing of S3 buckets, use the `ls` command. When you list all of the objects in your bucket, note that you must have the `s3:ListBucket` permission.

To use this example command, replace *DOC-EXAMPLE-BUCKET1* with the name of your bucket.

```
$ aws s3 ls s3://DOC-EXAMPLE-BUCKET1
```

The following example command lists all the Amazon S3 buckets in your account:

```
$ aws s3 ls
```

For more information and examples, see List bucket and objects.

#### Using the AWS SDKs

You can also access an Amazon S3 bucket by using the ListBuckets API operation. For examples of how to use this operation with different AWS SDKs, see Use `ListBuckets` with an AWS SDK or CLI.

## Creating a bucket

To upload your data to Amazon S3, you must first create an Amazon S3 bucket in one of the AWS Regions. When you create a bucket, you must choose a bucket name and Region. You can optionally choose other storage management options for the bucket. After you create a bucket, you cannot change the bucket name or Region. For information about naming buckets, see Bucket naming rules.

The AWS account that creates the bucket owns it. You can upload any number of objects to the bucket. By default, you can create up to 100 buckets in each of your AWS accounts. If you need more buckets, you can increase your account bucket limit to a maximum of 1,000 buckets by submitting a service limit increase. To learn how to submit a bucket limit increase, see AWS service quotas in the *AWS General Reference*. You can store any number of objects in a bucket.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use both to control ownership of objects that are uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using policies.

For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Server-side encryption with Amazon S3 managed keys (SSE-S3) is the base level of encryption configuration for every bucket in Amazon S3. All new objects uploaded to an S3 bucket are automatically encrypted with SSE-S3 as the base level of encryption setting. If you want to use a different type of default encryption, you can also specify server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) or customer-provided keys (SSE-C) to encrypt your data. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

You can use the Amazon S3 console, Amazon S3 APIs, AWS CLI, or AWS SDKs to create a bucket. For more information about the permissions required to create a bucket, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

## Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   > **Note**
   >
   > To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.

5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **General purpose**.
7. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, dots (.), and hyphens (-). For best compatibility, we recommend that you avoid using dots (.) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.

   After you create the bucket, you cannot change its name. For more information about naming buckets, see Bucket naming rules.

   > **Important**
   >
   > Avoid including sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

8. AWS Management Console allows you to copy an existing bucket's settings to your new bucket. If you do not want to copy the settings of an existing bucket, skip to the next step.

   > **Note**
   >
   > This option:
   >
   > - Is not available in the AWS CLI and is only available in console
   > - Is not available for directory buckets
   > - Does not copy the bucket policy from the existing bucket to the new bucket

   To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you would

like to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window re-opens.

Under **Copy settings from existing bucket**, you will now see the name of the bucket you selected. You will also see a **Restore defaults** option that you can use to remove the copied bucket settings. Review the remaining bucket settings, on the **Create bucket** page. You will see that they now match the settings of the bucket that you selected. You can skip to the final step.

9. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   **ACLs disabled**

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

     By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

   **ACLs enabled**

   - **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

     If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.
   - **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

10. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

    By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

    > **Note**
    >
    > To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

11. (Optional) Under **Bucket Versioning**, you can choose if you wish to keep variants of objects in your bucket. For more information about versioning, see Using versioning in S3 buckets.

    To disable or enable versioning on your bucket, choose either **Disable** or **Enable**.

12. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs used to categorize storage.

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

13. Under **Default encryption**, choose **Edit**.

14. To configure default encryption, under **Encryption type**, choose one of the following:

    - **Amazon S3 managed key (SSE-S3)**
    - **AWS Key Management Service key (SSE-KMS)**

> **Important**
>
> If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

Buckets and new objects are encrypted with server-side encryption with an **Amazon S3 managed key** as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

For more information about using Amazon S3 server-side encryption to encrypt your data, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

15. If you chose **AWS Key Management Service key (SSE-KMS)**, do the following:

    a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

      Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
    - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
    - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

      For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same

Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN. For more information on cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information on SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS).
When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS).

b. When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

   To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

16. (Optional) If you want to enable S3 Object Lock, do the following:

    a. Choose **Advanced settings**.

> **Important**
>
> Enabling Object Lock also enables versioning for the bucket. After enabling you must configure the Object Lock default retention and legal hold settings to protect new objects from being deleted or overwritten.

    b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

For more information, see Using S3 Object Lock.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning` and `s3:PutBucketObjectLockConfiguration`.

17. Choose **Create bucket**.

## Using the AWS SDKs

When you use the AWS SDKs to create a bucket, you must create a client and then use the client to send a request to create a bucket. As a best practice, you should create your client and bucket in the same AWS Region. If you don't specify a Region when you create a client or a bucket, Amazon S3 uses the default Region, US East (N. Virginia). If you want to constrain the bucket creation to a specific AWS Region, use the `LocationConstraint` condition key.

To create a client to access a dual-stack endpoint, you must specify an AWS Region. For more information, see Dual-stack endpoints. For a list of available AWS Regions, see Regions and endpoints in the *AWS General Reference*.

When you create a client, the Region maps to the Region-specific endpoint. The client uses this endpoint to communicate with Amazon S3: `s3.region.amazonaws.com`. If your Region launched after March 20, 2019, your client and bucket must be in the same Region. However, you can use a client in the US East (N. Virginia) Region to create a bucket in any Region that launched before March 20, 2019. For more information, see Legacy endpoints.

These AWS SDK code examples perform the following tasks:

- **Create a client by explicitly specifying an AWS Region** – In the example, the client uses the `s3.us-west-2.amazonaws.com` endpoint to communicate with Amazon S3. You can specify any AWS Region. For a list of AWS Regions, see Regions and endpoints in the *AWS General Reference*.
- **Send a create bucket request by specifying only a bucket name** – The client sends a request to Amazon S3 to create the bucket in the Region where you created a client.
- **Retrieve information about the location of the bucket** – Amazon S3 stores bucket location information in the *location* subresource that is associated with the bucket.

Java

This example shows how to create an Amazon S3 bucket using the AWS SDK for Java. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

import java.io.IOException;

public class CreateBucket2 {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region,
 the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));

                // Verify that the bucket was created by retrieving it and checking
 its
                // location.
                String bucketLocation = s3Client.getBucketLocation(new
 GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

For information about how to create and test a working sample, see AWS SDK for .NET Version 3 API Reference.

**Example**

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
```

```
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client,
 bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
 s3Client.PutBucketAsync(putBucketRequest);
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
 client.GetBucketLocationAsync(request);
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby

For information about how to create and test a working sample, see AWS SDK for Ruby - Version 3.

**Example**

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
```

```
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

### Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) to create an S3 bucket. For more information, see create-bucket in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

## Viewing the properties for an S3 bucket

You can view properties for any Amazon S3 bucket you own. These settings include the following:

- **Bucket Versioning** – Keep multiple versions of an object in one bucket by using versioning. By default, versioning is disabled for a new bucket. For information about enabling versioning, see Enabling versioning on buckets.
- **Tags** – With AWS cost allocation, you can use bucket tags to annotate billing for your use of a bucket. A tag is a key-value pair that represents a label that you assign to a bucket. For more information, see Using cost allocation S3 bucket tags.
- **Default encryption** – Enabling default encryption provides you with automatic server-side encryption. Amazon S3 encrypts an object before saving it to a disk and decrypts the object when you download it. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

- **Server access logging** – Get detailed records for the requests that are made to your bucket with server access logging. By default, Amazon S3 doesn't collect server access logs. For information about enabling server access logging, see Enabling Amazon S3 server access logging.
- **AWS CloudTrail data events** – Use CloudTrail to log data events. By default, trails don't log data events. Additional charges apply for data events. For more information, see Logging Data Events for Trails in the *AWS CloudTrail User Guide*.
- **Event notifications** – Enable certain Amazon S3 bucket events to send notification messages to a destination whenever the events occur. For more information, see Enabling and configuring event notifications using the Amazon S3 console.
- **Transfer acceleration** – Enable fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. For information about enabling transfer acceleration, see Enabling and using S3 Transfer Acceleration.
- **Object Lock** – Use S3 Object Lock to prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. For more information, see Using S3 Object Lock.
- **Requester Pays** – Enable Requester Pays if you want the requester (instead of the bucket owner) to pay for requests and data transfers. For more information, see Using Requester Pays buckets for storage transfers and usage.
- **Static website hosting** – You can host a static website on Amazon S3. For more information, see Hosting a static website using Amazon S3.

You can view bucket properties using the AWS Management Console, AWS CLI, or AWS SDKs

## Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to view the properties for.
3. Choose the **Properties** tab.
4. On the **Properties** page, you can configure the above properties for the bucket.

## Using the AWS CLI

### View bucket properties with the AWS CLI

The following commands show how you can use the AWS CLI to list different bucket properties.

The following returns the tag set associated with the bucket *amzn-s3-demo-bucket1*. For more information about bucket tags see, Using cost allocation S3 bucket tags.

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-tagging in the *AWS CLI Command Reference*.

The following returns the versioning state of the bucket *amzn-s3-demo-bucket1*. For information about the bucket versioning, see Using versioning in S3 buckets.

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-versioning in the *AWS CLI Command Reference*.

The following returns the default encryption configuration for the bucket *amzn-s3-demo-bucket1*. By default, all buckets have a default encryption configuration that uses server-side encryption with Amazon S3 managed keys (SSE-S3). For information about the bucket default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-encryption in the *AWS CLI Command Reference*.

The following returns the notification configuration of the bucket *amzn-s3-demo-bucket1*. For information about the bucket event notifications, see Amazon S3 Event Notifications.

```
aws s3api get-bucket-notification-configuration --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-notification-configuration in the *AWS CLI Command Reference*.

The following returns the logging status for the bucket *amzn-s3-demo-bucket1*. For information about the bucket logging, see Logging requests with server access logging.

```
aws s3api get-bucket-logging --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-logging in the *AWS CLI Command Reference*.

### Using the AWS SDKs

For examples of how to return bucket properties with the AWS SDKs, such as versioning, tags, and more, see Actions for Amazon S3 using AWS SDKs.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

## Emptying a bucket

You can empty a bucket's contents using the Amazon S3 console, AWS SDKs, or AWS Command Line Interface (AWS CLI). When you empty a bucket, you delete all the objects, but you keep the bucket. After you empty a bucket, it cannot be undone. Objects added to the bucket while the empty bucket action is in progress might be deleted. All objects (including all object versions and delete markers) in the bucket must be deleted before the bucket itself can be deleted.

When you empty a bucket that has S3 Versioning enabled or suspended, all versions of all the objects in the bucket are deleted. For more information, see Working with objects in a versioning-enabled bucket.

You can also specify a lifecycle configuration on a bucket to expire objects so that Amazon S3 can delete them. For more information, see Setting a lifecycle configuration on a bucket. To empty a large bucket, we recommend that you use an S3 Lifecycle configuration rule. Lifecycle expiration is an asynchronous process, so the rule might take some days to run before the bucket is empty. After the first time that Amazon S3 runs the rule, all objects that are eligible for expiration are marked for deletion. You're no longer charged for those objects that are marked for deletion. For more information, see How do I empty an Amazon S3 bucket using a lifecycle configuration rule?.

### Using the S3 console

You can use the Amazon S3 console to empty a bucket, which deletes all of the objects in the bucket without deleting the bucket.

#### To empty an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, select the option next to the name of the bucket that you want to empty, and then choose **Empty**.

3. On the **Empty bucket** page, confirm that you want to empty the bucket by entering the bucket name into the text field, and then choose **Empty**.
4. Monitor the progress of the bucket emptying process on the **Empty bucket: Status** page.

## Using the AWS CLI

You can empty a bucket using the AWS CLI only if the bucket does not have Bucket Versioning enabled. If versioning is not enabled, you can use the `rm` (remove) AWS CLI command with the `--recursive` parameter to empty the bucket (or remove a subset of objects with a specific key name prefix).

The following `rm` command removes objects that have the key name prefix `doc`, for example, `doc/doc1` and `doc/doc2`.

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

Use the following command to remove all objects without specifying a prefix.

```
$ aws s3 rm s3://bucket-name --recursive
```

For more information, see Using high-level S3 commands with the AWS CLI in the *AWS Command Line Interface User Guide*.

> **Note**
>
> You can't remove objects from a bucket that has versioning enabled. Amazon S3 adds a delete marker when you delete an object, which is what this command does. For more information about S3 Bucket Versioning, see Using versioning in S3 buckets.

## Using the AWS SDKs

You can use the AWS SDKs to empty a bucket or remove a subset of objects that have a specific key name prefix.

For an example of how to empty a bucket using AWS SDK for Java, see Deleting a bucket. The code deletes all objects, regardless of whether the bucket has versioning enabled, and then it deletes

the bucket. To just empty the bucket, make sure that you remove the statement that deletes the bucket.

For more information about using other AWS SDKs, see Tools for Amazon Web Services.

### Using a lifecycle configuration

To empty a large bucket, we recommend that you use an S3 Lifecycle configuration rule. Lifecycle expiration is an asynchronous process, so the rule might take some days to run before the bucket is empty. After the first time that Amazon S3 runs the rule, all objects that are eligible for expiration are marked for deletion. You're no longer charged for those objects that are marked for deletion. For more information, see How do I empty an Amazon S3 bucket using a lifecycle configuration rule?.

If you use a lifecycle configuration to empty your bucket, the configuration should include current versions, non-current versions, delete markers, and incomplete multipart uploads.

You can add lifecycle configuration rules to expire all objects or a subset of objects that have a specific key name prefix. For example, to remove all objects in a bucket, you can set a lifecycle rule to expire objects one day after creation.

Amazon S3 supports a bucket lifecycle rule that you can use to stop multipart uploads that don't complete within a specified number of days after being initiated. We recommend that you configure this lifecycle rule to minimize your storage costs. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

For more information about using a lifecycle configuration to empty a bucket, see Setting a lifecycle configuration on a bucket and Expiring objects.

## Emptying a bucket with AWS CloudTrail configured

AWS CloudTrail tracks object-level data events in an Amazon S3 bucket, such as deleting objects. If you use a bucket as a destination to log your CloudTrail events and are deleting objects from that same bucket you may be creating new objects while emptying your bucket. To prevent this, stop your AWS CloudTrail trails. For more information about stopping your CloudTrail trails from logging events, see Turning off logging for a trail in the *AWS CloudTrail User Guide*.

Another alternative to stopping CloudTrail trails from being added to the bucket is to add a deny `s3:PutObject` statement to your bucket policy. If you want to store new objects in the bucket at a

later time you will need to remove this deny `s3:PutObject` statement. For more information, see Object operations and IAM JSON policy elements: Effect in the *IAM User Guide*.

# Deleting a bucket

You can delete an empty Amazon S3 bucket. Before deleting a bucket, consider the following:

- Bucket names are unique. If you delete a bucket, another AWS user can use the name.
- If the bucket hosts a static website, and you created and configured an Amazon Route 53 hosted zone as described in Tutorial: Configuring a static website using a custom domain registered with Route 53, you must clean up the Route 53 hosted zone settings that are related to the bucket. For more information, see Step 2: Delete the Route 53 hosted zone.
- If the bucket receives log data from Elastic Load Balancing (ELB): We recommend that you stop the delivery of ELB logs to the bucket before deleting it. After you delete the bucket, if another user creates a bucket using the same name, your log data could potentially be delivered to that bucket. For information about ELB access logs, see Access logs in the *User Guide for Classic Load Balancers* and Access logs in the *User Guide for Application Load Balancers*.

### Troubleshooting

If you are unable to delete an Amazon S3 bucket, consider the following:

- **Make sure the bucket is empty** – You can only delete buckets that don't have any objects in them. Make sure the bucket is empty.
- **Make sure there aren't any access points attached** – You can only delete buckets that don't have any access points attached to them. Delete any access points that are attached to the bucket, before deleting the bucket.
- **AWS Organizations service control policies (SCPs)** – A service control policy can deny the delete permission on a bucket. For information about SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **s3:DeleteBucket permissions** – If you cannot delete a bucket, work with your IAM administrator to confirm that you have `s3:DeleteBucket` permissions. For information about how to view or update IAM permissions, see Changing permissions for an IAM user in the *IAM User Guide*.
- **s3:DeleteBucket deny statement** – If you have `s3:DeleteBucket` permissions in your IAM policy and you cannot delete a bucket, the bucket policy might include a deny statement for `s3:DeleteBucket`. Buckets created by ElasticBeanstalk have a policy containing this statement

by default. Before you can delete the bucket, you must delete this statement or the bucket policy.

> **Important**
>
> Bucket names are unique. If you delete a bucket, another AWS user can use the name. If you want to continue to use the same bucket name, don't delete the bucket. We recommend that you empty the bucket and keep it.

## Using the S3 console

### To delete an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, select the option next to the name of the bucket that you want to delete, and then choose **Delete** at the top of the page.
3. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name into the text field, and then choose **Delete bucket**.

   > **Note**
   >
   > If the bucket contains any objects, empty the bucket before deleting it by selecting the *empty bucket configuration* link in the **This bucket is not empty** error alert and following the instructions on the **Empty bucket** page. Then return to the **Delete bucket** page and delete the bucket.

4. To verify that you've deleted the bucket, open the **Buckets** list and enter the name of the bucket that you deleted. If the bucket can't be found, your deletion was successful.

## Using the AWS SDK for Java

The following example shows you how to delete a bucket using the AWS SDK for Java. First, the code deletes objects in the bucket and then it deletes the bucket. For information about other AWS SDKs, see Tools for Amazon Web Services.

Java

The following Java example deletes a bucket that contains objects. The example deletes all objects, and then it deletes the bucket. The example works for buckets with or without versioning enabled.

> **Note**
>
> For buckets without versioning enabled, you can delete all objects directly and then delete the bucket. For buckets with versioning enabled, you must delete all object versions before deleting the bucket.

For instructions on creating and testing a working sample, ssee Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;

public class DeleteBucket2 {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(clientRegion)
                    .build();

            // Delete all objects from the bucket. This is sufficient
            // for unversioned buckets. For versioned buckets, when you attempt to
delete
            // objects, Amazon S3 inserts
            // delete markers for all objects, but doesn't delete the object
versions.
            // To delete objects from versioned buckets, delete all of the object
versions
            // before deleting
            // the bucket (see below for an example).
            ObjectListing objectListing = s3Client.listObjects(bucketName);
            while (true) {
                Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
                while (objIter.hasNext()) {
                    s3Client.deleteObject(bucketName, objIter.next().getKey());
                }

                // If the bucket contains many objects, the listObjects() call
                // might not return all of the objects in the first listing. Check
to
                // see whether the listing was truncated. If so, retrieve the next
page of
                // objects
                // and delete them.
                if (objectListing.isTruncated()) {
                    objectListing = s3Client.listNextBatchOfObjects(objectListing);
                } else {
                    break;
                }
            }

            // Delete all object versions (required for versioned buckets).
            VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
            while (true) {
                Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
                while (versionIter.hasNext()) {
                    S3VersionSummary vs = versionIter.next();
```

```
                    s3Client.deleteVersion(bucketName, vs.getKey(),
 vs.getVersionId());
                }

                if (versionList.isTruncated()) {
                    versionList = s3Client.listNextBatchOfVersions(versionList);
                } else {
                    break;
                }
            }

            // After all objects and object versions are deleted, delete the bucket.
            s3Client.deleteBucket(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
 couldn't
            // parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Using the AWS CLI

You can delete a bucket that contains objects with the AWS CLI if it doesn't have versioning enabled. When you delete a bucket that contains objects, all the objects in the bucket are permanently deleted, including objects that are transitioned to the `S3 Glacier` storage class.

If your bucket does not have versioning enabled, you can use the `rb` (remove bucket) AWS CLI command with the `--force` parameter to delete the bucket and all the objects in it. This command deletes all objects first and then deletes the bucket.

If versioning is enabled versioned objects will not be deleted in this process which would cause the bucket deletion to fail because the bucket would not be empty. For more information about deleting versioned objects, see Deleting object versions.

```
$ aws s3 rb s3://bucket-name --force
```

For more information, see Using High-Level S3 Commands with the AWS Command Line Interface in the AWS Command Line Interface User Guide.

# Setting default server-side encryption behavior for Amazon S3 buckets

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

All Amazon S3 buckets have encryption configured by default, and objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption setting applies to all objects in your Amazon S3 buckets.

If you need more control over your keys, such as managing key rotation and access policy grants, you can choose to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information about editing KMS keys, see Editing keys in *AWS Key Management Service Developer Guide*.

**Note**

We've changed buckets to encrypt new object uploads automatically. If you previously created a bucket without default encryption, Amazon S3 will enable encryption by default for the bucket using SSE-S3. There will be no changes to the default encryption configuration for an existing bucket that already has SSE-S3 or SSE-KMS configured. If you

want to encrypt your objects with SSE-KMS, you must change the encryption type in your bucket settings. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).

When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys to decrease request traffic from Amazon S3 to AWS KMS and reduce the cost of encryption. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To identify buckets that have SSE-KMS enabled for default encryption, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to protect your data.

When you use server-side encryption, Amazon S3 encrypts an object before saving it to disk and decrypts it when you download the object. For more information about protecting data using server-side encryption and encryption-key management, see Protecting data with server-side encryption.

For more information about the permissions required for default encryption, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

You can configure the Amazon S3 default encryption behavior for an S3 bucket by using the Amazon S3 console, the AWS SDKs, the Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI).

**Encrypting existing objects**

To encrypt your existing unencrypted Amazon S3 objects, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on, and Batch Operations calls the respective API to perform the specified operation. You can use the Batch Operations Copy operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing large-scale batch operations on Amazon S3 objects and the *AWS Storage Blog* post Encrypting objects with Amazon S3 Batch Operations.

You can also encrypt existing objects by using the `CopyObject` API operation or the `copy-object` AWS CLI command. For more information, see the *AWS Storage Blog* post Encrypting existing Amazon S3 objects with the AWS CLI.

> **Note**
>
> Amazon S3 buckets with default bucket encryption set to SSE-KMS cannot be used as destination buckets for the section called "Logging server access". Only SSE-S3 default encryption is supported for server access log destination buckets.

## Using SSE-KMS encryption for cross-account operations

When using encryption for cross-account operations, be aware of the following:

- If an AWS KMS key Amazon Resource Name (ARN) or alias is not provided at request time or through the bucket's default encryption configuration, the AWS managed key (`aws/s3`) is used.
- If you're uploading or accessing S3 objects by using AWS Identity and Access Management (IAM) principals that are in the same AWS account as your KMS key, you can use the AWS managed key (`aws/s3`).
- If you want to grant cross-account access to your S3 objects, use a customer managed key. You can configure the policy of a customer managed key to allow access from another account.
- If you're specifying a customer managed KMS key, we recommend using a fully qualified KMS key ARN. If you use a KMS key alias instead, AWS KMS resolves the key within the requester's account. This behavior can result in data that's encrypted with a KMS key that belongs to the requester, and not the bucket owner.
- You must specify a key that you (the requester) have been granted `Encrypt` permission to. For more information, see Allow key users to use a KMS key for cryptographic operations in the *AWS Key Management Service Developer Guide*.

For more information about when to use customer managed keys and AWS managed KMS keys, see Should I use an AWS managed key or a customer managed key to encrypt my objects in Amazon S3?

## Using default encryption with replication

When you enable default encryption for a replication destination bucket, the following encryption behavior applies:

- If objects in the source bucket are not encrypted, the replica objects in the destination bucket are encrypted by using the default encryption settings of the destination bucket. As a result, the

entity tags (ETags) of the source objects differ from the ETags of the replica objects. If you have applications that use ETags, you must update those applications to account for this difference.

- If objects in the source bucket are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), the replica objects in the destination bucket use the same type of encryption as the source objects. The default encryption settings of the destination bucket are not used.

For more information about using default encryption with SSE-KMS, see Replicating encrypted objects.

## Using Amazon S3 Bucket Keys with default encryption

When you configure your bucket to use SSE-KMS as the default encryption behavior for new objects, you can also configure S3 Bucket Keys. S3 Bucket Keys decrease the number of transactions from Amazon S3 to AWS KMS to reduce the cost of SSE-KMS.

When you configure your bucket to use S3 Bucket Keys for SSE-KMS on new objects, AWS KMS generates a bucket-level key that is used to create a unique data key for objects in the bucket. This S3 Bucket Key is used for a time-limited period within Amazon S3, reducing the need for Amazon S3 to make requests to AWS KMS to complete encryption operations.

For more information about using S3 Bucket Keys, see Using Amazon S3 Bucket Keys.

## Configuring default encryption

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

Amazon S3 buckets have bucket encryption enabled by default, and new objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption applies to all new objects in your Amazon S3 buckets, and comes at no cost to you.

If you need more control over your encryption keys, such as managing key rotation and access policy grants, you can elect to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). For more information about DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)".

If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

When you set default bucket encryption to SSE-KMS, you can also configure an S3 Bucket Key to reduce your AWS KMS request costs. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

> **Note**
>
> If you use PutBucketEncryption to set your default bucket encryption to SSE-KMS, you should verify that your KMS key ID is correct. Amazon S3 does not validate the KMS key ID provided in PutBucketEncryption requests.

There are no additional charges for using default encryption for S3 buckets. Requests to configure the default encryption behavior incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing. For SSE-KMS and DSSE-KMS, AWS KMS charges apply and are listed at AWS KMS pricing.

Server-side encryption with customer-provided keys (SSE-C) is not supported for default encryption.

You can configure Amazon S3 default encryption for an S3 bucket by using the Amazon S3 console, the AWS SDKs, the Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI).

**Changes to note before enabling default encryption**

After you enable default encryption for a bucket, the following encryption behavior applies:

- There is no change to the encryption of the objects that existed in the bucket before default encryption was enabled.
- When you upload objects after enabling default encryption:
  - If your PUT request headers don't include encryption information, Amazon S3 uses the bucket's default encryption settings to encrypt the objects.
  - If your PUT request headers include encryption information, Amazon S3 uses the encryption information from the PUT request to encrypt objects before storing them in Amazon S3.
- If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> Objects uploaded before default encryption was enabled will not be encrypted. For information about encrypting existing objects, see the section called "Setting default bucket encryption".

### Using the S3 console

### To configure default encryption on an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want.
4. Choose the **Properties** tab.
5. Under **Default encryption**, choose **Edit**.
6. To configure encryption, under **Encryption type**, choose one of the following:
   - **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
   - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
   - **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**

> **Important**
>
> If you use the SSE-KMS or DSSE-KMS options for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

Buckets and new objects are encrypted by default with SSE-S3, unless you specify another type of default encryption for your buckets. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

For more information about using Amazon S3 server-side encryption to encrypt your data, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

7. If you chose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**, do the following:

   a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

     For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can only use KMS keys that are enabled in the same AWS Region as the bucket. When you choose **Choose from your KMS keys**, the S3 console only lists 100 KMS keys per Region. If you have more than 100 KMS keys in the same Region, you can only see the first 100 KMS keys in the S3 console. To use a KMS key that is not listed in the console, choose **Enter AWS KMS key ARN**, and enter the KMS key ARN.
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 only supports symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about using SSE-KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS). For more information about using DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)".

b. When you configure your bucket to use default encryption with SSE-KMS, you can also enable an S3 Bucket Key. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

   To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

> **Note**
>
> S3 Bucket Keys aren't supported for DSSE-KMS.

8. Choose **Save changes**.

### Using the AWS CLI

These examples show you how to configure default encryption by using SSE-S3 or by using SSE-KMS with an S3 Bucket Key.

For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about using the AWS CLI to configure default encryption, see put-bucket-encryption.

**Example – Default encryption with SSE-S3**

This example configures default bucket encryption with Amazon S3 managed keys.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

**Example – Default encryption with SSE-KMS using an S3 Bucket Key**

This example configures default bucket encryption with SSE-KMS using an S3 Bucket Key.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

**Using the REST API**

Use the REST API `PutBucketEncryption` operation to enable default encryption and to set the type of server-side encryption to use—SSE-S3, SSE-KMS, or DSSE-KMS.

For more information, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

# Monitoring default encryption with AWS CloudTrail and Amazon EventBridge

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

You can track default encryption configuration requests for Amazon S3 buckets by using AWS CloudTrail events. The following API event names are used in CloudTrail logs:

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

You can also create EventBridge rules to match the CloudTrail events for these API calls. For more information about CloudTrail events, see Enable logging for objects in a bucket using the console. For more information about EventBridge events, see Events from AWS services.

You can use CloudTrail logs for object-level Amazon S3 actions to track PUT and POST requests to Amazon S3. You can use these actions to verify whether default encryption is being used to encrypt objects when incoming PUT requests don't have encryption headers.

When Amazon S3 encrypts an object by using the default encryption settings, the log includes one of the following fields as the name-value pair: `"SSEApplied":"Default_SSE_S3"`, `"SSEApplied":"Default_SSE_KMS"`, or `"SSEApplied":"Default_DSSE_KMS"`.

When Amazon S3 encrypts an object by using the PUT encryption headers, the log includes one of the following fields as the name-value pair: `"SSEApplied":"SSE_S3"`, `"SSEApplied":"SSE_KMS"`, `"SSEApplied":"DSSE_KMS"`, or `"SSEApplied":"SSE_C"`.

For multipart uploads, this information is included in your `InitiateMultipartUpload` API operation requests. For more information about using CloudTrail and CloudWatch, see Monitoring Amazon S3.

# Working with Mountpoint for Amazon S3

Mountpoint for Amazon S3 is a high-throughput open source file client for mounting an Amazon S3 bucket as a local file system. With Mountpoint, your applications can access objects stored in Amazon S3 through file system operations, such as open and read. Mountpoint automatically translates these operations into S3 object API calls, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface.

Mountpoint for Amazon S3 is generally available for production use on your large-scale read-heavy applications: data lakes, machine learning training, image rendering, autonomous vehicle simulation, extract, transform, and load (ETL), and more.

Mountpoint supports basic file system operations, and can read files up to 5 TB in size. It can list and read existing files, and it can create new ones. It cannot modify existing files or delete directories, and it does not support symbolic links or file locking. Mountpoint is ideal for applications that do not need all of the features of a shared file system and POSIX-style permissions but require Amazon S3's elastic throughput to read and write large S3 datasets. For details, see Mountpoint file system behavior on GitHub. For workloads that require full POSIX support, we recommend Amazon FSx for Lustre and its support for linking S3 buckets.

Mountpoint for Amazon S3 is available only for Linux operating systems. You can use Mountpoint to access S3 objects in all storage classes except S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access Tier, and S3 Intelligent-Tiering Deep Archive Access Tier.

**Topics**



## Installing Mountpoint

You can download and install prebuilt packages of Mountpoint for Amazon S3 by using the command line. The instructions for downloading and installing Mountpoint vary, depending on which Linux operating system that you're using.

**Topics**



## RPM-based distributions (Amazon Linux, Fedora, CentOS, RHEL)

1. Copy the following download URL for your architecture.

   *x86_64*:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

   *ARM64 (Graviton)*:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
   ```

2. Download the Mountpoint for Amazon S3 package. Replace *download-link* with the appropriate download URL from the preceding step.

   ```
   wget download-link
   ```

3. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the appropriate signature URL for your architecture.

   *x86_64*:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
   ```

   *ARM64 (Graviton)*:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
   ```

   Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

4. Install the package by using the following command:

```
sudo yum install ./mount-s3.rpm
```

5. Verify that Mountpoint is successfully installed by entering the following command:

```
mount-s3 --version
```

You should see output similar to the following:

```
mount-s3 1.3.1
```

## DEB-based distributions (Debian, Ubuntu)

1. Copy the download URL for your architecture.

*x86_64*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
```

*ARM64 (Graviton)*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
```

2. Download the Mountpoint for Amazon S3 package. Replace *download-link* with the appropriate download URL from the preceding step.

```
wget download-link
```

3. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the signature URL for your architecture.

*x86_64*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
```

*ARM64 (Graviton)*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
```

Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

4. Install the package by using the following command:

```
sudo apt-get install ./mount-s3.deb
```

5. Verify that Mountpoint for Amazon S3 is successfully installed by running the following command:

```
mount-s3 --version
```

You should see output similar to the following:

```
mount-s3 1.3.1
```

## Other Linux distributions

1. Consult your operating system documentation to install the FUSE and `libfuse2` packages, which are required.
2. Copy the download URL for your architecture.

*x86_64*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
```

*ARM64 (Graviton)*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
```

3. Download the Mountpoint for Amazon S3 package. Replace *`download-link`* with the appropriate download URL from the preceding step.

```
wget download-link
```

4. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the signature URL for your architecture.

*x86_64*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
```

*ARM64 (Graviton)*:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
```

Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

5. Install the package by using the following command:

```
sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
mount-s3.tar.gz
```

6. Add the `mount-s3` binary to your `PATH` environment variable. In your `$HOME/.profile` file, append the following line:

```
export PATH=$PATH:/opt/aws/mountpoint-s3/bin
```

Save the `.profile` file, and run the following command:

```
source $HOME/.profile
```

7. Verify that Mountpoint for Amazon S3 is successfully installed by running the following command:

```
mount-s3 --version
```

You should see output similar to the following:

```
mount-s3 1.3.1
```

## Verifying the signature of the Mountpoint for Amazon S3 package

1. Install GnuPG (the gpg command). It is required to verify the authenticity and integrity of a downloaded Mountpoint for Amazon S3 package. GnuPG is installed by default on Amazon Linux Amazon Machine Images (AMIs). After you installGnuPG, proceed to step 2.
2. Download the Mountpoint public key by running the following command:

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. Import the Mountpoint public key into your keyring by running the following command:

```
gpg --import KEYS
```

4. Verify the fingerprint of the Mountpoint public key by running the following command:

```
gpg --fingerprint mountpoint-s3@amazon.com
```

Confirm that the displayed fingerprint string matches the following:

```
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A
```

If the fingerprint string doesn't match, do not finish installing Mountpoint, and contact AWS Support.

5. Download the package signature file. Replace *signature-link* with the appropriate signature link from the preceding sections.

```
wget signature-link
```

6. Verify the signature of the downloaded package by running the following command. Replace *signature-filename* with the file name from the previous step.

```
gpg --verify signature-filename
```

For example, on RPM-based distributions, including Amazon Linux, enter the following command:

```
gpg --verify mount-s3.rpm.asc
```

7. The output should include the phrase `Good signature`. If the output includes the phrase `BAD signature`, redownload the Mountpoint package file and repeat these steps. If the issue persists, do not finish installing Mountpoint, and contact AWS Support.

The output may include a warning about a trusted signature. This does not indicate a problem. It only means that you have not independently verified the Mountpoint public key.

# Configuring and using Mountpoint

To use Mountpoint for Amazon S3, your host needs valid AWS credentials with access to the bucket or buckets that you would like to mount. For different ways to authenticate, see Mountpoint AWS Credentials on GitHub.

For example, you can create a new AWS Identity and Access Management (IAM) user and role for this purpose. Make sure that this role has access to the bucket or buckets that you would like to mount. You can pass the IAM role to your Amazon EC2 instance with an instance profile.

## Using Mountpoint for Amazon S3

Use Mountpoint for Amazon S3 to do the following:

1. Mount buckets with the `mount-s3` command.

   In the following example, replace *DOC-EXAMPLE-BUCKET* with the name of your S3 bucket, and replace *~/mnt* with the directory on your host where you want your S3 bucket to be mounted.

   ```
   mkdir ~/mnt
   mount-s3 DOC-EXAMPLE-BUCKET ~/mnt
   ```

   Because the Mountpoint client runs in the background by default, the *~/mnt* directory now gives you access to the objects in your S3 bucket.

2. Access the objects in your bucket through Mountpoint.

   After you mount your bucket locally, you can use common Linux commands, such as `cat` or `ls`, to work with your S3 objects. Mountpoint for Amazon S3 interprets keys in your S3 bucket as file system paths by splitting them on the forward slash (/) character. For example, if you have the object key `Data/2023-01-01.csv` in your bucket, you will have a directory named `Data` in your Mountpoint file system, with a file named `2023-01-01.csv` inside it.

   Mountpoint for Amazon S3 intentionally does not implement the full POSIX standard specification for file systems. Mountpoint is optimized for workloads that need high-throughput read and write access to data stored in Amazon S3 through a file system interface, but that otherwise do not rely on file system features. For more information, see Mountpoint for Amazon S3 file system behavior on GitHub. Customers that need richer file system

semantics should consider other AWS file services, such as Amazon Elastic File System (Amazon EFS) or Amazon FSx.

3. Unmount your bucket by using the `umount` command. This command unmounts your S3 bucket and exits Mountpoint.

   To use the following example command, replace *~/mnt* with the directory on your host where your S3 bucket is mounted.

   ```
   umount ~/mnt
   ```

   **Note**

   To get a list of options for this command, run `umount --help`.

For additional Mountpoint configuration details, see S3 bucket configuration, and file system configuration on GitHub.

## Configuring caching in Mountpoint

When you use Mountpoint for Amazon S3, you can configure it to cache the most recently accessed data from your S3 buckets on Amazon EC2 instance storage or an attached Amazon EBS volume. Caching this data can help to accelerate performance and reduce the cost of repeated data access. Caching in Mountpoint is ideal for use cases where you repeatedly read the same data that doesn't change during the multiple reads. For example, you can use caching with machine learning training jobs that need to read a training dataset multiple times to improve model accuracy.

When you mount an S3 bucket, you can optionally enable caching through flags. You can configure the location and size of the data cache and the amount of time metadata is retained in the cache. When you mount a bucket and caching is enabled, Mountpoint creates an empty sub-directory at the configured cache location, if that sub-directory doesn't already exist. When you first mount a bucket and when you unmount, Mountpoint deletes the contents of the cache location. For more information about configuring and using caching in Mountpoint, see Mountpoint for Amazon S3 Caching configuration on GitHub.

When you mount an S3 bucket, you can enable caching with the `--cache` *CACHE_PATH* flag. In the following example, replace *CACHE_PATH* with the filepath to the directory that you want to

cache your data in. Replace *DOC-EXAMPLE-BUCKET* with the name of your S3 bucket, and replace *~/mnt* with the directory on your host where you want your S3 bucket to be mounted.

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH DOC-EXAMPLE-BUCKET ~/mnt
```

**Important**

If you enable caching, Mountpoint will persist unencrypted object content from your S3 bucket at the caching location configured at mount. In order to protect your data, we recommend that you restrict access to the data cache location.

## Troubleshooting Mountpoint

Mountpoint for Amazon S3 is backed by AWS Support. If you need assistance, contact the AWS Support Center.

You can also review and submit Mountpoint Issues on GitHub.

If you discover a potential security issue in this project, we ask that you notify AWS Security through our vulnerability reporting page. Do not create a public GitHub issue.

If your application behaves unexpectedly with Mountpoint, you can inspect your log information to diagnose the problem.

### Logging

By default, Mountpoint emits high-severity log information to syslog.

To view logs on most modern Linux distributions, including Amazon Linux, run the following `journald` command:

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

On other Linux systems, `syslog` entries are likely written to a file such as `/var/log/syslog`.

You can use these logs to troubleshoot your application. For example, if your application tries to overwrite an existing file, the operation fails, and you will see a line similar to the following in the log:

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

For more information, see Mountpoint for Amazon S3 Logging on GitHub.

# Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration is a bucket-level feature that enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration is designed to optimize transfer speeds from across the world into S3 buckets. Transfer Acceleration takes advantage of the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, the data is routed to Amazon S3 over an optimized network path.

When you use Transfer Acceleration, additional data transfer charges might apply. For more information about pricing, see Amazon S3 pricing.

## Why use Transfer Acceleration?

You might want to use Transfer Acceleration on a bucket for various reasons:

- Your customers upload to a centralized bucket from all over the world.
- You transfer gigabytes to terabytes of data on a regular basis across continents.
- You can't use all of your available bandwidth over the internet when uploading to Amazon S3.

For more information about when to use Transfer Acceleration, see Amazon S3 FAQs.

## Requirements for using Transfer Acceleration

The following are required when you are using Transfer Acceleration on an S3 bucket:

- Transfer Acceleration is only supported on virtual-hosted style requests. For more information about virtual-hosted style requests, see Making requests using the REST API.
- The name of the bucket used for Transfer Acceleration must be DNS-compliant and must not contain periods (".").
- Transfer Acceleration must be enabled on the bucket. For more information, see Enabling and using S3 Transfer Acceleration.

After you enable Transfer Acceleration on a bucket, it might take up to 20 minutes before the data transfer speed to the bucket increases.

> **Note**
>
> Transfer Acceleration is currently supported for buckets located in the following Regions:
>
> - Asia Pacific (Tokyo) (ap-northeast-1)
> - Asia Pacific (Seoul) (ap-northeast-2)
> - Asia Pacific (Mumbai) (ap-south-1)
> - Asia Pacific (Singapore) (ap-southeast-1)
> - Asia Pacific (Sydney) (ap-southeast-2)
> - Canada (Central) (ca-central-1)
> - Europe (Frankfurt) (eu-central-1)
> - Europe (Ireland) (eu-west-1)
> - Europe (London) (eu-west-2)
> - Europe (Paris) (eu-west-3)
> - South America (São Paulo) (sa-east-1)
> - US East (N. Virginia) (us-east-1)
> - US East (Ohio) (us-east-2)
> - US West (N. California) (us-west-1)
> - US West (Oregon) (us-west-2)

- To access the bucket that is enabled for Transfer Acceleration, you must use the endpoint *bucketname*`.s3-accelerate.amazonaws.com`. Or, use the dual-stack endpoint *bucketname*`.s3-accelerate.dualstack.amazonaws.com` to connect to the enabled bucket over IPv6. You can continue to use the regular endpoints for standard data transfer.
- You must be the bucket owner to set the transfer acceleration state. The bucket owner can assign permissions to other users to allow them to set the acceleration state on a bucket. The `s3:PutAccelerateConfiguration` permission permits users to enable or disable Transfer Acceleration on a bucket. The `s3:GetAccelerateConfiguration` permission permits users to return the Transfer Acceleration state of a bucket, which is either `Enabled` or `Suspended`.

The following sections describe how to get started and use Amazon S3 Transfer Acceleration for transferring data.

**Topics**



## Getting started with Amazon S3 Transfer Acceleration

You can use Amazon S3 Transfer Acceleration for fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration uses the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path.

To get started using Amazon S3 Transfer Acceleration, perform the following steps:

1. **Enable Transfer Acceleration on a bucket**

   You can enable Transfer Acceleration on a bucket any of the following ways:

   - Use the Amazon S3 console.
   - Use the REST API PUT Bucket accelerate operation.
   - Use the AWS CLI and AWS SDKs. For more information, see Developing with Amazon S3 using the AWS SDKs.

   For more information, see Enabling and using S3 Transfer Acceleration.

   > **Note**
   >
   > For your bucket to work with transfer acceleration, the bucket name must conform to DNS naming requirements and must not contain periods (".").

2. **Transfer data to and from the acceleration-enabled bucket**

   Use one of the following s3-accelerate endpoint domain names:

   - To access an acceleration-enabled bucket, use *`bucketname`*`.s3-accelerate.amazonaws.com`.

- To access an acceleration-enabled bucket over IPv6, use *bucketname*.s3-accelerate.dualstack.amazonaws.com.

  Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. The Transfer Acceleration dual-stack endpoint only uses the virtual hosted-style type of endpoint name. For more information, see Getting started making requests over IPv6 and Using Amazon S3 dual-stack endpoints.

> **Note**
>
> Your data transfer application must use one of the following two types of endpoints to access the bucket for faster data transfer: .s3-accelerate.amazonaws.com or .s3-accelerate.dualstack.amazonaws.com for the dual-stack endpoint. If you want to use standard data transfer, you can continue to use the regular endpoints.

You can point your Amazon S3 PUT object and GET object requests to the s3-accelerate endpoint domain name after you enable Transfer Acceleration. For example, suppose that you currently have a REST API application using PUT Object that uses the hostname mybucket.s3.us-east-1.amazonaws.com in the PUT request. To accelerate the PUT, you change the hostname in your request to mybucket.s3-accelerate.amazonaws.com. To go back to using the standard upload speed, change the name back to mybucket.s3.us-east-1.amazonaws.com.

After Transfer Acceleration is enabled, it can take up to 20 minutes for you to realize the performance benefit. However, the accelerate endpoint is available as soon as you enable Transfer Acceleration.

You can use the accelerate endpoint in the AWS CLI, AWS SDKs, and other tools that transfer data to and from Amazon S3. If you are using the AWS SDKs, some of the supported languages use an accelerate endpoint client configuration flag so you don't need to explicitly set the endpoint for Transfer Acceleration to *bucketname*.s3-accelerate.amazonaws.com. For examples of how to use an accelerate endpoint client configuration flag, see Enabling and using S3 Transfer Acceleration.

You can use all Amazon S3 operations through the transfer acceleration endpoints *except* for the following:

- GET Service (list buckets)
- PUT Bucket (create bucket)
- DELETE Bucket

Also, Amazon S3 Transfer Acceleration does not support cross-Region copies using PUT Object - Copy.

## Enabling and using S3 Transfer Acceleration

You can use Amazon S3 Transfer Acceleration transfer files quickly and securely over long distances between your client and an S3 bucket. You can enable Transfer Acceleration using the S3 console, the AWS Command Line Interface (AWS CLI), API, or the AWS SDKs.

This section provides examples of how to enable Amazon S3 Transfer Acceleration on a bucket and use the acceleration endpoint for the enabled bucket.

For more information about Transfer Acceleration requirements, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

### Using the S3 console

> **Note**
>
> If you want to compare accelerated and non-accelerated upload speeds, open the Amazon S3 Transfer Acceleration Speed Comparison tool.
> The Speed Comparison tool uses multipart upload to transfer a file from your browser to various AWS Regions with and without Amazon S3 transfer acceleration. You can compare the upload speed for direct uploads and transfer accelerated uploads by Region.

#### To enable transfer acceleration for an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable transfer acceleration for.
3. Choose **Properties**.
4. Under **Transfer acceleration**, choose **Edit**.

5. Choose **Enable**, and choose **Save changes**.

### To access accelerated data transfers

1. After Amazon S3 enables transfer acceleration for your bucket, view the **Properties** tab for the bucket.
2. Under **Transfer acceleration**, **Accelerated endpoint** displays the transfer acceleration endpoint for your bucket. Use this endpoint to access accelerated data transfers to and from your bucket.

   If you suspend transfer acceleration, the accelerate endpoint no longer works.

### Using the AWS CLI

The following are examples of AWS CLI commands used for Transfer Acceleration. For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

#### Enabling Transfer Acceleration on a bucket

Use the AWS CLI put-bucket-accelerate-configuration command to enable or suspend Transfer Acceleration on a bucket.

The following example sets `Status=Enabled` to enable Transfer Acceleration on a bucket. You use `Status=Suspended` to suspend Transfer Acceleration.

#### Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-
configuration Status=Enabled
```

#### Using Transfer Acceleration

You can direct all Amazon S3 requests made by s3 and s3api AWS CLI commands to the accelerate endpoint: `s3-accelerate.amazonaws.com`. To do this, set the configuration value `use_accelerate_endpoint` to `true` in a profile in your AWS Config file. Transfer Acceleration must be enabled on your bucket to use the accelerate endpoint.

All requests are sent using the virtual style of bucket addressing: `my-bucket.s3-accelerate.amazonaws.com`. Any `ListBuckets`, `CreateBucket`, and `DeleteBucket` requests are not sent to the accelerate endpoint because the endpoint doesn't support those operations.

For more information about `use_accelerate_endpoint`, see AWS CLI S3 Configuration in the *AWS CLI Command Reference*.

The following example sets `use_accelerate_endpoint` to `true` in the default profile.

**Example**

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

If you want to use the accelerate endpoint for some AWS CLI commands but not others, you can use either one of the following two methods:

- Use the accelerate endpoint for any s3 or s3api command by setting the `--endpoint-url` parameter to `https://s3-accelerate.amazonaws.com`.
- Set up separate profiles in your AWS Config file. For example, create one profile that sets `use_accelerate_endpoint` to `true` and a profile that does not set `use_accelerate_endpoint`. When you run a command, specify which profile you want to use, depending upon whether you want to use the accelerate endpoint.

### Uploading an object to a bucket enabled for Transfer Acceleration

The following example uploads a file to a bucket enabled for Transfer Acceleration by using the default profile that has been configured to use the accelerate endpoint.

**Example**

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

The following example uploads a file to a bucket enabled for Transfer Acceleration by using the `--endpoint-url` parameter to specify the accelerate endpoint.

**Example**

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url https://s3-accelerate.amazonaws.com
```

### Using the AWS SDKs

The following are examples of using Transfer Acceleration to upload objects to Amazon S3 using the AWS SDK. Some of the AWS SDK supported languages (for example, Java and .NET) use an accelerate endpoint client configuration flag so you don't need to explicitly set the endpoint for Transfer Acceleration to *bucketname*.s3-accelerate.amazonaws.com.

Java

#### Example

The following example shows how to use an accelerate endpoint to upload an object to Amazon S3. The example does the following:

- Creates an `AmazonS3Client` that is configured to use accelerate endpoints. All buckets that the client accesses must have Transfer Acceleration enabled.
- Enables Transfer Acceleration on a specified bucket. This step is necessary only if the bucket you specify doesn't already have Transfer Acceleration enabled.
- Verifies that transfer acceleration is enabled for the specified bucket.
- Uploads a new object to the specified bucket using the bucket's accelerate endpoint.

For more information about using Transfer Acceleration, see Getting started with Amazon S3 Transfer Acceleration. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

public class TransferAcceleration {
    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate
 endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .enableAccelerateMode()
                    .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(
                    new SetBucketAccelerateConfigurationRequest(bucketName,
                            new BucketAccelerateConfiguration(
                                    BucketAccelerateStatus.Enabled)));

            // Verify that transfer acceleration is enabled for the bucket.
            String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                    new GetBucketAccelerateConfigurationRequest(bucketName))
                    .getStatus();
            System.out.println("Bucket accelerate status: " + accelerateStatus);

            // Upload a new object using the accelerate endpoint.
            s3Client.putObject(bucketName, keyName, "Test object for transfer
 acceleration");
            System.out.println("Object \"" + keyName + "\" uploaded with transfer
 acceleration.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following example shows how to use the AWS SDK for .NET to enable Transfer Acceleration on a bucket. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

**Example**

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await
 s3Client.PutBucketAccelerateConfigurationAsync(putRequest);
```

```
                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
 s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ",
 response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer
 acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
}
```

When uploading an object to a bucket that has Transfer Acceleration enabled, you specify using the acceleration endpoint at the time of creating a client.

```
var client = new AmazonS3Client(new AmazonS3Config
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

Javascript

For an example of enabling Transfer Acceleration by using the AWS SDK for JavaScript, see Calling the putBucketAccelerateConfiguration operation in the *AWS SDK for JavaScript API Reference*.

Python (Boto)

For an example of enabling Transfer Acceleration by using the SDK for Python, see put_bucket_accelerate_configuration in the *AWS SDK for Python (Boto3) API Reference*.

Other

For information about using other AWS SDKs, see Sample Code and Libraries.

#### Using the REST API

Use the REST API `PutBucketAccelerateConfiguration` operation to enable accelerate configuration on an existing bucket.

For more information, see PutBucketAccelerateConfiguration in the *Amazon Simple Storage Service API Reference*.

## Using the Amazon S3 Transfer Acceleration Speed Comparison tool

You can use the Amazon S3 Transfer Acceleration Speed Comparison tool to compare accelerated and non-accelerated upload speeds across Amazon S3 Regions. The Speed Comparison tool uses multipart uploads to transfer a file from your browser to various Amazon S3 Regions with and without using Transfer Acceleration.

You can access the Speed Comparison tool using either of the following methods:

- Copy the following URL into your browser window, replacing *`region`* with the AWS Region that you are using (for example, `us-west-2`) and *`yourBucketName`* with the name of the bucket that you want to evaluate:

  ```
  https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/
  accelerate-speed-comparsion.html?
  region=region&origBucketName=yourBucketName
  ```

  For a list of the Regions supported by Amazon S3, see Amazon S3 endpoints and quotas in the *AWS General Reference*.
- Use the Amazon S3 console.

# Using Requester Pays buckets for storage transfers and usage

In general, bucket owners pay for all Amazon S3 storage and data transfer costs that are associated with their bucket. However, you can configure a bucket to be a *Requester Pays* bucket. With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data.

Typically, you configure buckets to be Requester Pays buckets when you want to share data but not incur charges associated with others accessing the data. For example, you might use Requester Pays buckets when making available large datasets, such as zip code directories, reference data, geospatial information, or web crawling data.

> **Important**
>
> If you enable Requester Pays on a bucket, anonymous access to that bucket is not allowed.

You must authenticate all requests involving Requester Pays buckets. The request authentication enables Amazon S3 to identify and charge the requester for their use of the Requester Pays bucket.

When the requester assumes an AWS Identity and Access Management (IAM) role before making their request, the account to which the role belongs is charged for the request. For more information about IAM roles, see IAM roles in the *IAM User Guide*.

After you configure a bucket to be a Requester Pays bucket, requesters must show they understand that they will be charged for the request and for the data download. To show they accept the charges, requesters must either include `x-amz-request-payer` as a header in their API request for DELETE, GET, HEAD, POST, and PUT requests, or add the `RequestPayer` parameter in their REST request. For CLI requests, requesters can use the `--request-payer` parameter.

**Example – Using Requester Pays when deleting an object**

To use the following DeleteObjectVersion API example, replace the *`user input placeholders`* with your own information.

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

If the requester restores objects by using the RestoreObject API, Requester Pays is supported as long as the `x-amz-request-payer` header or the `RequestPayer` parameter are in the request; however, the requester only pays for the cost of the request. The bucket owner pays the retrieval charges.

Requester Pays buckets do not support the following:

- Anonymous requests
- SOAP requests
- Using a Requester Pays bucket as the target bucket for end-user logging, or vice versa. However, you can turn on end-user logging on a Requester Pays bucket where the target bucket is not a Requester Pays bucket.

## How Requester Pays charges work

The charge for successful Requester Pays requests is straightforward: The requester pays for the data transfer and the request, and the bucket owner pays for the data storage. However, the bucket owner is charged for the request under the following conditions:

- The request returns an `AccessDenied` (HTTP `403 Forbidden`) error and the request is initiated inside the bucket owner's individual AWS account or AWS organization.
- The request is a SOAP request.

For more information about Requester Pays, see the following topics.

**Topics**

- Configuring Requester Pays on a bucket
- Retrieving the requestPayment configuration using the REST API
- Downloading objects from Requester Pays buckets

## Configuring Requester Pays on a bucket

You can configure an Amazon S3 bucket to be a *Requester Pays* bucket so that the requester pays the cost of the request and data download instead of the bucket owner.

This section provides examples of how to configure Requester Pays on an Amazon S3 bucket using the console and the REST API.

### Using the S3 console

#### To enable Requester Pays for an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable Requester Pays for.
3. Choose **Properties**.
4. Under **Requester pays**, choose **Edit**.
5. Choose **Enable**, and choose **Save changes**.

   Amazon S3 enables Requester Pays for your bucket and displays your **Bucket overview**. Under **Requester pays**, you see **Enabled**.

### Using the REST API

Only the bucket owner can set the `RequestPaymentConfiguration.payer` configuration value of a bucket to `BucketOwner` (the default) or `Requester`. Setting the `requestPayment` resource is optional. By default, the bucket is not a Requester Pays bucket.

To revert a Requester Pays bucket to a regular bucket, you use the value `BucketOwner`. Typically, you would use `BucketOwner` when uploading data to the Amazon S3 bucket, and then you would set the value to `Requester` before publishing the objects in the bucket.

#### To set requestPayment

- Use a PUT request to set the `Payer` value to `Requester` on a specified bucket.

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

If the request succeeds, Amazon S3 returns a response similar to the following.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

You can set Requester Pays only at the bucket level. You can't set Requester Pays for specific objects within the bucket.

You can configure a bucket to be `BucketOwner` or `Requester` at any time. However, there might be a few minutes before the new configuration value takes effect.

> **Note**
>
> Bucket owners who give out presigned URLs should consider carefully before configuring a bucket to be Requester Pays, especially if the URL has a long lifetime. The bucket owner is charged each time the requester uses a presigned URL that uses the bucket owner's credentials.

## Retrieving the requestPayment configuration using the REST API

You can determine the `Payer` value that is set on a bucket by requesting the resource `requestPayment`.

**To return the requestPayment resource**

- Use a GET request to obtain the `requestPayment` resource, as shown in the following request.

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

If the request succeeds, Amazon S3 returns a response similar to the following.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

This response shows that the `payer` value is set to `Requester`.

## Downloading objects from Requester Pays buckets

Because requesters are charged for downloading data from Requester Pays buckets, the requests must contain a special parameter, `x-amz-request-payer`, which confirms that the requester knows that they will be charged for the download. To access objects in Requester Pays buckets, requests must include one of the following.

- For DELETE, GET, HEAD, POST, and PUT requests, include `x-amz-request-payer : requester` in the header
- For signed URLs, include `x-amz-request-payer=requester` in the request

If the request succeeds and the requester is charged, the response includes the header `x-amz-request-charged:requester`. If `x-amz-request-payer` is not in the request, Amazon S3 returns a 403 error and charges the bucket owner for the request.

> **Note**
>
> Bucket owners do not need to add `x-amz-request-payer` to their requests.
> Ensure that you have included `x-amz-request-payer` and its value in your signature calculation. For more information, see Constructing the CanonicalizedAmzHeaders Element.

### Using the REST API

#### To download objects from a Requester Pays bucket

- Use a GET request to download an object from a Requester Pays bucket, as shown in the following request.

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

If the GET request succeeds and the requester is charged, the response includes `x-amz-request-charged:requester`.

Amazon S3 can return an `Access Denied` error for requests that try to get objects from a Requester Pays bucket. For more information, see Error Responses in the *Amazon Simple Storage Service API Reference*.

### Using the AWS CLI

To download objects from a Requester Pays bucket using the AWS CLI, you specify `--request-payer requester` as part of your `get-object` request. For more information, see get-object in the *AWS CLI Reference*.

## Bucket quotas, restrictions, and limitations

An Amazon S3 bucket is owned by the AWS account that created it. Bucket ownership is not transferable to another account.

### Bucket quota limits

By default, you can create up to 100 buckets in each of your AWS accounts. If you need additional buckets, you can increase your account bucket quota to a maximum of 1,000 buckets by submitting a quota increase request. There is no difference in performance whether you use many buckets or just a few.

> **Note**
>
> You do not need to submit multiple quota increase requests for each AWS Region. Your bucket quota is applied to your AWS account.

For information about how to increase your bucket quota, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

**Objects and bucket limitations**

There is no max bucket size or limit to the number of objects that you can store in a bucket. You can store all of your objects in a single bucket, or you can organize them across several buckets. However, you can't create a bucket from within another bucket.

**Bucket naming limits**

When you create a bucket, you choose its name and the AWS Region to create it in. After you create a bucket, you can't change its name or Region.

When naming a bucket, choose a name that is relevant to you or your business. Avoid using names associated with others. For example, you should avoid using `AWS` or `Amazon` in your bucket name.

**Reusing bucket names**

If a bucket is empty, you can delete it. After a bucket is deleted, the name becomes available for reuse. However, after you delete the bucket, you might not be able to reuse the name for various reasons.

For example, when you delete the bucket and the name becomes available for reuse, another AWS account might create a bucket with that name. In addition, some time might pass before you can reuse the name of a deleted bucket. If you want to use the same bucket name, we recommend that you don't delete the bucket.

For more information about bucket names, see Bucket naming rules.

**Bucket naming and automatically created buckets**

If your application automatically creates buckets, choose a bucket naming scheme that is unlikely to cause naming conflicts. Ensure that your application logic will choose a different bucket name if a bucket name is already taken.

For more information about bucket naming, see Bucket naming rules.

### Bucket operations

The high availability engineering of Amazon S3 is focused on *get*, *put*, *list*, and *delete* operations. Because bucket operations work against a centralized, global resource space, it is not recommended to create, delete, or configure buckets on the high availability code path of your application. It's better to create, delete, or configure buckets in a separate initialization or setup routine that you run less often.

# Uploading, downloading, and working with objects in Amazon S3

To store your data in Amazon S3, you work with resources known as buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to a bucket. When the object is in the bucket, you can open it, download it, and copy it. When you no longer need an object or a bucket, you can clean up these resources.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

> **Important**
>
> In the Amazon S3 console, when you choose **Open** or **Download As** for an object, these operations create presigned URLs. For the duration of five minutes, your object will be accessible to anyone who has access to these presigned URLs. For more information about presigned URLs, see Using presigned URLS.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

**Topics**

- Copying, moving, and renaming objects
- Downloading objects
- Checking object integrity
- Deleting Amazon S3 objects
- Organizing, listing, and working with your objects
- Working with presigned URLs
- Transforming objects with S3 Object Lambda

# Amazon S3 objects overview

Amazon S3 is an object store that uses unique key-values to store as many objects as you want. You store these objects in one or more buckets, and each object can be up to 5 TB in size. An object consists of the following:

Key

The name that you assign to an object. You use the object key to retrieve the object. For more information, see Working with object metadata.

Version ID

Within a bucket, a key and version ID uniquely identify an object. The version ID is a string that Amazon S3 generates when you add an object to a bucket. For more information, see Using versioning in S3 buckets.

Value

The content that you are storing.

An object value can be any sequence of bytes. Objects can range in size from zero to 5 TB. For more information, see Uploading objects.

Metadata

A set of name-value pairs with which you can store information regarding the object. You can assign metadata, referred to as user-defined metadata, to your objects in Amazon S3. Amazon S3 also assigns system-metadata to these objects, which it uses for managing objects. For more information, see Working with object metadata.

Subresources

Amazon S3 uses the subresource mechanism to store object-specific additional information. Because subresources are subordinates to objects, they are always associated with some other entity such as an object or a bucket. For more information, see Object subresources.

Access control information

You can control access to the objects you store in Amazon S3. Amazon S3 supports both the resource-based access control, such as an access control list (ACL) and bucket policies, and user-based access control. For more information about access control, see the following:

- Access Management
- Identity and Access Management for Amazon S3
- Configuring ACLs

Your Amazon S3 resources (for example, buckets and objects) are private by default. You must explicitly grant permission for others to access these resources. For more information about sharing objects, see Sharing objects with presigned URLs.

Tags

You can use tags to categorize your stored objects, for access control, or cost allocation. For more information, see Categorizing your storage using tags.

## Object subresources

Amazon S3 defines a set of subresources associated with buckets and objects. Subresources are subordinates to objects. This means that subresources don't exist on their own. They are always associated with some other entity, such as an object or a bucket.

The following table lists the subresources associated with Amazon S3 objects.

| Subresource | Description |
| --- | --- |
| acl | Contains a list of grants identifying the grantees and the permissions granted. When you create an object, the `acl` identifies the object owner as having full control over the object. You can retrieve an object ACL or replace it with an |

| Subresource | Description |
| --- | --- |
| | updated list of grants. Any update to an ACL requires you to replace the existing ACL. For more information about ACLs, see Access control list (ACL) overview. |

## Creating object key names

The *object key* (or key name) uniquely identifies the object in an Amazon S3 bucket. *Object metadata* is a set of name-value pairs. For more information about object metadata, see Working with object metadata.

When you create an object, you specify the key name, which uniquely identifies the object in the bucket. For example, on the Amazon S3 console, when you highlight a bucket, a list of objects in your bucket appears. These names are the *object keys*. The object key name is a sequence of Unicode characters with UTF-8 encoding of up to 1,024 bytes long. Object key names are case sensitive.

> **Note**
>
> Object key names with the value "soap" aren't supported for virtual-hosted-style requests. For object key name values where "soap" is used, a path-style URL must be used instead.

The Amazon S3 data model is a flat structure: You create a bucket, and the bucket stores objects. There is no hierarchy of subbuckets or subfolders. However, you can infer logical hierarchy using key name prefixes and delimiters as the Amazon S3 console does. The Amazon S3 console supports a concept of folders. For more information about how to edit metadata from the Amazon S3 console, see Editing object metadata in the Amazon S3 console.

Suppose that your bucket (`admin-created`) has four objects with the following object keys:

`Development/Projects.xls`

`Finance/statement1.pdf`

`Private/taxdocument.pdf`

`s3-dg.pdf`

The console uses the key name prefixes (`Development/`, `Finance/`, and `Private/`) and delimiter ('/') to present a folder structure. The `s3-dg.pdf` key does not have a prefix, so its object appears directly at the root level of the bucket. If you open the `Development/` folder, you see the `Projects.xlsx` object in it.

- Amazon S3 supports buckets and objects, and there is no hierarchy. However, by using prefixes and delimiters in an object key name, the Amazon S3 console and the AWS SDKs can infer hierarchy and introduce the concept of folders.
- The Amazon S3 console implements folder object creation by creating a zero-byte object with the folder *prefix and delimiter* value as the key. These folder objects don't appear in the console. Otherwise they behave like any other objects and can be viewed and manipulated through the REST API, AWS CLI, and AWS SDKs.

## Object key naming guidelines

You can use any UTF-8 character in an object key name. However, using certain characters in key names can cause problems with some applications and protocols. The following guidelines help you maximize compliance with DNS, web-safe characters, XML parsers, and other APIs.

### Safe characters

The following character sets are generally safe for use in key names.

| | |
|---|---|
| Alphanumeric characters | • 0-9<br>• a-z<br>• A-Z |
| Special characters | • Exclamation point (!)<br>• Hyphen (-)<br>• Underscore (_)<br>• Period (.)<br>• Asterisk (*)<br>• Single quote (')<br>• Open parenthesis (()<br>• Close parenthesis ()) |

The following are examples of valid object key names:

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

> **Note**
>
> Objects with key names ending with period(s) "." downloaded using the Amazon S3 console will have the period(s) "." removed from the key name of the downloaded object. To download an object with the key name ending in period(s) "." retained in the downloaded object, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. In addition, be aware of the following prefix limitations:
>
> - Objects with a prefix of "./" must be uploaded or downloaded with the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. You cannot use the Amazon S3 console.
> - Objects with a prefix of "../" cannot be uploaded using the AWS Command Line Interface (AWS CLI) or Amazon S3 console.

## Characters that might require special handling

The following characters in a key name might require additional code handling and likely need to be URL encoded or referenced as HEX. Some of these are non-printable characters that your browser might not handle, which also requires special handling:

- Ampersand ("&")
- Dollar ("$")
- ASCII character ranges 00–1F hex (0–31 decimal) and 7F (127 decimal)
- 'At' symbol ("@")
- Equals ("=")
- Semicolon (";")
- Forward slash ("/")
- Colon (":")
- Plus ("+")

- Space – Significant sequences of spaces might be lost in some uses (especially multiple spaces)
- Comma (",")
- Question mark ("?")

### Characters to avoid

We recommend that you don't use the following characters in a key name because of significant special character handling, which isn't consistent across all applications.

- Backslash ("\")
- Left curly brace ("{")
- Non-printable ASCII characters (128–255 decimal characters)
- Caret ("^")
- Right curly brace ("}")
- Percent character ("%")
- Grave accent / back tick ("`")
- Right square bracket ("]")
- Quotation marks
- 'Greater Than' symbol (">")
- Left square bracket ("[")
- Tilde ("~")
- 'Less Than' symbol ("<")
- 'Pound' character ("#")
- Vertical bar / pipe ("|")

### XML related object key constraints

As specified by the XML standard on end-of-line handling, all XML text is normalized such that single carriage returns (ASCII code 13) and carriage returns immediately followed by a line feed (ASCII code 10) are replaced by a single line feed character. To ensure the correct parsing of object keys in XML requests, carriage returns and other special characters must be replaced with their equivalent XML entity code when they are inserted within XML tags. The following is a list of such special characters and their equivalent entity codes:

- ' as '
- " as "
- & as &
- < as <
- > as >
- \r as  or 
- \n as 
 or 


**Example**

The following example illustrates the use of an XML entity code as a substitution for a carriage return. This `DeleteObjects` request deletes an object with the `key` parameter: `/some/prefix/objectwith\rcarriagereturn` (where the \r is the carriage return).

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
</Delete>
```

# Working with object metadata

You can set object metadata in Amazon S3 at the time you upload the object. Object metadata is a set of name-value pairs. After you upload the object, you cannot modify object metadata. The only way to modify object metadata is to make a copy of the object and set the metadata.

When you create an object, you also specify the key name, which uniquely identifies the object in the bucket. The *object key* (or *key name*) uniquely identifies the object in an Amazon S3 bucket. For more information, see Creating object key names.

There are two kinds of metadata in Amazon S3: *system-defined metadata* and *user-defined metadata*. The following sections provide more information about system-defined and user-defined metadata. For more information about editing metadata using the Amazon S3 console, see Editing object metadata in the Amazon S3 console.

# System-defined object metadata

For each object stored in a bucket, Amazon S3 maintains a set of system metadata. Amazon S3 processes this system metadata as needed. For example, Amazon S3 maintains object-creation date and size metadata and uses this information as part of object management.

There are two categories of system metadata:

- **System controlled** – Metadata such as the object-creation date is system controlled, meaning that only Amazon S3 can modify the value.
- **User controlled** – Other system metadata, such as the storage class configured for the object and whether the object has server-side encryption enabled, are examples of system metadata whose values you control. If your bucket is configured as a website, sometimes you might want to redirect a page request to another page or to an external URL. In this case, a webpage is an object in your bucket. Amazon S3 stores the page redirect value as system metadata whose value you control.

  When you create objects, you can configure the values of these system metadata items or update the values when you need to. For more information about storage classes, see Using Amazon S3 storage classes.

  Amazon S3 uses AWS KMS keys to encrypt your Amazon S3 objects. AWS KMS encrypts only the object data. The checksum, along with the specified algorithm, are stored as part of the object's metadata. If server-side encryption is requested for the object, then the checksum is stored in encrypted form. For more information about server-side encryption, see Protecting data with encryption.

> **Note**
>
> The PUT request header is limited to 8 KB in size. Within the PUT request header, the system-defined metadata is limited to 2 KB in size. The size of system-defined metadata is measured by taking the sum of the number of bytes in the US-ASCII encoding of each key and value.

The following table provides a list of system-defined metadata and whether you can update it.

| Name | Description | Can user modify the value? |
| --- | --- | --- |
| `Date` | The current date and time. | No |
| `Cache-Control` | A general header field used to specify caching policies. | Yes |
| `Content-Disposition` | Object presentational information. | Yes |
| `Content-Length` | The object size in bytes. | No |
| `Content-Type` | The object type. | Yes |
| `Last-Modified` | The object creation date or the last modified date, whichever is the latest. For multipart uploads, the object creation date is the date of initiation of the multipart upload. | No |
| `ETag` | An entity tag (ETag) that represents a specific version of an object. For objects that are not uploaded as a multipart upload and are either unencrypted or encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3), the ETag is an MD5 digest of the data. | No |
| `x-amz-server-side-encryption` | A header that indicates whether server-side encryption is enabled for the object, and whether that encryption is using the AWS Key Management Service (AWS KMS) keys (SSE-KMS) or using Amazon S3 managed encryption keys (SSE-S3). For more information, see Protecting data with server-side encryption. | Yes |
| `x-amz-checksum-crc32` , `x-amz-checksum-crc32c` , `x-amz-` | Headers that contain the checksum or digest of the object. At most, one of these headers will be set at a time, depending on the checksum algorithm that you instruct Amazon S3 to use. For more information about | No |

| Name | Description | Can user modify the value? |
| --- | --- | --- |
| `checksum-sha1` , `x-amz-checksum-sha256` | choosing the checksum algorithm, see Checking object integrity. | |
| `x-amz-version-id` | The object version. When you enable versioning on a bucket, Amazon S3 assigns a version ID to objects added to the bucket. For more information, see Using versioning in S3 buckets. | No |
| `x-amz-delete-marker` | A Boolean marker that indicates whether the object is a delete marker. This marker is used only in buckets that have versioning enabled, | No |
| `x-amz-storage-class` | The storage class used for storing the object. For more information, see Using Amazon S3 storage classes. | Yes |
| `x-amz-website-redirect-location` | A header that redirects requests for the associated object to another object in the same bucket or to an external URL. For more information, see (Optional) Configuring a webpage redirect. | Yes |
| `x-amz-server-side-encryption-aws-kms-key-id` | A header that indicates the ID of the AWS KMS symmetric encryption KMS key that was used to encrypt the object. This header is used only when the `x-amz-server-side-encryption` header is present and has the value of `aws:kms`. | Yes |
| `x-amz-server-side-encryption-customer-algorithm` | A header that indicates whether server-side encryption with customer-provided encryption keys (SSE-C) is enabled. For more information, see Using server-side encryption with customer-provided keys (SSE-C). | Yes |
| `x-amz-tagging` | The tag-set for the object. The tag-set must be encoded as URL Query parameters. | Yes |

# User-defined object metadata

When uploading an object, you can also assign metadata to the object. You provide this optional information as a name-value (key-value) pair when you send a PUT or POST request to create the object. When you upload objects using the REST API, the optional user-defined metadata names must begin with `x-amz-meta-` to distinguish them from other HTTP headers. When you retrieve the object using the REST API, this prefix is returned. When you upload objects using the SOAP API, the prefix is not required. When you retrieve the object using the SOAP API, the prefix is removed, regardless of which API you used to upload the object.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

When metadata is retrieved through the REST API, Amazon S3 combines headers that have the same name (ignoring case) into a comma-delimited list. If some metadata contains unprintable characters, it is not returned. Instead, the `x-amz-missing-meta` header is returned with a value of the number of unprintable metadata entries. The `HeadObject` action retrieves metadata from an object without returning the object itself. This operation is useful if you're only interested in an object's metadata. To use HEAD, you must have READ access to the object. For more information, see HeadObject in the *Amazon Simple Storage Service API Reference*.

User-defined metadata is a set of key-value pairs. Amazon S3 stores user-defined metadata keys in lowercase.

Amazon S3 allows arbitrary Unicode characters in your metadata values.

To avoid issues around the presentation of these metadata values, you should conform to using US-ASCII characters when using REST and UTF-8 when using SOAP or browser-based uploads through POST.

When using non–US-ASCII characters in your metadata values, the provided Unicode string is examined for non–US-ASCII characters. Values of such headers are character decoded as per RFC 2047 before storing and encoded as per RFC 2047 to make them mail-safe before returning. If the string contains only US-ASCII characters, it is presented as is.

The following is an example.

```
PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3
```

**Note**

The PUT request header is limited to 8 KB in size. Within the PUT request header, the user-defined metadata is limited to 2 KB in size. The size of user-defined metadata is measured by taking the sum of the number of bytes in the UTF-8 encoding of each key and value.

For information about changing the metadata of your object after it has been uploaded by creating a copy of the object, modifying it, and replacing the old object, or creating a new version, see Editing object metadata in the Amazon S3 console.

## Editing object metadata in the Amazon S3 console

You can use the Amazon S3 console to edit metadata of existing S3 objects. Some metadata is set by Amazon S3 when you upload the object. For example, `Content-Length` and `Last-Modified` are system-defined object metadata fields that can't be modified by a user.

You can also set some metadata when you upload the object and later edit it as your needs change. For example, you might have a set of objects that you initially store in the `STANDARD` storage class. Over time, you might no longer need this data to be highly available. So you change the storage

class to `GLACIER` by editing the value of the `x-amz-storage-class` key from `STANDARD` to `GLACIER`.

**Note**

Consider the following issues when you are editing object metadata in Amazon S3:

- This action creates a *copy* of the object with updated settings and the last-modified date. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. If S3 Versioning is not enabled, a new copy of the object replaces the original object. The AWS account associated with the IAM role that changes the property also becomes the owner of the new object or (object version).
- To use the Amazon S3 console to edit metadata for an object that has user-defined tags, you must also have the `s3:GetObjectTagging` permission. If you are using the Amazon S3 console to edit the metadata for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, the metadata for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.
- Editing metadata updates values for existing key names.
- Objects that are encrypted with customer-provided encryption keys (SSE-C) cannot be copied using the console. You must use the AWS CLI, AWS SDK, or the Amazon S3 REST API.

**Warning**

When editing metadata of folders, wait for the `Edit metadata` operation to finish before adding new objects to the folder. Otherwise, new objects might also be edited.

The following topics describe how to edit metadata of an object using the Amazon S3 console.

### Editing system-defined metadata

You can configure some, but not all, system metadata for an S3 object. For a list of system-defined metadata and whether you can modify their values, see System-defined object metadata.

**To edit system-defined metadata of an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Navigate to your Amazon S3 bucket or folder, and select the check box to the left of the names of the objects with metadata you want to edit.
3. On the **Actions** menu, choose **Edit actions**, and choose **Edit metadata**.
4. Review the objects listed, and choose **Add metadata**.
5. For metadata **Type**, select **System-defined**.
6. Specify a unique **Key** and the metadata **Value**.
7. To edit additional metadata, choose **Add metadata**. You can also choose **Remove** to remove a set of type-key-values.
8. When you are done, choose **Edit metadata** and Amazon S3 edits the metadata of the specified objects.

### Editing user-defined metadata

You can edit user-defined metadata of an object by combining the metadata prefix, `x-amz-meta-`, and a name you choose to create a custom key. For example, if you add the custom name `alt-name`, the metadata key would be `x-amz-meta-alt-name`.

User-defined metadata can be as large as 2 KB total. To calculate the total size of user-defined metadata, sum the number of bytes in the UTF-8 encoding for each key and value. Both keys and their values must conform to US-ASCII standards. For more information, see User-defined object metadata.

**To edit user-defined metadata of an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to add metadata to.

   You can also optionally navigate to a folder.
3. In the **Objects** list, select the check box next to the names of the objects that you want to add metadata to.

4. On the **Actions** menu, choose **Edit metadata**.
5. Review the objects listed, and choose **Add metadata**.
6. For metadata **Type**, choose **User-defined**.
7. Enter a unique custom **Key** following `x-amz-meta-`. Also enter a metadata **Value**.
8. To add additional metadata, choose **Add metadata**. You can also choose **Remove** to remove a set of type-key-values.
9. Choose **Edit metadata**.

   Amazon S3 edits the metadata of the specified objects.

# Uploading objects

When you upload a file to Amazon S3, it is stored as an S3 *object*. Objects consist of the file data and metadata that describes the object. You can have an unlimited number of objects in a bucket. Before you can upload files to an Amazon S3 bucket, you need write permissions for the bucket. For more information about access permissions, see Identity and Access Management for Amazon S3.

You can upload any file type—images, backups, data, movies, and so on—into an S3 bucket. The maximum size of a file that you can upload by using the Amazon S3 console is 160 GB. To upload a file larger than 160 GB, use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

If you upload an object with a key name that already exists in a versioning-enabled bucket, Amazon S3 creates another version of the object instead of replacing the existing object. For more information about versioning, see Using the S3 console.

Depending on the size of the data that you're uploading, Amazon S3 offers the following options:

- **Upload an object in a single operation by using the AWS SDKs, REST API, or AWS CLI** – With a single PUT operation, you can upload a single object up to 5 GB in size.
- **Upload a single object by using the Amazon S3 console –** With the Amazon S3 console, you can upload a single object up to 160 GB in size.
- **Upload an object in parts by using the AWS SDKs, REST API, or AWS CLI –** Using the multipart upload API operation, you can upload a single large object, up to 5 TB in size.

The multipart upload API operation is designed to improve the upload experience for larger objects. You can upload an object in parts. These object parts can be uploaded independently, in any order, and in parallel. You can use a multipart upload for objects from 5 MB to 5 TB in size. For more information, see Uploading and copying objects using multipart upload.

When you upload an object, the object is automatically encrypted using server-side encryption with Amazon S3 managed keys (SSE-S3) by default. When you download it, the object is decrypted. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets and Protecting data with encryption.

When you're uploading an object, if you want to use a different type of default encryption, you can also specify server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) in your S3 PUT requests or set the default encryption configuration in the destination bucket to use SSE-KMS to encrypt your data. For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

If you encounter an Access Denied (403 Forbidden) error in Amazon S3, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3 to learn more about its common causes.

## Using the S3 console

This procedure explains how to upload objects and folders to an Amazon S3 bucket by using the console.

When you upload an object, the object key name is the file name and any optional prefixes. In the Amazon S3 console, you can create folders to organize your objects. In Amazon S3, folders are represented as prefixes that appear in the object key name. If you upload an individual object to a folder in the Amazon S3 console, the folder name is included in the object key name.

For example, if you upload an object named `sample1.jpg` to a folder named `backup`, the key name is `backup/sample1.jpg`. However, the object is displayed in the console as `sample1.jpg` in the `backup` folder. For more information about key names, see Working with object metadata.

> **Note**
>
> If you rename an object or change any of the properties in the Amazon S3 console, for example **Storage Class**, **Encryption**, or **Metadata**, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).

When you upload a folder, Amazon S3 uploads all of the files and subfolders from the specified folder to your bucket. It then assigns an object key name that is a combination of the uploaded file name and the folder name. For example, if you upload a folder named `/images` that contains two files, `sample1.jpg` and `sample2.jpg`, Amazon S3 uploads the files and then assigns the corresponding key names, `images/sample1.jpg` and `images/sample2.jpg`. The key names include the folder name as a prefix. The Amazon S3 console displays only the part of the key name that follows the last `/`. For example, within an `images` folder, the `images/sample1.jpg` and `images/sample2.jpg` objects are displayed as `sample1.jpg` and a `sample2.jpg`.

**To upload folders and files to an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to upload your folders or files to.
4. Choose **Upload**.
5. In the **Upload** window, do one of the following:
   - Drag and drop files and folders to the **Upload** window.
   - Choose **Add file** or **Add folder**, choose the files or folders to upload, and choose **Open**.
6. To enable versioning, under **Destination**, choose **Enable Bucket Versioning**.
7. To upload the listed files and folders without configuring additional upload options, at the bottom of the page, choose **Upload**.

   Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

**To configure additional object properties**

1. To change access control list permissions, choose **Permissions**.
2. Under **Access control list (ACL)**, edit the permissions.

   For information about object access permissions, see Using the S3 console to set ACL permissions for an object. You can grant read access to your objects to the public (everyone in the world) for all of the files that you're uploading. However, we recommend not changing the default setting for public read access. Granting public read access is applicable to a small subset of use cases, such as when buckets are used for websites. You can always change the object permissions after you upload the object.
3. To configure other additional properties, choose **Properties**.
4. Under **Storage class**, choose the storage class for the files that you're uploading.

   For more information about storage classes, see Using Amazon S3 storage classes.
5. To update the encryption settings for your objects, under **Server-side encryption settings**, do the following.

   a. Choose **Specify an encryption key**.

   b. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.

   c. If you chose **Override bucket settings for default encryption**, you must configure the following encryption settings.

   - To encrypt the uploaded files by using keys that are managed by Amazon S3, choose **Amazon S3 managed key (SSE-S3)**.

     For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).
   - To encrypt the uploaded files by using keys stored in AWS Key Management Service (AWS KMS), choose **AWS Key Management Service key (SSE-KMS)**. Then choose one of the following options for **AWS KMS key**:
     - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN.
> Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

6. To use additional checksums, choose **On**. Then for **Checksum function**, choose the function that you would like to use. Amazon S3 calculates and stores the checksum value after it receives the entire object. You can use the **Precalculated value** box to supply a precalculated value. If you do, Amazon S3 compares the value that you provided to the value that it calculates. If the two values do not match, Amazon S3 generates an error.

   Additional checksums enable you to specify the checksum algorithm that you would like to use to verify your data. For more information about additional checksums, see Checking object integrity.

7. To add tags to all of the objects that you are uploading, choose **Add tag**. Enter a tag name in the **Key** field. Enter a value for the tag.

   Object tagging gives you a way to categorize storage. Each tag is a key-value pair. Key and tag values are case sensitive. You can have up to 10 tags per object. A tag key can be up to 128

Unicode characters in length, and tag values can be up to 255 Unicode characters in length. For more information about object tags, see Categorizing your storage using tags.

8. To add metadata, choose **Add metadata**.

   a. Under **Type**, choose **System defined** or **User defined**.

      For system-defined metadata, you can select common HTTP headers, such as **Content-Type** and **Content-Disposition**. For a list of system-defined metadata and information about whether you can add the value, see System-defined object metadata. Any metadata starting with the prefix `x-amz-meta-` is treated as user-defined metadata. User-defined metadata is stored with the object and is returned when you download the object. Both the keys and their values must conform to US-ASCII standards. User-defined metadata can be as large as 2 KB. For more information about system-defined and user-defined metadata, see Working with object metadata.

   b. For **Key**, choose a key.

   c. Type a value for the key.

9. To upload your objects, choose **Upload**.

   Amazon S3 uploads your object. When the upload completes, you can see a success message on the **Upload: status** page.

10. Choose **Exit**.

## Using the AWS SDKs

You can use the AWS SDKs to upload objects in Amazon S3. The SDKs provide wrapper libraries for you to upload data easily. For information, see the List of supported SDKs.

Here are some examples with a few select SDKs:

.NET

The following C# code example creates two objects with two `PutObjectRequest` requests:

- The first `PutObjectRequest` request saves a text string as sample object data. It also specifies the bucket and object key names.
- The second `PutObjectRequest` request uploads a file by specifying the file name. This request also specifies the `ContentType` header and optional object metadata (a title).

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        // For simplicity the example creates two objects from the same file.
        // You specify key names for these objects.
        private const string keyName1 = "*** key name for first object created ***";
        private const string keyName2 = "*** key name for second object created
 ***";
        private const string filePath = @"*** file path ***";
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };
```

```
                PutObjectResponse response1 = await
 client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
                    FilePath = filePath,
                    ContentType = "text/plain"
                };

                putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
                PutObjectResponse response2 = await
 client.PutObjectAsync(putRequest2);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object"
                    , e.Message);
            }
        }
    }
}
```

Java

The following example creates two objects. The first object has a text string as data, and the second object is a file. The example creates the first object by specifying the bucket name, object key, and text data directly in a call to `AmazonS3Client.putObject()`. The example creates the second object by using a `PutObjectRequest` that specifies the bucket name, object key, and file path. The `PutObjectRequest` also specifies the `ContentType` header and title metadata.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;
import java.io.IOException;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
```

```
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JavaScript

The following example uploads an existing file to an Amazon S3 bucket in a specific Region.

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

PHP

This example guides you through using classes from the AWS SDK for PHP to upload an object of up to 5 GB in size. For larger files, you must use the multipart upload API operation. For more information, see Uploading and copying objects using multipart upload.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

**Example — Creating an object in an Amazon S3 bucket by uploading data**

The following PHP example creates an object in a specified bucket by uploading data using the `putObject()` method.

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

try {
    // Upload data.
    $result = $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);

    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

The AWS SDK for Ruby - Version 3 has two ways of uploading an object to Amazon S3. The first uses a managed file uploader, which makes it easier to upload files of any size from disk. To use the managed file uploader method:

1. Create an instance of the `Aws::S3::Resource` class.

2. Reference the target object by bucket name and key. Objects live in a bucket and have unique keys that identify each object.
3. Call#`upload_file` on the object.

**Example**

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

The second way that the AWS SDK for Ruby - Version 3 can upload an object uses the `#put` method of `Aws::S3::Object`. This is useful if the object is a string or an I/O object that is not a file on disk. To use this method:

1. Create an instance of the `Aws::S3::Resource` class.
2. Reference the target object by bucket name and key.
3. Call`#put`, passing in the string or I/O object.

**Example**

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
```

```
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

### Using the REST API

You can send REST requests to upload an object. You can send a PUT request to upload data in a single operation. For more information, see PUT Object.

### Using the AWS CLI

You can send a PUT request to upload an object of up to 5 GB in a single operation. For more information, see the `PutObject` example in the *AWS CLI Command Reference*.

## Uploading and copying objects using multipart upload

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation.

Using multipart upload provides the following advantages:

- **Improved throughput** – You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** – Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- **Pause and resume object uploads** – You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- **Begin an upload before you know the final object size** – You can upload an object as you are creating it.

We recommend that you use multipart upload in the following ways:

- If you're uploading large objects over a stable high-bandwidth network, use multipart upload to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance.
- If you're uploading over a spotty network, use multipart upload to increase resiliency to network errors by avoiding upload restarts. When using multipart upload, you need to retry uploading only the parts that are interrupted during the upload. You don't need to restart uploading your object from the beginning.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets. For more information about using multipart upload with S3 Express One Zone and directory buckets, see Using multipart uploads with directory buckets.

## Multipart upload process

Multipart upload is a three-step process: You initiate the upload, you upload the object parts, and after you have uploaded all the parts, you complete the multipart upload. Upon receiving the complete multipart upload request, Amazon S3 constructs the object from the uploaded parts, and you can then access the object just as you would any other object in your bucket.

You can list all of your in-progress multipart uploads or get a list of the parts that you have uploaded for a specific multipart upload. Each of these operations is explained in this section.

### Multipart upload initiation

When you send a request to initiate a multipart upload, Amazon S3 returns a response with an upload ID, which is a unique identifier for your multipart upload. You must include this upload ID whenever you upload parts, list the parts, complete an upload, or stop an upload. If you want to provide any metadata describing the object being uploaded, you must provide it in the request to initiate multipart upload.

### Parts upload

When uploading a part, in addition to the upload ID, you must specify a part number. You can choose any part number between 1 and 10,000. A part number uniquely identifies a part and its position in the object you are uploading. The part number that you choose doesn't need to be in a consecutive sequence (for example, it can be 1, 5, and 14). If you upload a new part using the same part number as a previously uploaded part, the previously uploaded part is overwritten.

When you upload a part, Amazon S3 returns an *entity tag (ETag)* for the part as a header in the response. For each part upload, you must record the part number and the ETag value. You must include these values in the subsequent request to complete the multipart upload. Each part will have its own ETag at the time of upload. However, once the multipart upload is complete and all parts are consolidated, all the parts will be under one ETag as a checksum of checksums.

> **Note**
>
> After you initiate a multipart upload and upload one or more parts, you must either complete or stop the multipart upload to stop getting charged for storage of the uploaded parts. Only *after* you either complete or stop a multipart upload will Amazon S3 free up the parts storage and stop charging you for the parts storage.
> After stopping a multipart upload, you cannot upload any part using that upload ID again. If any part uploads were in-progress, they can still succeed or fail even after you stop the upload. To make sure you free all storage consumed by all parts, you must stop a multipart upload only after all part uploads have been completed.

### Multipart upload completion

When you complete a multipart upload, Amazon S3 creates an object by concatenating the parts in ascending order based on the part number. If any object metadata was provided in the *initiate multipart upload* request, Amazon S3 associates that metadata with the object. After a successful *complete* request, the parts no longer exist.

Your *complete multipart upload* request must include the upload ID and a list of both part numbers and corresponding ETag values. The Amazon S3 response includes an ETag that uniquely identifies the combined object data. This ETag is not necessarily an MD5 hash of the object data.

### Sample multipart upload calls

For this example, assume that you are generating a multipart upload for a 100 GB file. In this case, you would have the following API calls for the entire process. There would be a total of 1002 API calls.

- A `CreateMultipartUpload` call to start the process.
- 1000 individual `UploadPart` calls, each uploading a part of 100 MB, for a total size of 100 GB.
- A `CompleteMultipartUpload` call to finish the process.

#### Multipart upload listings

You can list the parts of a specific multipart upload or all in-progress multipart uploads. The list parts operation returns the parts information that you have uploaded for a specific multipart upload. For each list parts request, Amazon S3 returns the parts information for the specified multipart upload, up to a maximum of 1,000 parts. If there are more than 1,000 parts in the multipart upload, you must send a series of list part requests to retrieve all the parts. Note that the returned list of parts doesn't include parts that haven't finished uploading. Using the *list multipart uploads* operation, you can obtain a list of multipart uploads that are in progress.

An in-progress multipart upload is an upload that you have initiated, but have not yet completed or stopped. Each request returns at most 1,000 multipart uploads. If there are more than 1,000 multipart uploads in progress, you must send additional requests to retrieve the remaining multipart uploads. Use the returned listing only for verification. Do not use the result of this listing when sending a *complete multipart upload* request. Instead, maintain your own list of the part numbers that you specified when uploading parts and the corresponding ETag values that Amazon S3 returns.

## Checksums with multipart upload operations

When you upload an object to Amazon S3, you can specify a checksum algorithm for Amazon S3 to use. Amazon S3 uses MD5 by default to verify data integrity; however, you can specify an additional checksum algorithm to use. When using MD5, Amazon S3 calculates the checksum of the entire multipart object after the upload is complete. This checksum is not a checksum of the entire object, but rather a checksum of the checksums for each individual part.

When you instruct Amazon S3 to use additional checksums, Amazon S3 calculates the checksum value for each part and stores the values. You can use the API or SDK to retrieve the checksum value for individual parts by using `GetObject` or `HeadObject`. If you want to retrieve the checksum values for individual parts of multipart uploads still in process, you can use `ListParts`.

**Important**

If you are using a multipart upload with additional checksums, the multipart part numbers must use consecutive part numbers. When using additional checksums, if you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates HTTP `500 Internal Server Error` error.

For more information about how checksums work with multipart objects, see Checking object integrity.

## Concurrent multipart upload operations

In a distributed development environment, it is possible for your application to initiate several updates on the same object at the same time. Your application might initiate several multipart uploads using the same object key. For each of these uploads, your application can then upload parts and send a complete upload request to Amazon S3 to create the object. When the buckets have S3 Versioning enabled, completing a multipart upload always creates a new version. When you initiate multiple multipart uploads that use the same object key in a versioning-enabled bucket, the current version of the object is determined by which upload started most recently (`createdDate`). For example, suppose that you start a CreateMultipartUpload request for an object at 10:00 AM. Then you submit a second `CreateMultipartUpload` request for the same object at 11:00 AM. Because the second request was submitted most recently, the object uploaded by the 11:00 AM request will be the current version, even if the first upload is completed after the second one. For buckets that don't have versioning enabled, it is possible that some other request received between the time when a multipart upload is initiated and when it is completed might take precedence.

**Note**

It is possible for some other request received between the time you initiated a multipart upload and completed it to take precedence. For example, if another operation deletes a key after you initiate a multipart upload with that key, but before you complete it, the complete multipart upload response might indicate a successful object creation without you ever seeing the object.

## Multipart upload and pricing

After you initiate a multipart upload, Amazon S3 retains all the parts until you either complete or stop the upload. Throughout its lifetime, you are billed for all storage, bandwidth, and requests for this multipart upload and its associated parts.

These parts are charged according to the storage class specified when the parts were uploaded. An exception to this are parts uploaded to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. In-progress multipart parts for a PUT to the S3 Glacier Flexible Retrieval storage class are billed as S3 Glacier Flexible Retrieval Staging Storage at S3 Standard storage rates until the upload completes. In addition, both CreateMultipartUpload and UploadPart are billed at S3 Standard rates. Only the CompleteMultipartUpload request is billed at the S3 Glacier Flexible Retrieval rate. Similarly, in-progress multipart parts for a PUT to the S3 Glacier Deep Archive storage class are billed as S3 Glacier Flexible Retrieval Staging Storage at S3 Standard storage rates until the upload completes, with only the CompleteMultipartUpload request charged at S3 Glacier Deep Archive rates.

If you stop the multipart upload, Amazon S3 deletes upload artifacts and any parts that you have uploaded, and you are no longer billed for them. There are no early delete charges for deleting incomplete multipart uploads regardless of storage class specified. For more information about pricing, see Amazon S3 pricing.

> **Note**
>
> To minimize your storage costs, we recommend that you configure a lifecycle rule to delete incomplete multipart uploads after a specified number of days by using the `AbortIncompleteMultipartUpload` action. For more information about creating a lifecycle rule to delete incomplete multipart uploads, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

## API support for multipart upload

These libraries provide a high-level abstraction that makes uploading multipart objects easy. However, if your application requires, you can use the REST API directly. The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload.

For a multipart upload walkthrough that uses AWS Lambda functions, see Uploading large objects to Amazon S3 using multipart upload and transfer acceleration.

- Create Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

## AWS Command Line Interface support for multipart upload

The following topics in the AWS Command Line Interface describe the operations for multipart upload.

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

## AWS SDK support for multipart upload

You can use an AWS SDKs to upload an object in parts. For a list of AWS SDKs supported by API action see:

- Create Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload

- List Parts
- List Multipart Uploads

## Multipart upload API and permissions

You must have the necessary permissions to use the multipart upload operations. You can use access control lists (ACLs), the bucket policy, or the user policy to grant individuals permissions to perform these operations. The following table lists the required permissions for various multipart upload operations when using ACLs, a bucket policy, or a user policy.

| Action | Required permissions |
| --- | --- |
| Create Multipart Upload | You must be allowed to perform the `s3:PutObject` action on an object to create multipart upload.<br><br>The bucket owner can allow other principals to perform the `s3:PutObject` action. |
| Initiate Multipart Upload | You must be allowed to perform the `s3:PutObject` action on an object to initiate multipart upload.<br><br>The bucket owner can allow other principals to perform the `s3:PutObject` action. |
| Initiator | Container element that identifies who initiated the multipart upload. If the initiator is an AWS account, this element provides the same information as the Owner element. If the initiator is an IAM user, this element provides the user ARN and display name. |
| Upload Part | You must be allowed to perform the `s3:PutObject` action on an object to upload a part.<br><br>The bucket owner must allow the initiator to perform the `s3:PutObject` action on an object in order for the initiator to upload a part for that object. |
| Upload Part (Copy) | You must be allowed to perform the `s3:PutObject` action on an object to upload a part. Because you are uploading a part from an existing object, you must be allowed `s3:GetObject` on the source object. |

| Action | Required permissions |
| --- | --- |
| | For the initiator to upload a part for an object, the owner of the bucket must allow the initiator to perform the `s3:PutObject` action on the object. |
| Complete Multipart Upload | You must be allowed to perform the `s3:PutObject` action on an object to complete a multipart upload.<br><br>The bucket owner must allow the initiator to perform the `s3:PutObject` action on an object in order for the initiator to complete a multipart upload for that object. |
| Stop Multipart Upload | You must be allowed to perform the `s3:AbortMultipartUpload` action to stop a multipart upload.<br><br>By default, the bucket owner and the initiator of the multipart upload are allowed to perform this action as a part of IAM and bucket polices. If the initiator is an IAM user, that user's AWS account is also allowed to stop that multipart upload. With VPC endpoint policies, the initiator of the multipart upload does not automatically gain the permission to perform the `s3:AbortMultipartUpload` action.<br><br>In addition to these defaults, the bucket owner can allow other principals to perform the `s3:AbortMultipartUpload` action on an object. The bucket owner can deny any principal the ability to perform the `s3:AbortMultipartUpload` action. |

| Action | Required permissions |
| --- | --- |
| List Parts | You must be allowed to perform the `s3:ListMultipartUploadParts` action to list parts in a multipart upload.<br><br>By default, the bucket owner has permission to list parts for any multipart upload to the bucket. The initiator of the multipart upload has the permission to list parts of the specific multipart upload. If the multipart upload initiator is an IAM user, the AWS account controlling that IAM user also has permission to list parts of that upload.<br><br>In addition to these defaults, the bucket owner can allow other principals to perform the `s3:ListMultipartUploadParts` action on an object. The bucket owner can also deny any principal the ability to perform the `s3:ListMultipartUploadParts` action. |
| List Multipart Uploads | You must be allowed to perform the `s3:ListBucketMultipartUploads` action on a bucket to list multipart uploads in progress to that bucket.<br><br>In addition to the default, the bucket owner can allow other principals to perform the `s3:ListBucketMultipartUploads` action on the bucket. |
| AWS KMS Encrypt and Decrypt related permissions | To perform a multipart upload with encryption using an AWS Key Management Service (AWS KMS) KMS key, the requester must have permission to the `kms:Decrypt` and `kms:GenerateDataKey` actions on the key. The requester must also have permissions for the `kms:GenerateDataKey` action for the CreateMultipartUpload API. Then, the requester needs permissions for the `kms:Decrypt` action on the UploadPart and UploadPartCopy APIs. These permissions are required because Amazon S3 must decrypt and read data from the encrypted file parts before it completes the multipart upload.<br><br>If your IAM user or role is in the same AWS account as the KMS key, then you must have these permissions on the key policy. If your IAM user or role belongs to a different account than the KMS key, then you must have the permissions on both the key policy and your IAM user or role. |

For information on the relationship between ACL permissions and permissions in access policies, see Mapping of ACL permissions and access policy permissions. For information about IAM users, roles, and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.

**Topics**



## Configuring a bucket lifecycle configuration to delete incomplete multipart uploads

As a best practice, we recommend that you configure a lifecycle rule by using the `AbortIncompleteMultipartUpload` action to minimize your storage costs. For more information about aborting a multipart upload, see Aborting a multipart upload.

Amazon S3 supports a bucket lifecycle rule that you can use to direct Amazon S3 to stop multipart uploads that aren't completed within a specified number of days after being initiated. When a multipart upload isn't completed within the specified time frame, it becomes eligible for an abort operation. Amazon S3 then stops the multipart upload and deletes the parts associated with the multipart upload. This rule applies to both existing multipart uploads and those that you create later.

The following is an example lifecycle configuration that specifies a rule with the `AbortIncompleteMultipartUpload` action.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
```

```
            <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

In the example, the rule doesn't specify a value for the `Prefix` element (the object key name prefix). Therefore, the rule applies to all objects in the bucket for which you initiated multipart uploads. Any multipart uploads that were initiated and weren't completed within seven days become eligible for an abort operation. The abort action has no effect on completed multipart uploads.

For more information about the bucket lifecycle configuration, see Managing your storage lifecycle.

> **Note**
>
> If the multipart upload is completed within the number of days specified in the rule, the `AbortIncompleteMultipartUpload` lifecycle action does not apply (that is, Amazon S3 doesn't take any action). Also, this action doesn't apply to objects. No objects are deleted by this lifecycle action. Additionally, you will not incur early delete charges for S3 Lifecycle when you remove any incomplete multipart upload parts.

#### Using the S3 console

To automatically manage incomplete multipart uploads, you can use the S3 console to create a lifecycle rule to expire incomplete multipart upload bytes from your bucket after a specified number of days. The following procedure shows you how to add a lifecycle rule to delete incomplete multipart uploads after 7 days. For more information about adding lifecycle rules, see Setting a lifecycle configuration on a bucket.

**To add a lifecycle rule to abort incomplete multipart uploads that are more than 7 days old**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to create a lifecycle rule for.
3. Choose the **Management** tab, and choose **Create lifecycle rule**.
4. In **Lifecycle rule name**, enter a name for your rule.

   The name must be unique within the bucket.

5. Choose the scope of the lifecycle rule:

   - To create a lifecycle rule for all objects with a specific prefix, choose **Limit the scope of this rule using one or more filters**, and enter the prefix in the **Prefix** field.
   - To create a lifecycle rule for all objects in the bucket, choose **This rule applies to all objects in the bucket**, and choose **I acknowledge that this rule applies to all objects in the bucket**.
6. Under **Lifecycle rule actions**, select **Delete expired object delete markers or incomplete multipart uploads**.
7. Under **Delete expired object delete markers or incomplete multipart uploads**, select **Delete incomplete multipart uploads**.
8. In the **Number of days** field, enter the number of days after which to delete incomplete multipart uploads (for this example, 7 days).
9. Choose **Create rule**.

#### Using the AWS CLI

The following `put-bucket-lifecycle-configuration` AWS Command Line Interface (AWS CLI) command adds the lifecycle configuration for the specified bucket. To use this command, replace the *`user input placeholders`* with your information.

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket amzn-s3-demo-bucket1  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

The following example shows how to add a lifecycle rule to abort incomplete multipart uploads by using the AWS CLI. It includes an example JSON lifecycle configuration to abort incomplete multipart uploads that are more than 7 days old.

To use the CLI commands in this example, replace the *`user input placeholders`* with your information.

#### To add a lifecycle rule to abort incomplete multipart uploads

1. Set up the AWS CLI. For instructions, see Developing with Amazon S3 using the AWS CLI.
2. Save the following example lifecycle configuration in a file (for example, *`lifecycle.json`*). This example configuration specifies an empty prefix, and therefore it applies to all objects in the bucket. To restrict the configuration to a subset of objects, you can specify a prefix.

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. Run the following CLI command to set this lifecycle configuration on your bucket.

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket amzn-s3-demo-bucket1  \
--lifecycle-configuration file://lifecycle.json
```

4. To verify that the lifecycle configuration has been set on your bucket, retrieve the lifecycle configuration by using the following `get-bucket-lifecycle` command.

```
aws s3api get-bucket-lifecycle  \
--bucket amzn-s3-demo-bucket1
```

5. To delete the lifecycle configuration, use the following `delete-bucket-lifecycle` command.

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket1
```

## Uploading an object using multipart upload

You can use the multipart upload to programmatically upload a single object to Amazon S3.

For more information, see the following sections.

### Using the AWS SDKs (high-level API)

Some AWS SDKs expose a high-level API that simplifies multipart upload by combining the different API operations required to complete a multipart upload into a single operation. For more information, see Uploading and copying objects using multipart upload.

If you need to pause and resume multipart uploads, vary part sizes during the upload, or do not know the size of the data in advance, use the low-level API methods. The low-level API methods for multipart uploads offer additional functionality, for more information, see Using the AWS SDKs (low-level API).

Java

To upload large files, use the `TransferManager` class. This high-level API operation can upload data from a file or a stream. You can also set advanced options, such as the part size you want to use for the multipart upload, or the number of concurrent threads you want to use when uploading the parts. You can also set optional object properties, the storage class, or the access control list (ACL). You use the `PutObjectRequest` and the `TransferManagerConfiguration` classes to set these advanced options.

When possible, `TransferManager` tries to use multiple threads to upload multiple parts of a single upload at once. When dealing with large content sizes and high bandwidth, this can increase throughput significantly.

In addition to file-upload functionality, the `TransferManager` class enables you to stop an in-progress multipart upload. An upload is considered to be in progress after you initiate it and until you complete or stop it. The `TransferManager` stops all in-progress multipart uploads on a specified bucket that were initiated before a specified date and time.

> **Note**
>
> When you're using a stream for the source of data, the `TransferManager` class does not do concurrent uploads.

The following example loads an object using the high-level multipart upload Java API (the `TransferManager` class). For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import java.io.File;

public class HighLevelMultipartUpload {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** Path for file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(bucketName, keyName, new File(filePath));
            System.out.println("Object upload started");

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object upload complete");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

.NET

To upload a file to an S3 bucket, use the `TransferUtility` class. When uploading data from a file, you must provide the object's key name. If you don't, the API uses the file name for the key name. When uploading data from a stream, you must provide the object's key name.

To set advanced upload options—such as the part size, the number of threads when uploading the parts concurrently, metadata, the storage class, or ACL—use the `TransferUtilityUploadRequest` class.

> **Note**
>
> When you're using a stream for the source of data, the `TransferUtility` class does not do concurrent uploads.

The following C# example uploads a file to an Amazon S3 bucket in multiple parts. It shows how to use various `TransferUtility.Upload` overloads to upload a file. Each successive call to upload replaces the previous upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
```

```
        private const string keyName = "*** provide a name for the uploaded object
***";
        private const string filePath = "*** provide the full path name of the file
to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key
name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName,
keyName);
                Console.WriteLine("Upload 2 completed");

                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                                bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
```

```
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
                Console.WriteLine("Upload 4 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }

        }
    }
}
```

JavaScript

**Example**

Upload a large file.

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;
```

```
export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );

    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
              PartNumber: i + 1,
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
```

```
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open the
 file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

### Example

Download a large file.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
```

```
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Go

**Example**

Upload a large object by using an upload manager to break the data into parts and upload them concurrently.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
```

```
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

**Example**

Download a large object by using a download manager to get the data in parts and download them concurrently.

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey string)
 ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
```

```
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

PHP

This topic explains how to use the high-level `Aws\S3\Model\MultipartUpload\UploadBuilder` class from the AWS SDK for PHP for multipart file uploads. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example uploads a file to an Amazon S3 bucket. The example demonstrates how to set parameters for the `MultipartUploader` object.

```
 require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Python

The following example loads an object using the high-level multipart upload Python API (the `TransferManager` class).

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
```

```
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)
```

```
    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
```

```
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.
```

```
    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

### Using the AWS SDKs (low-level API)

The AWS SDK exposes a low-level API that closely resembles the Amazon S3 REST API for multipart uploads (see Uploading and copying objects using multipart upload. Use the low-level API when you need to pause and resume multipart uploads, vary part sizes during the upload, or do not

know the size of the upload data in advance. When you don't have these requirements, use the high-level API (see Using the AWS SDKs (high-level API)).

Java

The following example shows how to use the low-level Java classes to upload a file. It performs the following steps:

- Initiates a multipart upload using the `AmazonS3Client.initiateMultipartUpload()` method, and passes in an `InitiateMultipartUploadRequest` object.
- Saves the upload ID that the `AmazonS3Client.initiateMultipartUpload()` method returns. You provide this upload ID for each subsequent multipart upload operation.
- Uploads the parts of the object. For each part, you call the `AmazonS3Client.uploadPart()` method. You provide part upload information using an `UploadPartRequest` object.
- For each part, saves the ETag from the response of the `AmazonS3Client.uploadPart()` method in a list. You use the ETag values to complete the multipart upload.
- Calls the `AmazonS3Client.completeMultipartUpload()` method to complete the multipart upload.

**Example**

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String filePath = "*** Path to file to upload ***";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Create a list of ETag objects. You retrieve ETags for each object
 part
            // uploaded,
            // then, after each individual part has been uploaded, pass the list of
 ETags to
            // the request to complete the upload.
            List<PartETag> partETags = new ArrayList<PartETag>();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(bucketName, keyName);
            InitiateMultipartUploadResult initResponse =
 s3Client.initiateMultipartUpload(initRequest);

            // Upload the file parts.
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++) {
                // Because the last part could be less than 5 MB, adjust the part
 size as
                // needed.
                partSize = Math.min(partSize, (contentLength - filePosition));

                // Create the request to upload a part.
                UploadPartRequest uploadRequest = new UploadPartRequest()
                        .withBucketName(bucketName)
                        .withKey(keyName)
```

```
                        .withUploadId(initResponse.getUploadId())
                        .withPartNumber(i)
                        .withFileOffset(filePosition)
                        .withFile(file)
                        .withPartSize(partSize);

                // Upload the part and add the response's ETag to our list.
                UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
                partETags.add(uploadResult.getPartETag());

                filePosition += partSize;
            }

            // Complete the multipart upload.
            CompleteMultipartUploadRequest compRequest = new
 CompleteMultipartUploadRequest(bucketName, keyName,
                    initResponse.getUploadId(), partETags);
            s3Client.completeMultipartUpload(compRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following C# example shows how to use the low-level AWS SDK for .NET multipart upload API to upload a file to an S3 bucket. For information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload.

> **Note**
>
> When you use the AWS SDK for .NET API to upload large objects, a timeout might occur while data is being written to the request stream. You can set an explicit timeout using the `UploadPartRequest`.

The following C# example uploads a file to an S3 bucket using the low-level multipart upload API. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
       // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
```

```
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                        {
                            BucketName = bucketName,
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
                            FilePath = filePath
                        };

                    // Track upload progress.
                    uploadRequest.StreamTransferProgress +=
                        new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                    // Upload a part and add the response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Setup to complete the upload.
```

```
                CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        UploadId = initResponse.UploadId
                     };
                completeRequest.AddPartETags(uploadResponses);

                // Complete the upload.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
 exception.Message);

                // Abort the upload.
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    UploadId = initResponse.UploadId
                };
               await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

PHP

This topic shows how to use the low-level `uploadPart` method from version 3 of the AWS SDK for PHP to upload a file in multiple parts. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example uploads a file to an Amazon S3 bucket using the low-level PHP API multipart upload.

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
```

```
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part $partNumber of $filename." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of $filename failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded $filename to $url." . PHP_EOL;
```

### Using the AWS SDK for Ruby

The AWS SDK for Ruby version 3 supports Amazon S3 multipart uploads in two ways. For the first option, you can use managed file uploads. For more information, see Uploading Files to Amazon S3 in the *AWS Developer Blog*. Managed file uploads are the recommended method for uploading files to a bucket. They provide the following benefits:

- Manage multipart uploads for objects larger than 15MB.
- Correctly open files in binary mode to avoid encoding issues.
- Use multiple threads for uploading parts of large objects in parallel.

Alternatively, you can use the following multipart upload client operations directly:

- create_multipart_upload – Initiates a multipart upload and returns an upload ID.
- upload_part – Uploads a part in a multipart upload.
- upload_part_copy – Uploads a part by copying data from an existing object as data source.
- complete_multipart_upload – Completes a multipart upload by assembling previously uploaded parts.
- abort_multipart_upload – Stops a multipart upload.

**Using the REST API**

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload.

- Initiate Multipart Upload
- Upload Part
- Complete Multipart Upload
- Stop Multipart Upload
- List Parts
- List Multipart Uploads

**Using the AWS CLI**

The following sections in the AWS Command Line Interface (AWS CLI) describe the operations for multipart upload.

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

You can also use the REST API to make your own REST requests, or you can use one of the AWS SDKs. For more information about the REST API, see Using the REST API. For more information about the SDKs, see Uploading an object using multipart upload.

## Uploading a directory using the high-level .NET TransferUtility class

You can use the `TransferUtility` class to upload an entire directory. By default, the API uploads only the files at the root of the specified directory. You can, however, specify recursively uploading files in all of the subdirectories.

To select files in the specified directory based on filtering criteria, specify filtering expressions. For example, to upload only the .pdf files from a directory, specify the `"*.pdf"` filter expression.

When uploading files from a directory, you don't specify the key names for the resulting objects. Amazon S3 constructs the key names using the original file path. For example, assume that you have a directory called `c:\myfolder` with the following structure:

**Example**

```
C:\myfolder
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

When you upload this directory, Amazon S3 uses the following key names:

**Example**

```
a.txt
b.pdf
media/An.mp3
```

**Example**

The following C# example uploads a directory to an Amazon S3 bucket. It shows how to use various `TransferUtility.UploadDirectory` overloads to upload the directory. Each successive call to upload replaces the previous upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");

                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                                directoryPath,
                                                existingBucketName,
                                                wildCard,
                                                SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");
```

```
                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object", e.Message);
            }
        }
    }
}
```

## Listing multipart uploads

You can use the AWS SDKs (low-level API) to retrieve a list of in-progress multipart uploads in Amazon S3.

### Listing multipart uploads using the AWS SDK (low-level API)

Java

The following tasks guide you through using the low-level Java classes to list all in-progress multipart uploads on a bucket.

**Low-level API multipart uploads listing process**

| | |
|---|---|
| 1 | Create an instance of the `ListMultipartUploadsRequest` class and provide the bucket name. |
| 2 | Run the `AmazonS3Client.listMultipartUploads` method. The method returns an instance of the `MultipartUploadListing` class that gives you information about the multipart uploads in progress. |

The following Java code example demonstrates the preceding tasks.

**Example**

```
ListMultipartUploadsRequest allMultpartUploadsRequest =
     new ListMultipartUploadsRequest(existingBucketName);
MultipartUploadListing multipartUploadListing =
     s3Client.listMultipartUploads(allMultpartUploadsRequest);
```

.NET

To list all of the in-progress multipart uploads on a specific bucket, use the AWS SDK for .NET low-level multipart upload API's `ListMultipartUploadsRequest` class. The `AmazonS3Client.ListMultipartUploads` method returns an instance of the `ListMultipartUploadsResponse` class that provides information about the in-progress multipart uploads.

An in-progress multipart upload is a multipart upload that has been initiated using the initiate multipart upload request, but has not yet been completed or stopped. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload.

The following C# example shows how to use the AWS SDK for .NET to list all in-progress multipart uploads on a bucket. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
```

```
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

This topic shows how to use the low-level API classes from version 3 of the AWS SDK for PHP to list all in-progress multipart uploads on a bucket. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example demonstrates listing all in-progress multipart uploads on a bucket.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

### Listing multipart uploads using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for listing multipart uploads:

- ListParts-list the uploaded parts for a specific multipart upload.
- ListMultipartUploads-list in-progress multipart uploads.

### Listing multipart uploads using the AWS CLI

The following sections in the AWS Command Line Interface describe the operations for listing multipart uploads.

- list-parts-list the uploaded parts for a specific multipart upload.
- list-multipart-uploads-list in-progress multipart uploads.

## Tracking a multipart upload

The high-level multipart upload API provides a listen interface, `ProgressListener`, to track the upload progress when uploading an object to Amazon S3. Progress events occur periodically and notify the listener that bytes have been transferred.

Java

**Example**

```
TransferManager tm = new TransferManager(new ProfileCredentialsProvider());

PutObjectRequest request = new PutObjectRequest(
    existingBucketName, keyName, new File(filePath));

// Subscribe to the event and provide event handler.
request.setProgressListener(new ProgressListener() {
   public void progressChanged(ProgressEvent event) {
    System.out.println("Transferred bytes: " +
      event.getBytesTransfered());
            }
});
```

**Example**

The following Java code uploads a file and uses the `ProgressListener` to track the upload progress. For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import java.io.File;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.Upload;

public class TrackMPUProgressUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName            = "*** Provide object key ***";
        String filePath           = "*** file to upload ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = new PutObjectRequest(
          existingBucketName, keyName, new File(filePath));

        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        request.setGeneralProgressListener(new ProgressListener() {
   @Override
   public void progressChanged(ProgressEvent progressEvent) {
    System.out.println("Transferred bytes: " +
      progressEvent.getBytesTransferred());
   }
  });

        // TransferManager processes all transfers asynchronously,
        // so this call will return immediately.
        Upload upload = tm.upload(request);

        try {
         // You can block and wait for the upload to finish
         upload.waitForCompletion();
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload aborted.");
         amazonClientException.printStackTrace();
        }
```

```
    }
}
```

.NET

The following C# example uploads a file to an S3 bucket using the `TransferUtility` class, and tracks the progress of the upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object
 ***";
        private const string filePath = " *** provide the full path name of the file
 to upload **";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);
```

```
                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>
                        (uploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }

        static void uploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

## Aborting a multipart upload

After you initiate a multipart upload, you begin uploading parts. Amazon S3 stores these parts, but it creates the object from the parts only after you upload all of them and send a `successful`

request to complete the multipart upload (you should verify that your request to complete multipart upload is successful). Upon receiving the complete multipart upload request, Amazon S3 assembles the parts and creates an object. If you don't send the complete multipart upload request successfully, Amazon S3 does not assemble the parts and does not create any object.

You are billed for all storage associated with uploaded parts. For more information, see Multipart upload and pricing. So it's important that you either complete the multipart upload to have the object created or stop the multipart upload to remove any uploaded parts.

You can stop an in-progress multipart upload in Amazon S3 using the AWS Command Line Interface (AWS CLI), REST API, or AWS SDKs. You can also stop an incomplete multipart upload using a bucket lifecycle configuration.

**Using the AWS SDKs (high-level API)**

Java

The `TransferManager` class provides the `abortMultipartUploads` method to stop multipart uploads in progress. An upload is considered to be in progress after you initiate it and until you complete it or stop it. You provide a `Date` value, and this API stops all the multipart uploads on that bucket that were initiated before the specified `Date` and are still in progress.

The following tasks guide you through using the high-level Java classes to stop multipart uploads.

High-level API multipart uploads stopping process

| | |
|---|---|
| 1 | Create an instance of the `TransferManager` class. |
| 2 | Run the `TransferManager.abortMultipartUploads` method by passing the bucket name and a `Date` value. |

The following Java code stops all multipart uploads in progress that were initiated on a specific bucket over a week ago. For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import java.util.Date;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.services.s3.transfer.TransferManager;

public class AbortMPUUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide existing bucket name ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        int sevenDays = 1000 * 60 * 60 * 24 * 7;
  Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);

        try {
         tm.abortMultipartUploads(existingBucketName, oneWeekAgo);
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload was aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

> **Note**
>
> You can also stop a specific multipart upload. For more information, see Using the AWS SDKs (low-level API).

.NET

The following C# example stops all in-progress multipart uploads that were initiated on a specific bucket over a week ago. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
```

```
{
    class AbortMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified
 date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

> **Note**
>
> You can also stop a specific multipart upload. For more information, see Using the AWS SDKs (low-level API).

### Using the AWS SDKs (low-level API)

You can stop an in-progress multipart upload by calling the `AmazonS3.abortMultipartUpload` method. This method deletes any parts that were uploaded to Amazon S3 and frees up the resources. You must provide the upload ID, bucket name, and key name. The following Java code example demonstrates how to stop an in-progress multipart upload.

To stop a multipart upload, you provide the upload ID, and the bucket and key names that are used in the upload. After you have stopped a multipart upload, you can't use the upload ID to upload additional parts. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload.

Java

The following Java code example stops an in-progress multipart upload.

**Example**

```
InitiateMultipartUploadRequest initRequest =
    new InitiateMultipartUploadRequest(existingBucketName, keyName);
InitiateMultipartUploadResult initResponse =
               s3Client.initiateMultipartUpload(initRequest);

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(
            existingBucketName, keyName, initResponse.getUploadId()));
```

> **Note**
>
> Instead of a specific multipart upload, you can stop all your multipart uploads initiated before a specific time that are still in progress. This clean-up operation is useful to stop old multipart uploads that you initiated but did not complete or stop. For more information, see Using the AWS SDKs (high-level API).

.NET

The following C# example shows how to stop a multipart upload. For a complete C# sample that includes the following code, see Using the AWS SDKs (low-level API).

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

You can also abort all in-progress multipart uploads that were initiated prior to a specific time. This clean-up operation is useful for aborting multipart uploads that didn't complete or were aborted. For more information, see Using the AWS SDKs (high-level API).

PHP

This example shows how to use a class from version 3 of the AWS SDK for PHP to abort a multipart upload that is in progress. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2. The example the `abortMultipartUpload()` method.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
```

```
    'UploadId' => $uploadId,
]);
```

### Using the REST API

For more information about using the REST API to stop a multipart upload, see AbortMultipartUpload in the *Amazon Simple Storage Service API Reference*.

### Using the AWS CLI

For more information about using the AWS CLI to stop a multipart upload, see abort-multipart-upload in the *AWS CLI Command Reference*.

## Copying an object using multipart upload

The examples in this section show you how to copy objects greater than 5 GB using the multipart upload API. You can copy objects less than 5 GB in a single operation. For more information, see Copying, moving, and renaming objects.

### Using the AWS SDKs

To copy an object using the low-level API, do the following:

- Initiate a multipart upload by calling the `AmazonS3Client.initiateMultipartUpload()` method.
- Save the upload ID from the response object that the `AmazonS3Client.initiateMultipartUpload()` method returns. You provide this upload ID for each part-upload operation.
- Copy all of the parts. For each part that you need to copy, create a new instance of the `CopyPartRequest` class. Provide the part information, including the source and destination bucket names, source and destination object keys, upload ID, locations of the first and last bytes of the part, and part number.
- Save the responses of the `AmazonS3Client.copyPart()` method calls. Each response includes the `ETag` value and part number for the uploaded part. You need this information to complete the multipart upload.
- Call the `AmazonS3Client.completeMultipartUpload()` method to complete the copy operation.

Java

### Example

The following example shows how to use the Amazon S3 low-level Java API to perform a multipart copy. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(destBucketName,
                    destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);
```

```
            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
            ObjectMetadata metadataResult =
s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make
sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(sourceBucketName)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(destBucketName)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and
make the
            // copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    destBucketName,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

.NET

The following C# example shows how to use the AWS SDK for .NET to copy an Amazon S3 object that is larger than 5 GB from one source location to another, such as from one bucket to another. To copy objects that are smaller than 5 GB, use the single-operation copy procedure described in Using the AWS SDKs. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload.

This example shows how to copy an Amazon S3 object that is larger than 5 GB from one S3 bucket to another using the AWS SDK for .NET multipart upload API.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
```

```
   {
       private const string sourceBucket = "*** provide the name of the bucket with
source object ***";
       private const string targetBucket = "*** provide the name of the bucket to
copy the object to ***";
       private const string sourceObjectKey = "*** provide the name of object to
copy ***";
       private const string targetObjectKey = "*** provide the name of the object
copy ***";
       // Specify your bucket region (an example region is shown).
       private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
       private static IAmazonS3 s3Client;

       public static void Main()
       {
           s3Client = new AmazonS3Client(bucketRegion);
           Console.WriteLine("Copying an object");
           MPUCopyObjectAsync().Wait();
       }
       private static async Task MPUCopyObjectAsync()
       {
           // Create a list to store the upload part responses.
           List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();
           List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

           // Setup information required to initiate the multipart upload.
           InitiateMultipartUploadRequest initiateRequest =
               new InitiateMultipartUploadRequest
               {
                   BucketName = targetBucket,
                   Key = targetObjectKey
               };

           // Initiate the upload.
           InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

           // Save the upload ID.
           String uploadId = initResponse.UploadId;

           try
           {
```

```
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
                long objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    CopyPartRequest copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = targetBucket,
                        DestinationKey = targetObjectKey,
                        SourceBucket = sourceBucket,
                        SourceKey = sourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i
                    };

                    copyResponses.Add(await s3Client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                CompleteMultipartUploadRequest completeRequest =
                new CompleteMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey,
                    UploadId = initResponse.UploadId
```

```
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

### Using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload. For copying an existing object, use the Upload Part (Copy) API and specify the source object by adding the `x-amz-copy-source` request header in your request.

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

You can use these APIs to make your own REST requests, or you can use one of the SDKs we provide. For more information about using Multipart Upload with the AWS CLI, see Using the AWS CLI. For more information about the SDKs, see AWS SDK support for multipart upload.

## Amazon S3 multipart upload limits

The following table provides multipart upload core specifications. For more information, see Uploading and copying objects using multipart upload.

| Item | Specification |
| --- | --- |
| Maximum object size | 5 TiB |
| Maximum number of parts per upload | 10,000 |
| Part numbers | 1 to 10,000 (inclusive) |
| Part size | 5 MiB to 5 GiB. There is no minimum size limit on the last part of your multipart upload. |
| Maximum number of parts returned for a list parts request | 1000 |
| Maximum number of multipart uploads returned in a list multipart uploads request | 1000 |

# Copying, moving, and renaming objects

The `CopyObject` operation creates a copy of an object that is already stored in Amazon S3.

You can create a copy of an object up to 5 GB in a single atomic operation. However, to copy an object that is larger than 5 GB, you must use a multipart upload. For more information, see the section called "Copying an object".

Using the `CopyObject` operation, you can:

- Create additional copies of objects.
- Rename objects by copying them and deleting the original ones.

- Copy or move objects from one bucket to another, including across AWS Regions (for example, from `us-west-1` to `eu-west-2`). When you move an object, Amazon S3 copies the object to the specified destination and then deletes the source object.

  > **Note**
  >
  > Copying or moving objects across AWS Regions incurs bandwidth charges. For more information, see Amazon S3 Pricing.

- Change object metadata. Each Amazon S3 object has metadata. This metadata is a set of name-value pairs. You can set object metadata at the time you upload an object. After you upload the object, you cannot modify the object metadata. The only way to modify object metadata is to make a copy of the object and set the metadata. To do so, in the copy operation, set the same object as the source and target.

  Some object metadata is system metadata and other is user-defined. You can control some of the system metadata. For example, you can control the storage class and the type of server-side encryption to use for the object. When you copy an object, user-controlled system metadata and user-defined metadata are also copied. Amazon S3 resets the system-controlled metadata. For example, when you copy an object, Amazon S3 resets the creation date of the copied object. You don't need to set any of these system-controlled metadata values in your copy request.

  When copying an object, you might decide to update some of the metadata values. For example, if your source object is configured to use S3 Standard storage, you might choose to use S3 Intelligent-Tiering for the object copy. You might also decide to alter some of the user-defined metadata values present on the source object. If you choose to update any of the object's user-configurable metadata (system or user-defined) during the copy, then you must explicitly specify all of the user-configurable metadata present on the source object in your request, even if you are changing only one of the metadata values.

  For more information about the object metadata, see Working with object metadata.

### Copying archived and restored objects

If the source object is archived in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, you must first restore a temporary copy before you can copy the object to another bucket. For information about archiving objects, see Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival).

The **Copy** operation in the Amazon S3 console isn't supported for restored objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. To copy these restored objects, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

### Copying encrypted objects

Amazon S3 automatically encrypts all new objects that are copied to an S3 bucket. If you don't specify encryption information in your copy request, the encryption setting of the target object is set to the default encryption configuration of the destination bucket. By default, all buckets have a base level of encryption configuration that uses server-side encryption with Amazon S3 managed keys (SSE-S3). If the destination bucket has a default encryption configuration that uses server-side encryption with an AWS Key Management Service (AWS KMS) key (SSE-KMS), or a customer-provided encryption key (SSE-C), Amazon S3 uses the corresponding KMS key, or a customer-provided key to encrypt the target object copy.

When copying an object, if you want to use a different type of encryption setting for the target object, you can request that Amazon S3 encrypt the target object with a KMS key, an Amazon S3 managed key, or a customer-provided key. If the encryption setting in your request is different from the default encryption configuration of the destination bucket, the encryption setting in your request takes precedence. If the source object for the copy is encrypted with SSE-C, you must provide the necessary encryption information in your request so that Amazon S3 can decrypt the object for copying. For more information, see Protecting data with encryption.

### Using checksums when copying objects

When copying objects, you can choose to use a different checksum algorithm for the object. Whether you choose to use the same algorithm or a new one, Amazon S3 calculates a new checksum value after the object is copied. Amazon S3 does not directly copy the value of the checksum. The checksum value of objects that were loaded by using multipart uploads might change. For more information about how the checksum is calculated, see Using part-level checksums for multipart uploads.

### Copying multiple objects in a single request

To copy more than one Amazon S3 object with a single request, you can also use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

The S3 Batch Operations feature tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or REST API. For more information, see the section called "Batch Operations basics".

### Copying objects to directory buckets

For information about copying an object to a directory bucket, see Copying an object to a directory bucket. For information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

## To copy an object

To copy an object, use the following methods.

### Using the S3 console

**Note**

- When copying an object by using the Amazon S3 console, you must have the `s3:ListAllMyBuckets` permission. The console needs this permission to validate the **Copy** operation. For example policies that grant this permission, see the section called "Identity-based policy examples".

  If you're copying an object that has user-defined tags, you must also have the `s3:GetObjectTagging` permission. If you're copying an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the s3:`GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, the object will be copied without the user-defined tags, and you will receive an error.
- Objects encrypted with customer-provided encryption keys (SSE-C) cannot be copied by using the S3 console. To copy objects encrypted with SSE-C, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.
- Cross-Region copying of objects encrypted with SSE-KMS is not supported by the Amazon S3 console. To copy objects encrypted with SSE-KMS across Regions, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.

**To copy an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects that you want to copy.
3. Select the check box to the left of the names of the objects that you want to copy.
4. On the **Actions** menu, choose **Copy** from the list of options that appears.
5. Select the destination type and destination account. To specify the destination path, choose **Browse S3**, navigate to the destination, and select the check box to the left of the destination. Choose **Choose destination** in the lower-right corner.

   Alternatively, enter the destination path.
6. If you do *not* have bucket versioning enabled, you might be asked to acknowledge that existing objects with the same name are overwritten. If this is OK, select the check box and proceed. If you want to keep all versions of objects in this bucket, select **Enable Bucket Versioning**. You can also update the default encryption and S3 Object Lock properties.
7. Under **Additional checksums**, choose whether you want to copy the objects using the existing checksum function or replace the existing checksum function with a new one. When you uploaded the objects, you had the option to specify the checksum algorithm that was used to verify data integrity. When copying the object, you have the option to choose a new function. If you did not originally specify an additional checksum, you can use this section of the copy options to add one.

   > **Note**
   >
   > Even if you opt to use the same checksum function, your checksum value might change if you copy the object and it is over 16 MB in size. The checksum value might change because of how checksums are calculated for multipart uploads. For more information about how the checksum might change when copying the object, see Using part-level checksums for multipart uploads.

To change the checksum function, choose **Replace with a new checksum function**. Choose the new checksum function from the box. When the object is copied over, the new checksum is calculated and stored using the specified algorithm.

8. Choose **Copy** in the bottom-right corner. Amazon S3 copies your objects to the destination.

### Using the AWS SDKs

The examples in this section show how to copy objects up to 5 GB in a single operation. To copy objects larger than 5 GB, you must use a multipart upload. For more information, see Copying an object using multipart upload.

Java

#### Example

The following example copies an object in Amazon S3 using the AWS SDK for Java. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

import java.io.IOException;

public class CopyObjectSingleOperation {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key *** ";
        String destinationKey = "*** Destination object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName,
 sourceKey, bucketName, destinationKey);
            s3Client.copyObject(copyObjRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following C# example uses the high-level AWS SDK for .NET to copy objects that are as large as 5 GB in a single operation. For objects that are larger than 5 GB, use the multipart upload copy example described in Copying an object using multipart upload.

This example makes a copy of an object that is a maximum of 5 GB. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
```

```
        private const string destinationBucket = "*** provide the name of the bucket
to copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy
***";
        private const string destObjectKey = "*** provide the destination object key
name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await
s3Client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
    }
```

```
}
```

PHP

This topic guides you through using classes from version 3 of the AWS SDK for PHP to copy a single object and multiple objects within Amazon S3, from one bucket to another or within the same bucket.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example illustrates the use of the `copyObject()` method to copy a single object within Amazon S3. It also demonstrates how to make multiple copies of an object by using a batch of calls to `CopyObject` with the `getcommand()` method.

**Copying objects**

| | |
|---|---|
| 1 | Create an instance of an Amazon S3 client by using the `Aws\S3\S3Client` class constructor. |
| 2 | To make multiple copies of an object, you run a batch of calls to the Amazon S3 client getCommand() method, which is inherited from the Aws\CommandInterface class. You provide the `CopyObject` command as the first argument and an array containing the source bucket, source key name, target bucket, and target key name as the second argument. |

```
 require 'vendor/autoload.php';

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region' => 'us-east-1'
]);
```

```
// Copy an object.
$s3->copyObject([
    'Bucket' => $targetBucket,
    'Key' => "$sourceKeyname-copy",
    'CopySource' => "$sourceBucket/$sourceKeyname",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket' => $targetBucket,
        'Key' => "{targetKeyname}-$i",
        'CopySource' => "$sourceBucket/$sourceKeyname",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach ($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
    }
} catch (Exception $e) {
    // General error handling here
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
```

```
        self.object = s3_object
        self.key = self.object.key
```

```
   def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Ruby

The following tasks guide you through using the Ruby classes to copy an object in Amazon S3 from one bucket to another or within the same bucket.

### Copying objects

| | |
|---|---|
| 1 | Use the Amazon S3 modularized gem for version 3 of the AWS SDK for Ruby, require `aws-sdk-s3` , and provide your AWS credentials. For more information about how to provide your credentials, see Making requests using AWS account or IAM user credentials. |
| 2 | Provide the request information, such as the source bucket name, source key name, destination bucket name, and destination key. |

The following Ruby code example demonstrates the preceding tasks by using the `#copy_object` method to copy an object from one bucket to another.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
```

```
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

### Using the REST API

This example describes how to copy an object by using the Amazon S3 REST API. For more information about the REST API, see CopyObject.

This example copies the `flotsam` object from the *amzn-s3-demo-bucket1* bucket to the `jetsam` object of the *amzn-s3-demo-bucket2* bucket, preserving its metadata.

```
PUT /jetsam HTTP/1.1
Host: amzn-s3-demo-bucket2.s3.amazonaws.com
x-amz-copy-source: /amzn-s3-demo-bucket1/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

The signature was generated from the following information.

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n
```

```
x-amz-copy-source:/amzn-s3-demo-bucket1/flotsam\r\n
/amzn-s3-demo-bucket2/jetsam
```

Amazon S3 returns the following response that specifies the ETag of the object and when it was last modified.

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

### Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) to copy an S3 object. For more information, see copy-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

## To move an object

To move an object, use the following methods.

### Using the S3 console

**Note**

- If you're moving an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're moving an object that doesn't have user-

defined tags but is over 16 MB in size, you must also have the s3:GetObjectTagging permission.

If the destination bucket policy denies the s3:GetObjectTagging action, the object will be moved without the user-defined tags, and you will receive an error.

- Objects encrypted with customer-provided encryption keys (SSE-C) cannot be moved by using the Amazon S3 console. To move objects encrypted with SSE-C, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API.
- When moving folders, wait for the **Move** operation to finish before making additional changes in the folders.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.

**To move an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects that you want to move.
3. Select the check box to the left of the names of the objects that you want to move.
4. On the **Actions** menu, choose **Move**.
5. To specify the destination path, choose **Browse S3**, navigate to the destination, and select the check box to the left of the destination. Choose **Choose destination** in the lower-right corner.

   Alternatively, enter the destination path.
6. If you do *not* have bucket versioning enabled, you might be asked to acknowledge that existing objects with the same name are overwritten. If this is OK, select the check box and proceed. If you want to keep all versions of objects in this bucket, select **Enable Bucket Versioning**. You can also update the default encryption and Object Lock properties.
7. Choose **Move** in the bottom-right corner. Amazon S3 moves your objects to the destination.

**Note**

- This action creates a copy of all specified objects with updated settings, updates the last-modified date in the specified location, and adds a delete marker to the original object.
- This action updates metadata for bucket versioning, encryption, Object Lock features, and archived objects.

#### Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) to move an S3 object. For more information, see mv in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

## To rename an object

To rename an object, use the following procedure.

**Note**

- Renaming an object creates a copy of the object with a new last-modified date, and then adds a delete marker to the original object.
- Bucket settings for default encryption are automatically applied to any specified object that is unencrypted.
- You can't use the Amazon S3 console to rename objects with customer-provided encryption keys (SSE-C) . To rename objects encrypted with SSE-C, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API to copy those objects with new names.
- If this bucket uses the bucket owner enforced setting for S3 Object Ownership, object access control lists (ACLs) won't be copied.
- If you're renaming an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're renaming an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the s3:`GetObjectTagging` permission.

If the destination bucket policy denies the `s3:GetObjectTagging` action, the object will be renamed, but the user-defined tags will be removed from the object, and you will receive an error.

**To rename an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the object that you want to rename.
3. Select the check box to the left of the name of the object that you want to rename.
4. On the **Actions** menu, choose **Rename object**.
5. In the **New object name** box, enter the new name for the object.
6. Choose **Save changes** in the bottom-right corner. Amazon S3 renames your object.

# Downloading objects

This section explains how to download objects from an Amazon S3 bucket. With Amazon S3, you can store objects in one or more buckets, and each single object can be up to 5 TB in size. Any Amazon S3 object that is not archived is accessible in real time. Archived objects, however, must be restored before they can be downloaded. For information about downloading archived objects, see the section called "Downloading archived objects".

You can download a single object by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. To download an object from S3 without writing any code or running any commands, use the S3 console. For more information, see the section called "Downloading an object".

To download multiple objects, use AWS CloudShell, the AWS CLI, or the AWS SDKs. For more information, see the section called "Downloading multiple objects".

If you need to download part of an object, you use extra parameters with the AWS CLI or REST API to specify only the bytes that you want to download. For more information, see the section called "Downloading part of an object".

If you need to download an object that you don't own, ask the object owner to generate a presigned URL that allows you to download the object. For more information, see the section called “Downloading an object from another AWS account”.

When you download objects outside of the AWS network, data-transfer fees apply. Data transfer within the AWS network is free within the same AWS Region, but you will be charged for any GET requests. For more information about data-transfer costs and data-retrieval charges, see Amazon S3 pricing.

**Topics**



## Downloading an object

You can download an object by using the Amazon S3 console, AWS CLI, AWS SDKs, or REST API.

### Using the S3 console

This section explains how to use the Amazon S3 console to download an object from an S3 bucket.

> **Note**
>
> - You can download only one object at a time.
> - If you use the Amazon S3 console to download an object whose key name ends with a period (`.`), the period is removed from the key name of the downloaded object. To retain the period at the end of the name of the downloaded object, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

**To download an object from an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to download an object from.
3. You can download an object from an S3 bucket in any of the following ways:
   - Select the check box next to the object, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.
   - If you want to download a specific version of the object, turn on **Show versions** (located next to the search box). Select the check box next to the version of the object that you want, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.

### Using the AWS CLI

The following `get-object` example command shows how you can use the AWS CLI to download an object from Amazon S3. This command gets the object *`folder/my_image`* from the bucket *`amzn-s3-demo-bucket1`*. The object will be downloaded to a file named *`my_downloaded_image`*.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/
my_image my_downloaded_image
```

For more information and examples, see get-object in the *AWS CLI Command Reference*.

### Using the AWS SDKs

For examples of how to download an object with the AWS SDKs, see Use `GetObject` with an AWS SDK or CLI.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

### Using the REST API

You can use the REST API to retrieve objects from Amazon S3. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

# Downloading multiple objects

You can download multiple objects by using AWS CloudShell, the AWS CLI, or the AWS SDKs.

### Using AWS CloudShell in the AWS Management Console

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console.

For more information about AWS CloudShell, see What is CloudShell? in the *AWS CloudShell User Guide*.

> **Important**
>
> With AWS CloudShell, your home directory has storage up to 1GB per AWS Region. Therefore you cannot sync buckets with objects totaling over this amount. For more limitations, see Service quotas and restrictions in the *AWS CloudShell User Guide*.

### To download objects by using AWS CloudShell

1. Sign in to the AWS Management Console and open the CloudShell console at https://console.aws.amazon.com/cloudshell/.
2. Run the following command to sync objects in your bucket to CloudShell. The following command syncs objects from the bucket named *amzn-s3-demo-bucket1* and creates a folder named *temp* in CloudShell. CloudShell syncs your objects to this folder. To use this command, replace the *user input placeholders* with your own information.

   ```
   aws s3 sync s3://amzn-s3-demo-bucket1 ./temp
   ```

   > **Note**
   >
   > To perform pattern matching to either exclude or include particular objects, you can use the `--exclude "value"` and `--include "value"` parameters with the `sync` command.

3. Run the following command to zip your objects in the folder named *temp* to a file named *temp.zip*.

```
zip temp.zip -r temp/
```

4. Choose **Actions**, and then choose **Download file**.
5. Enter the file name **`temp.zip`** and then choose **Download**.
6. (Optional) Delete the *`temp.zip`* file and the objects that are synced to the *`temp`* folder in CloudShell. With AWS CloudShell, you have persistent storage of up to 1 GB for each AWS Region.

   You can use the following example command to delete your `.zip` file and your folder. To use this example command, replace the *`user input placeholders`* with your own information.

   ```
   rm temp.zip && rm -rf temp/
   ```

### Using the AWS CLI

The following example shows how you can use the AWS CLI to download all of the files or objects under the specified directory or prefix. This command copies all objects from the bucket *`amzn-s3-demo-bucket1`* to your current directory. To use this example command, use your bucket name in place of *`amzn-s3-demo-bucket1`*.

```
aws s3 cp s3://amzn-s3-demo-bucket1 . --recursive
```

The following command downloads all of the objects under the prefix *`logs`* in the bucket *`amzn-s3-demo-bucket1`* to your current directory. It also uses the `--exclude` and `--include` parameters to copy only objects with the suffix *`.log`*. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3 cp s3://amzn-s3-demo-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

For more information and examples, see cp in the *AWS CLI Command Reference*.

### Using the AWS SDKs

For examples of how to download all objects in an Amazon S3 bucket with the AWS SDKs, see Download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

# Downloading part of an object

You can download part of an object by using the AWS CLI or REST API. To do so, you use additional parameters to specify which part of an object that you want to download.

### Using the AWS CLI

The following example command performs a `GET` request for a range of bytes in the object named *`folder/my_data`* in the bucket named *`amzn-s3-demo-bucket1`*. In the request, the byte range must be prefixed with `bytes=`. The partial object is downloaded to the output file named *`my_data_range`*. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

For more information and examples, see get-object in the *AWS CLI Command Reference*.

For more information about the HTTP `Range` header, see RFC 9110 on the RFC Editor website.

> **Note**
>
> Amazon S3 doesn't support retrieving multiple ranges of data in a single `GET` request.

### Using the REST API

You can use the `partNumber` and `Range` parameters in the REST API to retrieve object parts from Amazon S3. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

# Downloading an object from another AWS account

You can use a presigned URL to grant others time-limited access to your objects without updating your bucket policy.

The presigned URL can be entered in a browser or used by a program to download an object. The credentials used by the URL are those of the AWS user who generated the URL. After the URL is

created, anyone with the presigned URL can download the corresponding object until the URL expires.

### Using a presigned URL in the S3 console

You can use the Amazon S3 console to generate a presigned URL for sharing an object by following these steps. When using the console, the maximum expiration time for a presigned URL is 12 hours from the time of creation.

**To generate a presigned URL by using the Amazon S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want a presigned URL for.
4. In the **Objects** list, select the object that you want to create a presigned URL for.
5. On the **Object actions** menu, choose **Share with a presigned URL**.
6. Specify how long you want the presigned URL to be valid.
7. Choose **Create presigned URL**.
8. When a confirmation message appears, the URL is automatically copied to your clipboard. You will see a button to copy the presigned URL if you need to copy it again.
9. To download the object, paste the URL into any browser, and the object will attempt to download.

For more information about presigned URLs and other methods for creating them, see Working with presigned URLs.

## Downloading archived objects

To reduce your storage costs for infrequently accessed objects, you can *archive* those objects. When you archive an object, it is moved into low-cost storage, which means that you can't access it in real time. To download an archived object, you must first restore it.

You can restore archived objects in minutes or hours, depending on the storage class. You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the Amazon S3 REST API, the AWS SDKs, and the AWS Command Line Interface (AWS CLI).

For instructions, see Restoring an archived object. After you restore the archived object, you can download it.

### Troubleshooting downloading objects

Insufficient permissions or incorrect bucket or AWS Identity and Access Management (IAM) user policies can cause errors when you're trying to download objects from Amazon S3. These problems can often cause Access Denied (403 Forbidden) errors, where Amazon S3 is unable to allow access to a resource.

For common causes of Access Denied (403 Forbidden) errors, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

## Checking object integrity

Amazon S3 uses checksum values to verify the integrity of data that you upload to or download from Amazon S3. In addition, you can request that another checksum value be calculated for any object that you store in Amazon S3. You can select from one of several checksum algorithms to use when uploading or copying your data. Amazon S3 uses this algorithm to compute an additional checksum value and store it as part of the object metadata. To learn more about how to use additional checksums to verify data integrity, see Tutorial: Checking the integrity of data in Amazon S3 with additional checksums.

When you upload an object, you can optionally include a precalculated checksum as part of your request. Amazon S3 compares the provided checksum to the checksum that it calculates by using your specified algorithm. If the two values don't match, Amazon S3 reports an error.

### Using supported checksum algorithms

Amazon S3 offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms:

- CRC32
- CRC32C
- SHA-1
- SHA-256

When you upload an object, you can specify the algorithm that you want to use:

- **When you're using the AWS Management Console**, you select the checksum algorithm that you want to use. When you do, you can optionally specify the checksum value of the object. When Amazon S3 receives the object, it calculates the checksum by using the algorithm that you specified. If the two checksum values don't match, Amazon S3 generates an error.
- **When you're using an SDK**, you can set the value of the `x-amz-sdk-checksum-algorithm` parameter to the algorithm that you want Amazon S3 to use when calculating the checksum. Amazon S3 automatically calculates the checksum value.
- **When you're using the REST API**, you don't use the `x-amz-sdk-checksum-algorithm` parameter. Instead, you use one of the algorithm-specific headers (for example, `x-amz-checksum-crc32`).

For more information about uploading objects, see Uploading objects.

To apply any of these checksum values to objects that are already uploaded to Amazon S3, you can copy the object. When you copy an object, you can specify whether you want to use the existing checksum algorithm or use a new one. You can specify a checksum algorithm when using any supported mechanism for copying objects, including S3 Batch Operations. For more information about S3 Batch Operations, see Performing large-scale batch operations on Amazon S3 objects.

> **Important**
>
> If you're using a multipart upload with additional checksums, the multipart part numbers must use consecutive part numbers. When using additional checksums, if you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an HTTP `500 Internal Server Error` error.

After uploading objects, you can get the checksum value and compare it to a precalculated or previously stored checksum value calculated using the same algorithm.

**Using the S3 console**

To learn more about using the console and specifying checksum algorithms to use when uploading objects, see Uploading objects and Tutorial: Checking the integrity of data in Amazon S3 with additional checksums.

### Using the AWS SDKs

The following example shows how you can use the AWS SDKs to upload a large file with multipart upload, download a large file, and validate a multipart upload file, all with using SHA-256 for file validation.

Java

#### Example Example: Uploading, downloading, and verifying a large file with SHA-256

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.Tagging;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.nio.ByteBuffer;
    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.ArrayList;
    import java.util.Base64;
    import java.util.List;

    public class LargeObjectValidation {
        private static String FILE_NAME = "sample.file";
        private static String BUCKET = "sample-bucket";
        //Optional, if you want a method of storing the full multipart object
 checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
```

```
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
```

```
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.

s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
```

```
                        break;
                    }
                }
                MessageDigest sha256FullObject =
 MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
 MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
 enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
 objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                    ResponseInputStream<GetObjectResponse> response =
 s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
 the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
 objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
```

```
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
 {
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
 s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
 ObjectAttributes.CHECKSUM).build());
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
 MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
 objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
```

```
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
 difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
```

```
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    }
```

**Using the REST API**

You can send REST requests to upload an object with a checksum value to verify the integrity of the data with PutObject. You can also retrieve the checksum value for objects using GetObject or HeadObject.

**Using the AWS CLI**

You can send a PUT request to upload an object of up to 5 GB in a single operation. For more information, see the `PutObject` in the *AWS CLI Command Reference*. You can also use `get-object` and `head-object` to retrieve the checksum of an already-uploaded object to verify the integrity of the data.

For information, see Amazon S3 CLI FAQ in the *AWS Command Line Interface User Guide*.

## Using Content-MD5 when uploading objects

Another way to verify the integrity of your object after uploading is to provide an MD5 digest of the object when you upload it. If you calculate the MD5 digest for your object, you can provide the digest with the PUT command by using the `Content-MD5` header.

After uploading the object, Amazon S3 calculates the MD5 digest of the object and compares it to the value that you provided. The request succeeds only if the two digests match.

Supplying an MD5 digest isn't required, but you can use it to verify the integrity of the object as part of the upload process.

## Using Content-MD5 and the ETag to verify uploaded objects

The entity tag (ETag) for an object represents a specific version of that object. Keep in mind that the ETag reflects changes only to the content of an object, not to its metadata. If only the metadata of an object changes, the ETag remains the same.

Depending on the object, the ETag of the object might be an MD5 digest of the object data:

- If an object is created by the `PutObject`, `PostObject`, or `CopyObject` operation, or through the AWS Management Console, and that object is also plaintext or encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3), that object has an ETag that is an MD5 digest of its object data.
- If an object is created by the `PutObject`, `PostObject`, or `CopyObject` operation, or through the AWS Management Console, and that object is encrypted by server-side encryption with customer-provided keys (SSE-C) or server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), that object has an ETag that is not an MD5 digest of its object data.
- If an object is created by either the `Multipart Upload` or `Part Copy` operation, the object's ETag is not an MD5 digest, regardless of the method of encryption. If an object is larger than 16 MB, the AWS Management Console uploads or copies that object as a multipart upload, and therefore the ETag isn't an MD5 digest.

For objects where the ETag is the `Content-MD5` digest of the object, you can compare the ETag value of the object with a calculated or previously stored `Content-MD5` digest.

## Using trailing checksums

When uploading objects to Amazon S3, you can either provide a precalculated checksum for the object or use an AWS SDK to automatically create trailing checksums on your behalf. If you decide to use a trailing checksum, Amazon S3 automatically generates the checksum by using your specified algorithm and uses it to validate the integrity of the object during upload.

To create a trailing checksum when using an AWS SDK, populate the `ChecksumAlgorithm` parameter with your preferred algorithm. The SDK uses that algorithm to calculate the checksum for your object (or object parts) and automatically appends it to the end of your upload request.

This behavior saves you time because Amazon S3 performs both the verification and upload of your data in a single pass.

> **Important**
>
> If you're using S3 Object Lambda, all requests to S3 Object Lambda are signed using `s3-object-lambda` instead of `s3`. This behavior affects the signature of trailing checksum values. For more information about S3 Object Lambda, see Transforming objects with S3 Object Lambda.

## Using part-level checksums for multipart uploads

When objects are uploaded to Amazon S3, they can either be uploaded as a single object or through the multipart upload process. Objects that are larger than 16 MB and uploaded through the console are automatically uploaded using multipart uploads. For more information about multipart uploads, see Uploading and copying objects using multipart upload.

When an object is uploaded as a multipart upload, the ETag for the object is not an MD5 digest of the entire object. Amazon S3 calculates the MD5 digest of each individual part as it is uploaded. The MD5 digests are used to determine the ETag for the final object. Amazon S3 concatenates the bytes for the MD5 digests together and then calculates the MD5 digest of these concatenated values. The final step for creating the ETag is when Amazon S3 adds a dash with the total number of parts to the end.

For example, consider an object uploaded with a multipart upload that has an ETag of `C9A5A6878D97B48CC965C1E41859F034-14`. In this case, `C9A5A6878D97B48CC965C1E41859F034` is the MD5 digest of all the digests concatenated together. The `-14` indicates that there are 14 parts associated with this object's multipart upload.

If you've enabled additional checksum values for your multipart object, Amazon S3 calculates the checksum for each individual part by using the specified checksum algorithm. The checksum for the completed object is calculated in the same way that Amazon S3 calculates the MD5 digest for the multipart upload. You can use this checksum to verify the integrity of the object.

To retrieve information about the object, including how many parts make up the entire object, you can use the GetObjectAttributes operation. With additional checksums, you can also recover information for each individual part that includes each part's checksum value.

For completed uploads, you can get an individual part's checksum by using the GetObject or HeadObject operations and specifying a part number or byte range that aligns to a single part. If you want to retrieve the checksum values for individual parts of multipart uploads still in progress, you can use ListParts.

Because of how Amazon S3 calculates the checksum for multipart objects, the checksum value for the object might change if you copy it. If you're using an SDK or the REST API and you call CopyObject, Amazon S3 copies any object up to the size limitations of the `CopyObject` API operation. Amazon S3 does this copy as a single action, regardless of whether the object was uploaded in a single request or as part of a multipart upload. With a copy command, the checksum of the object is a direct checksum of the full object. If the object was originally uploaded using a multipart upload, then the checksum value changes even though the data has not.

**Note**

Objects that are larger than the size limitations of the `CopyObject` API operation must use multipart copy commands.

**Important**

When you perform some operations using the AWS Management Console, Amazon S3 uses a multipart upload if the object is greater than 16 MB in size. In this case, the checksum is not a direct checksum of the full object, but rather a calculation based on the checksum values of each individual part.

For example, consider an object 100 MB in size that you uploaded as a single-part direct upload using the REST API. The checksum in this case is a checksum of the entire object. If you later use the console to rename that object, copy it, change the storage class, or edit the metadata, Amazon S3 uses the multipart upload functionality to update the object. As a result, Amazon S3 creates a new checksum value for the object that is calculated based on the checksum values of the individual parts.

**The preceding list of console operations is not a complete list of all the possible actions that you can take in the AWS Management Console that result in Amazon S3 updating the object using the multipart upload functionality.** Keep in mind that whenever you use the console to act on objects over 16 MB in size, the checksum value might not be the checksum of the entire object.

# Deleting Amazon S3 objects

You can delete one or more objects directly from Amazon S3 using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), or REST API. Because all objects in your S3 bucket incur storage costs, you should delete objects that you no longer need. For example, if you're collecting log files, it's a good idea to delete them when they're no longer needed. You can set up a lifecycle rule to automatically delete objects such as log files. For more information, see the section called “Setting lifecycle configuration”.

For information about Amazon S3 features and pricing, see Amazon S3 pricing.

You have the following API options when deleting an object:

- **Delete a single object** – Amazon S3 provides the `DELETE` (`DeleteObject`) API operation that you can use to delete one object in a single HTTP request.
- **Delete multiple objects** – Amazon S3 provides the Multi-Object Delete (`DeleteObjects`) API operation that you can use to delete up to 1,000 objects in a single HTTP request.

When deleting objects from a bucket where versioning is not enabled, you provide only the object key name. However, when deleting objects from a versioning-enabled bucket, you can optionally provide the version ID of the object to delete a specific version of the object.

## Programmatically deleting objects from a versioning-enabled bucket

If your bucket has versioning enabled, multiple versions of the same object can exist in the bucket. When working with versioning-enabled buckets, the delete API operations enable the following options:

- **Specify a non-versioned delete request** – Specify only the object's key, and not the version ID. In this case, Amazon S3 creates a delete marker and returns its version ID in the response. This makes your object disappear from the bucket. For information about object versioning and the delete marker concept, see Using versioning in S3 buckets.
- **Specify a versioned delete request** – Specify both the key and also a version ID. In this case the following two outcomes are possible:
  - If the version ID maps to a specific object version, Amazon S3 deletes the specific version of the object.

- If the version ID maps to the delete marker of that object, Amazon S3 deletes the delete marker. This makes the object reappear in your bucket.

## Deleting objects from an MFA-enabled bucket

When deleting objects from a multi-factor authentication (MFA)-enabled bucket, note the following:

- If you provide an MFA token that isn't valid, the request always fails.
- If you have an MFA-enabled bucket and you make a versioned delete request (you provide an object key and version ID), the request fails if you don't provide a valid MFA token. In addition, when using the Multi-Object Delete API operation on an MFA-enabled bucket, if any of the deletes are a versioned delete request (that is, you specify an object key and version ID), the entire request fails if you don't provide an MFA token.

However, in the following cases, the request succeeds:

- If you have an MFA-enabled bucket and you make a non-versioned delete request (you are not deleting a versioned object), and you don't provide an MFA token, the delete succeeds.
- If you have a Multi-Object Delete request that specifies only non-versioned objects to delete from an MFA-enabled bucket and you don't provide an MFA token, the deletions succeed.

For information about MFA delete, see Configuring MFA delete.

**Topics**



## Deleting a single object

You can use the Amazon S3 console or the DELETE API to delete a single existing object from an S3 bucket. For more information about deleting objects in Amazon S3, see Deleting Amazon S3 objects.

Because all objects in your S3 bucket incur storage costs, you should delete objects that you no longer need. For example, if you are collecting log files, it's a good idea to delete them when

they're no longer needed. You can set up a lifecycle rule to automatically delete objects such as log files. For more information, see the section called “Setting lifecycle configuration”.

For information about Amazon S3 features and pricing, see Amazon S3 pricing.

### Using the S3 console

Follow these steps to use the Amazon S3 console to delete a single object from a bucket.

> **Warning**
>
> When you permanently delete an object or specified object version in the Amazon S3 console, the deletion can't be undone.

### To delete an object that has versioning enabled or suspended

> **Note**
>
> If the version ID for an object in a versioning-suspended bucket is marked as NULL, S3 permanently deletes the object since no previous versions exist. However, if a valid version ID is listed for the object in a versioning-suspended bucket, then S3 creates a delete marker for the deleted object, while retaining the previous versions of the object.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete an object from.
3. Select the object and then choose **Delete**.
4. To confirm deletion of the objects list under **Specified objects** in the **Delete objects?** text box, enter **`delete`**.

**To permanently delete a specific object version in a versioning-enabled bucket**

> **Warning**
>
> When you permanently delete a specific object version in Amazon S3, the deletion can't be undone.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete an object from.
3. Select the object that you want to delete.
4. Choose the **Show versions** toggle.
5. Select the object version and then choose **Delete**.
6. To confirm permanent deletion of the specific object versions listed under **Specified objects**, in the **Delete objects?** text box, enter **Permanently delete**. Amazon S3 permanently deletes the specific object version.

**To permanently delete an object in an Amazon S3 bucket that *doesn't* have versioning enabled**

> **Warning**
>
> When you permanently delete an object in Amazon S3, the deletion can't be undone. Also, for any buckets without versioning enabled, deletions are permanent.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete an object from.
3. Select the object and then choose **Delete**.
4. To confirm permanent deletion of the object listed under **Specified objects**, in the **Delete objects?** text box, enter **permanently delete**.

**Note**

If you're experiencing any issues with deleting your object, see I want to permanently delete versioned objects.

### Using the AWS SDKs

The following examples show how you can use the AWS SDKs to delete an object from a bucket. For more information, see DELETE Object in the *Amazon Simple Storage Service API Reference*

If you have S3 Versioning enabled on the bucket, you have the following options:

- Delete a specific object version by specifying a version ID.
- Delete an object without specifying a version ID, in which case Amazon S3 adds a delete marker to the object.

For more information about S3 Versioning, see Using versioning in S3 buckets.

Java

**Example Example 1: Deleting an object (non-versioned bucket)**

The following example assumes that the bucket is not versioning-enabled and the object doesn't have any version IDs. In the delete request, you specify only the object key and not a version ID.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

import java.io.IOException;
```

```
public class DeleteObjectNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

**Example Example 2: Deleting an object (versioned bucket)**

The following example deletes an object from a versioned bucket. The example deletes a specific object version by specifying the object key name and version ID.

The example does the following:

1. Adds a sample object to the bucket. Amazon S3 returns the version ID of the newly added object. The example uses this version ID in the delete request.
2. Deletes the object version by specifying both the object key name and a version ID. If there are no other versions of that object, Amazon S3 deletes the object entirely. Otherwise, Amazon S3 only deletes the specified version.

> **Note**
>
> You can get the version IDs of an object by sending a `ListVersions` request.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;

public class DeleteObjectVersionEnabledBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED))
 {
                System.out.printf("Bucket %s is not versioning-enabled.",
 bucketName);
            } else {
                // Add an object.
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName,
```

```
                        "Sample content for deletion example.");
                System.out.printf("Object %s added to bucket %s\n", keyName,
 bucketName);

                // Delete the version of the object that we just created.
                System.out.println("Deleting versioned object " + keyName);
                s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,
 putResult.getVersionId()));
                System.out.printf("Object %s, version %s deleted\n", keyName,
 putResult.getVersionId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following examples show how to delete an object from both versioned and non-versioned buckets. For more information about S3 Versioning, see Using versioning in S3 buckets.

**Example Deleting an object from a non-versioned bucket**

The following C# example deletes an object from a non-versioned bucket. The example assumes that the objects don't have version IDs, so you don't specify version IDs. You specify only the object key.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;
```

```
namespace Amazon.DocSamples.S3
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
}
```

### Example Deleting an object from a versioned bucket

The following C# example deletes an object from a versioned bucket. It deletes a specific version of the object by specifying the object key name and version ID.

The code performs the following tasks:

1. Enables S3 Versioning on a bucket that you specify (if S3 Versioning is already enabled, this has no effect).
2. Adds a sample object to the bucket. In response, Amazon S3 returns the version ID of the newly added object. The example uses this version ID in the delete request.
3. Deletes the sample object by specifying both the object key name and a version ID.

> **Note**
>
> You can also get the version ID of an object by sending a `ListVersions` request.
>
> ```
> var listResponse = client.ListVersions(new ListVersionsRequest { BucketName
>  = bucketName, Prefix = keyName });
> ```

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
        private const string bucketName = "*** versioning-enabled bucket name ***";
        private const string keyName = "*** Object Key Name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
```

```
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID
                };
                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!"
            };
```

```
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
}
```

PHP

This example shows how to use classes from version 3 of the AWS SDK for PHP to delete an object from a non-versioned bucket. For information about deleting an object from a versioned bucket, see Using the REST API.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example deletes an object from a bucket. Because this example shows how to delete objects from non-versioned buckets, it provides only the bucket name and object key (not a version ID) in the delete request.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;

    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);
```

```
    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

### Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

### Using the AWS CLI

To delete one object per request, use the DELETE API. For more information, see DELETE Object. For more information about using the CLI to delete an object, see delete-object.

### Using the REST API

You can use the AWS SDKs to delete an object. However, if your application requires it, you can send REST requests directly. For more information, see DELETE Object in the *Amazon Simple Storage Service API Reference*.

## Deleting multiple objects

Because all objects in your S3 bucket incur storage costs, you should delete objects that you no longer need. For example, if you are collecting log files, it's a good idea to delete them when they're no longer needed. You can set up a lifecycle rule to automatically delete objects such as log files. For more information, see the section called “Setting lifecycle configuration”.

For information about Amazon S3 features and pricing, see Amazon S3 pricing.

You can use the Amazon S3 console, AWS SDKs, or the REST API to delete multiple objects simultaneously from an S3 bucket.

### Using the S3 console

Follow these steps to use the Amazon S3 console to delete multiple objects from a bucket.

**Warning**

- Deleting a specified object cannot be undone.
- This action deletes all specified objects. When deleting folders, wait for the delete action to finish before adding new objects to the folder. Otherwise, new objects might be deleted as well.
- When deleting objects in a bucket without versioning enabled, Amazon S3 will permanently delete the objects.

- When deleting objects in a bucket with bucket versioning **enabled** or **suspended**, Amazon S3 creates delete markers. For more information, see Working with delete markers.

**To delete objects that have versioning enabled or suspended**

> **Note**
>
> If the version IDs for the object in a versioning-suspended bucket are marked as NULL, S3 permanently deletes the objects since no previous versions exist. However, if a valid version ID is listed for the objects in a versioning-suspended bucket, then S3 creates delete markers for the deleted objects, while retaining the previous versions of the objects.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete the objects from.
3. Select the objects and then choose **Delete**.
4. To confirm deletion of the objects list under **Specified objects** in the **Delete objects?** text box, enter **`delete`**.

**To permanently delete specific object versions in a versioning-enabled bucket**

> **Warning**
>
> When you permanently delete specific object versions in Amazon S3, the deletion can't be undone.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete the objects from.
3. Select the objects that you want to delete.

4. Choose the **Show versions** toggle.
5. Select the object versions and then choose **Delete**.
6. To confirm permanent deletion of the specific object versions listed under **Specified objects**, in the **Delete objects?** text box, enter **Permanently delete**. Amazon S3 permanently deletes the specific object versions.

**To permanently delete the objects in an Amazon S3 bucket that *don't* have versioning enabled**

> **Warning**
>
> When you permanently delete an object in Amazon S3, the deletion can't be undone. Also, for any buckets without versioning enabled, deletions are permanent.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete the objects from.
3. Select the objects and then choose **Delete**.
4. To confirm permanent deletion of the objects listed under **Specified objects**, in the **Delete objects?** text box, enter **permanently delete**.

> **Note**
>
> If you're experiencing any issues with deleting your objects, see I want to permanently delete versioned objects.

### Using the AWS SDKs

For examples of how to delete multiple objects with the AWS SDKs, see Use `DeleteObjects` with an AWS SDK or CLI.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

### Using the REST API

You can use the AWS SDKs to delete multiple objects using the Multi-Object Delete API. However, if your application requires it, you can send REST requests directly.

For more information, see Delete Multiple Objects in the *Amazon Simple Storage Service API Reference*.

# Organizing, listing, and working with your objects

In Amazon S3, you can use prefixes to organize your storage. A prefix is a logical grouping of the objects in a bucket. The prefix value is similar to a directory name that enables you to store similar data under the same directory in a bucket. When you programmatically upload objects, you can use prefixes to organize your data.

In the Amazon S3 console, prefixes are called folders. You can view all your objects and folders in the S3 console by navigating to a bucket. You can also view information about each object, including object properties.

For more information about listing and organizing your data in Amazon S3, see the following topics.

### Topics

- Organizing objects using prefixes
- Listing object keys programmatically
- Organizing objects in the Amazon S3 console by using folders
- Viewing an object overview in the Amazon S3 console
- Viewing object properties in the Amazon S3 console

## Organizing objects using prefixes

You can use prefixes to organize the data that you store in Amazon S3 buckets. A prefix is a string of characters at the beginning of the object key name. A prefix can be any length, subject to the maximum length of the object key name (1,024 bytes). You can think of prefixes as a way to organize your data in a similar way to directories. However, prefixes are not directories.

Searching by prefix limits the results to only those keys that begin with the specified prefix. The delimiter causes a list operation to roll up all the keys that share a common prefix into a single summary list result.

The purpose of the prefix and delimiter parameters is to help you organize and then browse your keys hierarchically. To do this, first pick a delimiter for your bucket, such as slash (/), that doesn't occur in any of your anticipated key names. You can use another character as a delimiter. There is nothing unique about the slash (/) character, but it is a very common prefix delimiter. Next, construct your key names by concatenating all containing levels of the hierarchy, separating each level with the delimiter.

For example, if you were storing information about cities, you might naturally organize them by continent, then by country, then by province or state. Because these names don't usually contain punctuation, you might use slash (/) as the delimiter. The following examples use a slash (/) delimiter.

- Europe/France/Nouvelle-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

If you stored data for every city in the world in this manner, it would become awkward to manage a flat key namespace. By using `Prefix` and `Delimiter` with the list operation, you can use the hierarchy that you've created to list your data. For example, to list all the states in USA, set `Delimiter='/'` and `Prefix='North America/USA/'`. To list all the provinces in Canada for which you have data, set `Delimiter='/'` and `Prefix='North America/Canada/'`.

For more information about delimiters, prefixes, and nested folders, see Difference between prefixes and nested folders.

## Listing objects using prefixes and delimiters

If you issue a list request with a delimiter, you can browse your hierarchy at only one level, skipping over and summarizing the (possibly millions of) keys nested at deeper levels. For example, assume that you have a bucket (*DOC-EXAMPLE-BUCKET*) with the following keys:

```
sample.jpg
```

```
photos/2006/January/sample.jpg

photos/2006/February/sample2.jpg

photos/2006/February/sample3.jpg

photos/2006/February/sample4.jpg
```

The sample bucket has only the `sample.jpg` object at the root level. To list only the root level objects in the bucket, you send a GET request on the bucket with the slash (/) delimiter character. In response, Amazon S3 returns the `sample.jpg` object key because it does not contain the / delimiter character. All other keys contain the delimiter character. Amazon S3 groups these keys and returns a single `CommonPrefixes` element with the prefix value `photos/`, which is a substring from the beginning of these keys to the first occurrence of the specified delimiter.

**Example**

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>DOC-EXAMPLE-BUCKET</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>displayname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

For more information about listing object keys programmatically, see Listing object keys programmatically.

# Listing object keys programmatically

In Amazon S3, keys can be listed by prefix. You can choose a common prefix for the names of related keys and mark these keys with a special character that delimits hierarchy. You can then use the list operation to select and browse keys hierarchically. This is similar to how files are stored in directories within a file system.

Amazon S3 exposes a list operation that lets you enumerate the keys contained in a bucket. Keys are selected for listing by bucket and prefix. For example, consider a bucket named "`dictionary`" that contains a key for every English word. You might make a call to list all the keys in that bucket that start with the letter "q". List results are always returned in UTF-8 binary order.

Both the SOAP and REST list operations return an XML document that contains the names of matching keys and information about the object identified by each key.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

Groups of keys that share a prefix terminated by a special delimiter can be rolled up by that common prefix for the purposes of listing. This enables applications to organize and browse their keys hierarchically, much like how you would organize your files into directories in a file system.

For example, to extend the dictionary bucket to contain more than just English words, you might form keys by prefixing each word with its language and a delimiter, such as "`French/logical`". Using this naming scheme and the hierarchical listing feature, you could retrieve a list of only French words. You could also browse the top-level list of available languages without having to iterate through all the lexicographically intervening keys. For more information about this aspect of listing, see Organizing objects using prefixes.

**REST API**

If your application requires it, you can send REST requests directly. You can send a GET request to return some or all of the objects in a bucket or you can use selection criteria to return a subset of the objects in a bucket. For more information, see GET Bucket (List Objects) Version 2 in the *Amazon Simple Storage Service API Reference*.

### List implementation efficiency

List performance is not substantially affected by the total number of keys in your bucket. It's also not affected by the presence or absence of the `prefix`, `marker`, `maxkeys`, or `delimiter` arguments.

### Iterating through multipage results

As buckets can contain a virtually unlimited number of keys, the complete results of a list query can be extremely large. To manage large result sets, the Amazon S3 API supports pagination to split them into multiple responses. Each list keys response returns a page of up to 1,000 keys with an indicator indicating if the response is truncated. You send a series of list keys requests until you have received all the keys. AWS SDK wrapper libraries provide the same pagination.

## Examples

The following code examples show how to use `ListObjects`.

CLI

#### AWS CLI

The following example uses the `list-objects` command to display the names of all the objects in the specified bucket:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

The example uses the `--query` argument to filter the output of `list-objects` down to the key value and size for each object

For more information about objects, see Working with Amazon S3 Objects in the *Amazon S3 Developer Guide*.

- For API details, see ListObjects in *AWS CLI Command Reference*.

PowerShell

#### Tools for PowerShell

**Example 1: This command retrieves the information about all of the items in the bucket "test-files".**

```
Get-S3Object -BucketName test-files
```

**Example 2: This command retrieves the information about the item "sample.txt" from bucket "test-files".**

```
Get-S3Object -BucketName test-files -Key sample.txt
```

**Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "test-files".**

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- For API details, see ListObjects in *AWS Tools for PowerShell Cmdlet Reference*.

## Organizing objects in the Amazon S3 console by using folders

In Amazon S3, buckets and objects are the primary resources, and objects are stored in buckets. Amazon S3 has a flat structure instead of a hierarchy like you would see in a file system. However, for the sake of organizational simplicity, the Amazon S3 console supports the *folder* concept as a means of grouping objects. The console does this by using a shared name *prefix* for the grouped objects. In other words, the grouped objects have names that begin with a common string. This common string, or shared prefix, is the folder name. Object names are also referred to as *key names*.

For example, you can create a folder in the console named `photos` and store an object named `myphoto.jpg` in it. The object is then stored with the key name `photos/myphoto.jpg`, where `photos/` is the prefix.

Here are two more examples:

- If you have three objects in your bucket—`logs/date1.txt`, `logs/date2.txt`, and `logs/date3.txt`—the console will show a folder named `logs`. If you open the folder in the console, you will see three objects: `date1.txt`, `date2.txt`, and `date3.txt`.
- If you have an object named `photos/2017/example.jpg`, the console will show you a folder named `photos` containing the folder `2017`. The folder `2017` will contain the object `example.jpg`.

You can have folders within folders, but not buckets within buckets. You can upload and copy objects directly into a folder. Folders can be created, deleted, and made public, but they cannot be renamed. Objects can be copied from one folder to another.

> **Important**
>
> When you create a folder in Amazon S3, S3 creates a 0-byte object with a key that's set to the folder name that you provided. For example, if you create a folder named `photos` in your bucket, the Amazon S3 console creates a 0-byte object with the key `photos/`. The console creates this object to support the idea of folders.
>
> The Amazon S3 console treats all objects that have a forward slash (/) character as the last (trailing) character in the key name as a folder (for example, `examplekeyname/`). You can't upload an object that has a key name with a trailing / character by using the Amazon S3 console. However, you can upload objects that are named with a trailing / with the Amazon S3 API by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.
>
> An object that is named with a trailing / appears as a folder in the Amazon S3 console. The Amazon S3 console does not display the content and metadata for such an object. When you use the console to copy an object named with a trailing /, a new folder is created in the destination location, but the object's data and metadata are not copied.

**Topics**



## Creating a folder

This section describes how to use the Amazon S3 console to create a folder.

> **Important**
>
> If your bucket policy prevents uploading objects to this bucket without tags, metadata, or access control list (ACL) grantees, you can't create a folder by using the following

procedure. Instead, upload an empty folder and specify the following settings in the upload configuration.

**To create a folder**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to create a folder in.
4. If your bucket policy prevents uploading objects to this bucket without encryption, you must choose **Enable** under **Server-side encryption**.
5. Choose **Create folder**.
6. Enter a name for the folder (for example, **`favorite-pics`**). Then choose **Create folder**.

## Making folders public

We recommend blocking all public access to your Amazon S3 folders and buckets unless you specifically require a public folder or bucket. When you make a folder public, anyone on the internet can view all the objects that are grouped in that folder.

In the Amazon S3 console, you can make a folder public. You can also make a folder public by creating a bucket policy that limits data access by prefix. For more information, see Identity and Access Management for Amazon S3.

**Warning**

After you make a folder public in the Amazon S3 console, you can't make it private again. Instead, you must set permissions on each individual object in the public folder so that the objects have no public access. For more information, see Configuring ACLs.

**Topics**

- Calculating folder size
- Deleting folders

## Calculating folder size

This section describes how to use the Amazon S3 console to calculate a folder's size.

### To calculate a folder's size

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket in which your folder is stored.
4. In the **Objects** list, select the check box next to the name of the folder.
5. Choose **Actions**, and then choose **Calculate total size**.

> **Note**
>
> When you navigate away from the page, the folder information (including the total size) will no longer be available. You must calculate the total size again if you want to see it again.

> **Important**
>
> When you use the **Calculate total size** action on specified objects or folders within your bucket, Amazon S3 calculates the total number of objects and the total storage size. However, incomplete or in-progress multipart uploads and previous or noncurrent versions aren't calculated in the total number of objects or the total size. This action calculates only the total number of objects and the total size for the current or newest version of each object that is stored in the bucket.
>
> For example, if there are two versions of an object in your bucket, then the storage calculator in Amazon S3 counts them as only one object. As a result, the total number of objects that is calculated in the Amazon S3 console can differ from the **Object Count** metric shown in S3 Storage Lens and from the number reported by the Amazon CloudWatch metric, `NumberOfObjects`. Likewise, the total storage size can also differ from the **Total Storage** metric shown in S3 Storage Lens and from the `BucketSizeBytes` metric shown in CloudWatch.

## Deleting folders

This section explains how to use the Amazon S3 console to delete folders from an S3 bucket.

For information about Amazon S3 features and pricing, see Amazon S3.

### To delete folders from an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to delete folders from.
3. In the **Objects** list, select the check box next to the folders and objects that you want to delete.
4. Choose **Delete**.
5. On the **Delete objects** page, verify that the names of the folders you selected for deletion are listed.
6. In the **Delete objects** box, enter **`delete`**, and choose **Delete objects**.

> **Warning**
>
> This action deletes all specified objects. When deleting folders, wait for the delete action to finish before adding new objects to the folder. Otherwise, new objects might be deleted as well.

# Viewing an object overview in the Amazon S3 console

You can use the Amazon S3 console to view an overview of an object. The console provides all the essential information for an object in one place.

### To open the details page for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object for which you want an overview.

The object details page opens.

4. To download the object, choose **Object actions**, and then choose **Download**. To copy the path of the object to the clipboard, under **Object URL**, choose the URL.
5. If versioning is enabled on the bucket, choose **Versions** to list the versions of the object.

   - To download an object version, select the check box next to the version ID, choose **Actions**, and then choose **Download**.
   - To delete an object version, select the check box next to the version ID, and choose **Delete**.

> **Important**
>
> You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted.

## Viewing object properties in the Amazon S3 console

You can use the Amazon S3 console to view the properties of an object, including storage class, encryption settings, tags, and metadata.

**To view the properties of an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object you want to view properties for.

   The **Object overview** for your object opens. You can scroll down to view the object properties.
4. On the **Object overview** page, you can configure the following properties for the object.

> **Note**
>
> - If you change the **Storage Class**, **Encryption**, or **Metadata** properties, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object or (object version).

- If you change the **Storage Class**, **Encryption**, or **Metadata** properties for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing these properties for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, these properties for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.

a. **Storage class** – Each object in Amazon S3 has a storage class associated with it. The storage class that you choose to use depends on how frequently you access the object. The default storage class for S3 objects is STANDARD. You choose which storage class to use when you upload an object. For more information about storage classes, see Using Amazon S3 storage classes.

   To change the storage class after you upload an object, choose **Storage class**. Choose the storage class that you want, and then choose **Save**.

b. **Server-side encryption settings** – You can use server-side encryption to encrypt your S3 objects. For more information, see Specifying server-side encryption with AWS KMS (SSE-KMS) or Specifying server-side encryption with Amazon S3 managed keys (SSE-S3).

c. **Metadata** – Each object in Amazon S3 has a set of name-value pairs that represents its metadata. For information about adding metadata to an S3 object, see Editing object metadata in the Amazon S3 console.

d. **Tags** – You categorize storage by adding tags to an S3 object. For more information, see Categorizing your storage using tags.

e. **Object lock legal hold and retention** – You can prevent an object from being deleted. For more information, see Using S3 Object Lock.

# Working with presigned URLs

You can use presigned URLs to grant time-limited access to objects in Amazon S3 without updating your bucket policy. A presigned URL can be entered in a browser or used by a program to download an object. The credentials used by the presigned URL are those of the AWS user who generated the URL.

You can also use presigned URLs to allow someone to upload a specific object to your Amazon S3 bucket. This allows an upload without requiring another party to have AWS security credentials or permissions. If an object with the same key already exists in the bucket as specified in the presigned URL, Amazon S3 replaces the existing object with the uploaded object.

You can use the presigned URL multiple times, up to the expiration date and time.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An Amazon S3 bucket
- An object key (if downloading this object will be in your Amazon S3 bucket, if uploading this is the file name to be uploaded)
- An HTTP method (`GET` for downloading objects or `PUT` for uploading)
- An expiration time interval

Currently, Amazon S3 presigned URLs don't support using the following data-integrity checksum algorithms (CRC32, CRC32C, SHA-1, SHA-256) when you upload objects. To verify the integrity of your object after uploading, you can provide an MD5 digest of the object when you upload it with a presigned URL. For more information about object integrity, see Checking object integrity.

**Topics**



## Who can create a presigned URL

Anyone with valid security credentials can create a presigned URL. But for someone to successfully access an object, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon.

The following are the types of credentials that you can use to create a presigned URL:

- **IAM instance profile** – Valid up to 6 hours.
- **AWS Security Token Service** – Valid up to maximum 36 hours when signed with long-term security credentials or the duration of the temporary credential, whichever ends first.
- **IAM user** – Valid up to 7 days when you're using AWS Signature Version 4.

  To create a presigned URL that's valid for up to 7 days, first delegate IAM user credentials (the access key and secret key) to the method you're using to create the presigned URL.

> **Note**
>
> If you created a presigned URL using a temporary credential, the URL expires when the credential expires. In general, a presigned URL expires when the credential you used to create it is revoked, deleted, or deactivated. This is true even if the URL was created with a later expiration time. For temporary security credentials lifetimes, see Comparing AWS STS API operations in the *IAM User Guide*.

## Expiration time for presigned URLs

A presigned URL remains valid for the period of time specified when the URL is generated. If you create a presigned URL with the Amazon S3 console, the expiration time can be set between 1 minute and 12 hours. If you use the AWS CLI or AWS SDKs, the expiration time can be set as high as 7 days.

If you created a presigned URL by using a temporary token, then the URL expires when the token expires. In general, a presigned URL expires when the credential you used to create it is revoked, deleted, or deactivated. This is true even if the URL was created with a later expiration time. For more information about how the credentials you use affect the expiration time, see Who can create a presigned URL.

Amazon S3 checks the expiration date and time of a signed URL at the time of the HTTP request. For example, if a client begins to download a large file immediately before the expiration time, the download continues even if the expiration time passes during the download. However, if the connection drops and the client tries to restart the download after the expiration time passes, the download fails.

# Limiting presigned URL capabilities

The capabilities of a presigned URL are limited by the permissions of the user who created it. In essence, presigned URLs are bearer tokens that grant access to those who possess them. As such, we recommend that you protect them appropriately. The following are some methods that you can use to restrict the use of your presigned URLs.

### AWS Signature Version 4 (SigV4)

To enforce specific behavior when presigned URL requests are authenticated by using AWS Signature Version 4 (SigV4), you can use condition keys in bucket policies and access point policies. For example, the following bucket policy uses the `s3:signatureAge` condition to deny any Amazon S3 presigned URL request on objects in the *`amzn-s3-demo-bucket1`* bucket if the signature is more than 10 minutes old. To use this example, replace the *`user input placeholders`* with your own information.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Deny a presigned URL request if the signature is more than 10 min
 old",
         "Effect": "Deny",
         "Principal": {"AWS":"*"},
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition": {
            "NumericGreaterThan": {
               "s3:signatureAge": 600000
            }
         }
      }
   ]
}
```

For more information about policy keys related AWS Signature Version 4, see AWS Signature Version 4 Authentication in the *Amazon Simple Storage Service API Reference*.

### Network path restriction

If you want to restrict the use of presigned URLs and all Amazon S3 access to particular network paths, you can write AWS Identity and Access Management (IAM) policies. You can set these policies on the IAM principal that makes the call, the Amazon S3 bucket, or both.

A network-path restriction on the IAM principal requires the user of those credentials to make requests from the specified network. A restriction on the bucket or access point requires that all requests to that resource originate from the specified network. These restrictions also apply outside of the presigned URL scenario.

The IAM global condition key that you use depends on the type of endpoint. If you're using the public endpoint for Amazon S3, use `aws:SourceIp`. If you're using a virtual private cloud (VPC) endpoint to Amazon S3, use `aws:SourceVpc` or `aws:SourceVpce`.

The following IAM policy statement requires the principal to access AWS only from the specified network range. With this policy statement, all access must originate from that range. This includes the case of someone who's using a presigned URL for Amazon S3. To use this example, replace the *`user input placeholders`* with your own information.

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

## Sharing objects with presigned URLs

By default, all Amazon S3 objects are private, only the object owner has permission to access them. However, the object owner may share objects with others by creating a presigned URL. A presigned URL uses security credentials to grant time-limited permission to download objects. The URL can be entered in a browser or used by a program to download the object. The credentials used by the presigned URL are those of the AWS user who generated the URL.

For general information about presigned URLs, see Working with presigned URLs.

You can create a presigned URL for sharing an object without writing any code by using the Amazon S3 console, AWS Explorer for Visual Studio (Windows), or AWS Toolkit for Visual Studio

Code. You can also generate a presigned URL programmatically by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs.

### Using the S3 console

You can use the Amazon S3 console to generate a presigned URL for sharing an object by following these steps. When using the console the maximum expiration time for a presigned URL is 12 hours from the time of creation.

**To generate a presigned URL by using the Amazon S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want a presigned URL for.
4. In the **Objects** list, select the object that you want to create a presigned URL for.
5. On the **Object actions** menu, choose **Share with a presigned URL**.
6. Specify how long you want the presigned URL to be valid.
7. Choose **Create presigned URL**.
8. When a confirmation appears, the URL is automatically copied to your clipboard. You will see a button to copy the presigned URL if you need to copy it again.

### Using the AWS CLI

The following example AWS CLI command generates a presigned URL for sharing an object from an Amazon S3 bucket. When you use the AWS CLI, the maximum expiration time for a presigned URL is 7 days from the time of creation. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3 presign s3://amzn-s3-demo-bucket1/mydoc.txt --expires-in 604800
```

> **Note**
>
> For all AWS Regions launched after March 20, 2019 you need to specify the `endpoint-url` and `AWS Region` with the request. For a list of all the Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.

```
aws s3 presign s3://amzn-s3-demo-bucket1/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

For more information, see presign in the *AWS CLI Command Reference*.

**Using the AWS SDKs**

For examples of using the AWS SDKs to generate a presigned URL for sharing an object, see Create a presigned URL for Amazon S3 by using an AWS SDK.

When you use the AWS SDKs to generate a presigned URL, the maximum expiration time is 7 days from the time of creation.

> **Note**
>
> For all AWS Regions launched after March 20, 2019 you need to specify the `endpoint-url` and `AWS Region` with the request. For a list of all the Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.

> **Note**
>
> When using the AWS SDKs, the Tagging attribute must be a header and not a query parameter. All other attributes can be passed as a parameter for the presigned URL.

**Using the AWS Toolkit for Visual Studio (Windows)**

> **Note**
>
> At this time, the AWS Toolkit for Visual Studio does not support Visual Studio for Mac.

1. Install the AWS Toolkit for Visual Studio using the following instructions, Installing and setting up the Toolkit for Visual Studio in the *AWS Toolkit for Visual Studio User Guide*.
2. Connect to AWS using the following steps, Connecting to AWS in the *AWS Toolkit for Visual Studio User Guide*.
3. In the left side panel labeled **AWS Explorer**, double-click the bucket containing your object.
4. Right-click the object you wish to have a presigned URL generated for and select **Create Pre-Signed URL...**.
5. In the pop-up window, set the expiration date and time for your presigned URL.
6. The **Object Key**, should pre-populate based on the object you selected.
7. Choose **GET** to specify that this presigned URL will be used for downloading an object.
8. Choose the **Generate** button.
9. To copy the URL to the clipboard, choose **Copy**.
10. To use the generated presigned URL, paste the URL into any browser.

#### Using AWS Toolkit for Visual Studio Code

If you're using Visual Studio Code, you can generate a presigned URL to share an object without writing any code by using AWS Toolkit for Visual Studio Code. For general information, see AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.

For instructions on how to install the AWS Toolkit for Visual Studio Code, see Installing the AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.

1. Connect to AWS using the following steps, Connecting to AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.
2. Select the AWS logo on the left panel in Visual Studio Code.
3. Under **EXPLORER**, select **S3**.
4. Choose a bucket and file and open the context menu (right-click).
5. Choose **Generate presigned URL**, and then set the expiration time (in minutes).
6. Press Enter, and the presigned URL will be copied to your clipboard.

## Uploading objects with presigned URLs

You may use presigned URLs to allow someone to upload an object to your Amazon S3 bucket. Using a presigned URL will allow an upload without requiring another party to have AWS security credentials or permissions. A presigned URL is limited by the permissions of the user who creates it. That is, if you receive a presigned URL to upload an object, you can upload an object only if the creator of the URL has the necessary permissions to upload that object.

When someone uses the URL to upload an object, Amazon S3 creates the object in the specified bucket. If an object with the same key that is specified in the presigned URL already exists in the bucket, Amazon S3 replaces the existing object with the uploaded object. After upload, the bucket owner will own the object.

For general information about presigned URLs, see Working with presigned URLs.

You can create a presigned URL for uploading an object without writing any code by using AWS Explorer for Visual Studio. You can also generate a presigned URL programmatically by using the AWS SDKs.

**Using the AWS Toolkit for Visual Studio (Windows)**

> **Note**
>
> At this time, the AWS Toolkit for Visual Studio does not support Visual Studio for Mac.

1. Install the AWS Toolkit for Visual Studio using the following instructions, Installing and setting up the Toolkit for Visual Studio in the *AWS Toolkit for Visual Studio User Guide*.
2. Connect to AWS using the following steps, Connecting to AWS in the *AWS Toolkit for Visual Studio User Guide*.
3. In the left side panel labeled **AWS Explorer**, right-click the bucket you wish to have an object uploaded to.
4. Choose **Create Pre-Signed URL...**.
5. In the pop-up window, set the expiration date and time for your presigned URL.
6. For **Object Key**, set the name of the file to be uploaded. The file you're uploading must match this name exactly. If an object with the same object key already exists in the bucket, Amazon S3 will replace the existing object with the newly uploaded object.
7. Choose **PUT** to specify that this presigned URL will be used for uploading an object.

8. Choose the **Generate** button.
9. To copy the URL to the clipboard, choose **Copy**.
10. To use this URL you can send a PUT request with the `curl` command. Include the full path to your file and the presigned URL itself.

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

#### Using the AWS SDKs

For examples of using the AWS SDKs to generate a presigned URL for uploading an object, see Create a presigned URL for Amazon S3 by using an AWS SDK.

When you use the AWS SDKs to generate a presigned URL, the maximum expiration time is 7 days from the time of creation.

> **Note**
>
> For all AWS Regions launched after March 20, 2019 you need to specify the `endpoint-url` and `AWS Region` with the request. For a list of all the Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.

## Transforming objects with S3 Object Lambda

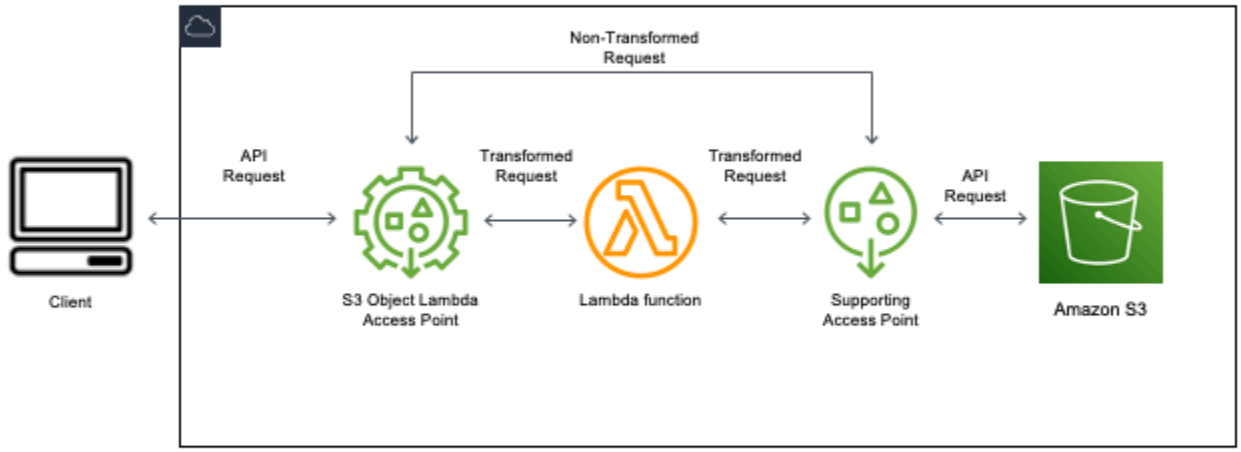
With Amazon S3 Object Lambda, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it is returned to an application. You can use custom code to modify the data returned by S3 `GET` requests to filter rows, dynamically resize and watermark images, redact confidential data, and more. You can also use S3 Object Lambda to modify the output of S3 `LIST` requests to create a custom view of all objects in a bucket and S3 `HEAD` requests to modify object metadata such as object name and size. You can use S3 Object Lambda as an origin for your Amazon CloudFront distribution to tailor data for end users, such as automatically resizing images, transcoding older formats (like from JPEG to WebP), or stripping metadata. For more information, see the AWS Blog post Use Amazon S3 Object Lambda with Amazon CloudFront. Powered by AWS Lambda functions, your code runs on infrastructure that is fully managed by AWS. Using S3 Object Lambda reduces the need to create and store derivative copies of your data or to run proxies, all with no need to change your applications.

### How S3 Object Lambda works

S3 Object Lambda uses AWS Lambda functions to automatically process the output of standard S3 `GET`, `LIST`, or `HEAD` requests. AWS Lambda is a serverless compute service that runs customer-defined code without requiring management of underlying compute resources. You can author and run your own custom Lambda functions, tailoring the data transformation to your specific use cases.

After you configure a Lambda function, you attach it to an S3 Object Lambda service endpoint, known as an *Object Lambda Access Point*. The Object Lambda Access Point uses a standard S3 access point, known as a *supporting access point*, to access Amazon S3.

When you send a request to your Object Lambda Access Point, Amazon S3 automatically calls your Lambda function. Any data retrieved by using an S3 `GET`, `LIST`, or `HEAD` request through the Object Lambda Access Point returns a transformed result back to the application. All other requests are processed as normal, as illustrated in the following diagram.




The topics in this section describe how to work with S3 Object Lambda.

### Topics

## Creating Object Lambda Access Points

An Object Lambda Access Point is associated with exactly one standard access point and thus one Amazon S3 bucket. To create an Object Lambda Access Point, you need the following resources:

- **An Amazon S3 bucket.** For information about creating buckets, see the section called "Creating a bucket".
- **A standard S3 access point.** When you're working with Object Lambda Access Points, this standard access point is known as a *supporting access point*. For information about creating standard access points, see the section called "Creating access points".
- **An AWS Lambda function.** You can either create your own Lambda function, or you can use a prebuilt function. For more information about creating Lambda functions, see the section called "Writing Lambda functions". For more information about prebuilt functions, see Using AWS built Lambda functions.
- **(Optional) An AWS Identity and Access Management (IAM) policy.** Amazon S3 access points support IAM resource policies that you can use to control the use of the access point by resource, user, or other conditions. For more information about creating these policies, see the section called "Configuring IAM policies".

The following sections describe how to create an Object Lambda Access Point by using:

- The AWS Management Console
- The AWS Command Line Interface (AWS CLI)
- An AWS CloudFormation template
- The AWS Cloud Development Kit (AWS CDK)

For information about how to create an Object Lambda Access Point by using the REST API, see `CreateAccessPointForObjectLambda` in the *Amazon Simple Storage Service API Reference*.

## Create an Object Lambda Access Point

Use one of the following procedures to create your Object Lambda Access Point.

### Using the S3 console

#### To create an Object Lambda Access Point by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Object Lambda Access Points**.
4. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
5. For **Object Lambda Access Point name**, enter the name that you want to use for the access point.

   As with standard access points, there are rules for naming Object Lambda Access Points. For more information, see Rules for naming Amazon S3 access points.
6. For **Supporting Access Point**, enter or browse to the standard access point that you want to use. The access point must be in the same AWS Region as the objects that you want to transform. For information about creating standard access points, see the section called "Creating access points".
7. Under **Transformation configuration**, you can add a function that transforms your data for your Object Lambda Access Point. Do one of the following:

   - If you already have a AWS Lambda function in your account you can choose it under **Invoke Lambda function**. Here you may enter the Amazon Resource Name (ARN) of an Lambda function in your AWS account or choose a Lambda function from the drop-down menu.
   - If you wish to use a AWS built function choose the function name under **AWS built function** and select **Create Lambda function**. This will take you to the Lambda console where you can deploy a built function into your AWS account. For more information about built functions, see Using AWS built Lambda functions.

Under **S3 APIs**, choose one or more API operations to invoke. For each API selected you must specify a Lambda function to invoke.

8. (Optional) Under **Payload**, add JSON text that you want to provide to your Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

   > **Important**
   >
   > When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

9. (Optional) For **Range and part number**, you must enable this option if you want to process GET and HEAD requests with range and part number headers. Enabling this option confirms that your Lambda function can recognize and process these requests. For more information about range headers and part numbers, see Working with Range and partNumber headers.
10. (Optional) For **Request metrics**, choose **Enable** or **Disable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate.
11. (Optional) Under **Object Lambda Access Point policy**, set a resource policy. Resource policies grant permissions for the specified Object Lambda Access Point and can control the use of the access point by resource, user, or other conditions. For more information about Object Lambda Access Point resource policies see, Configuring IAM policies for Object Lambda Access Points.
12. Under **Block Public Access settings for this Object Lambda Access Point**, select the block public access settings that you want to apply. All block public access settings are enabled by default for new Object Lambda Access Points, and we recommend that you leave default settings enabled. Amazon S3 currently doesn't support changing an Object Lambda Access Point's block public access settings after the Object Lambda Access Points has been created.

    For more information about using Amazon S3 Block Public Access, see Managing public access to access points.
13. Choose **Create Object Lambda Access Point**.

### Using the AWS CLI

#### To create an Object Lambda Access Point by using an AWS CloudFormation template

> **Note**
>
> To use the following commands, replace the *user input placeholders* with your own information.

1. Download the AWS Lambda function deployment package `s3objectlambda_deployment_package.zip` at S3 Object Lambda default configuration.
2. Run the following `put-object` command to upload the package to an Amazon S3 bucket.

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. Download the AWS CloudFormation template `s3objectlambda_defaultconfig.yaml` at S3 Object Lambda default configuration.
4. Run the following `deploy` command to deploy the template to your AWS account.

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name AWS CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

You can configure this AWS CloudFormation template to invoke Lambda for `GET`, `HEAD`, and `LIST` API operations. For more information about modifying the template's default configuration, see the section called “Automate S3 Object Lambda setup with AWS CloudFormation”.

### To create an Object Lambda Access Point by using the AWS CLI

> **Note**
>
> To use the following commands, replace the *`user input placeholders`* with your own information.

The following example creates an Object Lambda Access Point named *`my-object-lambda-ap`* for the bucket *`amzn-s3-demo-bucket1`* in the account *`111122223333`*. This example assumes that a standard access point named *`example-ap`* has already been created. For information about creating a standard access point, see the section called "Creating access points".

This example uses the AWS prebuilt function `decompress`. For more information about prebuilt functions, see the section called "Using AWS built functions".

1. Create a bucket. In this example, we will use *`amzn-s3-demo-bucket1`*. For information about creating buckets, see the section called "Creating a bucket".
2. Create a standard access point and attach it to your bucket. In this example, we will use *`example-ap`*. For information about creating standard access points, see the section called "Creating access points".
3. Do one of the following:

   - Create a Lambda function in your account that you would like to use to transform your Amazon S3 object. For more information about creating Lambda functions, see the section called "Writing Lambda functions". To use your custom function with the AWS CLI, see Using Lambda with the AWS CLI in the *AWS Lambda Developer Guide*.
   - Use an AWS prebuilt Lambda function. For more information about prebuilt functions, see Using AWS built Lambda functions.
4. Create a JSON configuration file named `my-olap-configuration.json`. In this configuration, provide the supporting access point and the Amazon Resource Name (ARN) for the Lambda function that you created in the previous steps or the ARN for the prebuilt function that you're using.

   **Example**

```
{
```

```
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. Run the `create-access-point-for-object-lambda` command to create your Object Lambda Access Point.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. (Optional) Create a JSON policy file named `my-olap-policy.json`.

   Adding an Object Lambda Access Point resource policy can control the use of the access point by resource, user, or other conditions. This resource policy grants the `GetObject` permission for account *444455556666* to the specified Object Lambda Access Point.

   **Example**

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
```

```
}
```

7. (Optional) Run the `put-access-point-policy-for-object-lambda` command to set your resource policy.

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. (Optional) Specify a payload.

   A payload is optional JSON that you can provide to your AWS Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

   The following Object Lambda Access Point configuration shows a payload with two parameters.

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
 }]
}
```

   The following Object Lambda Access Point configuration shows a payload with one parameter, and with `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-Range`, and `HeadObject-PartNumber` enabled.

```
{
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
```

```
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
        }
    }]
}
```

> **Important**
>
> When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

### Using the AWS CloudFormation console and template

You can create an Object Lambda Access Point by using the default configuration provided by Amazon S3. You can download an AWS CloudFormation template and Lambda function source code from the GitHub repository and deploy these resources to set up a functional Object Lambda Access Point.

For information about modifying the AWS CloudFormation template's default configuration, see the section called “Automate S3 Object Lambda setup with AWS CloudFormation”.

For information about configuring Object Lambda Access Points by using AWS CloudFormation without the template, see `AWS::S3ObjectLambda::AccessPoint` in the *AWS CloudFormation User Guide*.

### To upload the Lambda function deployment package

1. Download the AWS Lambda function deployment package `s3objectlambda_deployment_package.zip` at S3 Object Lambda default configuration.
2. Upload the package to an Amazon S3 bucket.

**To create an Object Lambda Access Point by using the AWS CloudFormation console**

1. Download the AWS CloudFormation template `s3objectlambda_defaultconfig.yaml` at S3 Object Lambda default configuration.
2. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Do one of the following:
   - If you've never used AWS CloudFormation before, on the AWS CloudFormation home page, choose **Create stack**.
   - If you have used AWS CloudFormation before, in the left navigation pane, choose **Stacks**. Choose **Create stack**, then choose **With new resources (standard)**.
4. For **Prerequisite - Prepare template**, choose **Template is ready**.
5. For **Specify template**, choose **Upload a template file** and upload `s3objectlambda_defaultconfig.yaml`.
6. Choose **Next**.
7. On the **Specify stack details** page, enter a name for the stack.
8. In the **Parameters** section, specify the following parameters that are defined in the stack template:
   a. For **CreateNewSupportingAccessPoint**, do one of the following:
      - If you already have a supporting access point for the S3 bucket where you uploaded the template, choose **false**.
      - If you want to create a new access point for this bucket, choose **true**.
   b. For **EnableCloudWatchMonitoring**, choose **true** or **false**, depending on whether you want to enable Amazon CloudWatch request metrics and alarms.
   c. (Optional) For **LambdaFunctionPayload**, add JSON text that you want to provide to your Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

> **Important**
>
> When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

d. For **LambdaFunctionRuntime**, enter your preferred runtime for the Lambda function. The available choices are `nodejs14.x`, `python3.9`, `java11`.

e. For **LambdaFunctionS3BucketName**, enter the Amazon S3 bucket name where you uploaded the deployment package.

f. For **LambdaFunctionS3Key**, enter the Amazon S3 object key where you uploaded the deployment package.

g. For **LambdaFunctionS3ObjectVersion**, enter the Amazon S3 object version where you uploaded the deployment package.

h. For **ObjectLambdaAccessPointName**, enter a name for your Object Lambda Access Point.

i. For **S3BucketName**, enter the Amazon S3 bucket name that will be associated with your Object Lambda Access Point.

j. For **SupportingAccessPointName**, enter the name of your supporting access point.

> **Note**
>
> This is an access point that is associated with the Amazon S3 bucket that you chose in the previous step. If you do not have any access points associated with your Amazon S3 bucket, you can configure the template to create one for you by choosing **true** for **CreateNewSupportingAccessPoint**.

9. Choose **Next**.
10. On the **Configure stack options page**, choose **Next**.

    For more information about the optional settings on this page, see Setting AWS CloudFormation stack options in the *AWS CloudFormation User Guide*.
11. On the **Review** page, choose **Create stack**.

### Using the AWS Cloud Development Kit (AWS CDK)

For more information about configuring Object Lambda Access Points by using the AWS CDK, see `AWS::S3ObjectLambda` Construct Library in the *AWS Cloud Development Kit (AWS CDK) API Reference*.

## Automate S3 Object Lambda setup with a CloudFormation template

You can use an AWS CloudFormation template to quickly create an Amazon S3 Object Lambda Access Point. The CloudFormation template automatically creates relevant resources, configures AWS Identity and Access Management (IAM) roles, and sets up an AWS Lambda function that automatically handles requests through the Object Lambda Access Point. With the CloudFormation template, you can implement best practices, improve your security posture, and reduce errors caused by manual processes.

This GitHub repository contains the CloudFormation template and Lambda function source code. For instructions on how to use the template, see the section called "Creating Object Lambda Access Points".

The Lambda function provided in the template does not run any transformation. Instead, it returns your objects as-is from your S3 bucket. You can clone the function and add your own transformation code to modify and process data as it is returned to an application. For more information about modifying your function, see the section called "Modifying the Lambda function" and the section called "Writing Lambda functions".

### Modifying the template

#### Creating a new supporting access point

S3 Object Lambda uses two access points, an Object Lambda Access Point and a standard S3 access point, which is referred to as the *supporting access point*. When you make a request to an Object Lambda Access Point, S3 either invokes Lambda on your behalf, or it delegates the request to the supporting access point, depending upon the S3 Object Lambda configuration. You can create a new supporting access point by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
CreateNewSupportingAccessPoint=true
```

#### Configuring a function payload

You can configure a payload to provide supplemental data to the Lambda function by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
LambdaFunctionPayload="format=json"
```

### Enabling Amazon CloudWatch monitoring

You can enable CloudWatch monitoring by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
EnableCloudWatchMonitoring=true
```

This parameter enables your Object Lambda Access Point for Amazon S3 request metrics and creates two CloudWatch alarms to monitor client-side and server-side errors.

> **Note**
>
> Amazon CloudWatch usage will incur additional costs. For more information about Amazon S3 request metrics, see Monitoring and logging access points.
> For pricing details, see CloudWatch pricing.

### Configuring provisioned concurrency

To reduce latency, you can configure provisioned concurrency for the Lambda function that's backing the Object Lambda Access Point by editing the template to include the following lines under `Resources`.

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
            ProvisionedConcurrentExecutions: Integer
```

> **Note**
>
> You will incur additional charges for provisioning concurrency. For more information about provisioned concurrency, see Managing Lambda provisioned concurrency in the *AWS Lambda Developer Guide*.
> For pricing details, see AWS Lambda pricing.

### Modifying the Lambda function

#### Changing header values for a `GetObject` request

By default, the Lambda function forwards all headers, except `Content-Length` and `ETag`, from the presigned URL request to the `GetObject` client. Based on your transformation code in the Lambda function, you can choose to send new header values to the `GetObject` client.

You can update your Lambda function to send new header values by passing them in the `WriteGetObjectResponse` API operation.

For example, if your Lambda function translates text in Amazon S3 objects to a different language, you can pass a new value in the `Content-Language` header. You can do this by modifying the `writeResponse` function as follows:

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
}
```

For a full list of supported headers, see `WriteGetObjectResponse` in the *Amazon Simple Storage Service API Reference*.

### Returning metadata headers

You can update your Lambda function to send new header values by passing them in the `WriteGetObjectResponse` API operation request.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
    ...headers
  }).promise();
}
```

### Returning a new status code

You can return a custom status code to the `GetObject` client by passing it in the `WriteGetObjectResponse` API operation request.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
```

```
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

For a full list of supported status codes, see `WriteGetObjectResponse` in the *Amazon Simple Storage Service API Reference*.

**Applying Range and `partNumber` parameters to the source object**

By default, the Object Lambda Access Point created by the CloudFormation template can handle the `Range` and `partNumber` parameters. The Lambda function applies the range or part number requested to the transformed object. To do so, the function must download the whole object and run the transformation. In some cases, your transformed object ranges might map exactly to your source object ranges. This means that requesting byte range A-B on your source object and running the transformation might produce the same result as requesting the whole object, running the transformation, and returning byte range A-B on the transformed object.

In such cases, you can change the Lambda function implementation to apply the range or part number directly to the source object. This approach reduces the overall function latency and memory required. For more information, see the section called “Working with Range and partNumber headers”.

**Disabling Range and `partNumber` handling**

By default, the Object Lambda Access Point created by the CloudFormation template can handle the `Range` and `partNumber` parameters. If you don't need this behavior, you can disable it by removing the following lines from the template:

```
AllowedFeatures:
  - GetObject-Range
  - GetObject-PartNumber
  - HeadObject-Range
  - HeadObject-PartNumber
```

**Transforming large objects**

By default, the Lambda function processes the entire object in memory before it can start streaming the response to S3 Object Lambda. You can modify the function to stream the response

as it performs the transformation. Doing so helps reduce the transformation latency and the Lambda function memory size. For an example implementation, see the Stream compressed content example.

## Using Amazon S3 Object Lambda Access Points

Making requests through Amazon S3 Object Lambda Access Points works the same as making requests through other access points. For more information about how to make requests through an access point, see Using access points. You can make requests through Object Lambda Access Points by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

> **Important**
>
> The Amazon Resource Names (ARNs) for Object Lambda Access Points use a service name of `s3-object-lambda`. Thus, Object Lambda Access Point ARNs begin with `arn:aws::s3-object-lambda`, instead of `arn:aws::s3`, which is used with other access points.

### How to find the ARN for your Object Lambda Access Point

To use an Object Lambda Access Point with the AWS CLI or AWS SDKs, you need to know the Amazon Resource Name (ARN) of the Object Lambda Access Point. The following examples show how to find the ARN for an Object Lambda Access Point by using the Amazon S3 console or AWS CLI.

#### Using the S3 console

#### To find the ARN for your Object Lambda Access Point by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. Choose the option button next to the Object Lambda Access Point whose ARN you want to copy.
4. Choose **Copy ARN**.

### Using the AWS CLI

#### To find the ARN for your Object Lambda Access Point by using the AWS CLI

1. To retrieve a list of the Object Lambda Access Points that are associated with your AWS account, run the following command. Before running the command, replace the account ID *111122223333* with your AWS account ID.

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. Review the command output to find the Object Lambda Access Point ARN that you want to use. The output of the previous command should look similar to the following example.

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

## How to use a bucket-style alias for your S3 bucket Object Lambda Access Point

When you create an Object Lambda Access Point, Amazon S3 automatically generates a unique alias for your Object Lambda Access Point. You can use this alias instead of an Amazon S3 bucket name or the Object Lambda Access Point Amazon Resource Name (ARN) in a request for access point data plane operations. For a list of these operations, see Access point compatibility with AWS services.

An Object Lambda Access Point alias name is created within the same namespace as an Amazon S3 bucket. This alias name is automatically generated and cannot be changed. For an existing Object Lambda Access Point, an alias is automatically assigned for use. An Object Lambda Access Point alias name meets all the requirements of a valid Amazon S3 bucket name and consists of the following parts:

*Object Lambda Access Point name prefix*-*metadata*--ol-s3

> **Note**
>
> The `--ol-s3` suffix is reserved for Object Lambda Access Point alias names and can't be used for bucket or Object Lambda Access Point names. For more information about Amazon S3 bucket-naming rules, see Bucket naming rules.

The following examples show the ARN and the Object Lambda Access Point alias for an Object Lambda Access Point named *my-object-lambda-access-point*:

- **ARN** – `arn:aws:s3-object-lambda:`*region*`:`*account-id*`:accesspoint/`*my-object-lambda-access-point*
- **Object Lambda Access Point alias** – *my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a*`--ol-s3`

When you use an Object Lambda Access Point, you can use the Object Lambda Access Point alias name without requiring extensive code changes.

When you delete an Object Lambda Access Point, the Object Lambda Access Point alias name becomes inactive and unprovisioned.

**How to find the alias for your Object Lambda Access Point**

**Using the S3 console**

**To find the alias for your Object Lambda Access Point by using the console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. For the Object Lambda Access Point that you want to use, copy the **Object Lambda Access Point alias** value.

**Using the AWS CLI**

When you create an Object Lambda Access Point, Amazon S3 automatically generates an Object Lambda Access Point alias name, as shown in the following example command. To run this command, replace the *user input placeholders* with your own information. For information

about how to create an Object Lambda Access Point by using the AWS CLI, see To create an Object Lambda Access Point by using the AWS CLI.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

The generated Object Lambda Access Point alias name has two fields:

- The `Value` field is the alias value of the Object Lambda Access Point.
- The `Status` field is the status of the Object Lambda Access Point alias. If the status is `PROVISIONING`, Amazon S3 is provisioning the Object Lambda Access Point alias, and the alias is not yet ready for use. If the status is `READY`, the Object Lambda Access Point alias has been successfully provisioned and is ready for use.

For more information about the `ObjectLambdaAccessPointAlias` data type in the REST API, see CreateAccessPointForObjectLambda and ObjectLambdaAccessPointAlias in the *Amazon Simple Storage Service API Reference*.

**How to use the Object Lambda Access Point alias**

You can use an Object Lambda Access Point alias instead of an Amazon S3 bucket name for the operations listed in Access point compatibility with AWS services.

The following AWS CLI example for the `get-bucket-location` command uses the bucket's access point alias to return the AWS Region that the bucket is in. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3

{
    "LocationConstraint": "us-west-2"
```

```
}
```

If the Object Lambda Access Point alias in a request isn't valid, the error code `InvalidAccessPointAliasError` is returned. For more information about `InvalidAccessPointAliasError`, see List of Error Codes in the *Amazon Simple Storage Service API Reference*.

The limitations of an Object Lambda Access Point alias are the same as those of an access point alias. For more information about the limitations of an access point alias, see Limitations.

## Security considerations for S3 Object Lambda Access Points

With Amazon S3 Object Lambda, you can perform custom transformations on data as it leaves Amazon S3 by using the scale and flexibility of AWS Lambda as a compute platform. S3 and Lambda remain secure by default, but to maintain this security, special consideration by the Lambda function author is required. S3 Object Lambda requires that all access be made by authenticated principals (no anonymous access) and over HTTPS.

To mitigate security risks, we recommend the following:

- Scope the Lambda execution role to the smallest set of permissions possible.
- Whenever possible, make sure your Lambda function accesses Amazon S3 through the provided presigned URL.

### Configuring IAM policies

S3 access points support AWS Identity and Access Management (IAM) resource policies that allow you to control the use of the access point by resource, user, or other conditions. For more information, see Configuring IAM policies for Object Lambda Access Points.

### Encryption behavior

Because Object Lambda Access Points use both Amazon S3 and AWS Lambda, there are differences in encryption behavior. For more information about default S3 encryption behavior, see Setting default server-side encryption behavior for Amazon S3 buckets.

- When you're using S3 server-side encryption with Object Lambda Access Points, the object is decrypted before being sent to Lambda. After the object is sent to Lambda, it is processed unencrypted (in the case of a GET or HEAD request).

- To prevent the encryption key from being logged, S3 rejects `GET` and `HEAD` requests for objects that are encrypted by using server-side encryption with customer-provided keys (SSE-C). However, the Lambda function might still retrieve these objects if it has access to the client-provided key.
- When using S3 client-side encryption with Object Lambda Access Points, make sure that Lambda has access to the encryption key so that it can decrypt and re-encrypt the object.

## Access points security

S3 Object Lambda uses two access points, an Object Lambda Access Point and a standard S3 access point, which is referred to as the *supporting access point*. When you make a request to an Object Lambda Access Point, S3 either invokes Lambda on your behalf, or it delegates the request to the supporting access point, depending upon the S3 Object Lambda configuration. When Lambda is invoked for a request S3 generates a presigned URL to your object on your behalf through the supporting access point. Your Lambda function receives this URL as input when the function is invoked.

You can set your Lambda function to use this presigned URL to retrieve the original object, instead of invoking S3 directly. By using this model, you can apply better security boundaries to your objects. You can limit direct object access through S3 buckets or S3 access points to a limited set of IAM roles or users. This approach also protects your Lambda functions from being subject to the confused deputy problem, where a misconfigured function with different permissions than the invoker could allow or deny access to objects when it should not.

## Object Lambda Access Point public access

S3 Object Lambda does not allow anonymous or public access because Amazon S3 must authorize your identity to complete any S3 Object Lambda request. When invoking requests through an Object Lambda Access Point, you must have the `lambda:InvokeFunction` permission for the configured Lambda function. Similarly, when invoking other API operations through an Object Lambda Access Point, you must have the required `s3:*` permissions.

Without these permissions, requests to invoke Lambda or delegate to S3 will fail with HTTP 403 (Forbidden) errors. All access must be made by authenticated principals. If you require public access, you can use Lambda@Edge as a possible alternative. For more information, see Customizing at the edge with Lambda@Edge in the *Amazon CloudFront Developer Guide*.

## Object Lambda Access Point IP addresses

The `describe-managed-prefix-lists` subnets support gateway virtual private cloud (VPC) endpoints and are related to the routing table of VPC endpoints. Since Object Lambda Access Point does not support gateway VPC its IP ranges are missing. The missing ranges belong to Amazon S3, but are not supported by gateway VPC endpoints. For more information about `describe-managed-prefix-lists`, see DescribeManagedPrefixLists in the *Amazon EC2 API Reference* and AWS IP address ranges in the *AWS General Reference*.

## Configuring IAM policies for Object Lambda Access Points

Amazon S3 access points support AWS Identity and Access Management (IAM) resource policies that you can use to control the use of the access point by resource, user, or other conditions. You can control access through an optional resource policy on your Object Lambda Access Point, or a resource policy on supporting access point. For step-by-step examples, see Tutorial: Transforming data for your application with S3 Object Lambda and Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.

The following four resources must have permissions granted to work with Object Lambda Access Points:

- The IAM identity, such as user or role. For more information about IAM identities and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.
- The bucket and its associated standard access point. When you're working with Object Lambda Access Points, this standard access point is known as a *supporting access point*.
- The Object Lambda Access Point.
- The AWS Lambda function.

> **Important**
>
> Before you save your policy, make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that are functional and conform to security best practices.

To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

The following policy examples assume that you have the following resources:

- An Amazon S3 bucket with the following Amazon Resource Name (ARN):

  `arn:aws:s3:::amzn-s3-demo-bucket1`
- An Amazon S3 standard access point on this bucket with the following ARN:

  `arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`
- An Object Lambda Access Point with the following ARN:

  `arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`
- An AWS Lambda function with the following ARN:

  `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

**Note**

If you're using a Lambda function from your account, you must include the specific function version in your policy statement. In the following example ARN, the version is indicated by *1*:
`arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`
Lambda doesn't support adding IAM policies to the version $LATEST. For more information about Lambda function versions, see Lambda function versions in the *AWS Lambda Developer Guide*.

### Example – Bucket policy that delegates access control to standard access points

The following S3 bucket policy example delegates access control for a bucket to the bucket's standard access points. This policy allows full access to all access points that are owned by the

bucket owner's account. Thus, all access to this bucket is controlled by the policies that are attached to its access points. Users can read from the bucket only through an access point, which means that operations can be invoked only through access points. For more information, see Delegating access control to access points.

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS":"account-ARN"},
        "Action" : "*",
        "Resource" : [
            "arn:aws:s3:::amzn-s3-demo-bucket1",
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

**Example – IAM policy that grants a user the necessary permissions to use an Object Lambda Access Point**

The following IAM policy grants a user permissions to the Lambda function, the standard access point, and the Object Lambda Access Point.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
```

```
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-
object-lambda-ap"
    }
  ]
}
```

### Enable permissions for Lambda execution roles

When GET requests are made to an Object Lambda Access Point, your Lambda function needs permission to send data to S3 Object Lambda Access Point. This permission is provided by enabling the `s3-object-lambda:WriteGetObjectResponse` permission on your Lambda function's execution role. You can create a new execution role or update an existing one.

> **Note**
>
> Your function needs the `s3-object-lambda:WriteGetObjectResponse` permission only if you're making a `GET` request.

**To create an execution role in the IAM console**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Under **Common use cases**, choose **Lambda**.
5. Choose **Next**.
6. On the **Add permissions** page, search for the AWS managed policy `AmazonS3ObjectLambdaExecutionRolePolicy`, and then select the check box beside the policy name.

   This policy should contain the `s3-object-lambda:WriteGetObjectResponse` Action.
7. Choose **Next**.
8. On the **Name, review, and create** page, for **Role name**, enter **`s3-object-lambda-role`**.
9. (Optional) Add a description and tags for this role.
10. Choose **Create role**.
11. Apply the newly created **`s3-object-lambda-role`** as your Lambda function's execution role. This can be done during or after Lambda function creation in the Lambda console.

For more information about execution roles, see Lambda execution role in the *AWS Lambda Developer Guide*.

**Using context keys with Object Lambda Access Points**

S3 Object Lambda will evaluate context keys such as `s3-object-lambda:TlsVersion` or `s3-object-lambda:AuthType` that are related to the connection or signing of the request. All other context keys, such as `s3:prefix`, are evaluated by Amazon S3.

### Object Lambda Access Point CORS support

When S3 Object Lambda receives a request from a browser or the request includes an `Origin` header, S3 Object Lambda always adds an `“AllowedOrigins":"*"` header field.

For more information, see Using cross-origin resource sharing (CORS).

## Writing Lambda functions for S3 Object Lambda Access Points

This section details how to write AWS Lambda functions for use with Amazon S3 Object Lambda Access Points.

To learn about complete end-to-end procedures for some S3 Object Lambda tasks, see the following:

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

**Topics**



### Working with `GetObject` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `GetObject`. S3 Object Lambda includes the Amazon S3 API operation, `WriteGetObjectResponse`, which enables the Lambda function to provide customized data and response headers to the `GetObject` caller.

`WriteGetObjectResponse` gives you extensive control over the status code, response headers, and response body, based on your processing needs. You can use `WriteGetObjectResponse` to respond with the whole transformed object, portions of the transformed object, or other responses

based on the context of your application. The following section shows unique examples of using the `WriteGetObjectResponse` API operation.

- **Example 1:** Respond with HTTP status code 403 (Forbidden)
- **Example 2:** Respond with a transformed image
- **Example 3:** Stream compressed content

### Example 1: Respond with HTTP status code 403 (Forbidden)

You can use `WriteGetObjectResponse` to respond with the HTTP status code 403 (Forbidden) based on the content of the object.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();

        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
```

```
            s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                    .withRequestRoute(event.outputRoute())
                    .withRequestToken(event.outputToken())
                    .withStatusCode(403)
                    .withContentLength(0L).withInputStream(new
 ByteArrayInputStream(new byte[0]))
                    .withErrorCode("MissingRequiredToken")
                    .withErrorMessage("The required token was not present in the
 request."));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(presignedResponse.body()));
    }
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
```

```
    get_context = event["getObjectContext"]
    user_request_headers = event["userRequest"]["headers"]

    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    is_token_present = "SuperSecretToken" in user_request_headers

    if is_token_present:
        # If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        response = requests.get(s3_url)
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 Body=response.content)
    else:
        # If the token is not present, we send an error back to the user.
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 StatusCode=403,
        ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
 secret enough.")

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;
```

```
    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
        const presignedResponse = await axios.get(inputS3Url);
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

### Example 2: Respond with a transformed image

When performing an image transformation, you might find that you need all the bytes of the source object before you can start processing them. In this case, your `WriteGetObjectResponse` request returns the whole object to the requesting application in one call.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);
```

```
        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(new ByteArrayInputStream(baos.toByteArray())));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)
```

```
    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

### Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
        .toBuffer();

    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();
```

```
    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

### Example 3: Stream compressed content

When you're compressing objects, compressed data is produced incrementally. Consequently, you can use your `WriteGetObjectResponse` request to return the compressed data as soon as it's ready. As shown in this example, you don't need to know the length of the completed transformation.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
```

```
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(bodyStream));
    }

}
```

Python

```
import boto3
import requests
import zlib
from botocore.config import Config


"""
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data

    def __iter__(self):
        while True:
            data = next(self.content)
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break
```

```
            yield chunk

        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
    s3 = boto3.client('s3', config=my_config)

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")
```

```
    # Gracefully exit the Lambda function.
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
        }).promise()
    );
```

```
    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

> **Note**
>
> Although S3 Object Lambda allows up to 60 seconds to send a complete response to the caller through the `WriteGetObjectResponse` request, the actual amount of time available might be less. For example, your Lambda function timeout might be less than 60 seconds. In other cases, the caller might have more stringent timeouts.

For the original caller to receive a response other than HTTP status code 500 (Internal Server Error), the `WriteGetObjectResponse` call must be completed. If the Lambda function returns, with an exception or otherwise, before the `WriteGetObjectResponse` API operation is called, the original caller receives a 500 (Internal Server Error) response. Exceptions thrown during the time it takes to complete the response result in truncated responses to the caller. If the Lambda function receives an HTTP status code 200 (OK) response from the `WriteGetObjectResponse` API call, then the original caller has sent the complete request. The Lambda function's response, whether an exception is thrown or not, is ignored by S3 Object Lambda.

When calling the `WriteGetObjectResponse` API operation, Amazon S3 requires the route and request token from the event context. For more information, see Event context format and usage.

The route and request token parameters are required to connect the `WriteGetObjectResult` response with the original caller. Even though it is always appropriate to retry 500 (Internal Server Error) responses, because the request token is a single-use token, subsequent attempts to use it might result in HTTP status code 400 (Bad Request) responses. Although the call to `WriteGetObjectResponse` with the route and request tokens doesn't need to be made from the invoked Lambda function, it must be made by an identity in the same account. The call also must be completed before the Lambda function finishes execution.

## Working with `HeadObject` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `HeadObject`. Lambda will receive a JSON payload that contains a key called `headObjectContext`. Inside the context, there is a single property called `inputS3Url`, which is a presigned URL for the supporting access point for `HeadObject`.

The presigned URL will include the following properties if they're specified:

- `versionId` (in the query parameters)
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

Other properties won't be presigned, and thus won't be included. Non-signed options sent as headers can be added manually to the request when calling the presigned URL that's found in the `userRequest` headers. Server-side encryption options are not supported for `HeadObject`.

For the request syntax URI parameters, see `HeadObject` in the *Amazon Simple Storage Service API Reference*.

The following example shows a Lambda JSON input payload for `HeadObject`.

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
  },
  "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
   },
   "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
             "attributes": {
             "mfaAuthenticated": "false",
             "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
        },
        "sessionIssuer": {
             "type": "Role",
             "principalId": "principalId",
             "arn": "arn:aws:iam::111122223333:role/Admin",
             "accountId": "111122223333",
             "userName": "Admin"
             }
        }
    },
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the headers and values that will be returned for the `HeadObject` call.

The following example shows the structure of the Lambda response JSON for `HeadObject`.

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
```

```
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
    };
}
```

The following example shows how to use the presigned URL to populate your response by modifying the header values as needed before returning the JSON.

Python

```
import requests

def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
```

```
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
        "headers": response_headers
        }
```

## Working with `ListObjects` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `ListObjects`. Lambda will receive the JSON payload with a new object named `listObjectsContext`. `listObjectsContext`contains a single property, `inputS3Url`, which is a presigned URL for the supporting access point for `ListObjects`.

Unlike `GetObject` and `HeadObject`, the presigned URL will include the following properties if they're specified:

- All the query parameters
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

For the request syntax URI parameters, see `ListObjects` in the *Amazon Simple Storage Service API Reference*.

> **Important**
>
> We recommend that you use the newer version, ListObjectsV2, when developing applications. For backward compatibility, Amazon S3 continues to support `ListObjects`.

The following example shows the Lambda JSON input payload for `ListObjects`.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
```

```
    },
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the status code, list XML result, or error information that will be returned from S3 Object Lambda.

S3 Object Lambda does not process or validate `listResultXml`, but instead forwards it to `ListObjects` caller. For `listBucketResult`, S3 Object Lambda expects certain properties to be of a specific type and will throw exceptions if it cannot parse them. `listResultXml` and `listBucketResult` can not be provided at the same time.

The following example demonstrates how to use the presigned URL to call Amazon S3 and use the result to populate a response, including error checking.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""
```

```
    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

The following example shows the structure of the Lambda response JSON for `ListObjects`.

```
{
```

```
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
        "name": <string>,  // Required for 'listBucketResult'
        "prefix": <string>,
        "marker": <string>,
        "nextMarker": <string>,
        "maxKeys": <int>,   // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>,  // Required for 'listBucketResult'
        "contents": [  {
            "key": <string>,  // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
            "size": <int>,   // Required for 'content'
            "owner": {
                "displayName": <string>,  // Required for 'owner'
                "id": <string>,  // Required for 'owner'
            },
            "storageClass": <string>
            },
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

## Working with `ListObjectsV2` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `ListObjectsV2`. Lambda will receive the JSON payload with a new object named

`listObjectsV2Context`. `listObjectsV2Context` contains a single property, `inputS3Url`, which is a presigned URL for the supporting access point for `ListObjectsV2`.

Unlike `GetObject` and `HeadObject`, the presigned URL will include the following properties, if they're specified:

- All the query parameters
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

For the request syntax URI parameters, see `ListObjectsV2` in the *Amazon Simple Storage Service API Reference*.

The following example shows the Lambda JSON input payload for `ListObjectsV2`.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
```

```
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the status code, list XML result, or error information that will be returned from S3 Object Lambda.

S3 Object Lambda does not process or validate `listResultXml`, but instead forwards it to `ListObjectsV2` caller. For `listBucketResult`, S3 Object Lambda expects certain properties to be of a specific type and will throw exceptions if it cannot parse them. `listResultXml` and `listBucketResult` can not be provided at the same time.

The following example demonstrates how to use the presigned URL to call Amazon S3 and use the result to populate a response, including error checking.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)
```

```
    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
```

```
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

The following example shows the structure of the Lambda response JSON for `ListObjectsV2`.

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
        "prefix": <string>,
        "startAfter": <string>,
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
```

```
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

## Event context format and usage

Amazon S3 Object Lambda provides context about the request that's being made in the event that's passed to your AWS Lambda function. The following shows an example request. Descriptions of the fields are included after the example.

```
{
    "xAmzRequestId": "requestId",
    "getObjectContext": {
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
        "outputToken": "OutputToken"
    },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
```

```
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "protocolVersion": "1.00"
}
```

The following fields are included in the request:

- `xAmzRequestId` – The Amazon S3 request ID for this request. We recommend that you log this value to help with debugging.
- `getObjectContext` – The input and output details for connections to Amazon S3 and S3 Object Lambda.
  - `inputS3Url` – A presigned URL that can be used to fetch the original object from Amazon S3. The URL is signed by using the original caller's identity, and that user's permissions will apply when the URL is used. If there are signed headers in the URL, the Lambda function must include these headers in the call to Amazon S3, except for the `Host` header.
  - `outputRoute` – A routing token that is added to the S3 Object Lambda URL when the Lambda function calls `WriteGetObjectResponse`.
  - `outputToken` – An opaque token that's used by S3 Object Lambda to match the `WriteGetObjectResponse` call with the original caller.
- `configuration` – Configuration information about the Object Lambda Access Point.
  - `accessPointArn` – The Amazon Resource Name (ARN) of the Object Lambda Access Point that received this request.

  - `supportingAccessPointArn` – The ARN of the supporting access point that is specified in the Object Lambda Access Point configuration.
  - `payload` – Custom data that is applied to the Object Lambda Access Point configuration. S3 Object Lambda treats this data as an opaque string, so it might need to be decoded before use.
- `userRequest` – Information about the original call to S3 Object Lambda.
  - `url` – The decoded URL of the request as received by S3 Object Lambda, excluding any authorization-related query parameters.
  - `headers` – A map of string to strings containing the HTTP headers and their values from the original call, excluding any authorization-related headers. If the same header appears multiple times, the values from each instance of the same header are combined into a comma-delimited list. The case of the original headers is retained in this map.
- `userIdentity` – Details about the identity that made the call to S3 Object Lambda. For more information, see Logging data events for trails in the *AWS CloudTrail User Guide*.
  - `type` – The type of identity.
  - `accountId` – The AWS account to which the identity belongs.
  - `userName` – The friendly name of the identity that made the call.
  - `principalId` – The unique identifier for the identity that made the call.
  - `arn` – The ARN of the principal who made the call. The last section of the ARN contains the user or role that made the call.
  - `sessionContext` – If the request was made with temporary security credentials, this element provides information about the session that was created for those credentials.
  - `invokedBy` – The name of the AWS service that made the request, such as Amazon EC2 Auto Scaling or AWS Elastic Beanstalk.
  - `sessionIssuer` – If the request was made with temporary security credentials, this element provides information about how the credentials were obtained.
- `protocolVersion` – The version ID of the context provided. The format of this field is `{Major Version}.{Minor Version}`. The minor version numbers are always two-digit numbers. Any removal or change to the semantics of a field necessitates a major version bump and requires active opt-in. Amazon S3 can add new fields at any time, at which point you might experience a minor version bump. Because of the nature of software rollouts, you might see multiple minor versions in use at once.

## Working with Range and partNumber headers

When working with large objects in Amazon S3 Object Lambda, you can use the `Range` HTTP header to download a specified byte range from an object. To fetch different byte ranges from within the same object, you can use concurrent connections to Amazon S3. You can also specify the `partNumber` parameter (an integer between 1 and 10,000), which performs a ranged request for the specified part of the object.

Because there are multiple ways that you might want to handle a request that includes the `Range` or `partNumber` parameters, S3 Object Lambda doesn't apply these parameters to the transformed object. Instead, your AWS Lambda function must implement this functionality as needed for your application.

To use the `Range` and `partNumber` parameters with S3 Object Lambda, you do the following:

- Enable these parameters in your Object Lambda Access Point configuration.
- Write a Lambda function that can handle requests that include these parameters.

The following steps describe how to accomplish this.

### Step 1: Configure your Object Lambda Access Point

By default, Object Lambda Access Points respond with an HTTP status code 501 (Not Implemented) error to any `GetObject` or `HeadObject` request that contains a `Range` or `partNumber` parameter, either in the headers or query parameters.

To enable an Object Lambda Access Point to accept such requests, you must include `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-Range`, or `HeadObject-PartNumber` in the `AllowedFeatures` section of your Object Lambda Access Point configuration. For more information about updating your Object Lambda Access Point configuration, see Creating Object Lambda Access Points.

### Step 2: Implement Range or `partNumber` handling in your Lambda function

When your Object Lambda Access Point invokes your Lambda function with a ranged `GetObject` or `HeadObject` request, the `Range` or `partNumber` parameter is included in the event context. The location of the parameter in the event context depends on which parameter was used and how it was included in the original request to the Object Lambda Access Point, as explained in the following table.

| Parameter | Event context location |
| --- | --- |
| Range (header) | `userRequest.headers.Range` |
| Range (query parameter) | `userRequest.url` (query parameter Range) |
| `partNumber` | `userRequest.url` (query parameter `partNumber` ) |

**Important**

The provided presigned URL for your Object Lambda Access Point doesn't contain the Range or `partNumber` parameter from the original request. See the following options on how to handle these parameters in your AWS Lambda function.

After you extract the Range or `partNumber` value, you can take one of the following approaches, based on your application's needs:

A. **Map the requested Range or `partNumber` to the transformed object (recommended).**

The most reliable way to handle Range or `partNumber` requests is to do the following:

- Retrieve the full object from Amazon S3.
- Transform the object.
- Apply the requested Range or `partNumber` parameters to the transformed object.

To do this, use the provided presigned URL to fetch the entire object from Amazon S3 and then process the object as needed. For an example Lambda function that processes a Range parameter in this way, see this sample in the AWS Samples GitHub repository.

B. **Map the requested Range to the presigned URL.**

In some cases, your Lambda function can map the requested Range directly to the presigned URL to retrieve only part of the object from Amazon S3. This approach is appropriate only if your transformation meets both of the following criteria:

1. Your transformation function can be applied to partial object ranges.

2. Applying the Range parameter before or after the transformation function results in the same transformed object.

For example, a transformation function that converts all characters in an ASCII-encoded object to uppercase meets both of the preceding criteria. The transformation can be applied to part of an object, and applying the Range parameter before the transformation achieves the same result as applying it after the transformation.

By contrast, a function that reverses the characters in an ASCII-encoded object doesn't meet these criteria. Such a function meets criterion 1, because it can be applied to partial object ranges. However, it doesn't meet criterion 2, because applying the Range parameter before the transformation achieves different results than applying the parameter after the transformation.

Consider a request to apply the function to the first three characters of an object with the contents `abcdefg`. Applying the Range parameter before the transformation retrieves only `abc` and then reverses the data, returning `cba`. But if the parameter is applied after the transformation, the function retrieves the entire object, reverses it, and then applies the Range parameter, returning `gfe`. Because these results are different, this function should not apply the Range parameter when retrieving the object from Amazon S3. Instead, it should retrieve the entire object, perform the transformation, and only then apply the Range parameter.

> **Warning**
>
> In many cases, applying the Range parameter to the presigned URL will result in unexpected behavior by the Lambda function or the requesting client. Unless you are sure that your application will work properly when retrieving only a partial object from Amazon S3, we recommend that you retrieve and transform full objects as described earlier in approach A.

If your application meets the criteria described earlier in approach B, you can simplify your AWS Lambda function by fetching only the requested object range and then running your transformation on that range.

The following Java code example demonstrates how to do the following:

- Retrieve the Range header from the `GetObject` request.

- Add the Range header to the presigned URL that Lambda can use to retrieve the requested range from Amazon S3.

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

## Using AWS built Lambda functions

AWS provides some prebuilt AWS Lambda functions that you can use with Amazon S3 Object Lambda to detect and redact personally identifiable information (PII) and decompress S3 objects. These Lambda functions are available in the AWS Serverless Application Repository. You can select these functions through the AWS Management Console when you create your Object Lambda Access Point.

For more information about how to deploy serverless applications from the AWS Serverless Application Repository, see Deploying Applications in the *AWS Serverless Application Repository Developer Guide*.

> **Note**
>
> The following examples can be used only with `GetObject` requests.

### Example 1: PII access control

This Lambda function uses Amazon Comprehend, a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. This function automatically detects personally identifiable information (PII), such as names, addresses, dates, credit card numbers, and social security numbers in documents in your Amazon S3 bucket. If you have documents in your

bucket that include PII, you can configure the PII Access Control function to detect these PII entity types and restrict access to unauthorized users.

To get started, deploy the following Lambda function in your account and add the Amazon Resource Name (ARN) for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: ComprehendPiiAccessControlS3ObjectLambda.

To view this function on GitHub, see Amazon Comprehend S3 Object Lambda.

### Example 2: PII redaction

This Lambda function uses Amazon Comprehend, a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. This function automatically redacts personally identifiable information (PII), such as names, addresses, dates, credit card numbers, and social security numbers from documents in your Amazon S3 bucket.

If you have documents in your bucket that include information such as credit card numbers or bank account information, you can configure the PII Redaction S3 Object Lambda function to detect PII and then return a copy of these documents in which PII entity types are redacted.

To get started, deploy the following Lambda function in your account and add the ARN for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: ComprehendPiiRedactionS3ObjectLambda.

To view this function on GitHub, see Amazon Comprehend S3 Object Lambda.

To learn about complete end-to-end procedures for some S3 Object Lambda tasks in PII redaction, see Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.

### Example 3: Decompression

The Lambda function `S3ObjectLambdaDecompression` can decompress objects that are stored in Amazon S3 in one of six compressed file formats: `bzip2`, `gzip`, `snappy`, `zlib`, `zstandard`, and `ZIP`.

To get started, deploy the following Lambda function in your account and add the ARN for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: S3ObjectLambdaDecompression.

To view this function on GitHub, see S3 Object Lambda Decompression.

## Best practices and guidelines for S3 Object Lambda

When using S3 Object Lambda, follow these best practices and guidelines to optimize operations and performance.

**Topics**

- Working with S3 Object Lambda
- AWS services used in connection with S3 Object Lambda
- Range and partNumber headers
- Transforming the expiry-date
- Working with the AWS CLI and AWS SDKs

### Working with S3 Object Lambda

S3 Object Lambda supports processing only `GET`, `LIST`, and `HEAD` requests. Any other requests don't invoke AWS Lambda and instead return standard, non-transformed API responses. You can create a maximum of 1,000 Object Lambda Access Points per AWS account per Region. The AWS

Lambda function that you use must be in the same AWS account and Region as the Object Lambda Access Point.

S3 Object Lambda allows up to 60 seconds to stream a complete response to its caller. Your function is also subject to AWS Lambda default quotas. For more information, see Lambda quotas in the *AWS Lambda Developer Guide*.

When S3 Object Lambda invokes your specified Lambda function, you are responsible for ensuring that any data that is overwritten or deleted from Amazon S3 by your specified Lambda function or application is intended and correct.

You can use S3 Object Lambda only to perform operations on objects. You cannot use S3 Object Lambda to perform other Amazon S3 operations, such as modifying or deleting buckets. For a complete list of S3 operations that support access points, see Access point compatibility with S3 operations.

In addition to this list, Object Lambda Access Points do not support the `POST Object`, `CopyObject` (as the source), and `SelectObjectContent` API operations.

### AWS services used in connection with S3 Object Lambda

S3 Object Lambda connects Amazon S3, AWS Lambda, and optionally, other AWS services of your choosing to deliver objects relevant to the requesting applications. All AWS services used with S3 Object Lambda are governed by their respective Service Level Agreements (SLAs). For example, if any AWS service does not meet its Service Commitment, you are eligible to receive a Service Credit, as documented in the service's SLA.

### Range and `partNumber` headers

When working with large objects, you can use the `Range` HTTP header to download a specified byte-range from an object. When you use the `Range` header, your request fetches only the specified portion of the object. You can also use the `partNumber` header to perform a ranged request for the specified part from the object.

For more information see, Working with Range and partNumber headers.

### Transforming the `expiry-date`

You can open or download transformed objects from your Object Lambda Access Point on the AWS Management Console. These objects must be non-expired. If your Lambda function transforms

the `expiry-date` of your objects, you might see expired objects that cannot be opened or downloaded. This behavior applies only to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive restored objects.

### Working with the AWS CLI and AWS SDKs

AWS Command Line Interface (AWS CLI) S3 subcommands (`cp`, `mv`, and `sync`) and the use of the AWS SDK for Java `TransferManager` class are not supported for use with S3 Object Lambda.

## S3 Object Lambda tutorials

The following tutorials present complete end-to-end procedures for some S3 Object Lambda tasks.

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

## Debugging S3 Object Lambda

Requests to Amazon S3 Object Lambda access points might result in a new error response when something goes wrong with the Lambda function invocation or execution. These errors follow the same format as standard Amazon S3 errors. For information about S3 Object Lambda errors, see S3 Object Lambda Error Code List in the *Amazon Simple Storage Service API Reference*.

For more information about general Lambda function debugging, see Monitoring and troubleshooting Lambda applications in the *AWS Lambda Developer Guide*.

For information about standard Amazon S3 errors, see Error Responses in the *Amazon Simple Storage Service API Reference*.

You can enable request metrics in Amazon CloudWatch for your Object Lambda Access Points. These metrics help you monitor the operational performance of your access point. You can enable request metrics during or after creation of your Object Lambda Access Point. For more information, see S3 Object Lambda request metrics in CloudWatch.

To get more granular logging about requests made to your Object Lambda Access Points, you can enable AWS CloudTrail data events. For more information, see Logging data events for trails in the *AWS CloudTrail User Guide*.

For S3 Object Lambda tutorials, see the following:

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

For more information about standard access points, see Managing data access with Amazon S3 access points.

For information about working with buckets, see Buckets overview. For information about working with objects, see Amazon S3 objects overview.

# What is S3 Express One Zone?

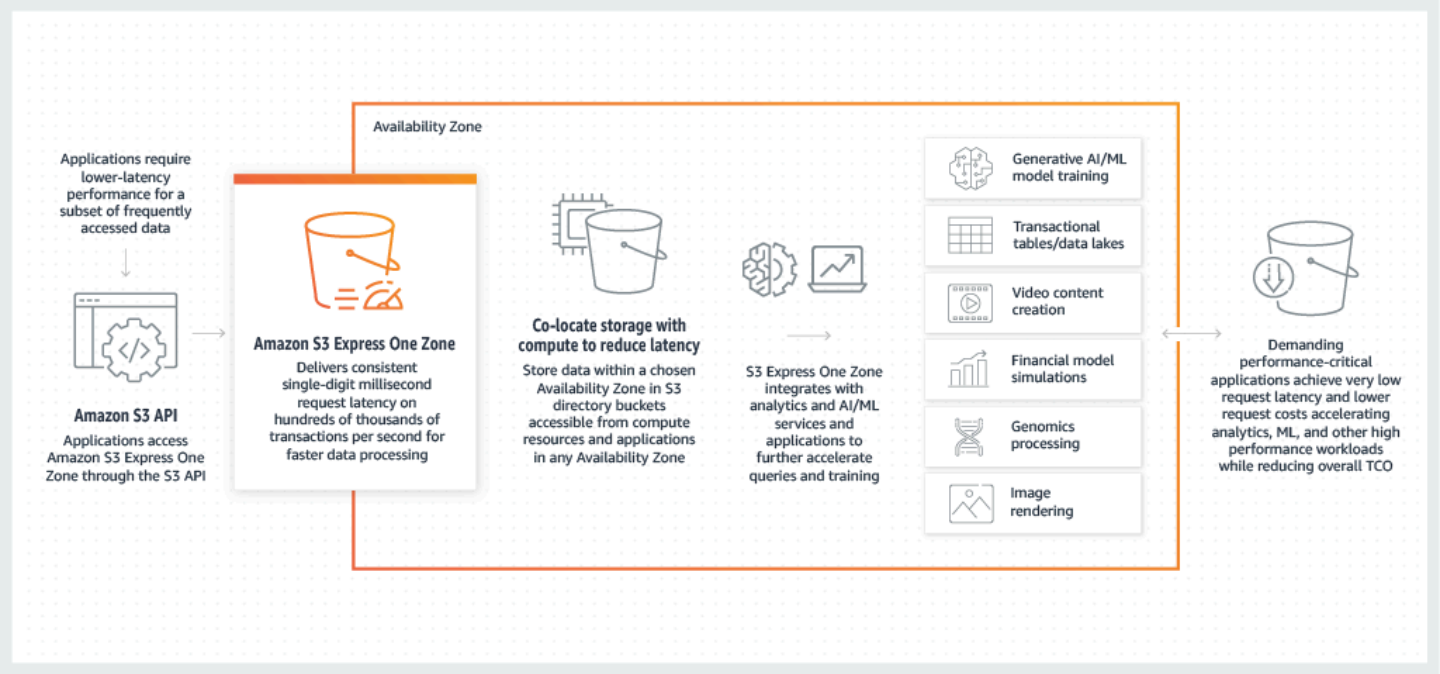
Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud-object storage class available today, with data access speeds up to 10x faster and with request costs 50 percent lower than S3 Standard. Applications can benefit immediately from requests being completed up to an order of magnitude faster. S3 Express One Zone provides similar performance elasticity as other S3 storage classes.

As with other Amazon S3 storage classes, you don't need to plan or provision capacity or throughput requirements in advance. You can scale your storage up or down, based on need, and access your data through the Amazon S3 API.

S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. Additionally, to further increase access speed and support hundreds of thousands of requests per second, data in S3 Express One Zone storage class is stored in a new bucket type: an Amazon S3 directory bucket. Each directory bucket can support hundreds of thousands of transactions per second (TPS), irrespective of key names or access pattern.

The Amazon S3 Express One Zone storage class is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy. If the existing device encounters a failure, S3 Express One Zone automatically shifts requests to new devices within an Availability Zone. This redundancy helps ensure uninterrupted access to your data within an Availability Zone.

S3 Express One Zone is ideal for any application where it's important to minimize the latency required to access an object. Such applications can be human-interactive workflows, like video editing, where creative professionals need responsive access to content from their user interfaces. S3 Express One Zone also benefits analytics and machine learning workloads that have similar responsiveness requirements from their data, especially workloads with lots of smaller accesses or large numbers of random accesses. S3 Express One Zone can be used with other AWS services to support analytics and artificial intelligence and machine learning (AI/ML) workloads, such as Amazon EMR, Amazon SageMaker, and Amazon Athena.




When using S3 Express One Zone, you can interact with your directory bucket in a virtual private cloud (VPC) by using a gateway VPC endpoint. With a gateway endpoint, you can access S3 Express One Zone directory buckets from your VPC without an internet gateway or NAT device for your VPC, and at no additional cost.

You can use many of the same Amazon S3 API operations and features with directory buckets that you use with general purpose buckets and other storage classes. These include Mountpoint for Amazon S3, server-side encryption with Amazon S3 managed keys (SSE-S3), S3 Batch Operations, and S3 Block Public Access. You can access S3 Express One Zone by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, and the Amazon S3 REST API.

For more information about S3 Express One Zone, see the following topics.

- Overview
- Features of S3 Express One Zone
- Related services
- Next steps

# Overview

To optimize performance and reduce latency, S3 Express One Zone introduces the following new concepts.

## Single Availability Zone

The Amazon S3 Express One Zone storage class is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy. If the existing device encounters a failure, S3 Express One Zone automatically shifts requests to new devices within an Availability Zone. This redundancy helps ensure uninterrupted access to your data within an Availability Zone.

An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. When you create a directory bucket, you choose the Availability Zone and AWS Region where your bucket will be located.

## Directory buckets

There are two types of Amazon S3 buckets: S3 general purpose buckets and S3 directory buckets. General purpose buckets are the default Amazon S3 bucket type that is used for the vast majority of S3 use cases. Directory buckets use only the S3 Express One Zone storage class, which is designed for workloads or performance-critical applications that require consistent single-digit millisecond latency. Choose the bucket type that best fits your application and performance requirements.

Directory buckets organize data hierarchically into directories, as opposed to the flat storage structure of general purpose buckets. There aren't prefix limits for directory buckets, and individual directories can scale horizontally.

Directory buckets use the S3 Express One Zone storage class, which is built to be used by performance sensitive applications. With S3 Express One Zone, you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. This is unlike general purpose buckets, which redundantly store objects across multiple Availability Zones in AWS Regions.

For more information about directory buckets, see Directory buckets. For more information about general purpose buckets, see Buckets overview.

## Endpoints and gateway VPC endpoints

Bucket-management API operations for directory buckets are available through a Regional endpoint and are referred to as Regional endpoint API operations. Examples of Regional endpoint

API operations are `CreateBucket` and `DeleteBucket`. After you create a directory bucket, you can use Zonal endpoint API operations to upload and manage the objects in your directory bucket. Zonal endpoint API operations are available through a Zonal endpoint. Examples of Zonal endpoint API operations are `PutObject` and `CopyObject`.

You can access S3 Express One Zone from your VPC by using gateway VPC endpoints. After you create a gateway endpoint, you can add it as a target in your route table for traffic destined from your VPC to S3 Express One Zone. As with Amazon S3, there is no additional charge for using gateway endpoints. For more information about how to configure gateway VPC endpoints, see Networking for S3 Express One Zone

### Session-based authorization

With S3 Express One Zone, you authenticate and authorize requests through a new session-based mechanism that is optimized to provide the lowest latency. You can use `CreateSession` to request temporary credentials that provide low-latency access to your bucket. These temporary credentials are scoped to a specific S3 directory bucket. Session tokens are used only with Zonal (object-level) operations (with the exception of CopyObject). For more information, see `CreateSession` authorization.

The supported AWS SDKs for S3 Express One Zone handle session establishment and refreshment on your behalf. To protect your sessions, temporary security credentials expire after 5 minutes. After you download and install the AWS SDKs and configure the necessary AWS Identity and Access Management (IAM) permissions, you can immediately start using API operations.

## Features of S3 Express One Zone

The following S3 features are available for S3 Express One Zone. For a complete list of supported API operationss and unsupported features, see How is S3 Express One Zone different?.

### Access management and security

With directory buckets, you can use the following features to audit and manage access. By default, directory buckets are private and can be accessed only by users who are explicitly granted access. Unlike general purpose buckets, which can set the access control boundary at the bucket, prefix, or object tag level, the access control boundary for directory buckets is set only at the bucket level. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

- S3 Block Public Access – All S3 Block Public Access settings are enabled by default at the bucket level. This default setting can't be modified.
- S3 Object Ownership (bucket owner enforced by default) – Access control lists (ACLs) are not supported for directory buckets. Directory buckets automatically use the bucket owner enforced setting for S3 Object Ownership. Bucket owner enforced means that ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. This default setting can't be modified.
- AWS Identity and Access Management (IAM) – IAM helps you securely control access to your directory buckets. You can use IAM to grant access to bucket management (Regional) API operations and object management (Zonal) API operations through the `s3express:CreateSession` action. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone. Unlike object-management actions, bucket management actions cannot be cross-account. Only the bucket owner can perform those actions.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your directory buckets. You can also use IAM to control access to the `CreateSession` API operation, which allows you to use the Zonal, or object management, API operations. You can grant same-account or cross-account access to Zonal API operations. For more information about S3 Express One Zone permissions and policies, see AWS Identity and Access Management (IAM) for S3 Express One Zone.
- IAM Access Analyzer for S3 – Evaluate and monitor your access policies to make sure that the policies provide only the intended access to your S3 resources.

## Logging and monitoring

S3 Express One Zone uses the following S3 logging and monitoring tools that you can use to monitor and control how your resources are being used:

- Amazon CloudWatch metrics – Monitor your AWS resources and applications by using CloudWatch to collect and track metrics. S3 Express One Zone uses the same CloudWatch namespace as other Amazon S3 storage classes (`AWS/S3`) and supports daily storage metrics for directory buckets: `BucketSizeBytes` and `NumberOfObjects`. For more information, see Monitoring metrics with Amazon CloudWatch.
- AWS CloudTrail logs – AWS CloudTrail is an AWS service that helps you implement operational and risk auditing, governance, and compliance of your AWS account by recording the actions taken by a user, role, or an AWS service. For S3 Express One Zone, CloudTrail captures Regional endpoint API operations (for example, `CreateBucket` and `PutBucketPolicy`) as management

events and Zonal API operations (for example, `GetObject` and `PutObject`) as data events. These events include actions taken in the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, and AWS API operations. For more information, see Logging with AWS CloudTrail for S3 Express One Zone.

> **Note**
>
> Amazon S3 server access logs aren't supported with S3 Express One Zone.

## Object management

After you create a directory bucket, you can manage your object storage by using the Amazon S3 console, AWS SDKs, and AWS CLI. The following features are available for object management with S3 Express One Zone:

- S3 Batch Operations – Use Batch Operations to perform bulk operations on objects in directory buckets, for example, **Copy** and **Invoke AWS Lambda function**. For example, you can use Batch Operations to copy objects between directory buckets and general purpose buckets. With Batch Operations, you can manage billions of objects at scale with a single S3 request by using the AWS SDKs or AWS CLI or a few clicks in the Amazon S3 console.
- Import – After you create a directory bucket, you can populate your bucket with objects by using the import feature in the Amazon S3 console. Import is a streamlined method for creating Batch Operations jobs to copy objects from general purpose buckets to directory buckets.

## AWS SDKs and client libraries

After you create a directory bucket and upload an object to your bucket, you can manage your object storage by using the following.

- Mountpoint for Amazon S3 – Mountpoint for Amazon S3 is an open-source file client that delivers high-throughput access, lowering compute costs for data lakes on Amazon S3. Mountpoint for Amazon S3 translates local file system API calls to S3 object API calls like `GET` and `LIST`. It is ideal for read-heavy data lake workloads that process petabytes of data and need the high elastic throughput provided by Amazon S3 to scale up and down across thousands of instances.

- S3A – S3A is a recommended Hadoop-compatible interface for accessing data stores in Amazon S3. S3A replaces the S3N Hadoop file system client.
- PyTorch on AWS – PyTorch on AWS is an open-source deep-learning framework that makes it easier to develop machine learning models and deploy them to production.
- AWS SDKs – You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying Amazon S3 REST API. For more information about using the AWS SDKs with S3 Express One Zone, see the section called "AWS SDKs".

## Encryption and data protection

Objects stored in directory buckets are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). Directory buckets don't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), server-side encryption with customer-provided encryption keys (SSE-C), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information, see Data protection and encryption and Using server-side encryption with Amazon S3 managed keys (SSE-S3).

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

For more information, see S3 additional checksum best practices.

## AWS Signature Version 4 (SigV4)

S3 Express One Zone uses AWS Signature Version 4 (SigV4). SigV4 is a signing protocol used to authenticate requests to Amazon S3 over HTTPS. S3 Express One Zone signs requests by using AWS Sigv4. For more information, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

## Strong consistency

S3 Express One Zone provides strong read-after-write consistency for PUT and DELETE requests of objects in your directory buckets in all AWS Regions. For more information, see Amazon S3 data consistency model.

# Related services

You can use the following AWS services with the S3 Express One Zone storage class to support your specific low-latency use case.

- Amazon Elastic Compute Cloud (Amazon EC2) – Amazon EC2 provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- AWS Lambda – Lambda is a compute service that lets you run code without provisioning or managing servers. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.
- Amazon Elastic Kubernetes Service (Amazon EKS) – Amazon EKS is a managed service that eliminates the need to install, operate, and maintain your own Kubernetes control plane on AWS. Kubernetes is an open-source system that automates the management, scaling, and deployment of containerized applications.
- Amazon Elastic Container Service (Amazon ECS) – Amazon ECS is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications.
- Amazon Athena – Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 by using standard SQL. You can also use Athena to interactively run data analytics by using Apache Spark without having to plan for, configure, or manage resources. When you run Apache Spark applications on Athena, you submit Spark code for processing and receive the results directly.
- Amazon SageMaker Runtime Model Training – Amazon SageMaker Runtime is a fully managed machine learning service. With SageMaker Runtime, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment.
- AWS Glue – AWS Glue is a serverless data-integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue for analytics, machine learning, and application development. AWS Glue also includes additional productivity and data-ops tooling for authoring, running jobs, and implementing business workflows.

- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data.

## Next steps

For more information about working with the S3 Express One Zone storage class and directory buckets, see the following topics:

- How is S3 Express One Zone different?
- Tutorial: Getting started with S3 Express One Zone
- Networking for S3 Express One Zone
- Directory buckets
- Working with objects in a directory bucket
- Security for S3 Express One Zone
- Optimizing Amazon S3 Express One Zone performance
- Developing with S3 Express One Zone

# How is S3 Express One Zone different?

Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. Additionally, to further increase access speed and support hundreds of thousands of requests per second, S3 Express One Zone data is stored in a new bucket type: an Amazon S3 directory bucket.

For more information, see What is S3 Express One Zone? and Directory buckets.

You can create directory buckets and access your data in S3 Express One Zone by using the Amazon S3 API. The Amazon S3 API is compatible with S3 Express One Zone and directory buckets, with the exception of a few notable differences. For more information about how S3 Express One Zone is different, see the following topics.

**Topics**

- S3 Express One Zone differences
- API operations supported by S3 Express One Zone
- Amazon S3 features not supported by S3 Express One Zone

## S3 Express One Zone differences

- **Supported bucket type** – Objects in the S3 Express One Zone storage class can only be stored in directory buckets. For more information, see Directory buckets.
- **Durability** – With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95% availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Single Availability Zone.
- **`ListObjectsV2` behavior**
  - For directory buckets, `ListObjectsV2` does not return objects in lexicographical (alphabetical) order. Additionally, prefixes must end in a delimiter and only "/" can be specified as the delimiter.
  - For directory buckets, `ListObjectsV2` response includes the prefixes that are related only to in-progress multipart uploads.
- **Deletion behavior** – When you delete an object in a directory bucket, Amazon S3 recursively deletes any empty directories in the object path. For example, if you delete the object key `dir1/dir2/file1.txt`, Amazon S3 deletes `file1.txt`. If the `dir1/` and `dir2/` directories are empty and contain no other objects, Amazon S3 also deletes those directories.
- **ETags and checksums** – Entity tags (ETags) for S3 Express One Zone are random alphanumeric strings and not MD5 checksums. For more information about using additional checksums with S3 Express One Zone, see S3 additional checksum best practices.
- **Object keys in `DeleteObjects` requests**
  - Object keys in `DeleteObjects` requests must contain at least one non-white space character. Strings of all white space characters aren't supported in `DeleteObjects` requests.
  - Object keys in `DeleteObjects` requests cannot contain Unicode control characters, except for the newline (`\n`), tab (`\t`), and carriage return (`\r`) characters.
- **Regional and Zonal endpoints** – When using S3 Express One Zone, you must specify the Region in all client requests. For Regional endpoints, you specify the Region, for example, `s3express-control.us-west-2.amazonaws.com`. For Zonal endpoints, you specify both the Region and

the Availability Zone, for example, `s3express-usw2-az1.us-west-2.amazonaws.com`. For more information, see Regional and Zonal endpoints.

- **Multipart uploads** – As with other objects stored in Amazon S3, you can upload and copy large objects that are stored in the S3 Express One Zone storage class by using the multipart upload process. However, the following are some differences when using the multipart upload process with objects stored in S3 Express One Zone. For more information, see the section called "Using multipart uploads with directory buckets".
  - The object creation date is the completion date of the multipart upload.
  - Multipart part numbers must use consecutive part numbers. If you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an HTTP `400 (Bad Request)` error.
  - The initiator of a multipart upload can abort the multipart upload request only if they have been granted explicit allow access to `AbortMultipartUpload` through the `s3express:CreateSession` permission. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.
- **Emptying a directory bucket** – The `s3 rm` command through the AWS Command Line Interface (CLI), the `delete` operation through Mountpoint, and the **Empty** bucket option button through the AWS Management Console are unable to delete in-progress multipart uploads in a directory bucket. To delete these in-progress multipart uploads, use the `ListMultipartUploads` operation to list the in-progress multipart uploads in the bucket and use the `AbortMultupartUpload` operation to abort all the in-progress multipart uploads.

## API operations supported by S3 Express One Zone

The Amazon S3 Express One Zone storage class supports both Regional (bucket level, or control plane) and Zonal (object level, or data plane) endpoint API operations. For more information, see Networking for S3 Express One Zone and Endpoints and gateway VPC endpoints.

### Regional endpoint API operations

The following Regional endpoint API operations are supported for S3 Express One Zone:

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy

- ListDirectoryBuckets
- PutBucketPolicy

### Zonal endpoint API operations

The following Zonal endpoint API operations are supported for S3 Express One Zone:

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

## Amazon S3 features not supported by S3 Express One Zone

The following Amazon S3 features are not supported by S3 Express One Zone:

- AWS managed policies
- AWS PrivateLink for S3
- MD5 checksums
- Multi-factor authentication (MFA) delete

- S3 Object Lock
- Requester Pays
- S3 Access Grants
- S3 Access Points
- Bucket tags
- Amazon CloudWatch request metrics
- S3 Event Notifications
- S3 Lifecycle
- S3 Multi-Region Access Points
- S3 Object Lambda Access Points
- S3 Versioning
- S3 Inventory
- S3 Replication
- Object tags
- S3 Select
- Server access logs
- Static website hosting
- S3 Storage Lens
- S3 Storage Lens groups
- S3 Transfer Acceleration
- Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)
- The option to copy an existing bucket settings when creating a new bucket in AWS Management Console.

# Tutorial: Getting started with S3 Express One Zone

Amazon S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources which provides

the highest possible access speed. Data in S3 Express One Zone is stored in S3 directory buckets. For more information on directory buckets, see Directory buckets.

S3 Express One Zone is ideal for any application where it's critical to minimize request latency. Such applications can be human-interactive workflows, like video editing, where creative professionals need responsive access to content from their user interfaces. S3 Express One Zone also benefits analytics and machine learning workloads that have similar responsiveness requirements from their data, especially workloads with a lot of smaller accesses or a large numbers of random accesses. S3 Express One Zone can be used with other AWS services such as Amazon EMR, Amazon Athena, AWS Glue Data Catalog and Amazon SageMaker Model Training to support analytics, artificial intelligence and machine learning (AI/ML) workloads,. You can work with the S3 Express One Zone storage class and directory buckets by using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), and Amazon S3 REST API. For more information, see What is S3 Express One Zone? and How is S3 Express One Zone different?.
This is an S3 Express One Zone workflow diagram.

**Objective**

In this tutorial, you will learn how to create a gateway endpoint, create and attach an IAM policy, create a directory bucket and then use the Import action to populate your directory bucket with objects currently stored in your general purpose bucket. Alternatively, you can manually upload objects to your directory bucket.

**Topics**

# Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions.

**Substeps**

- Create an AWS account
- Create an IAM user in your AWS account (console)
- Create an IAM policy and attach it to an IAM user or role (console)

## Create an AWS account

To complete this tutorial, you need an AWS account. When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. You are charged only for the services that you use. For more information about pricing, see S3 pricing.

## Create an IAM user in your AWS account (console)

AWS Identity and Access Management (IAM) is an AWS service that helps administrators securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to access objects and use directory buckets in S3 Express One Zone. You can use IAM for no additional charge.

By default, users don't have permissions to access directory buckets and perform S3 Express One Zone operations. To grant access permissions for directory buckets and S3 Express One Zone operations, you can use IAM to create users or roles and attach permissions to those identities. For more information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*. For more information about how to create an IAM role, see Creating a role to delegate permissions to an IAM user in the *IAM User Guide*.

For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn more about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*.

> **Warning**
>
> IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

## Create an IAM policy and attach it to an IAM user or role (console)

By default, users don't have permissions for directory buckets and S3 Express One Zone operations. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. Directory buckets are the only resource that you can include in bucket policies or IAM identity policies for S3 Express One Zone access.

To use Regional endpoint API operations (bucket-level or control plane operations) with S3 Express One Zone, you use the IAM authorization model, which doesn't involve session management. Permissions are granted for actions individually. To use Zonal endpoint API operations (object-level or data plane operations), you use `CreateSession` to create and manage sessions that are optimized for low-latency authorization of data requests. To retrieve and use a session token, you must allow the `s3express:CreateSession` action for your directory bucket in an identity-based policy or a bucket policy. If you're accessing S3 Express One Zone in the Amazon S3 console, through the AWS Command Line Interface (AWS CLI), or by using the AWS SDKs, S3 Express One Zone creates a session on your behalf. For more information, see `CreateSession` authorization and AWS Identity and Access Management (IAM) for S3 Express One Zone .

**To create an IAM policy and attach the policy to an IAM user (or role)**

1. Sign in to the AWS Management Console and open the IAM Management Console.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**.
4. Select **JSON**.
5. Copy the policy below into the **Policy editor** window. Before you can create directory buckets or use S3 Express One Zone, you must grant the necessary permissions to your AWS Identity and Access Management (IAM) role or users. This example policy allows access to the `CreateSession` API operation (for use with other Zonal or object-level API operations) and all of the Regional endpoint (bucket-level) API operations. This policy allows the `CreateSession` API operation for use with all directory buckets, but the Regional

endpoint API operations are allowed only for use with the specified directory bucket. To use this example policy, replace the *user input placeholders* with your own information.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-name--azid--x-s3/*"
         },
         {
             "Sid": "AllowCreateSession",
             "Effect": "Allow",
             "Action": "s3express:CreateSession",
             "Resource": "*"
         }
     ]
 }
```

6. Choose **Next**.
7. Name the policy.

> **Note**
>
> Bucket tags are not supported for S3 Express One Zone.

8. Select **Create policy**.
9. Now that you've created an IAM policy, you can attach it to an IAM user. In the navigation pane, choose **Policies**.
10. In the **search bar**, enter the name of your policy.

11. From the **Actions** menu, select **Attach**.
12. Under **Filter by Entity Type**, select **IAM users** or **Roles**.
13. In the **search field**, type the name of the user or role you wish to use.
14. Choose **Attach Policy**.

## Step 1: Configure a gateway VPC endpoint

You can access both Zonal and Regional API operations through gateway virtual private cloud (VPC) endpoints. Gateway endpoints can allow traffic to reach S3 Express One Zone without traversing a NAT Gateway. We strongly recommend using gateway endpoints as they provide the most optimal networking path when working with S3 Express One Zone. You can access S3 Express One Zone directory buckets from your VPC without an internet gateway or NAT device for your VPC, and at no additional cost. Use the following procedure to configure a gateway endpoint that connects to S3 Express One Zone storage class objects and directory buckets.

To access S3 Express One Zone, you use Regional and Zonal endpoints that are different from standard Amazon S3 endpoints. Depending on the Amazon S3 API operation that you use, either a Zonal or Regional endpoint is required. For a complete list of supported API operations by endpoint type, see API operations supported by S3 Express One Zone . You must access both Zonal and Regional endpoints through a gateway virtual private cloud (VPC) endpoint.

Use the following procedure to create a gateway endpoint that connects to S3 Express One Zone storage class objects and directory buckets.

**To configure a gateway VPC endpoint**

1. Open the Amazon VPC Console at https://console.aws.amazon.com/vpc/.
2. In the side navigation pane under **Virtual private cloud**, choose **Endpoints**.
3. Choose **Create endpoint**.
4. Create a name for your endpoint.
5. For **Service category**, choose **AWS services**.
6. Under **Services**, search using the filter **Type=Gateway** and then choose the option button next to **com.amazonaws.*region*.s3express**.
7. For **VPC**, choose the VPC in which to create the endpoint.

8. For **Route tables**, select the route tables to be used by the endpoint. Amazon VPC automatically adds a route that points traffic destined for the service to the endpoint network interface.
9. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the permissions that principals have to perform actions on resources over the VPC endpoint.
10. Choose **Create endpoint**.

After creating a gateway endpoint, you can use Regional API endpoints and Zonal API endpoints to access Amazon S3 Express One Zone storage class objects and directory buckets.

## Step 2: Create a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   > **Note**
   >
   > To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **Directory**.

   > **Note**
   >
   > - If you've chosen a Region that doesn't support directory buckets, the **Bucket type** option disappears, and the bucket type defaults to a general purpose bucket. To

create a directory bucket, you must choose a supported Region. For a list of Regions that support directory buckets and the Amazon S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".

- After you create the bucket, you can't change the bucket type.

For **Availability Zone**, choose a Availability Zone local to your compute services. For a list of Availability Zones that support directory buckets and the S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".

> **Note**
>
> The Availability Zone can't be changed after the bucket is created.

7. Under **Availability Zone** , select the check box to acknowledge that in the event of an Availability Zone outage, your data might be unavailable or lost.

> **Important**
>
> Although directory buckets are stored across multiple devices within a single Availability Zone, directory buckets don't store data redundantly across Availability Zones.

8. For **Bucket name**, enter a name for your directory bucket.

   The following naming rules apply for directory buckets.

   - Be unique within the chosen AWS Region and Availability Zone.
   - Name must be between 3 (min) and 63 (max) characters long, including the suffix.
   - Consists only of lowercase letters, numbers and hyphens (-).
   - Begin and end with a letter or number.
   - Must include the following suffix: `--azid--x-s3`.
   - Bucket names must not start with the prefix `xn--`.
   - Bucket names must not start with the prefix `sthree-`.
   - Bucket names must not start with the prefix `sthree-configurator`.
   - Bucket names must not start with the prefix `amzn-s3-demo-`.

- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

A suffix is automatically added to the base name that you provide when you create a directory bucket using the console. This suffix includes the Availability Zone ID of the Availability Zone that you chose.

After you create the bucket, you can't change its name. For more information about naming buckets, see Bucket naming rules.

> **Important**
>
> Do not include sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

9. Under **Object Ownership**, the **Bucket owner enforced** setting is automatically enabled, and all access control lists (ACLs) are disabled. For directory buckets, ACLs can't be enabled.

   **ACLs disabled**

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

     A majority of modern use cases in Amazon S3 no longer require the use of ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

10. Under **Block Public Access settings for this bucket**, all Block Public Access settings for your directory bucket are automatically enabled. These settings can't be modified for directory

buckets. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

11. Under **Server-side encryption settings**, Amazon S3 applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for all S3 buckets. All object uploads to directory buckets are encrypted with SSE-S3. For directory buckets, the encryption type can't be modified. For more information about SSE-S3, see the section called "Amazon S3 managed encryption keys (SSE-S3)".
12. Choose **Create bucket**.

    After creating the bucket, you can add files and folders to the bucket. For more information, see the section called "Working with objects in a directory bucket".

The following step demonstrates how to use the Import action in the Amazon S3 console to populate your directory bucket with data.

## Step 3: Importing data into a directory bucket

To complete this step, you must have a general purpose bucket that contains objects and is located in the same AWS Region as your directory bucket.

After you create a directory bucket in Amazon S3, you can populate the new bucket with data by using the Import action in the Amazon S3 console. Import simplifies copying data into directory buckets by letting you choose a prefix or a general purpose bucket to Import data from without having to specify all of the objects to copy individually. Import uses S3 Batch Operations which copies the objects in the selected prefix or general purpose bucket. You can monitor the progress of the Import copy job through the S3 Batch Operations job details page.

**To use the Import action**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Buckets**, and then choose the **Directory buckets** tab. Choose the directory bucket that you want to import objects into.
4. Choose **Import**.

5. For **Source**, enter the general purpose bucket (or bucket path including prefix) that contains the objects that you want to import. To choose an existing general purpose bucket from a list, choose **Browse S3**.
6. In the **Permissions** section, you can choose to have an IAM role auto-generated. Alternatively, you can select an IAM role from a list, or directly enter an IAM role ARN.

   - To allow Amazon S3 to create a new IAM role on your behalf, choose **Create new IAM role**.

     > **Note**
     >
     > If your source objects are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), don't choose the **Create new IAM role** option. Instead, specify an existing IAM role that has the `kms:Decrypt` permission.
     > Amazon S3 will use this permission to decrypt your objects. During the import process, Amazon S3 will then re-encrypt those objects by using server-side encryption with Amazon S3 managed keys (SSE-S3).

   - To choose an existing IAM role from a list, choose **Choose from existing IAM roles**.
   - To specify an existing IAM role by entering its Amazon Resource Name (ARN), choose **Enter IAM role ARN**, then enter the ARN in the corresponding field.
7. Review the information that's displayed in the **Destination** and **Copied object settings** sections. If the information in the **Destination** section is correct, choose **Import** to start the copy job.

   The Amazon S3 console displays the status of your new job on the **Batch Operations** page. For more information about the job, choose the option button next to the job name, and then on the **Actions** menu, choose **View details**. To open the directory bucket that the objects will be imported into, choose **View import destination**.

## Step 4: Manually upload objects to your directory bucket

You can also manually upload objects to your directory bucket.

**To manually upload objects**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the upper right corner of the page, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Buckets**.
4. Choose the **Directory buckets** tab.
5. Choose the name of the bucket that you want to upload your folders or files to.

   > **Note**
   >
   > If you chose the same directory bucket that you used in previous steps of this tutorial, your directory bucket will contain the objects that were uploaded from the Import tool. Notice that these objects are now stored in the S3 Express One Zone storage class.

6. In the **Objects** list, choose **Upload**.
7. On the **Upload** page, do one of the following:

   - Drag and drop files and folders to the dotted upload area.
   - Choose **Add files** or **Add folder**, choose the files or folders to upload, and then choose **Open** or **Upload**.

8. Under **Checksums**, choose the **Checksum function** that you want to use.

   > **Note**
   >
   > We recommend using CRC32 and CRC32C for the best performance with the S3 Express One Zone storage class. For more information, see S3 additional checksum best practices.

   (Optional) If you're uploading a single object that's less than 16 MB in size, you can also specify a pre-calculated checksum value. When you provide a pre-calculated value, Amazon S3 compares it with the value that it calculates by using the selected checksum function. If the values don't match, the upload won't start.

9. The options in the **Permissions** and **Properties** sections are automatically set to default settings and can't be modified. Block Public Access is automatically enabled, and S3 Versioning and S3 Object Lock can't be enabled for directory buckets.

   (Optional) If you want to add metadata in key-value pairs to your objects, expand the **Properties** section, and then in the **Metadata** section, choose **Add metadata**.

10. To upload the listed files and folders, choose **Upload**.

    Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

    You have successfully created a directory bucket and uploaded objects to your bucket.

## Step 5: Empty your directory bucket

You can empty your Amazon S3 directory bucket by using the Amazon S3 console.

**To empty a directory bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the upper right corner of the page, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Buckets**.
4. Choose the **Directory buckets** tab.
5. Choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
6. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.
7. Monitor the progress of the bucket emptying process on the **Empty bucket: status** page.

## Step 6: Delete your directory bucket

After you empty your directory bucket and abort all in-progress multipart uploads, you can delete your bucket by using the Amazon S3 console.

**To delete a directory bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the upper right corner of the page, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Buckets**.
4. Choose the **Directory buckets** tab.
5. In the **Directory buckets** list, choose the option button next to the bucket that you want to delete.
6. Choose **Delete**.
7. On the **Delete bucket** page, enter the name of the bucket in the text field to confirm the deletion of your bucket.

   > **Important**
   >
   > Deleting a directory bucket can't be undone.

8. To delete your directory bucket, choose **Delete bucket**.

## Next steps

In this tutorial, you have learned how to create a directory bucket and use the S3 Express One Zone storage class. After completing this tutorial, you can explore related AWS services to use with the S3 Express One Zone storage class.

You can use the following AWS services with the S3 Express One Zone storage class to support your specific low-latency use case.

- Amazon Elastic Compute Cloud (Amazon EC2) – Amazon EC2 provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

- AWS Lambda – Lambda is a compute service that lets you run code without provisioning or managing servers. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.
- Amazon Elastic Kubernetes Service (Amazon EKS) – Amazon EKS is a managed service that eliminates the need to install, operate, and maintain your own Kubernetes control plane on AWS. Kubernetes is an open-source system that automates the management, scaling, and deployment of containerized applications.
- Amazon Elastic Container Service (Amazon ECS) – Amazon ECS is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications.
- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark on AWS to process and analyze vast amounts of data.
- Amazon Athena – Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 by using standard SQL. You can also use Athena to interactively run data analytics by using Apache Spark without having to plan for, configure, or manage resources. When you run Apache Spark applications on Athena, you submit Spark code for processing and receive the results directly.
- AWS Glue Data Catalog – AWS Glue is a serverless data-integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue for analytics, machine learning, and application development. AWS Glue Data Catalog is a centralized repository that stores metadata about your organization's data sets. It acts as an index to the location, schema, and run-time metrics of your data sources.
- Amazon SageMaker Runtime Model Training – Amazon SageMaker Runtime is a fully managed machine learning service. With SageMaker Runtime, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment.

For more information on S3 Express One Zone, see What is S3 Express One Zone? and How is S3 Express One Zone different?.

## Networking for S3 Express One Zone

To access Amazon S3 Express One Zone storage class objects and directory buckets, you use Regional and Zonal API endpoints that are different from the standard Amazon S3 endpoints. Depending on the S3 API operation that you use, either a Zonal or Regional endpoint is required.

For a complete list of API operations by endpoint type, see API operations supported by S3 Express One Zone.

You can access both Zonal and Regional API operations through gateway virtual private cloud (VPC) endpoints. To configure gateway VPC endpoints, see the section called "Configuring VPC gateway endpoints".

The following topics describe the networking requirements for accessing S3 Express One Zone by using a gateway VPC endpoint.

**Topics**



## Endpoints

You can access Amazon S3 Express One Zone storage class objects and directory buckets from your VPC by using gateway VPC endpoints. S3 Express One Zone uses Regional and Zonal API endpoints. Depending on the Amazon S3 API operation that you use, either a Regional or Zonal endpoint is required. There is no additional charge for using gateway endpoints.

Bucket-level (or control plane) API operations are available through Regional endpoints and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`. When you create a directory bucket, you choose a single Availability where your directory bucket will be created. After you create a directory bucket, you can use Zonal endpoint API operations to upload and manage the objects in your directory bucket.

Object-level (or data plane) API operations are available through Zonal endpoints and are referred to as Zonal endpoint API operations. Examples of Zonal endpoint API operations are `CreateSession` and `PutObject`.

The following table shows the Regional and Zonal API endpoints that are available for each Region and Availability Zone.

## Configuring VPC gateway endpoints

Use the following procedure to create a gateway endpoint that connects to Amazon S3 Express One Zone storage class objects and directory buckets.

**To configure a gateway VPC endpoint**

1. Open the Amazon VPC Console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. Create a name for your endpoint.
5. For **Service category**, choose **AWS services**.
6. For **Services**, add the filter **Type=Gateway** and then choose the option button next to **com.amazonaws.*region*.s3express**.
7. For **VPC**, choose the VPC in which to create the endpoint.
8. For **Route tables**, select the route tables to be used by the endpoint. Amazon VPC automatically adds a route that points traffic destined for the service to the endpoint network interface.
9. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the permissions that principals have to perform actions on resources over the VPC endpoint.
10. (Optional) To add a tag, choose **Add new tag**, and enter the tag key and the tag value.
11. Choose **Create endpoint**.

After creating a gateway endpoint, you can use Regional API endpoints and Zonal API endpoints to access Amazon S3 Express One Zone storage class objects and directory buckets.

# Directory buckets

There are two types of Amazon S3 buckets, general purpose buckets and directory buckets. Choose the bucket type that best fits your application and performance requirements:

- **General purpose buckets** are the original S3 bucket type and are recommended for most use cases and access patterns. General purpose buckets also allow objects that are stored across all storage classes, except S3 Express One Zone.
- **Directory buckets** use the S3 Express One Zone storage class, which is recommended if your application is performance sensitive and benefits from single-digit millisecond PUT and GET latencies.

Directory buckets are used for workloads or performance-critical applications that require consistent single-digit millisecond latency. Directory buckets organize data hierarchically into directories as opposed to the flat storage structure of general purpose buckets. There aren't prefix limits for directory buckets, and individual directories can scale horizontally.

Directory buckets use the S3 Express One Zone storage class, which stores data across multiple devices within a single Availability Zone but doesn't store data redundantly across Availability Zones. When you create a directory bucket, we recommend that you specify an AWS Region and an Availability Zone that's local to your Amazon EC2, Amazon Elastic Kubernetes Service, or Amazon Elastic Container Service (Amazon ECS) compute instances to optimize performance.

You can create up to 10 directory buckets in each of your AWS accounts, with no limit on the number of objects that you can store in a bucket. Your bucket quota is applied to each Region in your AWS account. If your application requires increasing this limit, contact AWS Support. For more information, visit the Service Quotas console.

**Important**

Directory buckets that have no request activity for a period of at least 90 days transition to an inactive state. While in an inactive state, a directory bucket is temporarily inaccessible for reads and writes. Inactive buckets retain all storage, object metadata, and bucket metadata. Existing storage charges apply to inactive buckets. If you make an access request to an inactive bucket, the bucket transitions to an active state, typically within a few minutes. During this transition period, reads and writes return an HTTP `503 (Service Unavailable)` error code.

The following topics provide information about directory buckets. For more information about general purpose buckets, see Buckets overview.

**Topics**

- Availability Zones
- Directory bucket names
- Directories
- Key names
- Access management
- Working with directory buckets

- Directory bucket naming rules
- Creating a directory bucket
- Viewing directory bucket properties
- Managing bucket policies for directory buckets
- Emptying a directory bucket
- Deleting a directory bucket
- Listing directory buckets
- Using HeadBucket with directory buckets

## Availability Zones

When you create a directory bucket, you choose the Availability Zone and AWS Region.

Directory buckets use the S3 Express One Zone storage class, which is built to be used by performance-sensitive applications. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed.

With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Single Availability Zone

## Directory bucket names

A directory bucket name consists of a base name that you provide and a suffix that contains the ID of the Availability Zone that your bucket is located in. Directory bucket names must use the following format and follow the naming rules for directory buckets:

```
bucket-base-name--azid--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

For more information, see Directory bucket naming rules.

## Directories

Directory buckets organize data hierarchically into directories as opposed to the flat sorting structure of general purpose buckets. Each S3 directory bucket can support hundreds of thousands of transactions per second (TPS), independent of the number of directories within the bucket.

With a hierarchical namespace, the delimiter in the object key is important. The only supported delimiter is a forward slash (/). Directories are determined by delimiter boundaries. For example, the object key `dir1/dir2/file1.txt` results in the directories `dir1/` and `dir2/` being automatically created, and the object `file1.txt` being added to the `/dir2` directory in the path `dir1/dir2/file1.txt`.

The directory bucket indexing model returns unsorted results for the `ListObjectsV2` API operation. If you need to limit your results to a subsection of your bucket, you can specify a subdirectory path in the `prefix` parameter, for example, `prefix=dir1/`.

## Key names

For directory buckets, subdirectories that are common to multiple object keys are created with the first object key. Additional object keys for the same subdirectory use the previously created subdirectory. This model gives you flexibility in choosing object keys that are best suited to the application, with equal support for sparse and dense directories.

## Access management

Directory buckets have all S3 Block Public Access settings enabled by default at the bucket level. S3 Object Ownership is set to bucket owner enforced and access control lists (ACLs) are disabled. These settings can't be modified.

By default, users don't have permissions for directory buckets and S3 Express One Zone operations. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

## Working with directory buckets

For more information about working with directory buckets, see the following topics.

### Topics

- Directory bucket naming rules

- Creating a directory bucket
- Viewing directory bucket properties
- Managing bucket policies for directory buckets
- Emptying a directory bucket
- Deleting a directory bucket
- Listing directory buckets
- Using HeadBucket with directory buckets

## Directory bucket naming rules

When you create a directory bucket in Amazon S3, the following bucket naming rules apply. For general purpose bucket naming rules, see Bucket naming rules.

A directory bucket name consists of a base name that you provide, and a suffix that contains the ID of the AWS Availability Zone that your bucket is located in and `--x-s3`.

```
base-name--azid--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> When you create a directory bucket by using the console a suffix is automatically added to the base name that you provide. This suffix includes the Availability Zone ID of the Availability Zone that you chose.
> When you create a directory bucket by using an API you must provide the full suffix, including the Availability Zone ID, in your request. For a list of Availability Zone IDs, see S3 Express One Zone Availability Zones and Regions.

The following naming rules apply for directory buckets.

- Be unique within the chosen AWS Region and Availability Zone.
- Name must be between 3 (min) and 63 (max) characters long, including the suffix.

- Consists only of lowercase letters, numbers and hyphens (-).
- Begin and end with a letter or number.
- Must include the following suffix: `--azid--x-s3`.
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `sthree-configurator`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

## Creating a directory bucket

To start using the Amazon S3 Express One Zone storage class, you create a directory bucket. The S3 Express One Zone storage class can be used only with directory buckets. The S3 Express One Zone storage class supports low-latency use cases and provides faster data processing within a single Availability Zone. If your application is performance sensitive and benefits from single-digit millisecond PUT and GET latencies, we recommend creating a directory bucket so that you can use the S3 Express One Zone storage class.

There are two types of Amazon S3 buckets, general purpose buckets and directory buckets. You should choose the bucket type that best fits your application and performance requirements. General purpose buckets are the original S3 bucket type. General purpose buckets are recommended for most use cases and access patterns and allow objects stored across all storage classes, except S3 Express One Zone. For more information about general purpose buckets, see Buckets overview.

Directory buckets use the S3 Express One Zone storage class, which is designed to be used for workloads or performance-critical applications that require consistent single-digit millisecond latency. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which

provides the highest possible access speed. When you create a directory bucket, you can optionally specify an AWS Region and an Availability Zone that's local to your Amazon EC2, Amazon Elastic Kubernetes Service, or Amazon Elastic Container Service (Amazon ECS) compute instances to optimize performance.

With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Single Availability Zone

Directory buckets organize data hierarchically into directories, as opposed to the flat storage structure of general purpose buckets. There aren't prefix limits for directory buckets, and individual directories can scale horizontally.

For more information about directory buckets, see Directory buckets.

**Directory bucket names**

Directory bucket names must follow this format and comply with the rules for directory bucket naming:

```
bucket-base-name--azid--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

For more information about directory bucket naming rules, see Directory bucket naming rules.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

> **Note**
>
> To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly

transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **Directory**.

> **Note**
>
> - If you've chosen a Region that doesn't support directory buckets, the **Bucket type** option disappears, and the bucket type defaults to a general purpose bucket. To create a directory bucket, you must choose a supported Region. For a list of Regions that support directory buckets and the Amazon S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".
> - After you create the bucket, you can't change the bucket type.

For **Availability Zone**, choose a Availability Zone local to your compute services. For a list of Availability Zones that support directory buckets and the S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".

> **Note**
>
> The Availability Zone can't be changed after the bucket is created.

7. Under **Availability Zone** , select the check box to acknowledge that in the event of an Availability Zone outage, your data might be unavailable or lost.

> **Important**
>
> Although directory buckets are stored across multiple devices within a single Availability Zone, directory buckets don't store data redundantly across Availability Zones.

8. For **Bucket name**, enter a name for your directory bucket.

   The following naming rules apply for directory buckets.

   - Be unique within the chosen AWS Region and Availability Zone.
   - Name must be between 3 (min) and 63 (max) characters long, including the suffix.
   - Consists only of lowercase letters, numbers and hyphens (-).
   - Begin and end with a letter or number.
   - Must include the following suffix: `--`*`azid`*`--x-s3`.
   - Bucket names must not start with the prefix `xn--`.
   - Bucket names must not start with the prefix `sthree-`.
   - Bucket names must not start with the prefix `sthree-configurator`.
   - Bucket names must not start with the prefix `amzn-s3-demo-`.
   - Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.
   - Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
   - Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

   A suffix is automatically added to the base name that you provide when you create a directory bucket using the console. This suffix includes the Availability Zone ID of the Availability Zone that you chose.

   After you create the bucket, you can't change its name. For more information about naming buckets, see Bucket naming rules.

   > **Important**
   >
   > Do not include sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

9. Under **Object Ownership**, the **Bucket owner enforced** setting is automatically enabled, and all access control lists (ACLs) are disabled. For directory buckets, ACLs can't be enabled.

   **ACLs disabled**

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

     A majority of modern use cases in Amazon S3 no longer require the use of ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

10. Under **Block Public Access settings for this bucket**, all Block Public Access settings for your directory bucket are automatically enabled. These settings can't be modified for directory buckets. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
11. Under **Server-side encryption settings**, Amazon S3 applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for all S3 buckets. All object uploads to directory buckets are encrypted with SSE-S3. For directory buckets, the encryption type can't be modified. For more information about SSE-S3, see the section called “Amazon S3 managed encryption keys (SSE-S3)”.
12. Choose **Create bucket**.

    After creating the bucket, you can add files and folders to the bucket. For more information, see the section called “Working with objects in a directory bucket”.

### Using the AWS SDKs

SDK for Go

This example shows how to create a directory bucket by using the AWS SDK for Go.

**Example**

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
```

```
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

This example shows how to create an directory bucket by using the AWS SDK for Java 2.x.

**Example**

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.AVAILABILITY_ZONE)
                     .name("usw2-az1").build()) //this must match the Region and
 Availability Zone in your bucket name
```

```
            .bucket(BucketInfo.builder()
                   .type(BucketType.DIRECTORY)
                   .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                   .build()).build();
    try {

            CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
    }

    catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }
    }
```

AWS SDK for JavaScript

This example shows how to create a directory bucket by using the AWS SDK for JavaScript.

**Example**

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
```

```
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

AWS SDK for .NET

This example shows how to create a directory bucket by using the AWS SDK for .NET.

**Example**

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

This example shows how to create a directory bucket by using the AWS SDK for PHP.

**Example**

```
require 'vendor/autoload.php';

$s3Client = new S3Client([

    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
```

```
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
]);
```

SDK for Python

This example shows how to create a directory bucket by using the AWS SDK for Python (Boto3).

**Example**

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
```

```
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

This example shows how to create an directory bucket by using the AWS SDK for Ruby.

**Example**

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

### Using the AWS CLI

This example shows how to create a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a directory bucket you must provide configuration details and use the following naming convention: *`bucket-base-name`*`--`*`azid`*`--x-s3`

```
aws s3api create-bucket
--bucket bucket-base-name--azid--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
```

```
--region us-west-2
```

For more information, see create-bucket in the AWS Command Line Interface.

## Viewing directory bucket properties

You can view and configure the properties for an Amazon S3 directory bucket by using the Amazon S3 console. For more information, see Directory buckets and What is S3 Express One Zone?.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. In the **Directory buckets** list, choose the name of the bucket that you want to view the properties for.
5. Choose the **Properties** tab.
6. On the **Properties** tab, you can view the following properties for the bucket:

   - **Directory bucket overview** – You can see the AWS Region, Availability Zone, Amazon Resource Name (ARN), and creation date for the bucket.
   - **Default encryption** – Amazon S3 applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for all S3 buckets. For directory buckets, this setting can't be modified. Amazon S3 encrypts an object before saving it to a disk and decrypts the object when you download it. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

     For more information about supported features for directory buckets, see Features of S3 Express One Zone.

## Managing bucket policies for directory buckets

You can add, delete, update, and view bucket policies for Amazon S3 directory buckets by using the Amazon S3 console and the AWS SDKs. For more information, see the following topics. For more information about supported AWS Identity and Access Management (IAM) actions and condition keys for S3 Express One Zone, see AWS Identity and Access Management (IAM) for S3 Express One

Zone. For example bucket policies for directory buckets, see Example directory bucket policies for S3 Express One Zone.

**Topics**



## Adding a bucket policy

To add a bucket policy to a directory bucket, you can use the Amazon S3 console, the AWS SDKs, or the AWS CLI.

### Using the S3 console

**To create or edit a bucket policy**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. In the **Directory buckets** list, choose the name of the bucket that you want to upload your folders or files to.
5. Choose the **Permissions** tab.
6. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.
7. To generate a policy automatically, choose **Policy generator**.

   If you choose **Policy generator**, the AWS Policy Generator opens in a new window.

   If you don't want to use the AWS Policy Generator, you can add or edit JSON statements in the **Policy** section.

   a. On the **AWS Policy Generator** page, for **Select Type of Policy**, choose **S3 Bucket Policy**.

   b. Add a statement by entering the information in the provided fields, and then choose **Add Statement**. Repeat this step for as many statements as you want to add. For more information about these fields, see the IAM JSON policy elements reference in the *IAM User Guide*.

> **Note**
>
> For your convenience, the **Edit bucket policy** page displays the **Bucket ARN** (Amazon Resource Name) of the current bucket above the **Policy** text field. You can copy this ARN for use in the statements on the **AWS Policy Generator** page.

c. After you finish adding statements, choose **Generate Policy**.

d. Copy the generated policy text, choose **Close**, and return to the **Edit bucket policy** page in the Amazon S3 console.

8. In the **Policy** box, edit the existing policy or paste the bucket policy from the AWS Policy Generator. Make sure to resolve security warnings, errors, general warnings, and suggestions before you save your policy.

> **Note**
>
> Bucket policies are limited to 20 KB in size.

9. Choose **Save changes**, which returns you to the **Permissions** tab.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

PutBucketPolicy AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

      //sample policy text
      /**
       * policy_statement = {
       *         'Version': '2012-10-17',
       *         'Statement': [
       *             {
       *                 'Sid': 'AdminPolicy',
       *                 'Effect': 'Allow',
       *                 'Principal': {
```

```
        *                     "AWS": "111122223333"
        *                 },
        *                 'Action': 's3express:*',
        *                 'Resource':
'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--azid--x-s3'
        *             }
        *         ]
        *     }
        */
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();
            s3Client.putBucketPolicy(policyReq);
            System.out.println("Done!");
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

### Using the AWS CLI

This example shows how to add a bucket policy to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api put-bucket-policy --bucket bucket-base-name--azid--x-s3 --policy file://
bucket_policy.json
```

bucket_policy.json:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/"
        }
    ]
}
```

For more information, see put-bucket-policy in the AWS Command Line Interface.

## Viewing a bucket policy

To view a bucket policy for a directory bucket, use the following examples.

### Using the AWS CLI

This example shows how to view the bucket policy attached to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api get-bucket-policy --bucket bucket-base-name--azid--x-s3
```

For more information, see get-bucket-policy in the AWS Command Line Interface.

## Deleting a bucket policy

To delete a bucket policy for a directory bucket, use the following examples.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
      try {
          DeleteBucketPolicyRequest deleteBucketPolicyRequest =
 DeleteBucketPolicyRequest
                  .builder()
                  .bucket(bucketName)
                  .build()
          s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
          System.out.println("Successfully deleted bucket policy");
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
```

#### Using the AWS CLI

This example shows how to delete a bucket policy for a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api delete-bucket-policy --bucket bucket-base-name--azid--x-s3
```

For more information, see delete-bucket-policy in the AWS Command Line Interface.

## Emptying a directory bucket

You can empty an Amazon S3 directory bucket by using the Amazon S3 console. For more information about directory buckets, see Directory buckets.

Before you empty a directory bucket, note the following:

- When you empty a directory bucket, you delete all the objects, but you keep the directory bucket.
- After you empty a directory bucket, the empty action can't be undone.
- Objects that are added to the directory bucket while the empty bucket action is in progress might be deleted.

If you also want to delete the bucket, note the following:

- All objects in the directory bucket must be deleted before the bucket itself can be deleted.
- In-progress multipart uploads in the directory bucket must be aborted before the bucket itself can be deleted.

> **Note**
>
> The `s3 rm` command through the AWS Command Line Interface (CLI), the `delete` operation through Mountpoint, and the **Empty** bucket option button through the AWS Management Console are unable to delete in-progress multipart uploads in a directory bucket. To delete these in-progress multipart uploads, use the `ListMultipartUploads` operation to list the in-progress multipart uploads in the bucket and use the `AbortMultipartUpload` operation to abort all the in-progress multipart uploads.

To delete a directory bucket, see Deleting a directory bucket. To abort an in-progress multipart upload, see the section called "Aborting a multipart upload".

To empty a general purpose bucket, see Emptying a bucket.

**Using the S3 console**

**To empty a directory bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. Choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
5. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.
6. Monitor the progress of the bucket emptying process on the **Empty bucket: status** page.

## Deleting a directory bucket

You can delete only empty Amazon S3 directory buckets. Before you delete your directory bucket, you must delete all objects in the bucket and abort all in-progress multipart uploads.

To empty a directory bucket, see Emptying a directory bucket. To abort an in-progress multipart upload, see the section called "Aborting a multipart upload".

To delete a general purpose bucket, see Deleting a bucket.

### Using the S3 console

After you empty your directory bucket and abort all in-progress multipart uploads, you can delete your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. In the **Directory buckets** list, choose the option button next to the bucket that you want to delete.
5. Choose **Delete**.
6. On the **Delete bucket** page, enter the name of the bucket in the text field to confirm the deletion of your bucket.

   > **Important**
   >
   > Deleting a directory bucket can't be undone.

7. To delete your directory bucket, choose **Delete bucket**.

### Using the AWS SDKs

The following examples delete a directory bucket by using the AWS SDK for Java 2.x and AWS SDK for Python (Boto3).

SDK for Java 2.x

**Example**

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
```

```
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

**Example**

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

### Using the AWS CLI

This example shows how to delete a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api delete-bucket --bucket bucket-base-name--azid--x-s3 --region us-west-2
```

For more information, see delete-bucket in the AWS Command Line Interface.

## Listing directory buckets

The following examples show how to list directory buckets by using the AWS SDKs and AWS CLI.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

The following example lists directory buckets by using the AWS SDK for Java 2.x.

```
public static void listBuckets(S3Client s3Client) {
       try {
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();
           ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
           if (response.hasBuckets()) {
               for (Bucket bucket: response.buckets()) {
                   System.out.println(bucket.name());
                   System.out.println(bucket.creationDate());
               }
           }
       }

       catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

SDK for Python

**Example**

The following example lists directory buckets by using the AWS SDK for Python (Boto3).

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region

:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
return True


if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

AWS SDK for .NET

**Example**

The following example lists directory buckets by using the AWS SDK for .NET.

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
```

```
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

**Example**

The following example lists directory buckets by using the AWS SDK for PHP.

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

**Example**

The following example lists directory buckets by using the AWS SDK for Ruby.

```
s3 = Aws::S3::Client.new(region:'us-west-1')
s3.list_directory_buckets
```

### Using the AWS CLI

The following `list-directory-buckets` example command shows how you can use the AWS CLI to list your directory buckets in the *`us-east-1`* region. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api list-directory-buckets --region us-east-1
```

For more information, see list-directory-buckets in the *AWS CLI Command Reference*.

## Using HeadBucket with directory buckets

The following AWS SDK examples show how to use the `HeadBucket` API operation to determine if an Amazon S3 directory bucket exists and if you have permission to access it.

### Using the AWS SDKs

The following AWS SDK for Java 2.x example shows how to determine if a bucket exists and if you have permission to access it.

SDK for Java 2.x

#### Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
   try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest
                .builder()
                .bucket(bucketName)
                .build();
        s3Client.headBucket(headBucketRequest);
        System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
   }

   catch (S3Exception e) {
       System.err.println(e.awsErrorDetails().errorMessage());
       System.exit(1);
   }
}
```

### Using the AWS CLI

The following `head-bucket` example command shows how you can use the AWS CLI to determine if a directory bucket exists and if you have permission to access it. To run this command, replace the user input placeholders with your own information.

```
aws s3api head-bucket --bucket bucket-base-name--azid--x-s3
```

For more information, see head-bucket in the *AWS CLI Command Reference*.

# Working with objects in a directory bucket

After you create an Amazon S3 directory bucket, you can work with objects by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and the AWS SDKs.

For more information about bulk object operations with objects stored in the S3 Express One Zone storage class, see Object management. For more information about importing, uploading, copying, deleting, and downloading objects and reading metadata from objects in directory buckets, see the following topics.

**Topics**



## Importing objects into a directory bucket

After you create a directory bucket in Amazon S3, you can populate the new bucket with data by using the import action. Import is a streamlined method for creating S3 Batch Operations jobs to copy objects from general purpose buckets to directory buckets.

> **Note**
>
> The following limitations apply to import jobs:
>
> - The source bucket and the destination bucket must be in the same AWS Region and account.
> - The source bucket cannot be a directory bucket.
> - Objects larger than 5GB are not supported and will be omitted from the copy operation.

- Objects in the Glacier Flexible Retrieval, Glacier Deep Archive, Intelligent-Tiering Archive Access tier, and Intelligent-Tiering Deep Archive tier storage classes must be restored before they can be imported.
- Imported objects with MD5 checksum algorithms are converted to use CRC32 checksums.
- Imported objects use server-side encryption with Amazon S3 managed keys (SSE-S3).
- Imported objects use the Express One Zone storage class, which has a different pricing structure than the storage classes used by general purpose buckets. Consider this difference in cost when importing large numbers of objects.

When you configure an import job, you specify the source bucket or prefix where the existing objects will be copied from. You also provide an AWS Identity and Access Management (IAM) role that has permissions to access the source objects. Amazon S3 then starts a Batch Operations job that copies the objects and automatically applies appropriate storage class and checksum settings.

To configure import jobs, you use the Amazon S3 console.

## Using the Amazon S3 console

### To import objects into a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and then choose the **Directory** buckets tab. Choose the option button next to the directory bucket that you want to import objects into.
3. Choose **Import**.
4. For **Source**, enter the general purpose bucket (or bucket path including prefix) that contains the objects that you want to import. To choose an existing general purpose bucket from a list, choose **Browse S3**.
5. For **Permission to access and copy source objects**, do one of the following to specify an IAM role with the permissions necessary to import your source objects:
   - To allow Amazon S3 to create a new IAM role on your behalf, choose **Create new IAM role**.

> **Note**
>
> If your source objects are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), don't choose the **Create new IAM role** option. Instead, specify an existing IAM role that has the `kms:Decrypt` permission.
> Amazon S3 will use this permission to decrypt your objects. During the import process, Amazon S3 will then re-encrypt those objects by using server-side encryption with Amazon S3 managed keys (SSE-S3).

- To choose an existing IAM role from a list, choose **Choose from existing IAM roles**.
- To specify an existing IAM role by entering its Amazon Resource Name (ARN), choose **Enter IAM role ARN**, then enter the ARN in the corresponding field.

6. Review the information that's displayed in the **Destination** and **Copied object settings** sections. If the information in the **Destination** section is correct, choose **Import** to start the copy job.

   The Amazon S3 console displays the status of your new job on the **Batch Operations** page. For more information about the job, choose the option button next to the job name, and then on the **Actions** menu, choose **View details**. To open the directory bucket that the objects will be imported into, choose **View import destination**.

## Using Batch Operations with S3 Express One Zone

You can use Amazon S3 Batch Operations to perform operations on objects stored in S3 buckets. To learn more about S3 Batch Operations, see Performing large-scale batch operations on Amazon S3 objects.

The following topics discuss performing batch operations on objects stored in the S3 Express One Zone storage class in directory buckets.

**Topics**

- Using Batch Operations with directory buckets
- Key differences

## Using Batch Operations with directory buckets

You can perform the **Copy** operation and the **Invoke AWS Lambda function** operations on objects that are stored in directory buckets. With **Copy**, you can copy objects between buckets of the same type (for example, from a directory bucket to a directory bucket). You can also copy between general purpose buckets and directory buckets. With **Invoke AWS Lambda function**, you can use a Lambda function to perform actions on objects in your directory bucket with code that you define.

### Copying objects

You can copy between the same bucket type or between directory buckets and general purpose buckets. When you copy to a directory bucket, you must use the correct Amazon Resource Name (ARN) format for this bucket type. The ARN format for a directory bucket is `arn:aws:s3express:region:account-id:bucket/bucket-base-name--x-s3`.

You can also populate your directory bucket with data by using the **Import** action in the S3 console. **Import** is a streamlined method for creating Batch Operations jobs to copy objects from general purpose buckets to directory buckets. For **Import** copy jobs from general purpose buckets to directory buckets, S3 automatically generates a manifest. For more information, see Importing objects to a directory bucket and Specifying a manifest.

### Invoking Lambda functions (`LambdaInvoke`)

There are special requirements for using Batch Operations to invoke Lambda functions that act on directory buckets. For example, you must structure your Lambda request by using a v2 JSON invocation schema, and specify InvocationSchemaVersion 2.0 when you create the job. For more information, see Invoke AWS Lambda function.

## Key differences

The following is a list of key differences when you're using Batch Operations to perform bulk operations on objects that are stored in directory buckets with the S3 Express One Zone storage class:

- Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. The default encryption configuration of an S3 bucket is always enabled and is at a minimum set to server-side encryption with Amazon S3 managed keys (SSE-S3). For directory buckets, only sSSE-S3 is supported. If you make a `CopyObject` request that sets server-side encryption with customer-provided keys (SSE-C) or server-side encryption with AWS Key Management Service

(AWS KMS) keys (SSE-KMS) on a directory bucket (source or destination), the response returns an HTTP `400 (Bad Request)` error.

- Objects in directory buckets can't be tagged. You can only specify an empty tag set. By default, Batch Operations copies tags. If you copy an object that has tags between general purpose buckets and directory buckets, you receive a `501 (Not Implemented)` response.
- S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during uploads or downloads. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.
- By default, all Amazon S3 buckets set the S3 Object Ownership setting to bucket owner enforced and access control lists (ACLs) are disabled. For directory buckets, this setting can't be modified. You can copy an object from general purpose buckets to directory buckets. However, you can't overwrite the default ACL when you copy to or from a directory bucket.
- Regardless of how you specify your manifest, the list itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from (or save generated manifests to) directory buckets. However, objects described within the manifest can be stored in directory buckets.
- Batch Operations can't specify a directory bucket as a location in an S3 Inventory report. Inventory reports don't support directory buckets. You can create a manifest file for objects within a directory bucket by using the `ListObjectsV2` API operation to list the objects. You can then insert the list in a CSV file.

**Granting access**

To perform copy jobs, you must have the following permissions:

- To copy objects from one directory bucket to another directory bucket, you must have the `s3express:CreateSession` permission.
- To copy objects from directory buckets to general purpose buckets, you must have the `s3express:CreateSession` permission and the `s3:PutObject` permission to write the object copy to the destination bucket.
- To copy objects from general purpose buckets to directory buckets, you must have the `s3express:CreateSession` permission and the `s3:GetObject` permission to read the source object that is being copied.

For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.

- To invoke a Lambda function, you must grant permissions to your resource based on your Lambda function. To determine which permissions are required, check the corresponding API operations.

## Uploading an object to a directory bucket

After you create an Amazon S3 directory bucket, you can upload objects to it. The following examples show how to upload an object to a directory bucket by using the S3 console and the AWS SDKs. For information about bulk object upload operations with S3 Express One Zone, see Object management.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. Choose the name of the bucket that you want to upload your folders or files to.
5. In the **Objects** list, choose **Upload**.
6. On the **Upload** page, do one of the following:
   - Drag and drop files and folders to the dotted upload area.
   - Choose **Add files** or **Add folder**, choose the files or folders to upload, and then choose **Open** or **Upload**.
7. Under **Checksums**, choose the **Checksum function** that you want to use.

   (Optional) If you're uploading a single object that's less than 16 MB in size, you can also specify a precalculated checksum value. When you provide a precalculated value, Amazon S3 compares it with the value that it calculates by using the selected checksum function. If the values don't match, the upload won't start.
8. The options in the **Permissions** and **Properties** sections are automatically set to default settings and can't be modified. Block Public Access is automatically enabled, and S3 Versioning and S3 Object Lock can't be enabled for directory buckets.

(Optional) If you want to add metadata in key-value pairs to your objects, expand the **Properties** section, and then in the **Metadata** section, choose **Add metadata**.

9. To upload the listed files and folders, choose **Upload**.

   Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   //.metadata(metadata)
                   .build();
           s3Client.putObject(putObj, filePath);
           System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

       }

       catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
}
```

SDK for Python

#### Example

```
import boto3
import botocore
```

```
from botocore.exceptions import ClientError


def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                                Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --azid--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

if __name__ == "__main__":
    main()
```

### Using the AWS CLI

The following `put-object` example command shows how you can use the AWS CLI to upload an object from Amazon S3. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api put-object --bucket bucket-base-name--azid--x-s3 --key sampleinut/file001.bin
 --body bucket-seed/file001.bin
```

For more information, see put-object in the *AWS CLI Command Reference*.

# Using multipart uploads with directory buckets

You can use the multipart upload process to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation.

Using multipart upload provides the following advantages:

- **Improved throughput** – You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** – Smaller part sizes minimize the impact of restarting a failed upload because of a network error.
- **Pause and resume object uploads** – You can upload object parts over time. After you initiate a multipart upload, there is no expiration date. You must explicitly complete or abort the multipart upload.
- **Begin an upload before you know the final object size** – You can upload an object as you are creating it.

We recommend that you use multipart uploads in the following ways:

- If you're uploading large objects over a stable high-bandwidth network, use multipart uploads to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance.
- If you're uploading over a spotty network, use multipart uploads to increase resiliency to network errors by avoiding upload restarts. When using multipart uploads, you need to retry uploading only the parts that are interrupted during the upload. You don't need to restart uploading your object from the beginning.

When you're using multipart uploads to upload objects to the Amazon S3 Express One Zone storage class in directory buckets, the multipart upload process is similar to the process of using multipart upload to upload objects to general purpose buckets. However, there are some notable differences.

For more information about using multipart uploads to upload objects to S3 Express One Zone, see the following topics.

**Topics**



## The multipart upload process

A multipart upload is a three-step process:

- You initiate the upload.
- You upload the object parts.
- After you have uploaded all of the parts, you complete the multipart upload.

Upon receiving the complete multipart upload request, Amazon S3 constructs the object from the uploaded parts, and you can then access the object as you would any other object in your bucket.

**Multipart upload initiation**

When you send a request to initiate a multipart upload, Amazon S3 returns a response with an upload ID, which is a unique identifier for your multipart upload. You must include this upload ID whenever you upload parts, list the parts, complete an upload, or abort an upload.

**Parts upload**

When uploading a part, in addition to the upload ID, you must specify a part number. When you're using a multipart upload with S3 Express One Zone, the multipart part numbers must be consecutive part numbers. If you try to complete a multipart upload request with nonconsecutive part numbers, an `HTTP 400 Bad Request` (Invalid Part Order) error is generated.

A part number uniquely identifies a part and its position in the object that you are uploading. If you upload a new part by using the same part number as a previously uploaded part, the previously uploaded part is overwritten.

Whenever you upload a part, Amazon S3 returns an entity tag (ETag) header in its response. For each part upload, you must record the part number and the ETag value. The ETag values for all object part uploads will remain the same, but each part will be assigned a different part number. You must include these values in the subsequent request to complete the multipart upload.

Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. When doing a multipart upload, if you don't specify encryption information in your request, the encryption setting of the uploaded parts is set to the default encryption configuration of the destination bucket. The default encryption configuration of an Amazon S3 bucket is always enabled and is at a minimum set to server-side encryption with Amazon S3 managed keys (SSE-S3). For directory buckets, only SSE-S3 is supported. For more information, see Server-side encryption with Amazon S3 managed keys (SSE-S3).

**Multipart upload completion**

When you complete a multipart upload, Amazon S3 creates the object by concatenating the parts in ascending order based on the part number. After a successful *complete* request, the parts no longer exist.

Your *complete multipart upload* request must include the upload ID and a list of both part numbers and their corresponding ETag values. The Amazon S3 response includes an ETag that uniquely identifies the combined object data. This ETag is not an MD5 hash of the object data.

**Multipart upload listings**

You can list the parts of a specific multipart upload or all in-progress multipart uploads. The list parts operation returns the parts information that you have uploaded for a specific multipart upload. For each list parts request, Amazon S3 returns the parts information for the specified multipart upload, up to a maximum of 1,000 parts. If there are more than 1,000 parts in the multipart upload, you must use pagination to retrieve all the parts.

The returned list of parts doesn't include parts that haven't finished uploading. Using the *list multipart uploads* operation, you can obtain a list of multipart uploads that are in progress.

An in-progress multipart upload is an upload that you have initiated, but have not yet completed or aborted. Each request returns at most 1,000 multipart uploads. If there are more than 1,000 multipart uploads in progress, you must send additional requests to retrieve the remaining multipart uploads. Use the returned listing only for verification. Do not use the result of this listing when sending a *complete multipart upload* request. Instead, maintain your own list of the part

numbers that you specified when uploading parts and the corresponding ETag values that Amazon S3 returns.

For more information about multipart upload listings, see ListParts in the *Amazon Simple Storage Service API Reference*.

## Checksums with multipart upload operations

When you upload an object to, you can specify a checksum algorithm to check object integrity. MD5 is not supported for directory buckets. You can specify one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms:

- CRC32
- CRC32C
- SHA-1
- SHA-256

You can use the Amazon S3 REST API or the AWS SDKs to retrieve the checksum value for individual parts by using `GetObject` or `HeadObject`. If you want to retrieve the checksum values for individual parts of multipart uploads still in process, you can use `ListParts`.

> **Important**
>
> When using the preceding checksum algorithms, the multipart part numbers must use consecutive part numbers. If you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an `HTTP 400 Bad Request` (Invalid Part Order) error.

For more information about how checksums work with multipart objects, see Checking object integrity.

## Concurrent multipart upload operations

In a distributed development environment, your application can initiate several updates on the same object at the same time. For example, your application might initiate several multipart uploads by using the same object key. For each of these uploads, your application can then upload

parts and send a complete upload request to Amazon S3 to create the object. For S3 Express One Zone, the object creation time is the completion date of the multipart upload.

> **Important**
>
> Versioning isn't supported for objects that are stored in directory buckets.

## Multipart uploads and pricing

After you initiate a multipart upload, Amazon S3 retains all the parts until you either complete or abort the upload. Throughout its lifetime, you are billed for all storage, bandwidth, and requests for this multipart upload and its associated parts. If you abort the multipart upload, Amazon S3 deletes the upload artifacts and any parts that you have uploaded, and you are no longer billed for them. There are no early delete charges for deleting incomplete multipart uploads, regardless of the storage class specified. For more information about pricing, see Amazon S3 pricing.

> **Important**
>
> If the complete multipart upload request isn't sent successfully, the object parts aren't assembled and an object isn't created. You are billed for all storage associated with uploaded parts. It's important that you either complete the multipart upload to have the object created or abort the multipart upload to remove any uploaded parts.
> Before you can delete a directory bucket, you must complete or abort all in-progress multipart uploads. Directory buckets don't support S3 Lifecycle configurations. If needed, you can list your active multipart uploads, then abort the uploads, and then delete your bucket.

## Multipart upload API operations and permissions

To allow access to object management API operations on a directory bucket, you grant the `s3express:CreateSession` permission in a bucket policy or an AWS Identity and Access Management (IAM) identity-based policy.

You must have the necessary permissions to use the multipart upload operations. You can use bucket policies or IAM identity-based policies to grant IAM principals permissions to perform these operations. The following table lists the required permissions for various multipart upload operations.

You can identify the initiator of a multipart upload through the `Initiator` element. If the initiator is an AWS account, this element provides the same information as the `Owner` element. If the initiator is an IAM user, this element provides the user ARN and display name.

| Action | Required permissions |
| --- | --- |
| Create a multipart upload | To create the multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket. |
| Initiate a multipart upload | To initiate the multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket. |
| Upload a part | To upload a part, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.<br><br>For the initiator to upload a part, the bucket owner must allow the initiator to perform the `s3express:CreateSession` action on the directory bucket. |
| Upload a part (copy) | To upload a part, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.<br><br>For the initiator to upload a part for an object, the owner of the bucket must allow the initiator to perform the `s3express:CreateSession` action on the object. |
| Complete a multipart upload | To complete a multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.<br><br>For the initiator to complete a multipart upload, the bucket owner must allow the initiator to perform the `s3express:CreateSession` action on the object. |
| Abort a multipart upload | To abort a multipart upload, you must be allowed to perform the `s3express:CreateSession` action.<br><br>For the initiator to abort a multipart upload, the initiator must be granted explicit allow access to perform the `s3express:CreateSession` action. |

| Action | Required permissions |
| --- | --- |
| List parts | To list the parts in a multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket. |
| List in-progress multipart uploads | To list the in-progress multipart uploads to a bucket, you must be allowed to perform the `s3:ListBucketMultipartUploads` action on that bucket. |

### API operation support for multipart uploads

The following sections in the Amazon Simple Storage Service API Reference describe the Amazon S3 REST API operations for multipart uploads.

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListParts
- ListMultipartUploads

## Examples

To use a multipart upload to upload an object to S3 Express One Zone in a directory bucket, see the following examples.

### Topics

- Creating a multipart upload
- Uploading the parts of a multipart upload
- Completing a multipart upload
- Aborting a multipart upload
- Creating a multipart upload copy operation
- Listing in-progress multipart uploads
- Listing the parts of a multipart upload

### Creating a multipart upload

The following examples show how to create a multipart upload.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {

     CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
             .bucket(bucketName)
             .key(key)
             .build();

     String uploadId = null;

     try {
         CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
         uploadId = response.uploadId();
     }
     catch (S3Exception e) {
         System.err.println(e.awsErrorDetails().errorMessage());
         System.exit(1);
     }
     return uploadId;
```

SDK for Python

**Example**

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''

   try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None
```

### Using the AWS CLI

This example shows how to create a multipart upload to a directory bucket by using the AWS CLI. This command starts a multipart upload to the directory bucket *bucket-base-name*--*azid*--x-s3 for the object *KEY_NAME*. To use the command replace the *user input placeholders* with your own information.

```
aws s3api create-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

For more information, see create-multipart-upload in the AWS Command Line Interface.

### Uploading the parts of a multipart upload

The following examples show how to upload parts of a multipart upload.

### Using the AWS SDKs

SDK for Java 2.x

The following example shows how to break a single object into parts and then upload those parts to a directory bucket by using the SDK for Java 2.x.

#### Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
 private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
 String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
 buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
```

```
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
    }
```

SDK for Python

The following example shows how to break a single object into parts and then upload those parts to a directory bucket by using the SDK for Python.

**Example**

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
 that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
 successfully, else None
    '''
```

```
    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
upload_part['ETag']})
                part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list
```

### Using the AWS CLI

This example shows how to break a single object into parts and then upload those parts to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api upload-part --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see upload-part in the AWS Command Line Interface.

### Completing a multipart upload

The following examples show how to complete a multipart upload.

### Using the AWS SDKs

SDK for Java 2.x

The following examples show how to complete a multipart upload by using the SDK for Java 2.x.

#### Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
 * @param key
 * @param uploadId
 * @param uploadParts
 */
 private static void completeMultipartUpload(S3Client s3, String bucketName, String
 key, String uploadId, ListCompletedPart uploadParts) {
        CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                .parts(uploadParts)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    public static void multipartUploadTest(S3Client s3, String bucketName, String
 key, String localFilePath)  {
        System.out.println("Starting multipart upload for: " + key);
        try {
            String uploadId = createMultipartUpload(s3, bucketName, key);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
 localFilePath);
```

```
            completeMultipartUpload(s3, bucketName, key, uploadId, parts);
            System.out.println("Multipart upload completed for: " + key);
        }

        catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

The following examples show how to complete a multipart upload by using the SDK for Python.

**Example**

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
```

```
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
to {bucket_name}')
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
{bucket_name}')
```

### Using the AWS CLI

This example shows how to complete a multipart upload for a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api complete-multipart-upload --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

This example takes a JSON structure that describes the parts of the multipart upload that should be reassembled into the complete file. In this example, the `file://` prefix is used to load the JSON structure from a file in the local folder named `parts`.

parts.json:

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
  ]
```

```
}
```

For more information, see complete-multipart-upload in the AWS Command Line Interface.

### Aborting a multipart upload

The following examples show how to abort a multipart upload.

### Using the AWS SDKs

SDK for Java 2.x

The following example shows how to abort a multipart upload by using the SDK for Java 2.x.

#### Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            ListMultipartUpload uploads = response.uploads();

            AbortMultipartUploadRequest abortMultipartUploadRequest;
            for (MultipartUpload upload: uploads) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
```

SDK for Python

The following example shows how to abort a multipart upload by using the SDK for Python.

**Example**

```
import logging
import boto3
from botocore.exceptions import ClientError


def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
```

```
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
 been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
 {bucket_name} bucket')
```

### Using the AWS CLI

The following example shows how to abort a multipart upload by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api abort-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

For more information, see abort-multipart-upload in the AWS Command Line Interface.

### Creating a multipart upload copy operation

The following examples show how to copy bjects from one bucket to another using a multipart upload.

### Using the AWS SDKs

SDK for Java 2.x

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to another by using the SDK for Java 2.x.

**Example**

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
 *
 * @param s3
 * @param bucketName
 * @param key
 * @return
```

```
*/
private static String createMultipartUpload(S3Client s3, String bucketName, String
key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();
       String uploadId = null;
       try {
           CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
           uploadId = response.uploadId();
       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return uploadId;
 }

 /**
  * Creates copy parts based on source object size and copies over individual parts
  *
  * @param s3
  * @param sourceBucket
  * @param sourceKey
  * @param destnBucket
  * @param destnKey
  * @param uploadId
  * @return
  * @throws IOException
  */
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

       // Get the object size to track the end of the copy operation.
       HeadObjectRequest headObjectRequest = HeadObjectRequest
               .builder()
               .bucket(sourceBucket)
               .key(sourceKey)
               .build();
       HeadObjectResponse response = s3.headObject(headObjectRequest);
       Long objectSize = response.contentLength();
```

```
        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        while (bytePosition < objectSize) {
            // The last part might be smaller than partSize, so check to make sure
            // that lastByte isn't beyond the end of the object.
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }


    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
srcKey, String destnBucket, String destnKey)  {
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
```

```
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
 srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to another by using the SDK for Python.

**Example**

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''

    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
        logging.error(e)
        return None

def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
```

```
    Create a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
    try:
        part_counter = 1
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
```

```
        while (part_counter - 1) * part_size <object_size:
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True
```

```
if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
 mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
```

### Using the AWS CLI

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to a directory bucket using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api upload-part-copy --bucket bucket-base-name--azid--x-s3 --key TARGET_KEY_NAME
 --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see upload-part-copy in the AWS Command Line Interface.

### Listing in-progress multipart uploads

To list in-progress multipart uploads to a directory bucket, you can use the AWS SDKs, or the AWS CLI.

### Using the AWS SDKs

SDK for Java 2.x

The following examples show how to list in-progress (incomplete) multipart uploads by using the SDK for Java 2.x.

**Example**

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
       try {
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

           ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
           List MultipartUpload uploads = response.uploads();
           for (MultipartUpload upload: uploads) {
               System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
           }
     }
     catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
     }
 }
```

SDK for Python

The following examples show how to list in-progress (incomplete) multipart uploads by using the SDK for Python.

**Example**

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_multipart_uploads(s3_client, bucket_name):
```

```
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

### Using the AWS CLI

The following examples show how to list in-progress (incomplete) multipart uploads by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3api list-multipart-uploads --bucket bucket-base-name--azid--x-s3
```

For more information, see list-multipart-uploads in the AWS Command Line Interface.

### Listing the parts of a multipart upload

The following examples show how to list the parts of a multipart upload to a directory bucket.

### Using the AWS SDKs

SDK for Java 2.x

The following examples show how to list the parts of a multipart upload to a directory bucket by using SDK for Java 2.x.

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python

The following examples show how to list the parts of a multipart upload to a directory bucket by using SDK for Python.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None
```

```
if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
 {bucket_name} bucket')
```

#### Using the AWS CLI

The following examples show how to list the parts of a multipart upload to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api list-parts --bucket bucket-base-name--azid--x-s3 --key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see list-parts in the AWS Command Line Interface.

## Copying an object to a directory bucket

The copy operation creates a copy of an object that is already stored in Amazon S3. You can copy objects between directory buckets and general purpose buckets. You can also copy objects within a bucket and across buckets of the same type, for example, from directory bucket to directory bucket.

You can create a copy of object up to 5 GB in a single atomic operation. However, to copy an object that is greater than 5 GB, you must use the multipart upload API operations. For more information, see Using multipart uploads with directory buckets.

#### Permissions

To copy objects, you must have the following permissions:

- To copy objects from one directory bucket to another directory bucket, you must have the `s3express:CreateSession` permission.

- To copy objects from directory buckets to general purpose buckets, you must have the `s3express:CreateSession` permission and the `s3:PutObject` permission to write the object copy to the destination bucket.
- To copy objects from general purpose buckets to directory buckets, you must have the `s3express:CreateSession` permission and `s3:GetObject` permission to read the source object that is being copied.

  For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.

**Encryption**

Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. The default encryption configuration of an S3 bucket is always enabled and is at a minimum set to server-side encryption with Amazon S3 managed keys (SSE-S3).

For directory buckets, only SSE-S3 is supported. For general purpose buckets, you can use SSE-S3 (the default), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C).

If you make a copy request that sets SSE-C, SSE-KMS, or DSSE-KMS parameters on a directory bucket as either the source or destination, the response returns an error,

**Tags**

Directory buckets don't support tags. If you copy an object that has tags from a general purpose bucket to a directory bucket, you receive an HTTP `501 (Not Implemented)` response. For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.

**ETags**

Entity tags (ETags) for S3 Express One Zone are random alphanumeric strings and are not MD5 checksums. To help ensure object integrity, use additional checksums.

**Additional checksums**

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32,

CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

For more information, see S3 additional checksum best practices.

**Supported features**

For more information about which Amazon S3 features are supported for S3 Express One Zone, see How is S3 Express One Zone different?.

**Using the S3 console (copy to a directory bucket)**

**To copy an object from a general purpose bucket or a directory bucket to a directory bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the bucket that you want to copy objects from:
   - To copy from a general purpose bucket, choose the **General purpose buckets** tab.
   - To copy from a directory bucket, choose the **Directory buckets** tab.
4. Choose the general purpose bucket or directory bucket that contains the objects that you want to copy.
5. Choose the **Objects** tab. On the **Objects** page, select the check box to the left of the names of the objects that you want to copy.
6. On the **Actions** menu, choose **Copy**.

   The **Copy** page appears.
7. Under **Destination**, choose **Directory bucket** for your destination type. To specify the destination path, choose **Browse S3**, navigate to the destination, and then choose the option button to the left of the destination. Choose **Choose destination** in the lower-right corner.

   Alternatively, enter the destination path.
8. Under **Checksums**, choose whether you want to copy the objects with their existing checksum functions or replace the existing checksum functions with a new one. When you uploaded the objects, you had the option to specify the checksum algorithm that was used to verify data integrity. When copying the object, you have the option to choose a new function. If you dn't originally specify an additional checksum, you can use e **Checksums** section to add one.

> **Note**
>
> Even if you opt to use the same checksum function, your checksum value might change if the object is over 16 MB in size. The checksum value might change because of how checksums are calculated for multipart uploads. For more information about how the checksum might change when copying the object, see Using part-level checksums for multipart uploads.

To change the checksum function, choose **Replace with a new checksum function**. Choose the new checksum function from the dropdown list. When the object is copied over, the new checksum is calculated and stored by using the specified algorithm.

9. Choose **Copy** in the bottom-right corner. Amazon S3 copies your objects to the destination.

**Using the S3 console (copy to a general purpose bucket)**

**To copy an object from a directory bucket to a general purpose bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. Choose the directory bucket that contains the objects that you want to copy.
5. Choose the **Objects** tab. On the **Objects** page, select the check box to the left of the names of the objects that you want to copy.
6. On the **Actions** menu, choose **Copy**.
7. Under **Destination**, choose **General purpose bucket** for your destination type. To specify the destination path, choose **Browse S3**, navigate to the destination, and choose the option button to the left of the destination. Choose **Choose destination** in the lower-right corner.

   Alternatively, enter the destination path.
8. Under **Checksums**, choose whether you want to copy the objects with their existing checksum functions or replace the existing checksum functions with a new one. When you uploaded the objects, you had the option to specify the checksum algorithm that was used to verify data

integrity. When copying the object, you have the option to choose a new function. If you didn't originally specify an additional checksum, you can use the **Checksums** section to add one.

> **Note**
>
> Even if you opt to use the same checksum function, your checksum value might change if the object is over 16 MB in size. The checksum value might change because of how checksums are calculated for multipart uploads. For more information about how the checksum might change when copying the object, see Using part-level checksums for multipart uploads.

To change the checksum function, choose **Replace with a new checksum function**. Choose the new checksum function from the dropdown list. When the object is copied over, the new checksum is calculated and stored by using the specified algorithm.

9. Choose **Copy** in the bottom-right corner. Amazon S3 copies your objects to the destination.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
     CopyObjectRequest copyReq = CopyObjectRequest.builder()
         .sourceBucket(sourceBucket)
         .sourceKey(objectKey)
         .destinationBucket(targetBucket)
         .destinationKey(objectKey)
         .build();
      String temp = "";

      try {
          CopyObjectResponse copyRes = s3.copyObject(copyReq);
          System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
      }

      catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
}
```

### Using the AWS CLI

The following `copy-object` example command shows how you can use the AWS CLI to copy an object from one bucket to another bucket. You can copy objects between bucket types. To run this command, replace the user input placeholders with your own information.

```
aws s3api copy-object --copy-source bucket SOURCE_BUCKET/SOURCE_KEY_NAME --
key TARGET_KEY_NAME --bucket TARGET_BUCKET_NAME
```

For more information, see copy-object in the *AWS CLI Command Reference*.

## Deleting an object in a directory bucket

You can delete objects from an Amazon S3 directory bucket by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), or AWS SDKs. For more information, see Directory buckets and What is S3 Express One Zone?.

**Warning**

- Deleting an object can't be undone.
- This action deletes all specified objects. When deleting folders, wait for the delete action to finish before adding new objects to the folder. Otherwise, new objects might be deleted as well.

**Note**

When you programmatically delete multiple objects from a directory bucket, note the following:

- Object keys in `DeleteObjects` requests must contain at least one non-white space character. Strings of all white space characters are not supported.

- Object keys in `DeleteObjects` requests cannot contain Unicode control characters, except for newline (`\n`), tab (`\t`), and carriage return (`\r`).

### Using the S3 console

#### To delete objects

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. Choose the directory bucket that contains the objects that you want to delete.
5. Choose the **Objects** tab. In the **Objects** list, select the check box to the left of the object or objects that you want to delete.
6. Choose **Delete**.
7. On the **Delete objects** page, enter **`permanently delete`** in the text box.
8. Choose **Delete objects**.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

The following example deletes objects in a directory bucket by using the AWS SDK for Java 2.x.

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();
```

```
            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

    }
```

SDK for Python

**Example**

The following example deletes objects in a directory bucket by using the AWS SDK for Python (Boto3).

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_objects(s3_client, bucket_name, objects):
    '''
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
```

```
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
    ]

    s3_client = boto3.client('s3', region_name = region)
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
 {bucket_name}')
```

### Using the AWS CLI

The following `delete-object` example command shows how you can use the AWS CLI to delete an object from a directory bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api delete-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

For more information, see delete-object in the *AWS CLI Command Reference*.

## Downloading an object in a directory bucket

The following code examples show how to read data from (download) an object in an Amazon S3 directory bucket by using the `GetObject` API operation.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

The following code example shows how to read data from an object in a directory bucket by using the AWS SDK for Java 2.x.

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

**Example**

The following code example shows how to read data from an object in a directory bucket by using the AWS SDK for Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
    """
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
        raise
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

if __name__ == "__main__":
     main()
```

### Using the AWS CLI

The following `get-object` example command shows how you can use the AWS CLI to download an object from Amazon S3. This command gets the object *KEY_NAME* from the directory

bucket *bucket-base-name*--*azid*--x-s3. The object will be downloaded to a file named *LOCAL_FILE_NAME*. To run this command, replace the *user input placeholders* with your own information.

```
aws s3api get-object --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

For more information, see get-object in the *AWS CLI Command Reference*.

## Using `HeadObject` with directory buckets

The following AWS SDK and AWS CLI examples show how to use the `HeadObject` API operation to retrieve metadata from an object in an Amazon S3 directory bucket without returning the object itself.

### Using the AWS SDKs

SDK for Java 2.x

#### Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
 ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

**Using the AWS CLI**

The following `head-object` example command shows how you can use the AWS CLI to retrieve metadata from an object. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api head-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

For more information, see head-object in the *AWS CLI Command Reference*.

# Security for S3 Express One Zone

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs.

  To learn about the compliance programs that apply to Amazon S3 Express One Zone, see AWS services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation will help you understand how to apply the shared responsibility model when using S3 Express One Zone. The following topics show you how to configure S3 Express One Zone to meet your security and compliance objectives. You will also learn how to use other AWS services that can help you monitor and secure your resources when you're working with S3 Express One Zone.

**Topics**

- Data protection and encryption
- AWS Identity and Access Management (IAM) for S3 Express One Zone

- IAM identity-based policies for S3 Express One Zone
- Example directory bucket policies for S3 Express One Zone
- CreateSession authorization
- Security best practices for S3 Express One Zone

## Data protection and encryption

For more information about how S3 Express One Zone encrypts and protects your data, see the following topics.

**Topics**

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Encryption in transit
- Additional checksums
- Data deletion

### Server-side encryption with Amazon S3 managed keys (SSE-S3)

By default, all objects stored in directory buckets are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). Unencrypted uploads to directory buckets aren't permitted. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3) and Protecting data with encryption.

Directory buckets don't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS), or server-side encryption with customer-provided encryption keys (SSE-C).

### Encryption in transit

S3 Express One Zone can only be accessed through HTTPS (TLS).

S3 Express One Zone uses Regional and Zonal API endpoints. Depending on the Amazon S3 API operation that you use, either a Regional or Zonal endpoint is required. You can access Zonal and Regional endpoints through a gateway virtual private cloud (VPC) endpoint. There is no additional charge for using gateway endpoints. To learn more about Regional and Zonal API endpoints, see Networking for S3 Express One Zone.

### Additional checksums

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

For more information, see S3 additional checksum best practices.

### Data deletion

You can delete one or more objects directly from S3 Express One Zone by using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), or Amazon S3 REST API. Because all objects in your directory buckets incur storage costs, we recommend deleting objects that you no longer need.

Deleting an object that's stored in a directory bucket also recursively deletes any parent directories, if those parent directories don't contain any objects other than the object that's being deleted.

> **Note**
>
> Multi-factor authentication (MFA) delete and S3 Versioning are not supported for S3 Express One Zone.

## AWS Identity and Access Management (IAM) for S3 Express One Zone

AWS Identity and Access Management (IAM) is an AWS service that helps administrators securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon S3 resources in S3 Express One Zone. You can use IAM for no additional charge.

By default, users don't have permissions for directory buckets and S3 Express One Zone operations. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. For more information about IAM, see Security best practices in IAM in the *IAM User Guide*.

To provide access, you can add permissions to your users, groups, or roles through the following means:

- **Users and groups in AWS IAM Identity Center** – Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- **Users managed in IAM through an identity provider** – Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.
- **IAM roles and users** – Create a role that your user can assume. Follow the instructions in Creating a role to delegate permissions to an IAM user in the *IAM User Guide*.

By default, directory buckets are private and can be accessed only by users who are explicitly granted access. The access control boundary for directory buckets is set only at the bucket level. In contrast, the access control boundary for general purpose buckets can be set at the bucket, prefix, or object tag level. This difference means that directory buckets are the only resource that you can include in bucket policies or IAM identity policies for S3 Express One Zone access.

With S3 Express One Zone, in addition to IAM authorization, you authenticate and authorize requests through a new session-based mechanism that's handled by the `CreateSession` API operation. You can use `CreateSession` to request temporary credentials that provide low-latency access to your bucket. These temporary credentials are scoped to a specific directory bucket.

To work with `CreateSession`, we recommend using the latest version of the AWS SDKs or using the AWS Command Line Interface (AWS CLI). The supported AWS SDKs and the AWS CLI handle session establishment, refreshment, and termination on your behalf.

You use session tokens with only Zonal (object-level) operations (except for `CopyObject` and `HeadBucket`) to distribute the latency that's associated with authorization over a number of requests in a session. For Regional endpoint API operations (bucket-level operations), you use IAM authorization, which doesn't involve managing a session. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone and `CreateSession` authorization.

For more information about IAM for S3 Express One Zone, see the following topics.

**Topics**

- Principals
- Resources
- Actions for S3 Express One Zone
- Condition keys for S3 Express One Zone
- How API operations are authorized and authenticated

### Principals

When you create a resource-based policy to grant access to your buckets, you must use the `Principal` element to specify the person or application that can make a request for an action or operation on that resource. For directory bucket policies, you can use the following principals:

- An AWS account
- An IAM user
- An IAM role
- A federated user

For more information, see Principal in the *IAM User Guide*.

### Resources

Amazon Resource Names (ARNs) for directory buckets contain the `s3express` namespace, the AWS Region, the AWS account ID, and the directory bucket name, which includes the Availability Zone ID. To access and perform actions on your directory bucket, you must use the following ARN format:

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--azid--x-s3
```

For more information about ARNs, see Amazon Resource Names (ARNs) in the *IAM User Guide*. For more information about resources, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.

### Actions for S3 Express One Zone

In an IAM identity-based policy or resource-based policy, you define which S3 actions are allowed or denied. S3 Express One Zone actions correspond to specific API operations. S3 Express One Zone has a unique IAM namespace that is distinct from the standard namespace for Amazon S3. This namespace is `s3express`.

When you allow the `s3express:CreateSession` permission, this enables the `CreateSession` API operation to retrieve session tokens when accessing Zonal endpoint API (or object level) operations . These session tokens return credentials that are used to grant access to all of the other Zonal endpoint API operations. As a result, you don't have to grant access permissions to Zonal API operations by using IAM policies. Instead, the session token enables access.

For more information about Zonal and Regional endpoint API operations, see Networking for S3 Express One Zone. To learn more about the `CreateSession` API operation, see CreateSession in the *Amazon Simple Storage Service API Reference*.

You can specify the following actions in the `Action` element of an IAM policy statement. Use policies to grant permissions to perform an operation in AWS. When you use an action in a policy, you usually allow or deny access to the API operation with the same name. However, in some cases, a single action controls access to more than one API operation. Access to bucket-level actions can be granted in only IAM identity-based policies (user or role) and not bucket policies.

**Actions and condition keys for S3 Express One Zone**

| Action | API | Description | Access level | Condition keys |
|---|---|---|---|---|
| `s3express:CreateBucket` | `CreateBucket` | Grants permission to create a new bucket. | Write | `s3express:authType`<br>`s3express:LocationName`<br>`s3express:ResourceAccount`<br>`s3express:signatureversion`<br>`s3express:TlsVersion`<br>`s3express:x-amz-content-sha256` |

| Action | API | Description | Access level | Condition keys |
| --- | --- | --- | --- | --- |
| s3express:CreateSession | CreateSession | Grants permission to create a session token, which is used for granting access to all Zonal (object-level) API operations, such as PutObject , GetObject , and so on. | Write | s3express:authType<br><br>s3express:SessionMode<br><br>s3express:ResourceAccount<br><br>s3express:signaturversion<br><br>s3express:signatureAge<br><br>s3express:TlsVersion<br><br>s3express:x-amz-content-sha256 |

| Action | API | Description | Access level | Condition keys |
| --- | --- | --- | --- | --- |
| s3express:DeleteBucket | DeleteBucket | Grants permission to delete the bucket named in the URI. | Write | s3express:authType<br>s3express:ResourceAccount<br>s3express:signatureversion<br>s3express:TlsVersion<br>s3express:x-amz-content-sha256 |

| Action | API | Description | Access level | Condition keys |
| --- | --- | --- | --- | --- |
| s3express:DeleteBucketPolicy | DeleteBucketPolicy | Grants permission to delete the policy on a specified bucket. | Permissions management | s3express:authType<br>s3express:ResourceAccount<br>s3express:signatureversion<br>s3express:TlsVersion<br>s3express:x-amz-content-sha256 |

| Action | API | Description | Access level | Condition keys |
| --- | --- | --- | --- | --- |
| s3express:GetBucketPolicy | GetBucketPolicy | Grants permission to return the policy of the specified bucket. | Read | s3express:authType<br>s3express:ResourceAccount<br>s3express:signatureversion<br>s3express:TlsVersion<br>s3express:x-amz-content-sha256 |

| Action | API | Description | Access level | Condition keys |
| --- | --- | --- | --- | --- |
| s3express:ListAllMyDirectoryBuckets | ListDirectoryBuckets | Grants permission to list all directory buckets owned by the authenticated sender of the request. | List | s3express:authType<br><br>s3express:ResourceAccount<br><br>s3express:signatureversion<br><br>s3express:TlsVersion<br><br>s3express:x-amz-content-sha256 |

| Action | API | Description | Access level | Condition keys |
|---|---|---|---|---|
| s3express:PutBucketPolicy | PutBucketPolicy | Grants permission to add or replace a bucket policy on a bucket. | Permissions management | s3express:authType<br>s3express:ResourceAccount<br>s3express:signatureversion<br>s3express:TlsVersion<br>s3express:x-amz-content-sha256 |

## Condition keys for S3 Express One Zone

S3 Express One Zone defines the following condition keys that can be used in the Condition element of an IAM policy. You can use these keys to further refine the conditions under which the policy statement applies.

| Condition key | Description | Type |
|---|---|---|
| s3express:authType | Filters access by authentication method. To restrict incoming requests to use a specific authentication method, you can use this optional condition key. For example, you can use this condition key to allow only the HTTP | String |

| Condition key | Description | Type |
| --- | --- | --- |
| | `Authorization` header to be used in request authentication.<br><br>**Valid values:** `REST-HEADER` , `REST-QUERY-STRING` | |
| `s3express:LocationName` | Filters access to the `CreateBucket` API operation by a specific Availability Zone ID (AZ ID), for example, `usw2-az1`.<br><br>**Example value:** `usw2-az1` | String |
| `s3express:ResourceAccount` | Filters access by the resource owner's AWS account ID.<br><br>To restrict user, role, or application access to the directory buckets that are owned by a specific AWS account ID, you can use either the `aws:ResourceAccount` or `s3express:ResourceAccount` condition key. You can use this condition key in either AWS Identity and Access Management (IAM) identity policies or virtual private cloud (VPC) endpoint policies. For example, you can use this condition key to restrict clients within your VPC from accessing buckets that you don't own.<br><br>**Example value:** `111122223333` | String |

| Condition key | Description | Type |
| --- | --- | --- |
| `s3express:SessionMode` | Filters access by the permission requested by the `CreateSession` API operation. By default, the session is `ReadWrite`. You can use this condition key to limit access to `ReadOnly` or to explicitly deny `ReadWrite` access. For more information, see Example directory bucket policies for S3 Express One Zone and CreateSession in the *Amazon Simple Storage Service API Reference*.<br><br>**Valid values:** `ReadWrite`, `ReadOnly` | String |
| `s3express:signatureAge` | Filters access by the age in milliseconds of the request signature. This condition works only for presigned URLs.<br><br>In AWS Signature Version 4, the signing key is valid for up to seven days. Therefore, the signatures are also valid for up to seven days. For more information, see Introduction to signing requests in the *Amazon Simple Storage Service API Reference*. You can use this condition to further limit the signature age.<br><br>**Example value:** `600000` | Numeric |
| `s3express:signatureversion` | Identifies the version of AWS Signature that you want to support for authenticated requests. For authenticated requests, S3 Express One Zone supports Signature Version 4.<br><br>**Valid value:** `"AWS4-HMAC-SHA256"` (identifies Signature Version 4) | String |

| Condition key | Description | Type |
| --- | --- | --- |
| `s3express:TlsVersion` | Filters access by the TLS version that's used by the client.<br><br>You can use the `s3:TlsVersion` condition key to write IAM, virtual private cloud endpoint (VPCE), or bucket policies that restrict user or application access to directory buckets based on the TLS version that's used by the client. You can also use this condition key to write policies that require a minimum TLS version.<br><br>**Example value:** `1.3` | Numeric |

| Condition key | Description | Type |
| --- | --- | --- |
| `s3express:x-amz-content-sha256` | Filters access by unsigned content in your bucket.<br><br>You can use this condition key to disallow unsigned content in your bucket.<br><br>When you use Signature Version 4 for requests that use the `Authorization` header, you add the `x-amz-content-sha256` header in the signature calculation and then set its value to the hash payload.<br><br>You can use this condition key in your bucket policy to deny any uploads where the payloads aren't signed. For example:<br><br>• Deny uploads that use the `Authorization` header to authenticate requests but don't sign the payload. For more information, see Transferring payload in a single chunk in the *Amazon Simple Storage Service API Reference*.<br>• Deny uploads that use presigned URLs. Presigned URLs always have an `UNSIGNED_PAYLOAD`. For more information, see Authenticating requests and Authentication methods in the *Amazon Simple Storage Service API Reference*.<br><br>**Valid value:** `UNSIGNED-PAYLOAD` | String |

## How API operations are authorized and authenticated

The following table lists authorization and authentication information for S3 Express One Zone API operations. For each API operation, the table shows the API operation name, IAM action, endpoint type (Regional or Zonal), and authorization mechanism (IAM or session-based). This table also

indicates where cross-account access is supported. Access to bucket-level actions can be granted only in IAM identity-based policies (user or role), not bucket policies.

| API | Endpoint type | IAM action | Cross-account access |
|---|---|---|---|
| CreateBucket | Regional | s3express:CreateBucket | No |
| DeleteBucket | Regional | s3express:DeleteBucket | No |
| ListDirectoryBuckets | Regional | s3express:ListAllMyDirectoryBuckets | No |
| PutBucketPolicy | Regional | s3express:PutBucketPolicy | No |
| GetBucketPolicy | Regional | s3express:GetBucketPolicy | No |
| DeleteBucketPolicy | Regional | s3express:DeleteBucketPolicy | No |
| CreateSession | Zonal | s3express:CreateSession | Yes |
| CopyObject | Zonal | s3express:CreateSession | Yes |
| DeleteObject | Zonal | s3express:CreateSession | Yes |
| DeleteObjects | Zonal | s3express:CreateSession | Yes |
| HeadObject | Zonal | s3express:CreateSession | Yes |
| PutObject | Zonal | s3express:CreateSession | Yes |
| GetObjectAttributes | Zonal | s3express:CreateSession | Yes |
| ListObjectsV2 | Zonal | s3express:CreateSession | Yes |
| HeadBucket | Zonal | s3express:CreateSession | Yes |

| API | Endpoint type | IAM action | Cross-account access |
|---|---|---|---|
| CreateMultipartUpload | Zonal | s3express:CreateSession | Yes |
| UploadPart | Zonal | s3express:CreateSession | Yes |
| UploadPartCopy | Zonal | s3express:CreateSession | Yes |
| CompleteMultipartUpload | Zonal | s3express:CreateSession | Yes |
| AbortMultipartUpload | Zonal | s3express:CreateSession | Yes |
| ListParts | Zonal | s3express:CreateSession | Yes |
| ListMultipartUploads | Zonal | s3express:CreateSession | Yes |

## IAM identity-based policies for S3 Express One Zone

Before you can create directory buckets or use Amazon S3 Express One Zone storage class, you must grant the necessary permissions to your AWS Identity and Access Management (IAM) role or users. This example policy allows access to the `CreateSession` API operation (for use with Zonal endpoint [object level] API operations) and all of the Regional endpoint (bucket-level) API operations. This policy allows the `CreateSession` API operation for use with all directory buckets, but the Regional endpoint API operations are allowed only for use with the specified directory bucket. To use this example policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowAccessRegionalEndpointAPIs",
            "Effect": "Allow",
            "Action": [
                "s3express:DeleteBucket",
                "s3express:DeleteBucketPolicy",
                "s3express:CreateBucket",
                "s3express:PutBucketPolicy",
                "s3express:GetBucketPolicy",
                "s3express:ListAllMyDirectoryBuckets"
            ],

            "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-
name--azid--x-s3/*"
        },
        {
            "Sid": "AllowCreateSession",
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "*"
        }
    ]
}
```

## Example directory bucket policies for S3 Express One Zone

This section provides example directory bucket policies for use with the Amazon S3 Express One Zone storage class. To use these policies, replace the *`user input placeholders`* with your own information.

The following example bucket policy allows AWS account ID *`111122223333`* to use the `CreateSession` API operation with the default `ReadWrite` session for the specified directory bucket. This policy grants access to the Zonal endpoint (object level) API operations.

**Example – Bucket policy to allow `CreateSession` calls with the default `ReadWrite` session**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:account-id:bucket/bucket-base-
name--azid--x-s3",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

### Example – Bucket policy to allow `CreateSession` calls with a `ReadOnly` session

The following example bucket policy allows AWS account ID *111122223333* to use the `CreateSession` API operation. This policy uses the `s3express:SessionMode` condition key with the `ReadOnly` value to set a read-only session.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

**Example – Bucket policy to allow cross-account access for `CreateSession` calls**

The following example bucket policy allows AWS account ID *111122223333* to use the `CreateSession` API operation for the specified directory bucket that's owned by AWS account ID *444455556666*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/bucket-base-name--azid--x-s3"

        }
    ]
}
```

## `CreateSession` authorization

Amazon S3 Express One Zone supports both AWS Identity and Access Management (AWS IAM) authorization and session-based authorization:

- To use Regional endpoint API operations (bucket-level, or control plane, operations) with S3 Express One Zone, you use the IAM authorization model, which doesn't involve session management. Permissions are granted for actions individually. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.
- To use Zonal endpoint API operations (object-level, or data plane, operations), you use the `CreateSession` API operation to create and manage sessions that are optimized for low-latency authorization of data requests. To retrieve and use a session token, you must allow the `s3express:CreateSession` action for your directory bucket in an identity-based policy or

a bucket policy. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone. If you're accessing S3 Express One Zone in the Amazon S3 console, through the AWS Command Line Interface (AWS CLI), or by using the AWS SDKs, S3 Express One Zone creates a session on your behalf.

If you use the Amazon S3 REST API, you can then use the `CreateSession` API operation to obtain temporary security credentials that include an access key ID, a secret access key, a session token, and an expiration time. The temporary credentials provide the same permissions as long-term security credentials, such as IAM user credentials, but temporary security credentials must include a session token.

**Session Mode**

Session mode defines the scope of the session. In your bucket policy, you can specify the `s3express:SessionMode` condition key to control who can create a `ReadWrite` or `ReadOnly` session. For more information about `ReadWrite` or `ReadOnly` sessions, see the `x-amz-create-session-mode` parameter for CreateSession in the *Amazon S3 API Reference*. For more information about the bucket policy to create, see Example directory bucket policies for S3 Express One Zone.

**Session Token**

When you make a call by using temporary security credentials, the call must include a session token. The session token is returned along with the temporary credentials. A session token is scoped to your directory bucket and is used to verify that the security credentials are valid and haven't expired. To protect your sessions, temporary security credentials expire after 5 minutes.

**`CopyObject` and `HeadBucket`**

Temporary security credentials are scoped to a specific directory bucket and are automatically enabled for all Zonal (object-level) operation API calls to a given directory bucket. Unlike other Zonal endpoint API operations, `CopyObject` and `HeadBucket` don't use `CreateSession` authentication. All `CopyObject` and `HeadBucket` requests must be authenticated and signed by using IAM credentials. However, `CopyObject` and `HeadBucket` are still authorized by `s3express:CreateSession`, like other Zonal endpoint API operations.

For more information, see CreateSession in the *Amazon Simple Storage Service API Reference*.

# Security best practices for S3 Express One Zone

Amazon S3 Express One Zone provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful recommendations rather than prescriptions.

## Default Block Public Access and Object Ownership settings

To use the S3 Express One Zone storage class, you must use an S3 directory bucket. Directory buckets support S3 Block Public Access and S3 Object Ownership. These S3 features are used to audit and manage access to your buckets and objects.

By default, all Block Public Access settings for directory buckets are enabled. In addition, Object Ownership is set to bucket owner enforced, which means that access control lists (ACLs) are disabled. These settings can't be modified. For more information about these features, see the section called “Blocking public access” and the section called “Controlling object ownership”.

> **Note**
>
> You can't grant access to objects stored in directory buckets. You can grant access only to your directory buckets. The authorization model for S3 Express One Zone is different than the authorization model for Amazon S3. For more information, see `CreateSession` authorization.

## Authentication and authorization

The authentication and authorization mechanisms for S3 Express One Zone differ, depending on whether you are making requests to Zonal endpoint API operations or Regional endpoint API operations. Zonal API operations are object-level (data plane) operations. Regional API operations are bucket-level (control plane) operations.

With S3 Express One Zone, you authenticate and authorize requests to Zonal endpoint API operations through a new session-based mechanism that is optimized to provide the lowest latency. With session-based authentication, the AWS SDKs use the `CreateSession` API operation to request temporary credentials that provide low-latency access to your directory bucket. These temporary credentials are scoped to a specific directory bucket and expire after 5 minutes. You can

use these temporary credentials to sign Zonal (object level) API calls. For more information, see `CreateSession` authorization.

**Signing requests with S3 Express One Zone credentials**

You use your S3 Express One Zone credentials to sign Zonal endpoint (object level) API requests with AWS Signature Version 4, with `s3express` as the service name. When you sign your requests, use the secret key that's returned from `CreateSession` and also provide the session token with the `x-amzn-s3session-token header`. For more information, see CreateSession.

The supported AWS SDKs for S3 Express One Zone class manage credentials and signing on your behalf. We recommend using the AWS SDKs for S3 Express One Zone to refresh credentials and sign requests for you.

**Signing requests with IAM credentials**

All Regional (bucket-level) API calls must be authenticated and signed by AWS Identity and Access Management (IAM) credentials instead of temporary session credentials. IAM credentials consist of the access key ID and secret access key for the IAM identities. All `CopyObject` and `HeadBucket` requests must also be authenticated and signed by using IAM credentials.

To achieve the lowest latency for your Zonal (object-level) operation calls, we recommend using S3 Express One Zone credentials obtained from calling `CreateSession` to sign your requests, except for requests to `CopyObject` and `HeadBucket`.

## Use AWS CloudTrail

AWS CloudTrail provides a record of the actions taken by a user, a role, or an AWS service in Amazon S3. You can use information collected by CloudTrail to determine the following:

- The request that was made to Amazon S3
- The IP address from which the request was made
- Who made the request
- When the request was made
- Additional details about the request

When you set up your AWS account, CloudTrail management events are enabled by default. The following Regional endpoint API operations (bucket-level, or control plane, API operations) are logged to CloudTrail.

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- ListMultipartUploads

> **Note**
>
> `ListMultipartUploads` is a Zonal endpoint API operation. However, it is logged to CloudTrail as a management event. For more information, see ListMultipartUploads in the *Amazon Simple Storage Service API Reference*.

By default, CloudTrail trails don't log data events, but you can configure trails to log data events for directory buckets that you specify, or to log data events for all the directory buckets in your AWS account. The following Zonal endpoint API operations (object-level, or data plane, API operations) are logged to CloudTrail.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateSession
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2

- ListParts
- PutObject
- UploadPart
- UploadPartCopy

For more information on using AWS CloudTrail with S3 Express One Zone, see Logging with AWS CloudTrail for S3 Express One Zone.

**Implement monitoring by using AWS monitoring tools**

Monitoring is an important part of maintaining the reliability, security, availability, and performance of Amazon S3 and your AWS solutions. AWS provides several tools and services to help you monitor Amazon S3 and your other AWS services. For example, you can monitor Amazon CloudWatch metrics for Amazon S3, particularly the `BucketSizeBytes` and `NumberOfObjects` storage metrics.

Objects stored in the S3 Express One Zone storage class won't be reflected in the `BucketSizeBytes` and `NumberOfObjects` storage metrics for Amazon S3. However, the `BucketSizeBytes` and `NumberOfObjects` storage metrics are supported for S3 Express One Zone. To see the metrics of your choice, you can differentiate between the Amazon S3 storage classes and the S3 Express One Zone storage class by specifying a `StorageType` dimension. For more information, see Monitoring metrics with Amazon CloudWatch.

For more information, see Monitoring metrics with Amazon CloudWatch and Monitoring Amazon S3.

# Logging with AWS CloudTrail for S3 Express One Zone

Amazon S3 is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, when it was made, and additional details. When a supported event activity occurs in Amazon S3, that activity is recorded in a CloudTrail event. You can use AWS CloudTrail trail to log management events and data events for S3 Express One Zone. For more information, see Amazon S3 CloudTrail events and What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

## CloudTrail management events for S3 Express One Zone

By default, CloudTrail logs bucket-level actions for directory buckets as management events. The `eventsource` for CloudTrail management events for S3 Express One Zone is `s3express.amazonaws.com`. When you set up your AWS account, CloudTrail management events are enabled by default. The following Regional endpoint API operations (bucket-level, or control plane, API operations) are logged to CloudTrail.

- `CreateBucket`
- `DeleteBucket`
- `DeleteBucketPolicy`
- `PutBucketPolicy`
- `GetBucketPolicy`
- `ListDirectoryBuckets`
- `ListMultipartUploads`

> **Note**
>
> `ListMultipartUploads` is a Zonal endpoint API operation. However, this API operation is logged to CloudTrail as a management event. For more information, see `ListMultipartUploads` in the *Amazon Simple Storage Service API Reference*.

For more information on CloudTrail management events, see Logging management events in the *AWS CloudTrail User Guide.*

## CloudTrail data events for S3 Express One Zone

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail trails don't log data events, but you can configure trails to log data events for objects stored in general purpose buckets and directory buckets. For more information, see Enable logging for objects in a bucket using the console .

When you log data events for a trail in CloudTrail, you can choose to use advanced event selectors or basic event selectors. To log data events for objects stored in directory buckets, you must use advanced event selectors. When configuring advanced resource selectors, you will choose or specify the resource type for S3 Express One Zone which is `AWS::S3Express::Object`.

The following Zonal endpoint API operations (object-level , or. data plane, API operations) are logged to CloudTrail.

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CreateSession`
- `CopyObject`
- `CreateMultipartUpload`
- `DeleteObject`
- `DeleteObjects`
- `GetObject`
- `GetObjectAttributes`
- `HeadBucket`
- `HeadObject`
- `ListObjectsV2`
- `ListParts`
- `PutObject`
- `UploadPart`
- `UploadPartCopy`

For more information on CloudTrail data events, see Logging data events in the *AWS CloudTrail User Guide*.

For additional information about CloudTrail events for S3 Express One Zone, see the following topics:

**Topics**

- CloudTrail log file examples for S3 Express One Zone

# CloudTrail log file examples for S3 Express One Zone

A CloudTrail log file includes information about the requested API operation, the date and time of the operation, request parameters, and so on. This topic features examples for CloudTrail data events and management events for S3 Express One Zone.

### Topics



## CloudTrail data event log file examples for Amazon S3 Express One Zone

The following example shows a CloudTrail log file example that demonstrates CreateSession.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
      "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "AROAIDPPEZS35WEXAMPLE",
          "arn": "arn:aws:iam::111122223333:role/RoleToBeAssumed",
          "accountId": "111122223333",
          "userName":"RoleToBeAssumed
        },

        "attributes": {
          "creationDate": "2024-07-02T00:21:16Z",
        "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2024-07-02T00:22:11Z",
    "eventSource": "s3express.amazonaws.com",
    "eventName": "CreateSession",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "72.21.198.68",
```

```
        "userAgent": "aws-sdk-java/2.20.160-SNAPSHOT
 Linux/5.10.216-225.855.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.23+9-LTS
 Java/11.0.23 vendor/Amazon.com_Inc. md/internal exec-env/AWS_Lambda_java11 io/sync
 http/Apache cfg/retry-mode/standard",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3".
            "host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
            "x-amz-create-session-mode": "ReadWrite"
        },
        "responseElements": {
            "credentials": {
                "accessKeyId": "AKIAI44QH8DHBEXAMPLE"
                "expiration": ""Mar 20, 2024, 11:16:09 PM",
                "sessionToken": "<session token string>"
           },
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 1815,
            "availabilityZone": "usw2-az1"
          },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333"
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
```

```
            "recipientAccountId": "111122223333",
            "eventCategory": "Data",
            "tlsDetails": {
              "tlsVersion": "TLSv1.3",
              "cipherSuite": "TLS_AES_128_GCM_SHA256",
              "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

To use Zonal endpoint API operations (object-level, or data plane, operations), you can use the `CreateSession` API operation to create and manage sessions that are optimized for low-latency authorization of data requests. You can also use `CreateSession` to reduce the amount of logging. To identify which Zonal API operations were performed during a session, you can match the `accessKeyId` under the `responseElements` in your `CreateSession` log file to the `accessKeyId` in the log file of other Zonal API operations. For more information, see [`CreateSession` authorization].

The following example shows a CloudTrail log file example that demonstrates the [`GetObject`] API operation that was authenticated by `CreateSession`.

```
    {
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
          "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "sessionContext": {
            "attributes": {
              "creationDate": "2024-07-02T00:21:49Z"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:01Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
```

```
        "userAgent": "aws-sdk-java/2.25.66 Linux/5.10.216-225.855.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/17.0.11+9-LTS Java/17.0.11 vendor/Amazon.com_Inc. md/internal
 exec-env/AWS_Lambda_java17 io/sync http/Apache cfg/retry-mode/legacy",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3",
          "x-amz-checksum-mode": "ENABLED",
          "Host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
          "key": "test-get-obj-with-checksum"
        },
        "responseElements": null,
        "additionalEventData": {
          "SignatureVersion": "Sigv4",
          "CipherSuite": "TLS_AES_128_GCM_SHA256",
          "bytesTransferredIn": 0,
          "AuthenticationMethod": "AuthHeader",
          "x-amz-id-2": "oOy6w8K7LFsyFN",
          "bytesTransferredOut": 9,
          "availabilityZone": "usw2-az1",
          "sessionModeApplied": "ReadWrite"
         },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333",
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
```

```
            "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
          }
        }
```

In the `GetObject` log file example above, the `accessKeyId`(AKIAI44QH8DHBEXAMPLE) matches the `accessKeyId` under the `responseElements` in the CreateSession log file example. The matching `accessKeyId` indicates the session in which `GetObject` operation was performed.

# Optimizing Amazon S3 Express One Zone performance

Amazon S3 Express One Zone is a high-performance, single Availability Zone (AZ) S3 storage class that's purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the first S3 storage class that gives you the option to co-locate high-performance object storage and AWS compute resources, such as Amazon Elastic Compute Cloud, Amazon Elastic Kubernetes Service, and Amazon Elastic Container Service, within a single Availability Zone. Co-locating your storage and compute resources optimizes compute performance and costs and provides increased data-processing speed.

S3 Express One Zone provides similar performance elasticity to other S3 storage classes, but with consistent single-digit millisecond first-byte read and write request latencies—up to 10x faster than S3 Standard. S3 Express One Zone is designed from the ground up to support burst throughput up to very high aggregate levels. The S3 Express One Zone storage class uses a custom-built architecture to optimize for performance and deliver consistently low request latency by storing data on high-performance hardware. The object protocol for S3 Express One Zone has been enhanced to streamline authentication and metadata overhead.

To further increase access speed and support hundreds of thousands of requests per second, S3 Express One Zone stores data in a new bucket type—an Amazon S3 directory bucket. Each S3 directory bucket can support hundreds of thousands of transactions per second (TPS).

The combination of high-performance, purpose-built hardware and software that delivers single-digit millisecond data access speed and directory buckets that scale for large numbers of transactions per second makes S3 Express One Zone the best Amazon S3 storage class for request-intensive operations or performance-critical applications.

The following topics describe best practice guidelines and design patterns for optimizing performance with applications that use the S3 Express One Zone storage class.

**Topics**



# Performance guidelines and design patterns for S3 Express One Zone

When building applications that upload and retrieve objects from Amazon S3 Express One Zone, follow our best practice guidelines to optimize performance. To use the S3 Express One Zone storage class, you must create an S3 directory bucket. The S3 Express One Zone storage class isn't supported for use with S3 general purpose buckets.

For performance guidelines for all other Amazon S3 storage classes and S3 general purpose buckets, see Best practices design patterns: optimizing Amazon S3 performance.

To obtain the best performance for your application when using the S3 Express One Zone storage class and directory buckets, we recommend the following guidelines and design patterns.

**Topics**



## Co-locate S3 Express One Zone storage with your AWS compute resources

Each directory bucket is stored in a single Availability Zone that you select when you create the bucket. You can get started by creating a new directory bucket in an Availability Zone local to your compute workloads or resources. You can then immediately begin very low-latency reads and writes. Directory buckets are the first S3 buckets where you can choose the Availability Zone in an AWS Region to reduce latency between compute and storage.

If you access directory buckets across Availability Zones, latency will increase. To optimize performance, we recommend that you access a directory bucket from Amazon Elastic Container

Service, Amazon Elastic Kubernetes Service, and Amazon Elastic Compute Cloud instances that are located in the same Availability Zone when possible.

## Directory buckets

Each directory bucket can support hundreds of thousands of transactions per second (TPS). Unlike general purpose buckets, directory buckets organize keys hierarchically into directories instead of prefixes. A prefix is a string of characters at the beginning of the object key name. You can think of prefixes as a way to organize your data in a similar way to directories. However, prefixes are not directories.

Prefixes organize data in a flat namespace within general purpose buckets, and there are no limits to the number of prefixes within a general purpose bucket. Each prefix can achieve at least 3,500 PUT/POST/DELETE or 5,500 GET/HEAD requests per second. You can also parallelize requests across multiple prefixes to scale performance. However, this scaling, in the case of both read and write operations, happens gradually and is not instantaneous. While general purpose buckets are scaling to your new higher request rate, you might receive some HTTP status code 503 (Service Unavailable) errors.

With a hierarchical namespace, the delimiter in the object key is important. The only supported delimiter is a forward slash (/). Directories are determined by delimiter boundaries. For example, the object key `dir1/dir2/file1.txt` results in the directories `dir1/` and `dir2/` being automatically created, and the object `file1.txt` being added to the `/dir2` directory in the path `dir1/dir2/file1.txt`.

The directories that are created when objects are uploaded to directory buckets have no per-prefix TPS limits and are automatically pre-scaled to reduce the chance of HTTP 503 (Service Unavailable) errors. This automatic scaling allows your applications to parallelize read and write requests within and across directories as needed.

## Directory bucket horizontal scaling request parallelization

You can achieve the best performance by issuing multiple concurrent requests to directory buckets to spread your requests over separate connections to maximize the accessible bandwidth. S3 Express One Zone doesn't have any limits for the number of connections made to your directory bucket. Individual directories can scale performance horizontally and automatically when large numbers of concurrent writes to the same directory are happening.

When an object key is initially created and its key name includes a directory, the directory is automatically created for the object. Subsequent object uploads to that same directory do not require the directory to be created, which reduces latency on object uploads to existing directories.

Although both shallow and deep directory structures are supported for storing objects within a directory bucket, directory buckets do automatically scale horizontally, with lower latency on concurrent uploads to the same directory or to parallel directory siblings.

## Use session-based authentication

S3 Express One Zone and directory buckets support a new session-based authorization mechanism to authenticate and authorize requests to a directory bucket. With session-based authentication, the AWS SDKs automatically use the `CreateSession` API operation to create a temporary session token that can be used for low-latency authorization of data requests to a directory bucket.

The AWS SDKs use the `CreateSession` API operation to request temporary credentials, and then automatically create and refresh tokens for you on your behalf every 5 minutes. To take advantage of the performance benefits of the S3 Express One Zone storage class, we recommended that you use the AWS SDKs to initiate and manage the `CreateSession` API request. For more information about this session-based model, see `CreateSession` authorization.

## S3 additional checksum best practices

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

CRC32 is the default checksum used by the AWS SDKs when transmitting data to or from S3 Express One Zone. We recommend using CRC32 and CRC32C for the best performance with the S3 Express One Zone storage class.

## Use the latest version of the AWS SDKs and common runtime libraries

Several of the AWS SDKs also provide the AWS Common Runtime (CRT) libraries to further accelerate performance in S3 clients. These SDKs include the AWS SDK for Java 2.x, the AWS SDK for C++, and the AWS SDK for Python (Boto3). The CRT-based S3 client transfers objects to and

from S3 Express One Zone with enhanced performance and reliability by automatically using the multipart upload API operation and byte-range fetches to automate horizontally scaling connections.

To achieve the highest performance with the S3 Express One Zone storage class, we recommend using the latest version of the AWS SDKs that include the CRT libraries or using the AWS Command Line Interface (AWS CLI).

## Performance troubleshooting

### Retry requests for latency-sensitive applications

S3 Express One Zone is purpose-built to deliver consistent levels of high-performance without additional tuning. However, setting aggressive timeout values and retries can further help drive consistent latency and performance. The AWS SDKs have configurable timeout and retry values that you can tune to the tolerances of your specific application.

### AWS Common Runtime (CRT) libraries and Amazon EC2 instance type pairing

Applications that perform a large number of read and write operations likely need more memory or computing capacity than applications that don't. When launching your Amazon Elastic Compute Cloud (Amazon EC2) instances for your performance-demanding workload, choose instance types that have the amount of these resources that your application needs. S3 Express One Zone high-performance storage is ideally paired with larger and newer instance types with larger amounts of system memory and more powerful CPUs and GPUs that can take advantage of higher-performance storage. We also recommend using the latest versions of the CRT-enabled AWS SDKs, which can better accelerate read and write requests in parallel.

### Use session-based authentication in AWS SDKs instead of the HTTP REST APIs

With Amazon S3, you can also optimize performance when you're using HTTP REST API requests by following the same best practices that are part of the AWS SDKs. However, with the session-based authorization and authentication mechanism that's used by S3 Express One Zone, we strongly recommend that you use the AWS SDKs to manage `CreateSession` and its managed session token. The AWS SDKs automatically create and refresh tokens on your behalf by using the `CreateSession` API operation. Using `CreateSession` saves on per-request round-trip latency to AWS Identity and Access Management (IAM) to authorize each request.

# Developing with S3 Express One Zone

Amazon S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. With S3 Express One Zone storage class, you use S3 directory buckets to store your data. Each directory bucket uses the S3 Express One Zone storage class to store objects in a single Availability Zone that you can select when you create the bucket.

After you've created your directory bucket, you can then immediately begin very low-latency reads and writes. You can communicate with your directory bucket by using an endpoint connection over a virtual private cloud (VPC), or you can use Zonal and Regional API operations to manage your objects and directory buckets. You can also use S3 Express One Zone storage class through the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, and the Amazon S3 REST API.

The Amazon S3 Express One Zone storage class is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy. If the existing device encounters a failure, S3 Express One Zone automatically shifts requests to new devices within an Availability Zone. This redundancy helps ensure uninterrupted access to your data within an Availability Zone.

**Topics**

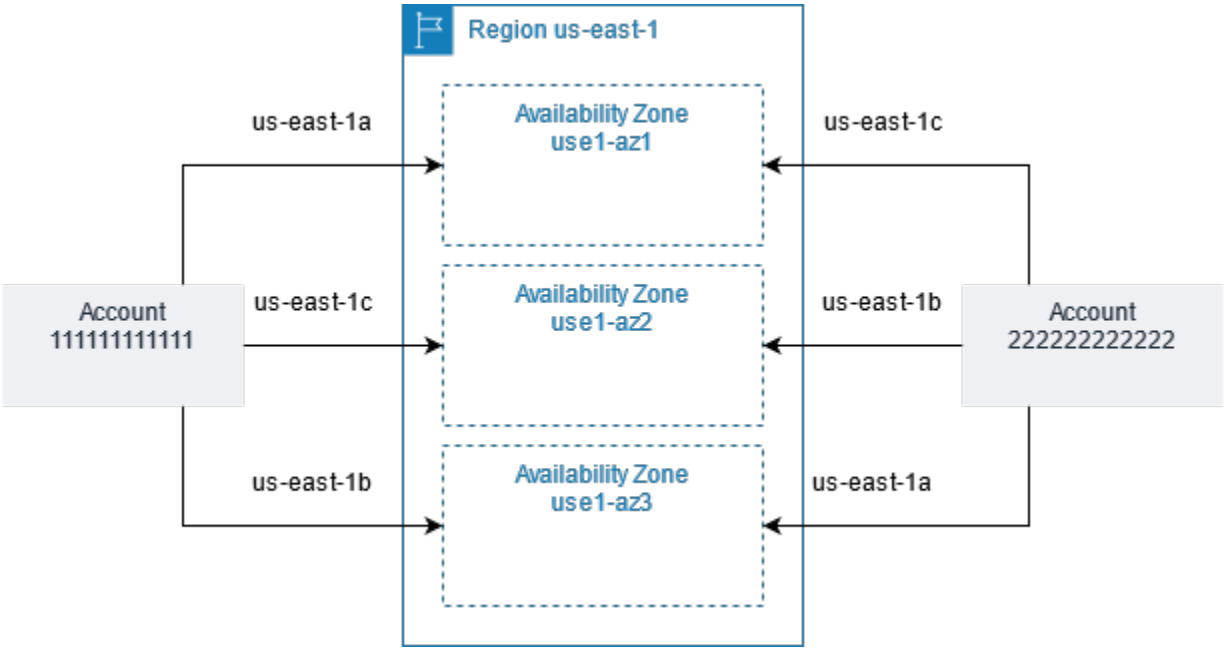


## S3 Express One Zone Availability Zones and Regions

An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. To optimize low-latency retrievals, objects in the Amazon S3 Express One Zone storage class are redundantly stored in S3 directory buckets in a single Availability Zone that's local to your compute workload. When you create a directory bucket, you choose the Availability Zone and AWS Region where your bucket will be located.

AWS maps the physical Availability Zones randomly to the Availability Zone names for each AWS account. This approach helps to distribute resources across the Availability Zones in an AWS Region, instead of resources likely being concentrated in the first Availability Zone for each Region. As a result, the Availability Zone `us-east-1a` for your AWS account might not represent the same physical location as `us-east-1a` for a different AWS account. For more information, see Regions and Availability Zones in the *Amazon EC2 User Guide*.

To coordinate Availability Zones across accounts, you must use the *AZ ID*, which is a unique and consistent identifier for an Availability Zone. For example, `use1-az1` is an AZ ID for the `us-east-1` Region and it has the same physical location in every AWS account. The following illustration shows how the AZ IDs are the same for every account, even though the Availability Zone names might be mapped differently for each account.




With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Single Availability Zone

S3 Express One Zone is supported in the following Regions and Availability Zones:

**S3 Express One Zone supported Regions and Availability Zones**

| Region name | Region code | Availability Zone ID |
|---|---|---|
| US East (N. Virginia) | us-east-1 | use1-az4 |
| | | use1-az5 |
| | | use1-az6 |
| US West (Oregon) | us-west-2 | usw2-az1 |
| | | usw2-az3 |
| | | usw2-az4 |
| Asia Pacific (Tokyo) | ap-northeast-1 | apne1-az1 |
| | | apne1-az4 |
| Europe (Stockholm) | eu-north-1 | eun1-az1 |
| | | eun1-az2 |
| | | eun1-az3 |

## Regional and Zonal endpoints

To access the Regional and Zonal endpoints for Amazon S3 Express One Zone from your virtual private cloud (VPC), you can use gateway VPC endpoints. After you create a gateway endpoint, you can add it as a target in your route table for traffic destined from your VPC to S3 Express One Zone. There is no additional charge for using gateway endpoints. For more information about how to configure gateway VPC endpoints, see Networking for S3 Express One Zone.

When you're working with S3 Express One Zone, bucket-level (control plane) API operations are available through a Regional endpoint and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`.

After you create a directory bucket, you can use Zonal (object level, or data plane endpoint API operations) to upload and manage the objects in your directory bucket. Zonal endpoint API operations are available through a Zonal endpoint. Examples of Zonal API operations are `PutObject` and `CopyObject`.

## Working with S3 Express One Zone by using the S3 console, AWS CLI, and AWS SDKs

You can work with the S3 Express One Zone storage class and directory buckets by using the AWS SDKs, Amazon S3 console, AWS Command Line Interface (AWS CLI), and Amazon S3 REST API.

### S3 Console

To get started using the S3 console, follow these steps:

- Creating a directory bucket
- Emptying a directory bucket
- Deleting a directory bucket

For a full tutorial, see Tutorial: Getting started with S3 Express One Zone.

### AWS SDKs

S3 Express One Zone supports the following AWS SDKs:

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript v3
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- AWS SDK for Kotlin

- AWS SDK for Rust

When you're working with S3 Express One Zone, we recommend using the latest version of the AWS SDKs. The supported AWS SDKs for S3 Express One Zone handle session establishment, refreshment, and termination on your behalf. This means that you can immediately start using API operations after you download and install the AWS SDKs and configure the necessary IAM permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

For information about the AWS SDKs, including how to download and install them, see Tools to Build on AWS.

For AWS SDK examples, see the following:

- Creating a directory bucket
- Emptying a directory bucket
- Deleting a directory bucket

### AWS Command Line Interface (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create directory buckets and use supported Regional and Zonal endpoint API operations for S3 Express One Zone.

To get started with the AWS CLI, see Get started with the AWS CLI in the *AWS CLI Command Reference*.

> **Note**
>
> To use directory buckets with the high-level `aws s3` commands, update your AWS CLI to the latest version. For more information about how to install and configure the AWS CLI, see Install or update the latest version of the AWS CLI in the *AWS CLI Command Reference*.

For AWS CLI examples, see the following:

- Creating a directory bucket
- Emptying a directory bucket
- Deleting a directory bucket

# S3 Express One Zone API operations

The Amazon S3 Express One Zone storage class supports both Regional (bucket level, or control plane) and Zonal (object level, or data plane) endpoint API operations. For more information, see Networking for S3 Express One Zone and Endpoints and gateway VPC endpoints.

### Regional endpoint API operations

The following Regional endpoint API operations are supported for S3 Express One Zone:

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

### Zonal endpoint API operations

The following Zonal endpoint API operations are supported for S3 Express One Zone:

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload

- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

# Managing data access with Amazon S3 access points

Amazon S3 access points simplify data access for any AWS service or customer application that stores data in S3. Access points are named network endpoints that are attached to buckets that you can use to perform S3 object operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions and network controls that S3 applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. You can configure any access point to accept requests only from a virtual private cloud (VPC) to restrict Amazon S3 data access to a private network. You can also configure custom block public access settings for each access point.

**Note**

- You can only use access points to perform operations on objects. You can't use access points to perform other Amazon S3 operations, such as modifying or deleting buckets. For a complete list of S3 operations that support access points, see Access point compatibility with AWS services.
- Access points work with some, but not all, AWS services and features. For example, you can't configure Cross-Region Replication to operate through an access point. For a complete list of AWS services that are compatible with S3 access points, see Access point compatibility with AWS services.

This section explains how to work with Amazon S3 access points. For information about working with buckets, see Buckets overview. For information about working with objects, see Amazon S3 objects overview.

**Topics**

# Configuring IAM policies for using access points

Amazon S3 access points support AWS Identity and Access Management (IAM) resource policies that allow you to control the use of the access point by resource, user, or other conditions. For an application or user to be able to access objects through an access point, both the access point and the underlying bucket must permit the request.

> **Important**
>
> Adding an S3 access point to a bucket doesn't change the bucket's behavior when the bucket is accessed directly through the bucket's name or Amazon Resource Name (ARN). All existing operations against the bucket will continue to work as before. Restrictions that you include in an access point policy apply only to requests made through that access point.

When you're using IAM resource policies, make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide recommendations to help you author policies that are functional and conform to security best practices.

To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

## Access point policy examples

The following examples demonstrate how to create IAM policies to control requests made through an access point.

> **Note**
>
> Permissions granted in an access point policy are effective only if the underlying bucket also allows the same access. You can accomplish this in two ways:
>
> 1. **(Recommended)** Delegate access control from the bucket to the access point, as described in Delegating access control to access points.

2. Add the same permissions contained in the access point policy to the underlying bucket's policy. The Example 1 access point policy example demonstrates how to modify the underlying bucket policy to allow the necessary access.

### Example 1 – Access point policy grant

The following access point policy grants IAM user *Jane* in account *123456789012* permissions to GET and PUT objects with the prefix *Jane/* through the access point *my-access-point* in account *123456789012*.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/Jane/*"
    }]
}
```

**Note**

For the access point policy to effectively grant access to *Jane*, the underlying bucket must also allow the same access to *Jane*. You can delegate access control from the bucket to the access point as described in Delegating access control to access points. Or, you can add the following policy to the underlying bucket to grant the necessary permissions to Jane. Note that the `Resource` entry differs between the access point and bucket policies.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
```

```
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/Jane/*"
    }]
}
```

### Example 2 – Access point policy with tag condition

The following access point policy grants IAM user *Mateo* in account *123456789012* permissions to GET objects through the access point *my-access-point* in the account *123456789012* that have the tag key *data* set with a value of *finance*.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect":"Allow",
        "Principal" : {
            "AWS": "arn:aws:iam::123456789012:user/Mateo"
        },
        "Action":"s3:GetObject",
        "Resource" : "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

### Example 3 – Access point policy that allows bucket listing

The following access point policy allows IAM user `Arnav` in the account *123456789012* permission to view the objects contained in the bucket underlying the access point *my-access-point* in the account *123456789012*.

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Arnav"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point"
    }]
}
```

### Example 4 – Service control policy

The following service control policy requires all new access points to be created with a virtual private cloud (VPC) network origin. With this policy in place, users in your organization can't create new access points that are accessible from the internet.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

### Example 5 – Bucket policy to limit S3 operations to VPC network origins

The following bucket policy limits access to all S3 object operations for the bucket *amzn-s3-demo-bucket* to access points with a VPC network origin.

> **Important**
>
> Before using a statement like the one shown in this example, make sure that you don't need to use features that aren't supported by access points, such as Cross-Region Replication.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

## Condition keys

S3 access points have condition keys that you can use in IAM policies to control access to your resources. The following condition keys represent only part of an IAM policy. For full policy examples, see Access point policy examples, the section called "Delegating access control to access points", and the section called "Granting permissions for cross-account access points".

### `s3:DataAccessPointArn`

This example shows a string that you can use to match on an access point ARN. The following example matches all access points for AWS account *123456789012* in Region *us-west-2*:

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

### `s3:DataAccessPointAccount`

This example shows a string operator that you can use to match on the account ID of the owner of an access point. The following example matches all access points that are owned by the AWS account *123456789012*.

```
"Condition" : {
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

### `s3:AccessPointNetworkOrigin`

This example shows a string operator that you can use to match on the network origin, either `Internet` or VPC. The following example matches only access points with a VPC origin.

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

For more information about using condition keys with Amazon S3, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

## Delegating access control to access points

You can delegate access control for a bucket to the bucket's access points. The following example bucket policy allows full access to all access points that are owned by the bucket owner's account. Thus, all access to this bucket is controlled by the policies attached to its access points. We recommend configuring your buckets this way for all use cases that don't require direct access to the bucket.

**Example 6 – Bucket policy that delegates access control to access points**

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

## Granting permissions for cross-account access points

To create an access point to a bucket that's owned by another account, you must first create the access point by specifying the bucket name and account owner ID. Then, the bucket owner must update the bucket policy to authorize requests from the access point. Creating an access point is similar to creating a DNS CNAME in that the access point doesn't provide access to the bucket contents. All bucket access is controlled by the bucket policy. The following example bucket policy allows `GET` and `LIST` requests on the bucket from an access point that's owned by a trusted AWS account.

Replace *`Bucket ARN`* with the ARN of the bucket.

**Example 7 – Bucket policy delegating permissions to another AWS account**

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point owner's
 account ID" }
        }
    }]
}
```

# Creating access points

Amazon S3 provides functionality for creating and managing access points. You can create S3 access points by using the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

By default, you can create up to 10,000 access points per Region for each of your AWS accounts. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS Service Quotas in the *AWS General Reference*.

**Note**

Because you might want to publicize your access point name so that other users can use the access point, avoid including sensitive information in the access point name. Access point names are published in a publicly accessible database known as the Domain Name System (DNS).

## Rules for naming Amazon S3 access points

Access point names must meet the following conditions:

- Must be unique within a single AWS account and Region
- Must comply with DNS naming restrictions
- Must begin with a number or lowercase letter
- Must be between 3 and 50 characters long
- Can't begin or end with a hyphen (-)
- Can't contain underscores (_), uppercase letters, or periods (.)
- Can't end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Using a bucket-style alias for your S3 bucket access point.

To create an access point, see the following topics.

**Topics**

- Creating an access point
- Creating access points restricted to a virtual private cloud
- Managing public access to access points

## Creating an access point

An access point is associated with exactly one Amazon S3 bucket. If you want to use a bucket in your AWS account, you must first create a bucket. For more information about creating buckets, see Creating, configuring, and working with Amazon S3 buckets.

You can also create a cross-account access point that's associated with a bucket in another AWS account, as long as you know the bucket name and the bucket owner's account ID. However, creating cross-account access points doesn't grant you access to data in the bucket until you are granted permissions from the bucket owner. The bucket owner must grant the access point owner's account (your account) access to the bucket through the bucket policy. For more information, see Granting permissions for cross-account access points.

By default, you can create up to 10,000 access points per Region for each of your AWS accounts. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS Service Quotas in the *AWS General Reference*.

The following examples demonstrate how to create an access point with the AWS CLI and the S3 console. For more information about how to create access points by using the REST API, see CreateAccessPoint in the *Amazon Simple Storage Service API Reference*.

### Using the S3 console

#### To create an access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create an access point.
3. In the left navigation pane, choose **Access Points**.
4. On the **Access Points** page, choose **Create access point**.
5. In the **Access point name** field, enter the name for the access point. For more information about naming access points, see Rules for naming Amazon S3 access points.
6. For **Bucket name**, specify the S3 bucket that you want to use with the access point.

   To use a bucket in your account, choose **Choose a bucket in this account**, and enter or browse for the bucket name.

   To use a bucket in a different AWS account, choose **Specify a bucket in another account**, and enter the AWS account ID and name of the bucket.

   > **Note**
   >
   > If you're using a bucket in a different AWS account, the bucket owner must update the bucket policy to authorize requests from the access point. For an example bucket policy, see Granting permissions for cross-account access points.

7. Choose a **Network origin**. If you choose **Virtual private cloud (VPC)**, enter the **VPC ID** that you want to use with the access point.

   For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.

8. Under **Block Public Access settings for this Access Point**, select the block public access settings that you want to apply to the access point. All block public access settings are enabled

by default for new access points. We recommend that you keep all settings enabled unless you know that you have a specific need to disable any of them.

> **Note**
>
> After you create an access point, you can't change its block public access settings.

For more information about using Amazon S3 Block Public Access with access points, see Managing public access to access points.

9. (Optional) Under **Access Point policy - *optional***, specify the access point policy. Before you save your policy, make sure to resolve any security warnings, errors, general warnings, and suggestions. For more information about specifying an access point policy, see Access point policy examples.
10. Choose **Create access point**.

#### Using the AWS CLI

The following example command creates an access point named *example-ap* for the bucket *amzn-s3-demo-bucket* in the account *111122223333*. To create the access point, you send a request to Amazon S3 that specifies the following:

- The access point name. For information about naming rules, see the section called "Rules for naming Amazon S3 access points".
- The name of the bucket that you want to associate the access point with.
- The account ID for the AWS account that owns the bucket.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket
```

When you're creating an access point by using a bucket in a different AWS account, include the `--bucket-account-id` parameter. The following example command creates an access point in the AWS account *111122223333*, using the bucket *amzn-s3-demo-bucket2*, which is in the AWS account *444455556666*.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket --bucket-account-id 444455556666
```

## Creating access points restricted to a virtual private cloud

When you create an access point, you can choose to make the access point accessible from the internet, or you can specify that all requests made through that access point must originate from a specific virtual private cloud (VPC). An access point that's accessible from the internet is said to have a network origin of `Internet`. It can be used from anywhere on the internet, subject to any other access restrictions in place for the access point, underlying bucket, and related resources, such as the requested objects. An access point that's only accessible from a specified VPC has a network origin of `VPC`, and Amazon S3 rejects any request made to the access point that doesn't originate from that VPC.

> **Important**
>
> You can only specify an access point's network origin when you create the access point. After you create the access point, you can't change its network origin.

To restrict an access point to VPC-only access, you include the `VpcConfiguration` parameter with the request to create the access point. In the `VpcConfiguration` parameter, you specify the VPC ID that you want to be able to use the access point. If a request is made through the access point, the request must originate from the VPC or Amazon S3 will reject it.

You can retrieve an access point's network origin using the AWS CLI, AWS SDKs, or REST APIs. If an access point has a VPC configuration specified, its network origin is `VPC`. Otherwise, the access point's network origin is `Internet`.

**Example**

***Example: Create an access point that's restricted to VPC access***

The following example creates an access point named `example-vpc-ap` for bucket `example-bucket` in account `123456789012` that allows access only from the `vpc-1a2b3c` VPC. The example then verifies that the new access point has a network origin of `VPC`.

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket example-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

To use an access point with a VPC, you must modify the access policy for your VPC endpoint. VPC endpoints allow traffic to flow from your VPC to Amazon S3. They have access control policies that control how resources within the VPC are allowed to interact with Amazon S3. Requests from your VPC to Amazon S3 only succeed through an access point if the VPC endpoint policy grants access to both the access point and the underlying bucket.

**Note**

To make resources accessible only within a VPC, make sure to create a private hosted zone for your VPC endpoint. To use a private hosted zone, modify your VPC settings so that the VPC network attributes `enableDnsHostnames` and `enableDnsSupport` are set to `true`.

The following example policy statement configures a VPC endpoint to allow calls to `GetObject` for a bucket named `awsexamplebucket1` and an access point named `example-vpc-ap`.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*",
            "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
        ]
    }]
}
```

**Note**

The `"Resource"` declaration in this example uses an Amazon Resource Name (ARN) to specify the access point. For more information about access point ARNs, see Using access points.

For more information about VPC endpoint policies, see Using endpoint policies for Amazon S3 in the *VPC User Guide*.

## Managing public access to access points

Amazon S3 access points support independent *block public access* settings for each access point. When you create an access point, you can specify block public access settings that apply to that access point. For any request made through an access point, Amazon S3 evaluates the block public access settings for that access point, the underlying bucket, and the bucket owner's account. If any of these settings indicate that the request should be blocked, Amazon S3 rejects the request.

For more information about the S3 Block Public Access feature, see Blocking public access to your Amazon S3 storage.

**Important**

- All block public access settings are enabled by default for access points. You must explicitly disable any settings that you don't want to apply to an access point.
- Amazon S3 currently doesn't support changing an access point's block public access settings after the access point has been created.

### Example

#### *Example: Create an access point with Custom Block Public Access Settings*

This example creates an access point named `example-ap` for bucket `example-bucket` in account `123456789012` with non-default Block Public Access settings. The example then retrieves the new access point's configuration to verify its Block Public Access settings.

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket example-bucket --public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
    "Name": "example-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

# Using access points

You can access the objects in an Amazon S3 bucket with an *access point* using the AWS Management Console, AWS CLI, AWS SDKs, or the S3 REST APIs.

Access points have Amazon Resource Names (ARNs). Access point ARNs are similar to bucket ARNs, but they are explicitly typed and encode the access point's Region and the AWS account ID of the access point's owner. For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

Access point ARNs use the format `arn:aws:s3:region:account-id:accesspoint/resource`. For example:

- **arn:aws:s3:us-west-2:123456789012:accesspoint/test** represents the access point named `test`, owned by account `123456789012` in Region `us-west-2`.
- **arn:aws:s3:us-west-2:123456789012:accesspoint/*** represents all access points under account `123456789012` in Region `us-west-2`.

ARNs for objects accessed through an access point use the format `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`. For example:

- **arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01** represents the object `unit-01`, accessed through the access point named `test`, owned by account `123456789012` in Region `us-west-2`.
- **arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/*** represents all objects for access point `test`, in account `123456789012` in Region `us-west-2`.
- **arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/*** represents all objects under prefix `unit-01/finance/` for access point `test`, in account `123456789012` in Region `us-west-2`.

## Accessing a bucket through S3 access points

S3 access points only support virtual-host-style addressing. To address a bucket through an access point, use the following format.

```
https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com
```

**Note**

- If your access point name includes dash (-) characters, include the dashes in the URL and insert another dash before the account ID. For example, to use an access point named `finance-docs` owned by account `123456789012` in Region `us-west-2`, the appropriate URL would be `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com`.
- S3 access points don't support access by HTTP, only secure access by HTTPS.

**Topics**



If you have a Virtual Private Cloud (VPC), see Managing Amazon S3 access with VPC endpoints and S3 Access Points.

## Monitoring and logging access points

Amazon S3 logs requests made through access points and requests made to the APIs that manage access points, such as `CreateAccessPoint` and `GetAccessPointPolicy`. To monitor and manage usage patterns, you can also configure Amazon CloudWatch Logs request metrics for access points.

**Topics**



### CloudWatch request metrics

To understand and improve the performance of applications that are using access points, you can use CloudWatch for Amazon S3 request metrics. Request metrics help you monitor Amazon S3 requests to quickly identify and act on operational issues.

By default, request metrics are available at the bucket level. However, you can define a filter for request metrics using a shared prefix, object tags, or an access point. When you create an access point filter, the request metrics configuration includes requests to the access point that you specify. You can receive metrics, set alarms, and access dashboards to view real-time operations performed through this access point.

You must opt in to request metrics by configuring them in the console or by using the Amazon S3 API. Request metrics are available at 1-minute intervals after some latency for processing. Request metrics are billed at the same rate as CloudWatch custom metrics. For more information, see Amazon CloudWatch pricing.

To create a request metrics configuration that filters by access point, see Creating a metrics configuration that filters by prefix, object tag, or access point.

### Request logs

You can log requests made through access points and requests made to the APIs that manage access points, such as `CreateAccessPoint` and `GetAccessPointPolicy`, by using server access logging and AWS CloudTrail.

CloudTrail log entries for requests made through access points include the access point ARN in the `resources` section of the log.

For example, suppose you have the following configuration:

- A bucket named `amzn-s3-demo-bucket1` in Region `us-west-2` that contains an object named `my-image.jpg`
- An access point named `my-bucket-ap` that is associated with `amzn-s3-demo-bucket1`
- An AWS account ID of `123456789012`

The following example shows the `resources` section of a CloudTrail log entry for the preceding configuration:

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1/my-image.jpg"
        },
        {"accountId": "123456789012",
```

```
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

For more information about S3 Server Access Logs, see Logging requests with server access logging. For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

## Using Amazon S3 access points with the Amazon S3 console

This section explains how to manage and use your Amazon S3 access points using the AWS Management Console. Before you begin, navigate to the detail page for the access point you want to manage or use, as described in the following procedure.

**Topics**



### Listing access points for your account

**To list all access points created in your AWS account**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **access points**.

4. On the **access points** page, under **access points**, view the access points in your AWS Region.
5. (Optional) Search for access points by name by entering a name into the text field next to the Region dropdown menu.
6. Choose the name of the access point you want to manage or use.

## Listing access points for a bucket

**To list all access points in you AWS account for a single bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region then choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Buckets**.
4. On the **Buckets** page, select the name of the bucket whose access points you want to list.
5. On the bucket detail page, choose the **access points** tab.
6. Choose the name of the access point you want to manage or use.

## Viewing configuration details for an access point

1. Navigate to the access point detail page for the access point whose details you want to view, as described in Listing access points for your account.
2. Under **access point overview**, view configuration details and properties for the selected access point.

## Using an access point

1. Navigate to the access point detail page for the access point you want to use, as described in Listing access points for your account.
2. Under the **Objects** tab, choose the name of an object or objects that you want to access through the access point. On the object operation pages, the console displays a label above the name of your bucket that shows the access point that you're currently using. While you're using the access point, you can only perform the object operations that are allowed by the access point permissions.

> **Note**
>
> - The console view always shows all objects in the bucket. Using an access point as described in this procedure restricts the operations you can perform on those objects, but not whether you can see that they exist in the bucket.
> - The S3 Management Console doesn't support using virtual private cloud (VPC) access points to access bucket resources. To access bucket resources from a VPC access point, use the AWS CLI, AWS SDKs, or Amazon S3 REST APIs.

## Viewing block public access settings for an access point

1. Navigate to the access point detail page for the access point whose settings you want to view, as described in Listing access points for your account.
2. Choose **Permissions**.
3. Under **access point policy**, review the access point's Block Public Access settings.

> **Note**
>
> You can't change the Block Public Access settings for an access point after the access point is created.

## Editing an access point policy

1. Navigate to the access point detail page for the access point whose policy you want to edit, as described in Listing access points for your account.
2. Choose **Permissions**.
3. Under **access point policy**, choose **Edit**.
4. Enter the access point policy in the text field. The console automatically displays the Amazon Resource Name (ARN) for the access point, which you can use in the policy.

### Deleting an access point

1. Navigate to the list of access points for your account or for a specific bucket, as described in Listing access points for your account.
2. Select the option button next to the name of the access point that you want to delete.
3. Choose **Delete**.
4. Confirm that you want to delete your access point by entering its name in the text field that appears, and choose **Delete**.

## Using a bucket-style alias for your S3 bucket access point

When you create an access point, Amazon S3 automatically generates an alias that you can use instead of an Amazon S3 bucket name for data access. You can use this access point alias instead of an Amazon Resource Name (ARN) for access point data plane operations. For a list of these operations, see Access point compatibility with AWS services.

The following shows an example ARN and access point alias for an access point named *my-access-point*.

- **ARN** – `arn:aws:s3:`*region*`:`*account-id*`:accesspoint/`*my-access-point*
- **Access point alias** – *my-access-point*`-`*hrzrlukc5m36ft7okagglf3gmwluquse1b*`-s3alias`

For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

### Access point alias names

An access point alias name is created within the same namespace as an Amazon S3 bucket. This alias name is automatically generated and cannot be changed. An access point alias name meets all the requirements of a valid Amazon S3 bucket name and consists of the following parts:

*access point prefix*`-`*metadata*`-s3alias`

**Note**

The `-s3alias` suffix is reserved for access point alias names and can't be used for bucket or access point names. For more information about Amazon S3 bucket-naming rules, see Bucket naming rules.

## Access point alias use cases and limitations

When adopting access points, you can use access point alias names without requiring extensive code changes.

When you create an access point, Amazon S3 automatically generates an access point alias name, as shown in the following example. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --bucket amzn-s3-demo-bucket1 --name my-access-point
 --account-id 111122223333
{
    "AccessPointArn":
    "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

You can use this access point alias name instead of an Amazon S3 bucket name in any data plane operation. For a list of these operations, see Access point compatibility with AWS services.

The following AWS CLI example for the `get-object` command uses the bucket's access point alias to return information about the specified object. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
```

```
    "Metadata": {}
}
```

### Limitations

- Aliases cannot be configured by customers.
- Aliases cannot be deleted or modified or disabled on an access point.
- You can use this access point alias name instead of an Amazon S3 bucket name in some data plane operations. For a list of these operations, see Access point compatibility with S3 operations.
- You can't use an access point alias name for Amazon S3 control plane operations. For a list of Amazon S3 control plane operations, see Amazon S3 Control in the *Amazon Simple Storage Service API Reference*.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.
- Aliases cannot be used in AWS Identity and Access Management (IAM) policies.
- Aliases cannot be used as a logging destination for S3 server access logs.
- Aliases cannot be used as a logging destination for AWS CloudTrail logs.
- Amazon SageMaker GroundTruth does not support access point aliases.

## Using access points with compatible Amazon S3 operations

The following examples demonstrate how to use access points with compatible operations in Amazon S3.

### Topics

## Access point compatibility with AWS services

Amazon S3 access point aliases allow applications that require an S3 bucket name to easily use an access point. You can use S3 access point aliases where you use S3 bucket names to access data in S3. For more information, see Access point alias use cases and limitations.

## Access point compatibility with S3 operations

You can use access points to access a bucket using the following subset of Amazon S3 APIs. All the operations listed below can accept either access point ARNs or access point aliases:

### S3 operations

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject (same-region copies only)
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetBucketAcl
- GetBucketCors
- GetBucketLocation
- GetBucketNotificationConfiguration
- GetBucketPolicy
- GetObject
- GetObjectAcl
- GetObjectAttributes
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects

- ListObjectsV2
- ListObjectVersions
- ListParts
- Presign
- PutObject
- PutObjectLegalHold
- PutObjectRetention
- PutObjectAcl
- PutObjectTagging
- RestoreObject
- UploadPart
- UploadPartCopy (same-region copies only)

### Request an object through an access point

The following example requests the object `my-image.jpg` through the access point `prod` owned by account ID `123456789012` in Region `us-west-2`, and saves the downloaded file as `download.jpg`.

AWS CLI

```
aws s3api get-object --key my-image.jpg --bucket arn:aws:s3:us-
west-2:123456789012:accesspoint/prod download.jpg
```

### Upload an object through an access point alias

The following example uploads the object `my-image.jpg` through the access point alias `my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias` owned by account ID `123456789012` in Region `us-west-2`.

AWS CLI

```
aws s3api put-object --bucket my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-
s3alias --key my-image.jpg --body my-image.jpg
```

## Delete an object through an access point

The following example deletes the object `my-image.jpg` through the access point `prod` owned by account ID `123456789012` in Region `us-west-2`.

AWS CLI

```
aws s3api delete-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg
```

## List objects through an access point alias

The following example lists objects through the access point alias `my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias` owned by account ID `123456789012` in Region `us-west-2`.

AWS CLI

```
aws s3api list-objects-v2 --bucket my-access-point-
hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias
```

## Add a tag set to an object through an access point

The following example adds a tag set to the existing object `my-image.jpg` through the access point `prod` owned by account ID `123456789012` in Region `us-west-2`.

AWS CLI

```
aws s3api put-object-tagging --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/
prod --key my-image.jpg --tagging TagSet=[{Key="finance",Value="true"}]
```

## Grant access permissions through an access point using an ACL

The following example applies an ACL to an existing object `my-image.jpg` through the access point `prod` owned by account ID `123456789012` in Region `us-west-2`.

AWS CLI

```
aws s3api put-object-acl --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg --acl private
```

## Access points restrictions and limitations

Amazon S3 access points have the following restrictions and limitations:

- Each access point is associated with exactly one bucket, which you must specify when you create the access point. After you create an access point, you can't associate it with a different bucket. However, you can delete an access point, and then create another one with the same name and associate that new access point with a different bucket.
- Access point names must meet certain conditions. For more information about naming access points, see Rules for naming Amazon S3 access points.
- After you create an access point, you can't change its virtual private cloud (VPC) configuration.
- Access point policies are limited to 20 KB in size.
- You can create a maximum of 10,000 access points per AWS account per Region. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS service quotas in the *AWS General Reference*.
- In AWS Regions where you have more than 1,000 access points, you can't search for an access point by name in the Amazon S3 console.
- You can't use an access point as a destination for S3 Replication. For more information about replication, see Replicating objects overview.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.
- You can address access points only by using virtual-host-style URLs. For more information about virtual-host-style addressing, see Accessing and listing an Amazon S3 bucket.
- API operations that control access point functionality (for example, `PutAccessPoint` and `GetAccessPointPolicy`) don't support cross-account calls.
- You must use AWS Signature Version 4 when making requests to an access point by using the REST APIs. For more information about authenticating requests, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

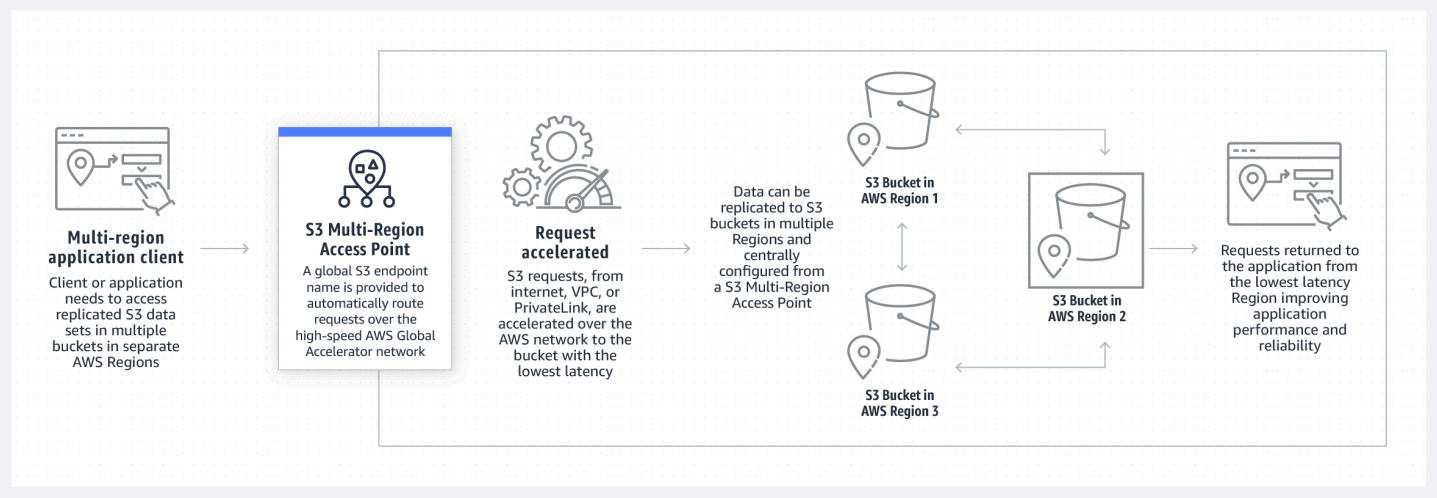
- Access points only support requests over HTTPS. Amazon S3 will automatically respond with an HTTP redirect for any requests made via HTTP, to upgrade the request to HTTPS.
- Access points don't support anonymous access.
- Cross-account access points don’t grant you access to data until you are granted permissions from the bucket owner. The bucket owner always retains ultimate control over access to the data and must update the bucket policy to authorize requests from the cross-account access point. To view a bucket policy example, see Configuring IAM policies for using access points.
- When you're viewing a cross-account access point in the Amazon S3 console, the **Access** column displays **Unknown**. The Amazon S3 console can't determine if public access is granted for the associated bucket and objects. Unless you require a public configuration for a specific use case, we recommend that you and the bucket owner block all public access to the access point and the bucket. For more information, see Blocking public access to your Amazon S3 storage.

# Multi-Region Access Points in Amazon S3

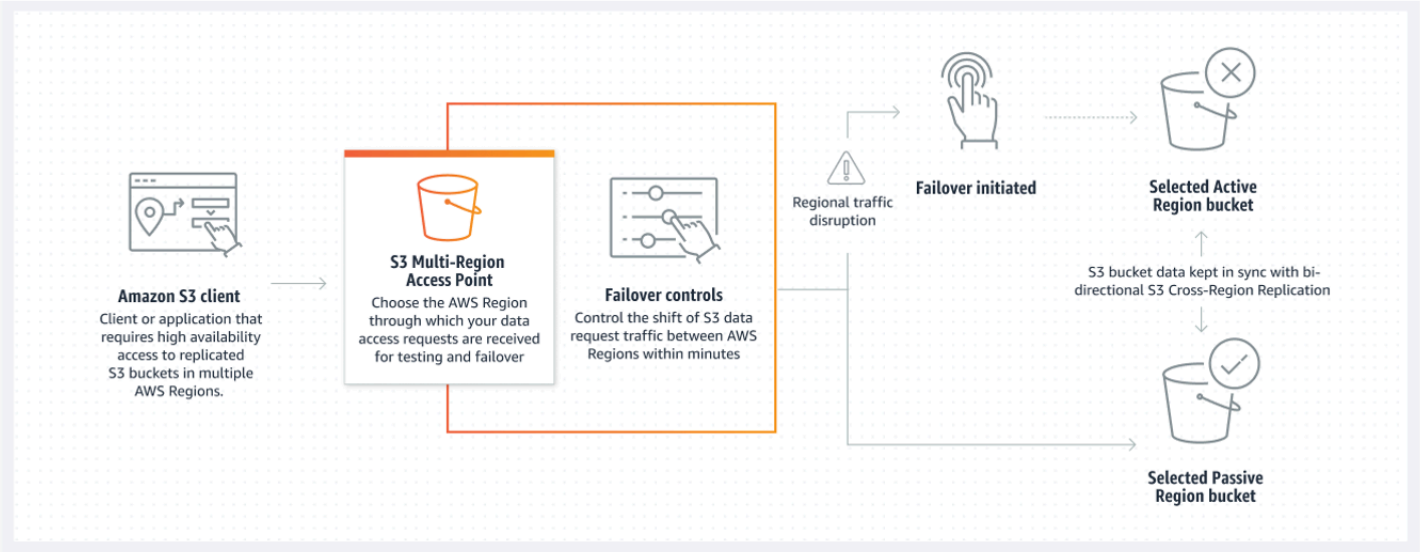
Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, Multi-Region Access Points provide built-in network resilience with acceleration of internet-based requests to Amazon S3. Application requests made to a Multi-Region Access Point global endpoint use AWS Global Accelerator to automatically route over the AWS global network to the closest-proximity S3 bucket with an active routing status.

When you create a Multi-Region Access Point, you specify a set of AWS Regions where you want to store data to be served through that Multi-Region Access Point. You can use S3 Cross-Region Replication (CRR) to synchronize data among buckets in those Regions. You can then request or write data through the Multi-Region Access Point global endpoint. Amazon S3 automatically serves requests to the replicated dataset from the closest available Region. Multi-Region Access Points are also compatible with applications that are running in Amazon virtual private clouds (VPCs), including those that are using AWS PrivateLink for Amazon S3.

The following image is a graphical representation of an Amazon S3 Multi-Region Access Point in an active-active configuration. The graphic shows how Amazon S3 requests are automatically routed to buckets in the closest active AWS Region.




The following image is a graphical representation of an Amazon S3 Multi-Region Access Point in an active-passive configuration. The graphic shows how you can control Amazon S3 data-access traffic to fail over between active and passive AWS Regions.




To learn more about how to use Multi-Region Access Points, see Tutorial: Getting started with Amazon S3 Multi-Region Access Points.

**Topics**

- Creating Multi-Region Access Points
- Configuring a Multi-Region Access Point for use with AWS PrivateLink
- Making requests through a Multi-Region Access Point

# Creating Multi-Region Access Points

To create a Multi-Region Access Point in Amazon S3, you do the following:

- Specify the name for the Multi-Region Access Point.
- Choose one bucket in each AWS Region that you want to serve requests for the Multi-Region Access Point.
- Configure the Amazon S3 Block Public Access settings for the Multi-Region Access Point.

You provide all of this information in a create request, which Amazon S3 processes asynchronously. Amazon S3 provides a token that you can use to monitor the status of the asynchronous creation request.

Make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer

runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that are functional and conform to security best practices. To learn more about validating policies using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

When you use the API, the request to create a Multi-Region Access Point is asynchronous. When you submit a request to create a Multi-Region Access Point, Amazon S3 synchronously authorizes the request. It then immediately returns a token that you can use to track the progress of the creation request. For more information about tracking asynchronous requests to create and manage Multi-Region Access Points, see Using Multi-Region Access Points with supported API operations.

After you create the Multi-Region Access Point, you can create an access control policy for it. Each Multi-Region Access Point can have an associated policy. A Multi-Region Access Point policy is a resource-based policy that you can use to limit the use of the Multi-Region Access Point by resource, user, or other conditions.

**Note**

For an application or user to be able to access an object through a Multi-Region Access Point, both of the following policies must permit the request:

- The access policy for the Multi-Region Access Point
- The access policy for the underlying bucket that contains the object

When the two policies are different, the more restrictive policy takes precedence. To simplify permissions management for Multi-Region Access Points, you can delegate access control from the bucket to the Multi-Region Access Point. For more information, see the section called “Multi-Region Access Point policy examples”.

Using a bucket with a Multi-Region Access Point doesn't change the bucket's behavior when the bucket is accessed through the existing bucket name or an Amazon Resource Name (ARN). All existing operations against the bucket continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests that are made through the Multi-Region Access Point.

You can update the policy for a Multi-Region Access Point after creating it, but you can't delete the policy. However, you can update the Multi-Region Access Point policy to deny all permissions.

**Topics**



## Rules for naming Amazon S3 Multi-Region Access Points

When you create a Multi-Region Access Point, you give it a name, which is a string that you choose. You can't change the name of the Multi-Region Access Point after it is created. The name must be unique in your AWS account, and it must conform to the naming requirements listed in Multi-Region Access Point restrictions and limitations. To help you identify the Multi-Region Access Point, use a name that is meaningful to you, to your organization, or that reflects the scenario.

You use this name when invoking Multi-Region Access Point management operations, such as `GetMultiRegionAccessPoint` and `PutMultiRegionAccessPointPolicy`. The name is not used to send requests to the Multi-Region Access Point, and it doesn't need to be exposed to clients who make requests by using the Multi-Region Access Point.

When Amazon S3 creates a Multi-Region Access Point, it automatically assigns an alias to it. This alias is a unique alphanumeric string that ends in `.mrap`. The alias is used to construct the hostname and the Amazon Resource Name (ARN) for a Multi-Region Access Point. The fully qualified name is also based on the alias for the Multi-Region Access Point.

You can't determine the name of a Multi-Region Access Point from its alias, so you can disclose an alias without risk of exposing the name, purpose, or owner of the Multi-Region Access Point. Amazon S3 selects the alias for each new Multi-Region Access Point, and the alias can't be changed. For more information about addressing a Multi-Region Access Point, see Making requests through a Multi-Region Access Point.

Multi-Region Access Point aliases are unique throughout time and aren't based on the name or configuration of a Multi-Region Access Point. If you create a Multi-Region Access Point, and then

delete it and create another one with the same name and configuration, the second Multi-Region Access Point will have a different alias than the first. New Multi-Region Access Points can never have the same alias as a previous Multi-Region Access Point.

## Rules for choosing buckets for Amazon S3 Multi-Region Access Points

Each Multi-Region Access Point is associated with the Regions where you want to fulfill requests. The Multi-Region Access Point must be associated with exactly one bucket in each of those Regions. You specify the name of each bucket in the request to create the Multi-Region Access Point. Buckets that support the Multi-Region Access Point can either be in the same AWS account that owns the Multi-Region Access Point, or they can be in other AWS accounts.

A single bucket can be used by multiple Multi-Region Access Points.

**Important**

- You can specify the buckets that are associated with a Multi-Region Access Point only at the time that you create it. After it is created, you can’t add, modify, or remove buckets from the Multi-Region Access Point configuration. To change the buckets, you must delete the entire Multi-Region Access Point and create a new one.
- You can't delete a bucket that is part of a Multi-Region Access Point. If you want to delete a bucket that's attached to a Multi-Region Access Point, delete the Multi-Region Access Point first.
- If you add a bucket that's owned by another account to your Multi-Region Access Point, the bucket owner must also update their bucket policy to grant access permissions to the Multi-Region Access Point. Otherwise, the Multi-Region Access Point won't be able to retrieve data from that bucket. For example policies that show how to grant such access, see Multi-Region Access Point policy examples.
- Not all Regions support Multi-Region Access Points. To see the list of supported Regions, see Multi-Region Access Point restrictions and limitations.

You can create replication rules to synchronize data between buckets. These rules enable you to automatically copy data from source buckets to destination buckets. Having buckets connected to a Multi-Region Access Point does not affect how replication works. Configuring replication with Multi-Region Access Points is described in a later section.

**Important**

When you make a request to a Multi-Region Access Point, the Multi-Region Access Point isn't aware of the data contents of the buckets in the Multi-Region Access Point. Therefore, the bucket that gets the request might not contain the requested data. To create consistent datasets in the Amazon S3 buckets that are associated with a Multi-Region Access Point, we recommend that you configure S3 Cross-Region Replication (CRR). For more information, see Configuring replication for use with Multi-Region Access Points.

## Create an Amazon S3 Multi-Region Access Point

The following example demonstrates how to create a Multi-Region Access Point by using the Amazon S3 console.

### Using the S3 console

#### To create a Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose **Create Multi-Region Access Points** to begin creating your Multi-Region Access Point.
4. On the **Multi-Region Access Point** page, supply a name for the Multi-Region Access Point in the **Multi-Region Access Point name** field.
5. Select the buckets that will be associated with this Multi-Region Access Point. You can choose buckets that are in your account, or you can choose buckets from other accounts.

**Note**

You must add at least one bucket from either your account or other accounts. Also, be aware that Multi-Region Access Points support only one bucket per AWS Region. Therefore, you can't add two buckets from the same Region. AWS Regions that are disabled by default are not supported.

- To add a bucket that is in your account, choose **Add buckets**. A list of all the buckets in your account displays. You can search for your bucket by name, or sort the bucket names in alphabetical order.
- To add a bucket from another account, choose **Add bucket from other accounts**. Make sure that you know the exact bucket name and AWS account ID because you can't search or browse for buckets in other accounts.

> **Note**
>
> You must enter a valid AWS account ID and bucket name. The bucket must also be in a supported Region, or you will encounter an error when you try to create your Multi-Region Access Point. For the list of Regions that support Multi-Region Access Points, see Multi-Region Access Points restrictions and limitations.

6. (Optional) If you need to remove a bucket that you added, choose **Remove**.

> **Note**
>
> You can’t add or remove buckets to this Multi-Region Access Point after you’ve finished creating it.

7. Under **Block Public Access settings for this Multi-Region Access Point**, select the Block Public Access settings that you want to apply to the Multi-Region Access Point. By default, all Block Public Access settings are enabled for new Multi-Region Access Points. We recommend that you leave all settings enabled unless you know that you have a specific need to disable any of them.

> **Note**
>
> You can't change the Block Public Access settings for a Multi-Region Access Point after the Multi-Region Access Point has been created. Therefore, if you're going to block public access, make sure that your applications work correctly without public access before you create a Multi-Region Access Point.

8. Choose **Create Multi-Region Access Point**.

> **Important**
>
> When you add a bucket that's owned by another account to your Multi-Region Access Point, the bucket owner must also update their bucket policy to grant access permissions to the Multi-Region Access Point. Otherwise, the Multi-Region Access Point won't be able to retrieve data from that bucket. For example policies that show how to grant such access, see Multi-Region Access Point policy examples.

### Using the AWS CLI

You can use the AWS CLI to create a Multi-Region Access Point. When you create the Multi-Region Access Point, you must provide all the buckets that it will support. You can't add buckets to the Multi-Region Access Point after it has been created.

The following example creates a Multi-Region Access Point with two buckets by using the AWS CLI. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "amzn-s3-demo-bucket1" },
            { "Bucket": "amzn-s3-demo-bucket2" }
        ]
    }' --region us-west-2
```

## Blocking public access with Amazon S3 Multi-Region Access Points

Each Multi-Region Access Point has distinct settings for Amazon S3 Block Public Access. These settings operate in conjunction with the Block Public Access settings for the AWS account that owns the Multi-Region Access Point and the underlying buckets.

When Amazon S3 authorizes a request, it applies the most restrictive combination of these settings. If the Block Public Access settings for any of these resources (the Multi-Region Access Point owner account, the underlying bucket, or the bucket owner account) block access for the requested action or resource, Amazon S3 rejects the request.

We recommend that you enable all Block Public Access settings unless you have a specific need to disable any of them. By default, all Block Public Access settings are enabled for a Multi-Region Access Point. If Block Public Access is enabled, the Multi-Region Access Point can't accept internet-based requests.

> **Important**
>
> You can't change the Block Public Access settings for a Multi-Region Access Point after it has been created.

For more information about Amazon S3 Block Public Access, see Blocking public access to your Amazon S3 storage.

## Viewing Amazon S3 Multi-Region Access Points configuration details

The following example demonstrates how to view Multi-Region Access Point configuration details by using the Amazon S3 console.

**Using the S3 console**

**To create a Multi-Region Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of the Multi-Region Access Point for which you want to view the configuration details.

   - The **Properties** tab lists all of the buckets that are associated with your Multi-Region Access Point, the creation date, the Amazon Resource Name (ARN), and the alias. The AWS account ID column also lists any buckets owned by external accounts that are associated with your Multi-Region Access Point.

- The **Permissions** tab lists the Block Public Access settings that are applied to the buckets associated with this Multi-Region Access Point. You can also view the Multi-Region Access Point policy for your Multi-Region Access Point, if you've created one. The **Info** alert on the **Permissions** page also lists all the buckets (in your account and other accounts) for this Multi-Region Access Point that have the **Public Access is blocked** setting enabled.
- The **Replication and failover** tab provides a map view of the buckets that are associated with your Multi-Region Access Point and the Regions that the buckets reside int. If there are buckets from another account that you don't have permission to pull data from, the Region will be marked in red on the **Replication summary** map, indicating that it is an **AWS Region with errors getting replication status**.

  > **Note**
  >
  > To retrieve replication status information from a bucket in an external account, the bucket owner must grant you the `s3:GetBucketReplication` permission in their bucket policy.

  This tab also provides the replication metrics, replication rules, and failover statuses for the Regions that are used with your Multi-Region Access Point.

#### Using the AWS CLI

You can use the AWS CLI to view the configuration details for a Multi-Region Access Point.

The following AWS CLI example gets your current Multi-Region Access Point configuration. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name amzn-s3-demo-bucket1
```

## Deleting a Multi-Region Access Point

The following procedure explains how to delete a Multi-Region Access Point by using the Amazon S3 console.

Deleting a Multi-Region Access Point does not delete the buckets associated with the Multi-Region Access Point, only the Multi-Region Access Point itself.

### Using the S3 console

#### To delete a Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Select the option button next to the name of your Multi-Region Access Point.
4. Choose **Delete**.
5. In the **Delete Multi-Region Access Point** dialog box, enter the name of the AWS bucket that you want to delete.

   > **Note**
   >
   > Make sure to enter a valid bucket name. Otherwise, the **Delete** button will be disabled.

6. Choose **Delete** to confirm deletion of your Multi-Region Access Point.

### Using the AWS CLI

You can use the AWS CLI to delete a Multi-Region Access Point. This action does not delete the buckets associated with the Multi-Region Access Point, only the Multi-Region Access Point itself. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

## Configuring a Multi-Region Access Point for use with AWS PrivateLink

You can use Multi-Region Access Points to route Amazon S3 request traffic between AWS Regions. Each Multi-Region Access Point global endpoint routes Amazon S3 data request traffic from multiple sources without your having to build complex networking configurations with separate endpoints. These data-request traffic sources include:

- Traffic originating in a virtual private cloud (VPC)

- Traffic from on-premises data centers traveling over AWS PrivateLink
- Traffic from the public internet

If you establish an AWS PrivateLink connection to an S3 Multi-Region Access Point, you can route S3 requests into AWS, or across multiple AWS Regions, over a private connection by using a simple network architecture and configuration. When you use AWS PrivateLink, you don't need to configure a VPC peering connection.

**Topics**



## Configuring a Multi-Region Access Point for use with AWS PrivateLink

AWS PrivateLink provides you with private connectivity to Amazon S3 using private IP addresses in your virtual private cloud (VPC). You can provision one or more interface endpoints inside your VPC to connect to Amazon S3 Multi-Region Access Points.

You can create **com.amazonaws.s3-global.accesspoint** endpoints for Multi-Region Access Points through the AWS Management Console, AWS CLI, or AWS SDKs. To learn more about how to configure an interface endpoint for Multi-Region Access Point, see Interface VPC endpoints in the *VPC User Guide*.

To make requests to a Multi-Region Access Point via interface endpoints, follow these steps to configure the VPC and the Multi-Region Access Point.

**To configure a Multi-Region Access Point to use with AWS PrivateLink**

1. Create or have an appropriate VPC endpoint that can connect to Multi-Region Access Points. For more information about creating VPC endpoints, see Interface VPC endpoints in the *VPC User Guide*.

   > **Important**
   >
   > Make sure to create a **com.amazonaws.s3-global.accesspoint** endpoint. Other endpoint types cannot access Multi-Region Access Points.

After this VPC endpoint is created, all Multi-Region Access Point requests in the VPC route through this endpoint if you have private DNS enabled for the endpoint. This is enabled by default.

2. If the Multi-Region Access Point policy does not support connections from VPC endpoints, you will need to update it.
3. Verify that the individual bucket policies will allow access to the users of the Multi-Region Access Point.

Remember that Multi-Region Access Points work by routing requests to buckets, not by fulfilling requests themselves. This is important to remember because the originator of the request must have permissions to the Multi-Region Access Point and be allowed to access the individual buckets in the Multi-Region Access Point. Otherwise, the request might be routed to a bucket where the originator doesn't have permissions to fulfill the request. A Multi-Region Access Point and the buckets associated can be owned by the same or another AWS account. However, VPCs from different accounts can use a Multi-Region Access Point if the permissions are configured correctly.

Because of this, the VPC endpoint policy must allow access both to the Multi-Region Access Point and to each underlying bucket that you want to be able to fulfill requests. For example, suppose that you have a Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`. It is backed by buckets `amzn-s3-demo-bucket1` and `amzn-s3-demo-bucket2`, all owned by AWS account `123456789012`. In this case, the following VPC endpoint policy would allow `GetObject` requests from the VPC made to `mfzwi23gnjvgw.mrap` to be fulfilled by either backing bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Read-buckets-and-MRAP-VPCE-policy",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "arn:aws:s3:::amzn-s3-demo-bucket2/*",
            "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
```

```
        ]
    }]
}
```

As mentioned previously, you also must make sure that the Multi-Region Access Point policy is configured to support access through a VPC endpoint. You don't need to specify the VPC endpoint that is requesting access. The following sample policy would grant access to any requester trying to use the Multi-Region Access Point for the `GetObject` requests.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
    }]
}
```

And of course, the individual buckets would each need a policy to support access from requests submitted through VPC endpoint. The following example policy grants read access to any anonymous users, which would include requests made through the VPC endpoint.

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect":"Allow",
       "Principal": "*",
       "Action": "s3:GetObject",
       "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket1",
           "arn:aws:s3:::amzn-s3-demo-bucket2/*"]
    }]
}
```

For more information about editing a VPC endpoint policy, see Control access to services with VPC endpoints in the *VPC User Guide*.

### Removing access to a Multi-Region Access Point from a VPC endpoint

If you own a Multi-Region Access Point and want to remove access to it from an interface endpoint, you must supply a new access policy for the Multi-Region Access Point that prevents access for requests coming through VPC endpoints. However, if the buckets in your Multi-Region Access Point support requests through VPC endpoints, they will continue to support these requests. If you want to prevent that support, you must also update the policies for the buckets. Supplying a new access policy to the Multi-Region Access Point prevents access only to the Multi-Region Access Point, not to the underlying buckets.

> **Note**
>
> You can't delete an access policy for a Multi-Region Access Point. To remove access to a Multi-Region Access Point, you must provide a new access policy with the modified access that you want.

Instead of updating the access policy for the Multi-Region Access Point, you can update the bucket policies to prevent requests through VPC endpoints. In this case, users can still access the Multi-Region Access Point through the VPC endpoint. However, if the Multi-Region Access Point request is routed to a bucket where the bucket policy prevents access, the request will generate an error message.

## Making requests through a Multi-Region Access Point

Like other resources, Amazon S3 Multi-Region Access Points have Amazon Resource Names (ARNs). You can use these ARNs to direct requests to Multi-Region Access Points by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 API. You can also use these ARNs to identify Multi-Region Access Points in access control policies. A Multi-Region Access Point ARN doesn't include or disclose the name of the Multi-Region Access Point. For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

> **Note**
>
> The Multi-Region Access Point alias and ARN cannot be used interchangeably.

Multi-Region Access Point ARNs use the following format:

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

The following are a few examples of Multi-Region Access Point ARNs:

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap` represents the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, which is owned by AWS account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/*` represents all Multi-Region Access Points under the account `123456789012`. This ARN matches all Multi-Region Access Points for account `123456789012`, but doesn't match any Regional Amazon S3 Access Points because the ARN doesn't include an AWS Region. In contrast, the ARN `arn:aws:s3:us-west-2:123456789012:accesspoint/*` matches all Regional Amazon S3 Access Points in the Region `us-west-2` for the account `123456789012`, but doesn't match any Multi-Region Access Points.

ARNs for objects that are accessed through a Multi-Region Access Point use the following format:

`arn:aws:s3::account_id:accesspoint/MultiRegionAccessPoint_alias//key`

As with Multi-Region Access Point ARNs, the ARNs for objects that are accessed through Multi-Region Access Points don't include an AWS Region. Here are some examples.

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01` represents the `-01`, which is accessed through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, which is owned by account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//*` represents all objects that can be accessed through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, in account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/*` represents all objects that can be accessed under the `-01/finance/` for the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, in account `123456789012`.

## Multi-Region Access Point hostnames

You can access data in Amazon S3 through a Multi-Region Access Point by using the hostname of the Multi-Region Access Point. Requests can be directed to this hostname from the public internet.

If you have configured one or more internet gateways for the Multi-Region Access Point, requests can also be directed to this hostname from a virtual private cloud (VPC). For more information about creating VPC interface endpoints to use with Multi-Region Access Points, see Configuring a Multi-Region Access Point for use with AWS PrivateLink.

To make requests through a Multi-Region Access Point from a VPC by using a VPC endpoint, you can use AWS PrivateLink. When you're making requests to a Multi-Region Access Point by using AWS PrivateLink, you cannot directly use an endpoint-specific Regional DOMAIN NAME SYSTEM (DNS) name that ends with *region*`.vpce.amazonaws.com`. This hostname will not have a certificate associated with it, so it cannot be used directly. You can still use the public DOMAIN NAME SYSTEM (DNS) name of the VPC endpoint as a `CNAME` or `ALIAS` target. Alternatively, you can enable private DOMAIN NAME SYSTEM (DNS) on the endpoint and use the standard Multi-Region Access Point *MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com` DOMAIN NAME SYSTEM (DNS) name, as described in this section.

When you make requests to the API for Amazon S3 data operations (for example, `GetObject`) through a Multi-Region Access Point, the hostname for the request is as follows:

*MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com`

For example, to make a `GetObject` request through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, make a request to the hostname `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. The `s3-global` portion of the hostname indicates that this hostname is not for a specific Region.

Making requests through a Multi-Region Access Point is similar to making requests through a single-Region access point. However, it's important to be aware of the following differences:

- Multi-Region Access Point ARNs don't include an AWS Region. They follow the format `arn:aws:s3::`*account-id*`:accesspoint/`*MultiRegionAccessPoint_alias*.
- For requests made through API operations (these requests don't require the use of an ARN), Multi-Region Access Points use a different endpoint scheme. The scheme is *MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com`—for example, `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. Note the differences compared to a single-Region access point:
  - Multi-Region Access Point hostnames use their alias, not the Multi-Region Access Point name.
  - Multi-Region Access Point hostnames don't include the owner's AWS account ID.
  - Multi-Region Access Point hostnames don't include an AWS Region.

- Multi-Region Access Point hostnames include `s3-global.amazonaws.com` instead of `s3.amazonaws.com`.
- Multi-Region Access Point requests must be signed by using Signature Version 4A (SigV4A). When you use the AWS SDKs, the SDK automatically converts a SigV4 to SigV4A. Therefore, make sure that your AWS SDK supports SigV4A as the signing implementation that is used to sign the global AWS Region requests. For more information about SigV4A, see Signing AWS API requests in the *AWS General Reference*.

## Multi-Region Access Points and Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration is a feature that enables fast transfer of data to buckets. Transfer Acceleration is configured on the individual bucket level. For more information about Transfer Acceleration, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Multi-Region Access Points use a similar accelerated transfer mechanism as Transfer Acceleration for sending large objects over the AWS network. Because of this, you don't need to use Transfer Acceleration when sending requests through a Multi-Region Access Point. This increased transfer performance is automatically incorporated into the Multi-Region Access Point.

**Topics**

- Permissions
- Multi-Region Access Point restrictions and limitations
- Multi-Region Access Point request routing
- Amazon S3 Multi-Region Access Points failover controls
- Configuring replication for use with Multi-Region Access Points
- Using Multi-Region Access Points with supported API operations
- Monitoring and logging requests made through a Multi-Region Access Point to underlying resources

## Permissions

Amazon S3 Multi-Region Access Points can simplify data access for Amazon S3 buckets in multiple AWS Regions. Multi-Region Access Points are named global endpoints that you can use to perform Amazon S3 data-access object operations, such as `GetObject` and `PutObject`. Each Multi-Region

Access Point can have distinct permissions and network controls for any request that is made through the global endpoint.

Each Multi-Region Access Point can also enforce a customized access policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For a request to succeed, all of the following must permit the operation:

- The Multi-Region Access Point policy
- The underlying AWS Identity and Access Management (IAM) policy
- The underlying bucket policy (where the request is routed to)

You can configure any Multi-Region Access Point policy to accept requests only from specific IAM users or groups. For an example of how to do this, see Example 2 in the section called "Multi-Region Access Point policy examples". To restrict Amazon S3 data access to a private network, you can configure the Multi-Region Access Point policy to accept requests only from a virtual private cloud (VPC).

For example, suppose that you make a `GetObject` request through a Multi-Region Access Point by using a user called `AppDataReader` in your AWS account. To help ensure that the request won't be denied, the `AppDataReader` user must be granted the `s3:GetObject` permission by the Multi-Region Access Point and by each bucket underlying the Multi-Region Access Point. `AppDataReader` won't be able to retrieve data from any bucket that doesn't grant this permission.

> **Important**
>
> Delegating access control for a bucket to a Multi-Region Access Point policy doesn't change the bucket's behavior when the bucket is accessed directly through its bucket name or Amazon Resource Name (ARN). All operations made directly against the bucket will continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests made through that Multi-Region Access Point.

## Managing public access to a Multi-Region Access Point

Multi-Region Access Points support independent Block Public Access settings for each Multi-Region Access Point. When you create a Multi-Region Access Point, you can specify the Block Public Access settings that apply to that Multi-Region Access Point.

> **Note**
>
> Any Block Public Access settings that are enabled under **Block Public Access settings for this account** (in your own account) or **Block Public Settings for external buckets** still apply even if the independent Block Public Access settings for your Multi-Region Access Point are disabled.

For any request that is made through a Multi-Region Access Point, Amazon S3 evaluates the Block Public Access settings for:

- The Multi-Region Access Point
- The underlying buckets (including external buckets)
- The account that owns the Multi-Region Access Point
- The account that owns the underlying buckets (including external accounts)

If any of these settings indicate that the request should be blocked, Amazon S3 rejects the request. For more information about the Amazon S3 Block Public Access feature, see Blocking public access to your Amazon S3 storage.

> **Important**
>
> By default, all Block Public Access settings are enabled for Multi-Region Access Points. You must explicitly turn off any settings that you don't want to apply to a Multi-Region Access Point.
> You can't change the Block Public Access settings for a Multi-Region Access Point after it has been created.

## Viewing Block Public Access settings for a Multi-Region Access Point

**To view the Block Public Access settings for a Multi-Region Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.

3. Choose the name of the Multi-Region Access Point that you want to review.
4. Choose the **Permissions** tab.
5. Under **Block Public Access settings for this Multi-Region Access Point**, review the Block Public Access settings for your Multi-Region Access Point.

> **Note**
>
> You can't edit the Block Public Access settings after the Multi-Region Access Point is created. Therefore, if you're going to block public access, make sure that your applications work correctly without public access before you create a Multi-Region Access Point.

## Using a Multi-Region Access Point policy

The following example Multi-Region Access Point policy grants an IAM user access to list and download files from your Multi-Region Access Point. To use this example policy, replace the *`user input placeholders`* with your own information.

```
 {
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
         },
         "Action":[
            "s3:ListBucket",
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/
*"
         ]
      }
   ]
}
```

To associate your Multi-Region Access Point policy with the specified Multi-Region Access Point by using the AWS Command Line Interface (AWS CLI), use the following `put-multi-region-access-point-policy` command. To use this example command, replace the *`user input placeholders`* with your own information. Each Multi-Region Access Point can have only one policy, so a request made to the `put-multi-region-access-point-policy` action replaces any existing policy that is associated with the specified Multi-Region Access Point.

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "amzn-s3-demo-bucket-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

To query your results for the previous operation, use the following command:

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
--account-id 111122223333
--request-token-arn requestArn
```

To retrieve your Multi-Region Access Point policy, use the following command:

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=amzn-s3-demo-bucket-MultiRegionAccessPoint
```

## Editing the Multi-Region Access Point policy

The Multi-Region Access Point policy (written in JSON) provides storage access to the Amazon S3 buckets that are used with this Multi-Region Access Point. You can allow or deny specific principals to perform various actions on your Multi-Region Access Point. When a request is routed to a bucket through the Multi-Region Access Point, both the access policies for the Multi-Region Access Point and the bucket apply. The more restrictive access policy always takes precedence.

> **Note**
>
> If a bucket contains objects that are owned by other accounts, the Multi-Region Access Point policy doesn't apply to the objects that are owned by other AWS accounts.

After you apply a Multi-Region Access Point policy, the policy cannot be deleted. You can either edit the policy or create a new policy that overwrites the existing one.

**To edit the Multi-Region Access Point policy**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of the Multi-Region Access Point that you want to edit the policy for.
4. Choose the **Permissions** tab.
5. Scroll down to the **Multi-Region Access Point policy** section. Choose **Edit** to update the policy (in JSON).
6. The **Edit Multi-Region Access Point policy** page appears. You can either enter the policy directly into the text field, or you can choose **Add statement** to select policy elements from a dropdown list.

   > **Note**
   >
   > The console automatically displays the Multi-Region Access Point Amazon Resource Name (ARN), which you can use in the policy. For example Multi-Region Access Point policies, see the section called “Multi-Region Access Point policy examples”.

## Multi-Region Access Point policy examples

Amazon S3 Multi-Region Access Points support AWS Identity and Access Management (IAM) resource policies. You can use these policies to control the use of the Multi-Region Access Point by resource, user, or other conditions. For an application or user to be able to access objects through a Multi-Region Access Point, both the Multi-Region Access Point and the underlying bucket must allow the same access.

To allow the same access to both the Multi-Region Access Point and the underlying bucket, do one of the following:

- **(Recommended)** To simplify access controls when using an Amazon S3 Multi-Region Access Point, delegate access control for the Amazon S3 bucket to the Multi-Region Access Point. For an example of how to do this, see Example 1 in this section.
- Add the same permissions contained in the Multi-Region Access Point policy to the underlying bucket policy.

> **Important**
>
> Delegating access control for a bucket to a Multi-Region Access Point policy doesn't change the bucket's behavior when the bucket is accessed directly through its bucket name or Amazon Resource Name (ARN). All operations made directly against the bucket will continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests made through that Multi-Region Access Point.

### Example 1 – Delegating access to specific Multi-Region Access Points in your bucket policy (for the same account or cross-account)

The following example bucket policy grants full bucket access to a specific Multi-Region Access Point. This means that all access to this bucket is controlled by the policies that are attached to the Multi-Region Access Point. We recommend configuring your buckets this way for all use cases that don't require direct access to the bucket. You can use this bucket policy structure for Multi-Region Access Points in either the same account or in another account.

```
{
    "Version": "2012-10-17",
    "Statement" : [
```

```
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" : "MultiRegionAccessPoint_ARN" }
        }
    }]
}
```

> **Note**
>
> If there are multiple Multi-Region Access Points that you're granting access to, make sure to list each Multi-Region Access Point.

### Example 2 – Granting an account access to a Multi-Region Access Point in your Multi-Region Access Point policy

The following Multi-Region Access Point policy allows account *123456789012* permission to list and read the objects contained in the Multi-Region Access Point defined by the *MultiRegionAccessPoint_ARN*.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
          "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
       },
       "Action":[
          "s3:ListBucket",
          "s3:GetObject"
       ],
       "Resource":[
          "MultiRegionAccessPoint_ARN",
          "MultiRegionAccessPoint_ARN/object/*"
       ]
     }
  ]
```

```
}
```

**Example 3 – Multi-Region Access Point policy that allows bucket listing**

The following Multi-Region Access Point policy allows account *123456789012* permission to list the objects contained in the Multi-Region Access Point defined by the *MultiRegionAccessPoint_ARN*.

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
       "Effect": "Allow",
       "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action": "s3:ListBucket",
        "Resource": "MultiRegionAccessPoint_ARN"
       }
    ]
}
```

## Multi-Region Access Point restrictions and limitations

Multi-Region Access Points in Amazon S3 have the following restrictions and limitations.

- Multi-Region Access Point names:
  - Must be unique within a single AWS account
  - Must begin with a number or lowercase letter
  - Must be between 3 and 50 characters long
  - Can't begin or end with a hyphen (-)
  - Can't contain underscores (_), uppercase letters, or periods (.)
  - Can't be edited after they are created
- Multi-Region Access Point aliases are generated by Amazon S3 and can't be edited or reused.
- You cannot access data through a Multi-Region Access Point by using gateway endpoints. However, you can access data through a Multi-Region Access Point by using interface endpoints. To use AWS PrivateLink, you must create VPC endpoints. For more information, see Configuring a Multi-Region Access Point for use with AWS PrivateLink.

- To use Multi-Region Access Points with Amazon CloudFront, you must configure the Multi-Region Access Point as a `Custom Origin` distribution type. For more information about various origin types, see Using various origins with CloudFront distributions. For more information about using Multi-Region Access Points with Amazon CloudFront, see Building an active-active, proximity-based application across multiple Regions on the *AWS Storage Blog*.
- Multi-Region Access Point minimum requirements:
  - Transport Layer Security (TLS) v1.2
  - Signature Version 4 (SigV4A)

    Multi-Region Access Points support Signature Version 4A. This version of SigV4 allows requests to be signed for multiple AWS Regions. This feature is useful in API operations that might result in data access from one of several Regions. When using an AWS SDK, you supply your credentials, and the requests to Multi-Region Access Points will use Signature Version 4A without additional configuration. Make sure to check your AWS SDK compatibility with the SigV4A algorithm. For more information about SigV4A, see Signing AWS API requests in the *AWS General Reference*.

    > **Note**
    >
    > To use SigV4A with temporary security credentials—for example, when using AWS Identity and Access Management (IAM) roles—you can request the temporary credentials from a Regional AWS Security Token Service (AWS STS) endpoint. If you request temporary credentials from the global AWS STS endpoint (`sts.amazonaws.com`), then you must first set the Region compatibility of session tokens for the global endpoint to be valid in all AWS Regions. For more information, see Managing AWS STS in an AWS Region in the *IAM User Guide*.
- Multi-Region Access Points don't support anonymous requests.
- Multi-Region Access Point limitations:
  - IPv6 is not supported.
  - Amazon S3 on Outposts buckets are not supported.
  - Multi-Region Access Points supports copy operations using Multi-Region Access Points only as a destination when using the Multi-Region Access Point ARN.
  - The S3 Batch Operations feature is not supported.

- Certain AWS SDKs are not supported. To confirm which AWS SDKs are supported for Multi-Region Access Points, see Compatibility with AWS SDKs.
- The service quotas for Multi-Region Access Points are as follows:
  - There is a maximum of 100 Multi-Region Access Points per account.
  - There is a limit of 17 Regions for a single Multi-Region Access Point.
- After you create a Multi-Region Access Point, you can't add, modify, or remove buckets from the Multi-Region Access Point configuration. To change the buckets, you must delete the entire Multi-Region Access Point and create a new one. If a cross-account bucket in your Multi-Region Access Point is deleted, the only way to reconnect this bucket is to recreate the bucket, using the same name and Region in that account.
- Underlying buckets (in the same account) that are used in a Multi-Region Access Point can be deleted only after a Multi-Region Access Point is deleted.
- All control plane requests to create or maintain Multi-Region Access Points must be routed to the `US West (Oregon)` Region. For Multi-Region Access Point data plane requests, Regions don't need to be specified.
- For the Multi-Region Access Point failover control plane, requests must be routed to one of these five supported Regions:
  - `US East (N. Virginia)`
  - `US West (Oregon)`
  - `Asia Pacific (Sydney)`
  - `Asia Pacific (Tokyo)`
  - `Europe (Ireland)`
- Your Multi-Region Access Point only supports buckets in the following AWS Regions:
  - `US East (N. Virginia)`
  - `US East (Ohio)`
  - `US West (N. California)`
  - `US West (Oregon)`
  - `Asia Pacific (Mumbai)`
  - `Asia Pacific (Osaka)`
  - `Asia Pacific (Seoul)`
  - `Asia Pacific (Singapore)`
  - `Asia Pacific (Sydney)`

- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

## Multi-Region Access Point request routing

When you make a request through a Multi-Region Access Point, Amazon S3 determines which of the buckets that are associated with the Multi-Region Access Point is closest to you. Amazon S3 then directs the request to that bucket, regardless of the AWS Region it is located in.

After the Multi-Region Access Point routes the request to the closest-proximity bucket, Amazon S3 processes the request as if you made it directly to that bucket. Multi-Region Access Points aren't aware of the data contents of an Amazon S3 bucket. Therefore, the bucket that gets the request might not contain the requested data. To create consistent datasets in the Amazon S3 buckets that are associated with a Multi-Region Access Point, you can configure S3 Cross-Region Replication (CRR). Then any bucket can fulfill the request successfully.

Amazon S3 directs Multi-Region Access Point requests according to the following rules:

- Amazon S3 optimizes requests to be fulfilled according to proximity. It looks at the buckets supported by the Multi-Region Access Point and relays the request to the bucket that has the closest proximity.
- If the request specifies an existing resource (for example, `GetObject`), Amazon S3 does *not* consider the name of the object when fulfilling the request. This means that even if an object exists in one bucket in the Multi-Region Access Point, your request can be routed to a bucket that doesn't contain the object. This situation will result in a 404 error message being returned to the client.

  To avoid 404 errors, we recommend that you configure S3 Cross-Region Replication (CRR) for your buckets. Replication helps resolve the potential issue when the object that you want is in a bucket in the Multi-Region Access Point, but it's not located in the specific bucket that your

request was routed to. For more information about configuring replication, see Configuring replication for use with Multi-Region Access Points.

To ensure that your requests are fulfilled by using the specific objects that you want, we also recommend that you turn on bucket versioning and include version IDs in your requests. This approach helps ensure that you have the correct version of the object that you are looking for. Versioning-enabled buckets can also help you recover objects from accidental overwrite. For more information, see Using S3 Versioning in S3 buckets.

- If the request is to create a resource (for example, `PutObject` or `CreateMultipartUpload`), Amazon S3 fulfills the request by using the closest-proximity bucket. For example, consider a video company that wants to support video uploads from anywhere in the world. When a user makes a PUT request to the Multi-Region Access Point, the object is put into the bucket with the closest proximity. To then make that uploaded video available to others around the world for download with the lowest latency, you can use CRR with bidirectional (two-way) replication. Using CRR with two-way replication keeps the contents of all the buckets that are associated with the Multi-Region Access Point synchronized. For more information about using replication with Multi-Region Access Points, see Configuring replication for use with Multi-Region Access Points.

## Amazon S3 Multi-Region Access Points failover controls

With Amazon S3 Multi-Region Access Point failover controls, you can maintain business continuity during Regional traffic disruptions, while also giving your applications a multi-Region architecture to fulfill compliance and redundancy needs. If your Regional traffic gets disrupted, you can use Multi-Region Access Point failover controls to select which AWS Regions behind an Amazon S3 Multi-Region Access Point will process data-access and storage requests.

To support failover, you can set up your Multi-Region Access Point in an active-passive configuration, with traffic flowing to the active Region during normal conditions, and a passive Region on standby for failover.

For example, to perform failover to an AWS Region of your choice, you shift traffic from your primary (active) Region to your secondary (passive) Region. In an active-passive configuration like this, one bucket is active and accepting traffic, while the other bucket is passive and not accepting traffic. The passive bucket is used for disaster recovery. When you initiate failover, all traffic (such as GET or PUT requests) is directed to the bucket in the active state (in one Region) and away from the bucket in the passive state (in another Region).

If you have S3 Cross-Region Replication (CRR) enabled with two-way replication rules, you can keep your buckets synchronized during a failover. In addition, if you have CRR enabled in an active-active configuration, Amazon S3 Multi-Region Access Points can also fetch data from the bucket location of closest proximity, which improves application performance.

## AWS Region support

With Amazon S3 Multi-Region Access Points failover controls, your S3 buckets can be in any of the 17 Regions where Multi-Region Access Points are supported. You can initiate failover across any two Regions at one time.

> **Note**
>
> Although failover is initiated between only two Regions at one time, you can separately update the routing statuses for multiple Regions at the same time in your Multi-Region Access Point.

The following topics demonstrate how to use and manage Amazon S3 Multi-Region Access Point failover controls.

**Topics**

- Amazon S3 Multi-Region Access Points routing states
- Using Amazon S3 Multi-Region Access Point failover controls
- Amazon S3 Multi-Region Access Point failover controls errors

## Amazon S3 Multi-Region Access Points routing states

Your Amazon S3 Multi-Region Access Points failover configuration determines the routing status of the AWS Regions that are used with the Multi-Region Access Point. You can configure your Amazon S3 Multi-Region Access Point to be in an active-active state or active-passive state.

- **Active-active** – In an active-active configuration, all requests are automatically sent to the closest proximity AWS Region in your Multi-Region Access Point. After the Multi-Region Access Point has been configured to be in an active-active state, all Regions can receive traffic. If traffic disruption occurs in an active-active configuration, network traffic will automatically be redirected to one of the active Regions.

- **Active-passive** – In an active-passive configuration, the active Regions in your Multi-Region Access Point receive traffic and the passive ones do not. If you intend to use S3 failover controls to initiate failover in a disaster situation, set up your Multi-Region Access Points in an active-passive configuration while you're testing and performing disaster-recovery planning.

## Using Amazon S3 Multi-Region Access Point failover controls

This section explains how to manage and use your Amazon S3 Multi-Region Access Points failover controls by using the AWS Management Console.

There are two failover controls in the **Failover configuration** section on your Multi-Region Access Point details page in the AWS Management Console: **Edit routing status** and **Failover**. You can use these controls as follows:

- **Edit routing status** – You can manually edit the routing statuses of up to 17 AWS Regions in a single request for your Multi-Region Access Point by choosing **Edit routing status**. You can use **Edit routing status** for the following purposes:
  - To set or edit the routing statuses of one or more Regions in your Multi-Region Access Point
  - To create a failover configuration for your Multi-Region Access Point by configuring two Regions to be in an active-passive state
  - To manually fail over your Regions
  - To manually switch traffic between Regions
- **Failover** – When you initiate failover by choosing **Failover**, you are only updating the routing statuses of two Regions that are already configured to be in an active-passive state. During a failover that you initiated by choosing **Failover**, the routing statuses between the two Regions are automatically switched.

### Editing the routing status of the Regions in your Multi-Region Access Point

You can manually update the routing statuses of up to 17 AWS Regions in a single request for your Multi-Region Access Point by choosing **Edit routing status** in the **Failover configuration** section on your Multi-Region Access Point details page. However, when you initiate failover by choosing **Failover**, you are only updating the routing statuses of two Regions that are already configured to be in an active-passive state. During a failover that you initiated by choosing **Failover**, the routing statuses between the two Regions are automatically switched.

You can use **Edit routing status** (as described in the following procedure) for the following purposes:

- To set or edit the routing statuses of one or more Regions in your Multi-Region Access Point
- To create a failover configuration for your Multi-Region Access Point by configuring two Regions to be in an active-passive state
- To manually fail over your Regions
- To manually switch traffic between Regions

### Using the S3 console

#### To update the routing status of the Regions in your Multi-Region Access Point

1. Sign in to the AWS Management Console.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to update.
5. Choose the **Replication and failover** tab.
6. Select one or more Regions that you want to edit the routing status of.

   > **Note**
   >
   > To initiate failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point.

7. Choose **Edit routing status**.
8. In the dialog box that appears, select **Active** or **Passive** for the **Routing status** for each Region.

   An active state allows traffic to be routed to the Region. A passive state stops any traffic from being directed to the Region.

   If you are creating a failover configuration for your Multi-Region Access Point or initiating failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point.
9. Choose **Save routing status**. It takes about 2 minutes for traffic to be redirected.

After you submit the routing status of the AWS Regions for your Multi-Region Access Point, you can verify your routing status changes. To verify these changes, go to Amazon CloudWatch at https://console.aws.amazon.com/cloudwatch/ to monitor the shift of your Amazon S3 data-request traffic (for example, GET and PUT requests) between active and passive Regions. Any existing connections will not be terminated during failover. Existing connections will continue until they reach a success or failure status.

### Using the AWS CLI

> **Note**
>
> You can run Multi-Region Access Point AWS CLI routing commands against any of these five Regions:
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

The following example command updates your current Multi-Region Access Point route configuration. To update the active or passive status of a bucket, set the TrafficDialPercentage value to 100 for active and to 0 for passive. In this example, *DOC-EXAMPLE-BUCKET-1* is set to active, and *DOC-EXAMPLE-BUCKET-2* is set to passive. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=DOC-EXAMPLE-BUCKET-1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET-2,TrafficDialPercentage=0
```

The following example command gets your updated Multi-Region Access Point routing configuration. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

### Initiating failover

When you initiate failover by choosing **Failover** in the **Failover configuration** section on your Multi-Region Access Point details page, Amazon S3 request traffic automatically gets shifted to an alternate AWS Region. The failover process is completed within 2 minutes.

You can initiate a failover across any two AWS Regions at one time (of the 17 Regions where Multi-Region Access Points are supported). Failover events are then logged in AWS CloudTrail. Upon failover completion, you can monitor Amazon S3 traffic and any traffic routing updates to the new active Region in Amazon CloudWatch.

> **Important**
>
> To keep all metadata and objects in sync across buckets during data replication, we recommend that you create two-way replication rules and enable replica modification sync before configuring your failover controls.
> Two-way replication rules help ensure that when data is written to the Amazon S3 bucket that traffic fails over to, that data is then replicated back to the source bucket. Replica modification sync helps ensure that object metadata is also synchronized between buckets during two-way replication.
> For more information about configuring replication to support failover, see the section called “Bucket replication”.

#### To initiate failover between replicated buckets

1. Sign in to the AWS Management Console.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to use to initiate failover.
5. Choose the **Replication and failover** tab.
6. Scroll down to the **Failover configuration** section and select two AWS Regions.

> **Note**
>
> To initiate failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point. An active state allows traffic to be directed to a Region. A passive state stops any traffic from being directed to the Region.

7. Choose **Failover**.
8. In the dialog box, choose **Failover** again to initiate the failover process. During this process, the routing statuses of the two Regions are automatically switched. All new traffic is directed to the Region that becomes active, and traffic stops being directed to the Region that becomes passive. It takes about 2 minutes for traffic to be redirected.

   After you initiate the failover process, you can verify your traffic changes. To verify these changes, go to Amazon CloudWatch at https://console.aws.amazon.com/cloudwatch/ to monitor the shift of your Amazon S3 data-request traffic (for example, GET and PUT requests) between active and passive Regions. Any existing connections will not be terminated during failover. Existing connections will continue until they reach a success or failure status.

### Viewing your Amazon S3 Multi-Region Access Point routing controls

#### Using the S3 console

#### To view the routing controls for your Amazon S3 Multi-Region Access Point

1. Sign in to the AWS Management Console.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to review.
5. Choose the **Replication and failover** tab. This page displays the routing configuration details and summary for your Multi-Region Access Point, associated replication rules, and replication metrics. You can see the routing status of your Regions in the **Failover configuration** section.

### Using the AWS CLI

The following example AWS CLI command gets your current Multi-Region Access Point route configuration for the specified Region. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> This command can only be executed against these five Regions:
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

## Amazon S3 Multi-Region Access Point failover controls errors

When you update the failover configuration for your Multi-Region Access Point, you might encounter one of these errors:

- HTTP 400 Bad Request: This error can occur if you enter an invalid Multi-Region Access Point ARN while updating your failover configuration. You can confirm your Multi-Region Access Point ARN by reviewing your Multi-Region Access Point policy. To review or update your Multi-Region Access Point policy, see Editing the Multi-Region Access Point policy. This error can also occur if you use an empty string or a random string while updating your Amazon S3 Multi-Region Access Point failover controls. Make sure to use the Multi-Region Access Point ARN format:

  `arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

- HTTP 503 Slow Down: This error occurs if you send too many requests in a short period of time. Rejected requests will result in an error.

- HTTP 409 Conflict: This error occurs when two or more concurrent route configuration update requests are targeting a single Multi-Region Access Point. The first request succeeds, but any other requests fail with an error.
- HTTP 405 Method Not Allowed: This error occurs when you've selected a Multi-Region Access Point with only one AWS Region when initiating failover. You must select two Regions before you can initiate failover. Otherwise, an error is returned.

## Configuring replication for use with Multi-Region Access Points

When you make a request to a Multi-Region Access Point endpoint, Amazon S3 automatically routes the request to the bucket that is closest to you. Amazon S3 doesn't consider the contents of the request when making this decision. If you make a request to `GET` an object, your request might be routed to a bucket that doesn't have a copy of this object. If that happens, you receive an HTTP status code 404 (Not Found) error. For more information about Multi-Region Access Point request routing, see the section called “Request routing”.

If you want the Multi-Region Access Point to be able to retrieve the object regardless of which bucket receives the request, you must configure Amazon S3 Cross-Region Replication (CRR).

For example, consider a Multi-Region Access Point with three buckets:

- A bucket named `my-bucket-usw2` in the Region `us-west-2` that contains the object `my-image.jpg`
- A bucket named `my-bucket-aps1` in the Region `ap-south-1` that contains the object `my-image.jpg`
- A bucket named `my-bucket-euc1` in the Region `eu-central-1` that doesn't contain the object `my-image.jpg`

In this situation, if you make a `GetObject` request for the object `my-image.jpg`, the success of that request depends upon which bucket receives your request. Because Amazon S3 doesn't consider the contents of the request, it might route your `GetObject` request to the `my-bucket-euc1` bucket if that bucket responds of closest proximity. Even though your object is in a bucket in the Multi-Region Access Point, you will get an HTTP 404 Not Found error because the individual bucket that received your request didn't have the object.

Enabling Cross-Region Replication (CRR) helps avoid this result. With appropriate replication rules, the `my-image.jpg` object is copied over to the `my-bucket-euc1` bucket. Therefore, if Amazon S3 routes your request to that bucket, you can now retrieve the object.

Replication works as normal with buckets that are assigned to a Multi-Region Access Point. Amazon S3 doesn't perform any special replication handling with buckets that are in Multi-Region Access Points. For more information about configuring replication in your buckets, see Setting up live replication.

**Recommendations for using replication with Multi-Region Access Points**

For the best replication performance when working with Multi-Region Access Points, we recommend the following:

- Configure S3 Replication Time Control (S3 RTC). To replicate your data across different Regions within a predictable time frame, you can use S3 RTC. S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called "Using S3 Replication Time Control". There are additional charges for S3 RTC. For information, see Amazon S3 pricing.
- Use two-way (bidirectional) replication to support keeping buckets synchronized when buckets are updated through the Multi-Region Access Point. For more information, see the section called "Create two-way replication rules for your Multi-Region Access Point".
- Create cross-account Multi-Region Access Points to replicate data to buckets in separate AWS accounts. This approach provides account-level separation, so that data can be accessed from and replicated across different accounts in different Regions other than the source bucket. Setting up cross-account Multi-Region Access Points comes at no additional cost. If you're a bucket owner but don't own the Multi-Region Access Point, you pay only for data transfer and request costs. Multi-Region Access Point owners pay for data routing and internet-acceleration costs. For more information, see Amazon S3 pricing.
- Enable replica modification sync for each replication rule to also keep metadata changes to your objects in sync. For more information, see Enabling replica modification sync.
- Enable Amazon CloudWatch metrics to monitor replication events. CloudWatch metrics fees apply. For more information, see Amazon CloudWatch pricing.

**Topics**

- Create one-way replication rules for your Multi-Region Access Point

- Create two-way replication rules for your Multi-Region Access Point
- View the replication rules for your Multi-Region Access Point

## Create one-way replication rules for your Multi-Region Access Point

Replication rules enable automatic and asynchronous copying of objects across buckets. A one-way replication rule helps ensure that data is fully replicated from a source bucket in one AWS Region to a destination bucket in another Region. When one-way replication is set up, a replication rule from the source bucket (DOC-EXAMPLE-BUCKET-1) to the destination bucket (DOC-EXAMPLE-BUCKET-2) is created. Like all replication rules, you can apply the one-way replication rule to the entire Amazon S3 bucket or to a subset of objects that are filtered by a prefix or object tags.

> **Important**
>
> We recommend using one-way replication if your users will only be consuming the objects in your destination buckets. If your users will be uploading or modifying the objects in your destination buckets, use two-way replication to keep all of your buckets in sync. We also recommend two-way replication if you plan to use your Multi-Region Access Point for failover. To set up two-way replication, see the section called "Create two-way replication rules for your Multi-Region Access Point".

### To create a one-way replication rule for your Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of your Multi-Region Access Point.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section, and then choose **Create replication rules**. Make sure that you have sufficient permissions to create the replication rule, or versioning will be disabled.

> **Note**
>
> You can create replication rules only for buckets in your own account. To create replication rules for external buckets, the bucket owners must create the replication rules for those buckets.

6. On the **Create replication rules** page, choose the **Replicate objects from one or more source buckets to one or more destination buckets** template.

> **Important**
>
> When you create replication rules by using this template, they replace any existing replication rules that are already assigned to the bucket.
> To add to or modify any existing replication rules instead of replacing them, go to each bucket's **Management** tab in the console, and then edit the rules in the **Replication rules** section. You can also add to or modify existing replication rules by using the AWS CLI, SDKs, or REST API. For more information, see Replication configuration.

7. In the **Source and destination** section, under **Source buckets**, select one or more buckets that you want to replicate objects from. All buckets (source and destination) that are chosen for replication must have S3 Versioning enabled, and each bucket must reside in a different AWS Region. For more information about S3 Versioning, see Using versioning in Amazon S3 buckets.

   Under **Destination buckets**, select one or more buckets that you want to replicate objects to.

> **Note**
>
> Make sure that you have the required read and replicate permissions to establish replication, or you will encounter errors. For more information, see Creating an IAM role.

8. In the **Replication rule configuration** section, choose whether the replication rule will be **Enabled** or **Disabled** when it's created.

> **Note**
>
> You can't enter a name in the **Replication rule name** box. Replication rule names are generated based on your configuration when you create the replication rule.

9. In the **Scope** section, choose the appropriate scope for your replication.

   - To replicate the whole bucket, choose **Apply to all objects in the bucket**.
   - To replicate a subset of the objects in the bucket, choose **Limit the scope of this rule using one or more filters**.

     You can filter your objects by using a prefix, object tags, or a combination of both.

     - To limit replication to all objects that have names that begin with the same string (for example `pictures`), enter a prefix in the **Prefix** box.

       If you enter a prefix that is the name of a folder, you must use a delimiter such as a `/` (forward slash) to indicate its level of hierarchy (for example, `pictures/`). For more information about prefixes, see Organizing objects using prefixes.
     - To replicate all objects that have one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Categorizing your storage using tags.

10. Scroll down to the **Additional replication options** section, and select the replication options that you want to apply.

> **Note**
>
> We recommend that you apply the following options:
>
> - **Replication time control (RTC)** – To replicate your data across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3 RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called “Using S3 Replication Time Control”.
> - **Replication metrics and notifications** – Enable Amazon CloudWatch metrics to monitor replication events.

- **Delete marker replication** – Delete markers created by S3 delete operations will be replicated. Delete markers created by lifecycle rules are not replicated. For more information, see Replicating delete markers between buckets.

There are additional charges for S3 RTC and CloudWatch replication metrics and notifications. For more information, see Amazon S3 Pricing and Amazon CloudWatch pricing.

11. If you're writing a new replication rule that replaces an existing one, select **I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten**.
12. Choose **Create replication rules** to create and save your new one-way replication rule.

## Create two-way replication rules for your Multi-Region Access Point

Replication rules enable automatic and asynchronous copying of objects across buckets. A two-way replication rule (also known as a bidirectional replication rule) ensures that data is fully synchronized between two or more buckets in different AWS Regions. When two-way replication is set up, a replication rule from the source bucket (DOC-EXAMPLE-BUCKET-1) to the bucket containing the replicas (DOC-EXAMPLE-BUCKET-2) is created. Then, a second replication rule from the bucket containing the replicas (DOC-EXAMPLE-BUCKET-2) to the source bucket (DOC-EXAMPLE-BUCKET-1) is created.

Like all replication rules, you can apply the two-way replication rule to the entire Amazon S3 bucket or to a subset of objects filtered by a prefix or object tags. You can also keep metadata changes to your objects in sync by enabling replica modification sync for each replication rule. You can enable replica modification sync through the Amazon S3 console, the AWS CLI, the AWS SDKs, the Amazon S3 REST API, or AWS CloudFormation.

To monitor the replication progress of objects and object metadata in Amazon CloudWatch, enable S3 Replication metrics and notifications. For more information, see Monitoring progress with replication metrics and Amazon S3 event notifications.

**To create a two-way replication rule for your Multi-Region Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.

3. Choose the name of the Multi-Region Access Point that you want to update.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section, and then choose **Create replication rules**.
6. On the **Create replication rules** page, choose the **Replicate objects among all specified buckets** template. The **Replicate objects among all specified buckets** template sets up two-way replication (with failover capabilities) for your buckets.

   > **Important**
   >
   > When you create replication rules by using this template, they replace any existing replication rules that are already assigned to the bucket.
   > To add to or modify any existing replication rules instead of replacing them, go to each bucket's **Management** tab in the console, and then edit the rules in the **Replication rules** section. You can also add to or modify existing replication rules by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information, see Replication configuration.

7. In the **Buckets** section, select at least two buckets that you want to replicate objects from. All buckets chosen for replication must have S3 Versioning enabled, and each bucket must reside in a different AWS Region. For more information about S3 Versioning, see Using versioning in Amazon S3 buckets.

   > **Note**
   >
   > Make sure that you have the required read and replicate permissions to establish replication, or you will encounter errors. For more information, see Creating an IAM role.

8. In the **Replication rule configuration** section, choose whether the replication rule will be **Enabled** or **Disabled** when it's created.

   > **Note**
   >
   > You can't enter a name in the **Replication rule name** box. Replication rule names are generated based on your configuration when you create the replication rule.

9. In the **Scope** section, choose the appropriate scope for your replication.

- To replicate the whole bucket, choose **Apply to all objects in the bucket**.
- To replicate a subset of the objects in the bucket, choose **Limit the scope of this rule using one or more filters**.

  You can filter your objects by using a prefix, object tags, or a combination of both.

  - To limit replication to all objects that have names that begin with the same string (for example `pictures`), enter a prefix in the **Prefix** box.

    If you enter a prefix that is the name of a folder, you must use a / (forward slash) as the last character (for example, `pictures/`).
  - To replicate all objects that have one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Categorizing your storage using tags.

10. Scroll down to the **Additional replication options** section, and select the replication options that you want to apply.

> **Note**
>
> We recommend that you apply the following options, especially if you intend to configure your Multi-Region Access Point to support failover:
>
> - **Replication time control (RTC)** – To replicate your data across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3 RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called "Using S3 Replication Time Control".
> - **Replication metrics and notifications** – Enable Amazon CloudWatch metrics to monitor replication events.
> - **Delete marker replication** – Delete markers created by S3 delete operations will be replicated. Delete markers created by lifecycle rules are not replicated. For more information, see Replicating delete markers between buckets.
> - **Replica modification sync** – Enable replica modification sync for each replication rule to also keep metadata changes to your objects in sync. For more information, see Enabling replica modification sync.

There are additional charges for S3 RTC and CloudWatch replication metrics and notifications. For more information, see Amazon S3 Pricing and Amazon CloudWatch pricing.

11. If you're writing a new replication rule that replaces an existing one, select **I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten**.
12. Choose **Create replication rules** to create and save your new two-way replication rules.

### View the replication rules for your Multi-Region Access Point

With Multi-Region Access Points, you can either set up one-way replication rules or two-way (bidirectional) replication rules. For information about how to manage your replication rules, see Managing replication rules by using the Amazon S3 console.

**To view the replication rules for your Multi-Region Access Point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of your Multi-Region Access Point.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section. This section lists all of the replication rules that have been created for your Multi-Region Access Point.

> **Note**
>
> If you've added a bucket from another account to this Multi-Region Access Point, you must have the `s3:GetBucketReplication` permission from the bucket owner to view the replication rules for that bucket.

## Using Multi-Region Access Points with supported API operations

Amazon S3 provides a set of operations to manage Multi-Region Access Points. Amazon S3 processes some of these operations synchronously and some asynchronously. When you invoke

an asynchronous operation, Amazon S3 first synchronously authorizes the requested operation. If authorization is successful, Amazon S3 returns a token that you can use to track the progress and results of the requested operation.

> **Note**
>
> Requests that are made through the Amazon S3 console are always synchronous. The console waits until the request is completed before allowing you to submit another request.

You can view the current status and results of asynchronous operations by using the console, or you can use `DescribeMultiRegionAccessPointOperation` in the AWS CLI, AWS SDKs, or REST API. Amazon S3 provides a tracking token in the response to an asynchronous operation. You include that tracking token as an argument to `DescribeMultiRegionAccessPointOperation`. When you include the tracking token, Amazon S3 then returns the current status and results of the specified operation, including any errors or relevant resource information. Amazon S3 performs `DescribeMultiRegionAccessPointOperation` operations synchronously.

All control plane requests to create or maintain Multi-Region Access Points must be routed to the `US West (Oregon)` Region. For Multi-Region Access Point data plane requests, Regions don't need to be specified. For the Multi-Region Access Point failover control plane, the request must be routed to one of the five supported Regions. For more information about Multi-Region Access Point supported Regions, see Multi-Region Access Point restrictions and limitations.

In addition, you must grant the `s3:ListAllMyBuckets` permission to the user, role, or other AWS Identity and Access Management (IAM) entity that makes a request to manage a Multi-Region Access Point.

The following examples demonstrate how to use Multi-Region Access Points with compatible operations in Amazon S3.

**Topics**

- Add an object to a bucket in your Multi-Region Access Point
- Retrieve objects from your Multi-Region Access Point
- List objects that are stored in a bucket underlying your Multi-Region Access Point
- Use a presigned URL with Multi-Region Access Points
- Use a bucket that's configured with Requester Pays with Multi-Region Access Points

## Multi-Region Access Point compatibility with AWS services and AWS SDKs

To use a Multi-Region Access Point with applications that require an Amazon S3 bucket name, use the Amazon Resource Name (ARN) of the Multi-Region Access Point when making requests by using an AWS SDK. To check which AWS SDKs are compatible with Multi-Region Access Points, see Compatibility with AWS SDKs.

## Multi-Region Access Point compatibility with S3 operations

You can use use the following Amazon S3 data plane API operations to perform actions on objects in buckets that are associated with your Multi-Region Access Point. The following S3 operations can accept Multi-Region Access Point ARNs:

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject

- PutObjectAcl
- PutObjectLegalHold
- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> Multi-Region Access Points supports copy operations using Multi-Region Access Points only as a destination when using the Multi-Region Access Point ARN.

You can use the following Amazon S3 control plane operations to create and manage your Multi-Region Access Points:

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

## View your Multi-Region Access Point routing configuration

AWS CLI

The following example command retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-multi-region-access-point-routes
```

```
--region eu-west-1
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

The following SDK for Java code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

The following SDK for JavaScript code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
```

```
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

The following SDK for Python code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

## Update your underlying Amazon S3 bucket policy

To grant proper access, you must also update the underlying Amazon S3 bucket policy. The following examples delegate access control to the Multi-Region Access Point policy. After you delegate access control to the Multi-Region Access Point policy, the bucket policy is no longer used for access control when requests are made through the Multi-Region Access Point.

Here's an example bucket policy that delegates access control to the Multi-Region Access Point policy. To use this example bucket policy, replace the *user input placeholders* with your own information. To apply this policy through the AWS CLI `put-bucket-policy` command, as shown in the next example, save the policy in a file, for example, `policy.json`.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": { "AWS": "*" },
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::111122223333/*", "arn:aws:s3:::amzn-s3-demo-bucket"],
    "Condition": {
      "StringEquals": {
```

```
        "s3:DataAccessPointAccount": "444455556666"
      }
    }
  }
}
```

The following `put-bucket-policy` example command associates the updated S3 bucket policy with your S3 bucket:

```
aws s3api put-bucket-policy
  --bucket amzn-s3-demo-bucket
  --policy file:///tmp/policy.json
```

## Update a Multi-Region Access Point route configuration

The following example command updates the Multi-Region Access Point route configuration. Multi-Region Access Point route commands can be run against the following five Regions:

- `ap-southeast-2`
- `ap-northeast-1`
- `us-east-1`
- `us-west-2`
- `eu-west-1`

In a Multi-Region Access Point routing configuration, you can set buckets to an active or passive routing status. Active buckets receive traffic, whereas passive buckets do not. You can set a bucket's routing status by setting the `TrafficDialPercentage` value for the bucket to `100` for active or `0` for passive.

AWS CLI

The following example command updates your Multi-Region Access Point routing configuration. In this example, *amzn-s3-demo-bucket1* is set to active status and *amzn-s3-demo-bucket2* is set to passive. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
```

```
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
                Bucket=amzn-s3-demo-bucket2,TrafficDialPercentage=0
```

SDK for Java

The following SDK for Java code updates your Multi-Region Access Point routing configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

The following SDK for JavaScript code updates your Multi-Region Access Point route configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
const REGION = 'ap-southeast-2'
```

```
const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'amzn-s3-demo-bucket1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

The following SDK for Python code updates your Multi-Region Access Point route configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
            'Bucket': DOC-EXAMPLE-BUCKET,
```

```
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

## Add an object to a bucket in your Multi-Region Access Point

To add an object to the bucket that's associated with the Multi-Region Access Point, you can use the PutObject operation. To keep all buckets in the Multi-Region Access Point in sync, enable Cross-Region Replication.

> **Note**
>
> To use this operation, you must have the `s3:PutObject` permission for the Multi-Region Access Point. For more information about Multi-Region Access Point permission requirements, see Permissions.

AWS CLI

The following example data plane request uploads *example.txt* to the specified Multi-Region Access Point. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt',
    Body='Hello S3!'
)
```

## Retrieve objects from your Multi-Region Access Point

To retrieve objects from the Multi-Region Access Point, you can use the GetObject operation.

> **Note**
>
> To use this API operation, you must have the `s3:GetObject` permission for the Multi-Region Access Point. For more information about Multi-Region Access Point permissions requirements, see Permissions.

AWS CLI

The following example data plane request retrieves *example.txt* from the specified Multi-Region Access Point and downloads it as *downloaded_example.txt*. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})

async function getObjectExample() {
    const command = new GetObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

## List objects that are stored in a bucket underlying your Multi-Region Access Point

To return a list of objects that are stored in a bucket underlying your Multi-Region Access Point, use the ListObjectsV2 operation. In the following example command, all objects for the specified Multi-Region Access Point are listed by using the ARN for the Multi-Region Access Point. In this case, the Multi-Region Access Point ARN is:

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> To use this API operation, you must have the `s3:ListBucket` permission for the Multi-Region Access Point and the underlying bucket. For more information about Multi-Region Access Point permissions requirements, see Permissions.

AWS CLI

The following example data plane request lists the objects in the bucket that underlies the Multi-Region Access Point that's specified by the ARN. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();
```

```
String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

## Use a presigned URL with Multi-Region Access Points

You can use a presigned URL to generate a URL that allows others to access your Amazon S3 buckets through an Amazon S3 Multi-Region Access Point. When you create a presigned URL, you associate it with a specific object action, such as an S3 upload (`PutObject`) or an S3 download

(`GetObject`). You can share the presigned URL, and anyone with access to it can perform the action embedded in the URL as if they were the original signing user.

Presigned URLs have an expiration date. When the expiration time is reached, the URL will no longer work.

Before you use S3 Multi-Region Access Points with presigned URLs, check the AWS SDK compatibility with the SigV4A algorithm. Verify that your SDK version supports SigV4A as the signing implementation that is used to sign the global AWS Region requests. For more information about using presigned URLs with Amazon S3, see Sharing objects by using presigned URLs.

The following examples show how you can use Multi-Region Access Points with presigned URLs. To use these examples, replace the *`user input placeholders`* with your own information.

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python

```
import logging
import boto3
from botocore.exceptions import ClientError

s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
```

```
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

**Note**

To use SigV4A with temporary security credentials—for example, when using IAM roles—make sure that you request the temporary credentials from a Regional endpoint in AWS Security Token Service (AWS STS), instead of a global endpoint. If you use the global endpoint for AWS STS (`sts.amazonaws.com`), AWS STS will generate temporary credentials from a global endpoint, which isn't supported by Sig4A. As a result, you'll get an error. To resolve this issue, use any of the listed Regional endpoints for AWS STS.

## Use a bucket that's configured with Requester Pays with Multi-Region Access Points

If an S3 bucket that is associated with your Multi-Region Access Points is configured to use Requester Pays, the requester will pay for the bucket request, the download, and any Multi-Region Access Points related costs. For more information, see Amazon S3 pricing.

Here's an example of a data plane request to a Multi-Region Access Point that is connected to a Requester Pays bucket.

AWS CLI

To download objects from a Multi-Region Access Point that is connected to a Requester Pays bucket, you must specify `--request-payer requester` as part of your get-object request. You must also specify the name of the file in the bucket and the location where the downloaded file should be stored.

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

To download objects from a Multi-Region Access Point that is connected to a Requester Pays bucket, you must specify the `RequestPayer.REQUESTER` as part of your `GetObject` request. You must also specify the name of the file in the bucket, as well as the location where it should be stored.

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

## Monitoring and logging requests made through a Multi-Region Access Point to underlying resources

Amazon S3 logs requests made through Multi-Region Access Points and requests made to the API operations that manage them, such as `CreateMultiRegionAccessPoint` and `GetMultiRegionAccessPointPolicy`. Requests made to Amazon S3 through a Multi-Region Access Point appear in your Amazon S3 server access logs and AWS CloudTrail logs with the Multi-Region Access Point hostname. An access point's hostname takes the form *`MRAP_alias`*`.accesspoint.s3-global.amazonaws.com`. For example, suppose that you have the following bucket and Multi-Region Access Point configuration:

- A bucket named `my-bucket-usw2` in the Region `us-west-2` that contains the object `my-image.jpg`.
- A bucket named `my-bucket-aps1` in the Region `ap-south-1` that contains the object `my-image.jpg`.
- A bucket named `my-bucket-euc1` in the Region `eu-central-1` that doesn't contain an object named `my-image.jpg`.
- A Multi-Region Access Point named `my-mrap` with the alias `mfzwi23gnjvgw.mrap` that is configured to fulfill requests from all three buckets.
- Your AWS account ID is `123456789012`.

A request made to retrieve `my-image.jpg` directly through any of the buckets appears in your logs with a hostname of *`bucket_name`*`.s3.`*`Region`*`.amazonaws.com`.

If you make the request through the Multi-Region Access Point instead, Amazon S3 first determines which of the buckets in the different Regions is closest. After Amazon S3 determines which bucket to use to fulfill the request, it sends the request to that bucket and logs the operation by using the Multi-Region Access Point hostname. In this example, if Amazon S3 relays the request to `my-bucket-aps1`, your logs will reflect a successful `GET` request for `my-image.jpg` from `my-bucket-aps1`, using a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`.

**Important**

Multi-Region Access Points aren't aware of the data contents of the underlying buckets. Therefore, the bucket that gets the request might not contain the requested data. For example, if Amazon S3 determines that the `my-bucket-euc1` bucket is the closest, your logs will reflect a failed `GET` request for `my-image.jpg` from `my-bucket-euc1`, using a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. If the request was routed to `my-bucket-usw2` instead, your logs would indicate a successful `GET` request.

For more information about Amazon S3 server access logs, see Logging requests with server access logging. For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

## Monitoring and logging requests made to Multi-Region Access Point management API operations

Amazon S3 provides several API operations to manage Multi-Region Access Points, such as `CreateMultiRegionAccessPoint` and `GetMultiRegionAccessPointPolicy`. When you make requests to these API operations by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API, Amazon S3 processes these requests asynchronously. Provided that you have the appropriate permissions for the request, Amazon S3 returns a token for these requests. You can use this token with `DescribeAsyncOperation` to help you to view the status of ongoing asynchronous operations. Amazon S3 processes `DescribeAsyncOperation` requests synchronously. To view the status of asynchronous requests, you can use the Amazon S3 console, AWS CLI, SDKs, or REST API.

> **Note**
>
> The console displays only the status of asynchronous requests that were made within the previous 14 days. To view the status of older requests, use the AWS CLI, SDKs, or REST API.

Asynchronous management operations can be in one of several states:

NEW

Amazon S3 has received the request and is preparing to perform the operation.

IN_PROGRESS

Amazon S3 is currently performing the operation.

SUCCESS

The operation succeeded. The response includes relevant information, such as the Multi-Region Access Point alias for a `CreateMultiRegionAccessPoint` request.

FAILED

The operation failed. The response includes an error message that indicates the reason for the request failure.

## Using AWS CloudTrail with Multi-Region Access Points

You can use AWS CloudTrail to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure. With Multi-Region Access Points and CloudTrail logging, you can identify the following:

- Who or what took which action
- Which resources were acted upon
- When the event occurred
- Other details regarding the event

You can use this logging information to help you analyze and respond to activity that occurred through your Multi-Region Access Points.

### How to set up AWS CloudTrail for Multi-Region Access Points

To enable CloudTrail logging for any operations related to creating or maintaining Multi-Region Access Points, you must configure CloudTrail logging to record the events in the US West (Oregon) Region. You must set up your logging configuration this way regardless of which Region you are in when making the request, or which Regions the Multi-Region Access Point supports. All requests to create or maintain a Multi-Region Access Point are routed through the US West (Oregon) Region. We recommend that you either add this Region to an existing trail or create a new trail that contains this Region and all the Regions associated with the Multi-Region Access Point.

Amazon S3 logs all requests made through a Multi-Region Access Point and requests made to the API operations that manage access points, such as `CreateMultiRegionAccessPoint` and `GetMultiRegionAccessPointPolicy`. When you log these requests through a Multi-Region Access Point, they appear in your AWS CloudTrail logs with the hostname of the Multi-Region Access Point. For example, if you make requests to a bucket through a Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, the entries in the CloudTrail log will have a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`.

Multi-Region Access Points route requests to the closest bucketBecause of this behavior, when you are looking at the CloudTrail logs for a Multi-Region Access Point, you will see requests being made to the underlying buckets. Some of those requests might be direct requests to the bucket that are not routed through the Multi-Region Access Point. Keep this fact in mind when reviewing traffic. When a bucket is in a Multi-Region Access Point, requests can still be made to that bucket directly without going through the Multi-Region Access Point.

There are asynchronous events involved with creating and managing Multi-Region Access Points. Asynchronous requests don't have completion events in the CloudTrail log. For more information about asynchronous requests, see Monitoring and logging requests made to Multi-Region Access Point management API operations.

For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

# Amazon S3 security

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

**Security of the cloud**

AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon S3, see AWS Services in Scope by Compliance Program.

**Security in the cloud**

Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations. For Amazon S3, your responsibility includes the following areas:

- Managing your data, including object ownership and encryption.
- Classifying your assets.
- Managing access to your data using IAM roles and other service configurations to apply the appropriate permissions.
- Enabling detective controls such as AWS CloudTrail or Amazon GuardDuty for Amazon S3.

This documentation will help you understand how to apply the shared responsibility model when using Amazon S3. The following topics show you how to configure Amazon S3 to meet your security and compliance objectives. You'll also learn how to use other AWS services that can help you monitor and secure your Amazon S3 resources.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



# Data protection in Amazon S3

Amazon S3 provides a highly durable storage infrastructure designed for mission-critical and primary data storage. S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive redundantly store objects on multiple devices across a minimum of three Availability Zones in an AWS Region. An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Availability Zones are physically separated by a meaningful distance, many kilometers, from any other Availability Zone, although all are within 100 km (60 miles) of each other. The S3 One Zone-IA storage class stores data redundantly across multiple devices within a single Availability Zone. These services are designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy, and they also regularly verify the integrity of your data using checksums.

Amazon S3 standard storage offers the following features:

- Backed with the Amazon S3 Service Level Agreement.
- Designed to provide 99.999999999% durability and 99.99% availability of objects over a given year.
- S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive are all designed to sustain data in the event of the loss of an entire Amazon S3 Availability Zone.

Amazon S3 further protects your data using versioning. You can use versioning to preserve, retrieve, and restore every version of every object that is stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. By default, requests retrieve the most recently written version. You can retrieve older versions of an object by specifying a version of the object in a request.

Apart from S3 Versioning, you can also use Amazon S3 Object Lock and S3 Replication to protect your data. For more information, see Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management, so that each user is given only the permissions necessary to fulfill their job duties.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

The following security best practices also address data protection in Amazon S3:

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3
- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

# Protecting data with encryption

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

Data protection refers to protecting data while it's in transit (as it travels to and from Amazon S3) and at rest (while it is stored on disks in Amazon S3 data centers). You can protect data in transit by using Secure Socket Layer/Transport Layer Security (SSL/TLS) or client-side encryption. For protecting data at rest in Amazon S3, you have the following options:

- **Server-side encryption** – Amazon S3 encrypts your objects before saving them on disks in AWS data centers and then decrypts the objects when you download them.

  All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

  If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

  For more information about each option for server-side encryption, see Protecting data with server-side encryption.

To configure server-side encryption, see:

- Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)
- Specifying server-side encryption with AWS KMS (SSE-KMS)
- the section called "Specifying DSSE-KMS"
- Specifying server-side encryption with customer-provided keys (SSE-C)

- **Client-side encryption** – You encrypt your data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, encryption keys, and related tools.

  To configure client-side encryption, see Protecting data by using client-side encryption.

To see which percentage of your storage bytes are encrypted, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

For more information about server-side encryption and client-side encryption, review the following topics.

**Topics**

- Protecting data with server-side encryption
- Protecting data by using client-side encryption

## Protecting data with server-side encryption

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API

response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in AWS data centers and decrypts it for you when you access it. As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects. For example, if you share your objects by using a presigned URL, that URL works the same way for both encrypted and unencrypted objects. Additionally, when you list objects in your bucket, the list API operations return a list of all objects, regardless of whether they are encrypted.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

**Note**

You can't apply different types of server-side encryption to the same object simultaneously.

If you need to encrypt your existing objects, use S3 Batch Operations and S3 Inventory. For more information, see Encrypting objects with Amazon S3 Batch Operations and Performing large-scale batch operations on Amazon S3 objects.

You have four mutually exclusive options for server-side encryption, depending on how you choose to manage the encryption keys and the number of encryption layers that you want to apply.

**Server-side encryption with Amazon S3 managed keys (SSE-S3)**

All Amazon S3 buckets have encryption configured by default. The default option for server-side encryption is with Amazon S3 managed keys (SSE-S3). Each object is encrypted with a unique key. As an additional safeguard, SSE-S3 encrypts the key itself with a root key that it regularly rotates. SSE-S3 uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

**Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)**

Server-side encryption with AWS KMS keys (SSE-KMS) is provided through an integration of the AWS KMS service with Amazon S3. With AWS KMS, you have more control over your keys. For example, you can view separate keys, edit control policies, and follow the keys in AWS CloudTrail. Additionally, you can create and manage customer managed keys or use AWS managed keys that are unique to you, your service, and your Region. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).

**Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**

Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) is similar to SSE-KMS, but DSSE-KMS applies two individual layers of object-level encryption instead of one layer. Because both layers of encryption are applied to an object on the server side, you can use a wide range of AWS services and tools to analyze data in S3 while using an encryption method that can satisfy your compliance requirements. For more information, see Using dual-layer server-side encryption with AWS KMS keys (DSSE-KMS).

**Server-side encryption with customer-provided keys (SSE-C)**

With server-side encryption with customer-provided keys (SSE-C), you manage the encryption keys, and Amazon S3 manages the encryption as it writes to disks and the decryption when you access your objects. For more information, see Using server-side encryption with customer-provided keys (SSE-C).

## Amazon S3 now automatically encrypts all new objects

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. SSE-S3, which uses 256-bit Advanced Encryption Standard (AES-256), is automatically applied to all new buckets and to any existing S3 bucket that doesn't already have

default encryption configured. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs.

The following sections answer questions about this update.

**Does Amazon S3 change the default encryption settings for my existing buckets that already have default encryption configured?**

No. There are no changes to the default encryption configuration for an existing bucket that already has SSE-S3 or server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) configured. For more information about how to set the default encryption behavior for buckets, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3 and SSE-KMS encryption settings, see Protecting data with server-side encryption.

**Is default encryption enabled on my existing buckets that don't have default encryption configured?**

Yes. Amazon S3 now configures default encryption on all existing unencrypted buckets to apply server-side encryption with S3 managed keys (SSE-S3) as the base level of encryption for new objects uploaded to these buckets. Objects that are already in an existing unencrypted bucket won't be automatically encrypted.

**How can I view the default encryption status of new object uploads?**

Currently, you can view the default encryption status of new object uploads in AWS CloudTrail logs, S3 Inventory, and S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs.

- To view your CloudTrail events, see Viewing CloudTrail events in the CloudTrail console in the *AWS CloudTrail User Guide*. CloudTrail logs provide API tracking for PUT and POST requests to Amazon S3. When default encryption is being used to encrypt objects in your buckets, the CloudTrail logs for PUT and POST API requests will include the following field as the name-value pair: `"SSEApplied":"Default_SSE_S3"`.
- To view the automatic encryption status of new object uploads in S3 Inventory, configure an S3 Inventory report to include the **Encryption** metadata field, and then see the encryption status of each new object in the report. For more information, see Setting up Amazon S3 Inventory.

- To view the automatic encryption status for new object uploads in S3 Storage Lens, configure an S3 Storage Lens dashboard and see the **Encrypted bytes** and **Encrypted object count** metrics in the **Data protection** category of the dashboard. For more information, see Creating an Amazon S3 Storage Lens dashboard and Viewing S3 Storage Lens metrics on the dashboards.
- To view the automatic bucket-level encryption status in the Amazon S3 console, check the **Default encryption** of your Amazon S3 buckets in the Amazon S3 console. For more information, see Configuring default encryption.
- To view the automatic encryption status as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs, check the response header `x-amz-server-side-encryption` when you use object action APIs, such as PutObject and GetObject.

**What do I have to do to take advantage of this change?**

You are not required to make any changes to your existing applications. Because default encryption is enabled for all of your buckets, all new objects uploaded to Amazon S3 are automatically encrypted.

**Can I disable encryption for the new objects being written to my bucket?**

No. SSE-S3 is the new base level of encryption that's applied to all the new objects being uploaded to your bucket. You can no longer disable encryption for new object uploads.

**Will my charges be affected?**

No. Default encryption with SSE-S3 is available at no additional cost. You will be billed for storage, requests, and other S3 features, as usual. For pricing, see Amazon S3 pricing.

**Will Amazon S3 encrypt my existing objects that are unencrypted?**

No. Beginning on January 5, 2023, Amazon S3 only automatically encrypts new object uploads. To encrypt existing objects, you can use S3 Batch Operations to create encrypted copies of your objects. These encrypted copies will retain the existing object data and name and will be encrypted by using the encryption keys that you specify. For more details, see Encrypting objects with Amazon S3 Batch Operations in the *AWS Storage Blog*.

**I did not enable encryption for my buckets before this release. Do I need to change the way that I access objects?**

No. Default encryption with SSE-S3 automatically encrypts your data as it's written to Amazon S3 and decrypts it for you when you access it. There is no change in the way that you access objects that are automatically encrypted.

**Do I need to change the way that I access my client-side encrypted objects?**

No. All client-side encrypted objects that are encrypted before being uploaded into Amazon S3 arrive as encrypted ciphertext objects within Amazon S3. These objects will now have an additional layer of SSE-S3 encryption. Your workloads that use client-side encrypted objects will not require any changes to your client services or authorization settings.

> **Note**
>
> HashiCorp Terraform users that aren't using an updated version of the AWS Provider might see an unexpected drift after creating new S3 buckets with no customer defined encryption configuration. To avoid this drift, update your Terraform AWS Provider version to one of the following versions: any 4.x release, 3.76.1, or 2.70.4.

## Using server-side encryption with Amazon S3 managed keys (SSE-S3)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

All new object uploads to Amazon S3 buckets are encrypted by default with server-side encryption with Amazon S3 managed keys (SSE-S3).

Server-side encryption protects data at rest. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a key that it rotates regularly. Amazon

S3 server-side encryption uses 256-bit Advanced Encryption Standard Galois/Counter Mode (AES-GCM) to encrypt all uploaded objects.

There are no additional fees for using server-side encryption with Amazon S3 managed keys (SSE-S3). However, requests to configure the default encryption feature incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing.

If you require your data uploads to be encrypted using only Amazon S3 managed keys, you can use the following bucket policy. For example, the following bucket policy denies permissions to upload an object unless the request includes the `x-amz-server-side-encryption` header to request server-side encryption:

```
{
  "Version": "2012-10-17",
  "Id": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    }
  ]
}
```

**Note**

Server-side encryption encrypts only the object data, not the object metadata.

### API support for server-side encryption

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3.

To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

To configure server-side encryption by using the object creation REST APIs, you must provide the `x-amz-server-side-encryption` request header. For information about the REST APIs, see Using the REST API.

The following Amazon S3 APIs support this header:

- **PUT operations** – Specify the request header when uploading data using the PUT API. For more information, see PUT Object.
- **Initiate Multipart Upload** – Specify the header in the initiate request when uploading large objects using the multipart upload API operation. For more information, see Initiate Multipart Upload.
- **COPY operations** – When you copy an object, you have both a source object and a target object. For more information, see PUT Object - Copy.

> **Note**
>
> When using a POST operation to upload an object, instead of providing the request header, you provide the same information in the form fields. For more information, see POST Object.

The AWS SDKs also provide wrapper APIs that you can use to request server-side encryption. You can also use the AWS Management Console to upload objects and request server-side encryption.

For more general information, see AWS KMS concepts in the *AWS Key Management Service Developer Guide*.

**Topics**

- Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)

### Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

You can specify SSE-S3 by using the S3 console, REST APIs, AWS SDKs, and AWS Command Line Interface (AWS CLI). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

#### Using the S3 console

This topic describes how to set or change the type of encryption an object by using the AWS Management Console. When you copy an object by using the console, Amazon S3 copies the object as is. That means that if the source object is encrypted, the target object is also encrypted. You can use the console to add or change encryption for an object.

**Note**

- If you change an object's encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).
- If you change the encryption type for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing the encryption type for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, the encryption type for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.

**To change encryption for an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object.
4. In the **Objects** list, choose the name of the object that you want to add or change encryption for.

   The object's details page appears, with several sections that display the properties for your object.
5. Choose the **Properties** tab.
6. Scroll down to the **Server-side encryption settings** section, and then choose **Edit**.
7. Under **Encryption settings**, choose **Use bucket default encryption settings** or **Override bucket default encryption settings**.
8. If you chose **Override bucket settings for default encryption**, configure the following encryption settings.

   - Under **Encryption type**, choose **Amazon S3 managed keys (SSE-S3)**. SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt

each object. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

9. Choose **Save changes**.

> **Note**
>
> This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

### Using the REST API

At the time of object creation—that is, when you are uploading a new object or making a copy of an existing object—you can specify if you want Amazon S3 to encrypt your data with Amazon S3 managed keys (SSE-S3) by adding the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `AES256`, which Amazon S3 supports. Amazon S3 confirms that your object is stored with SSE-S3 by returning the response header `x-amz-server-side-encryption`.

The following REST upload API operations accept the `x-amz-server-side-encryption` request header.

- PUT Object
- PUT Object - Copy
- POST Object
- Initiate Multipart Upload

When uploading large objects by using the multipart upload API operation, you can specify server-side encryption by adding the `x-amz-server-side-encryption` header to the Initiate Multipart Upload request. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object is not encrypted unless you explicitly request server-side encryption.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored using SSE-S3.

- PUT Object

- PUT Object - Copy
- POST Object
- Initiate Multipart Upload
- Upload Part
- Upload Part - Copy
- Complete Multipart Upload
- Get Object
- Head Object

> **Note**
>
> Do not send encryption request headers for GET requests and HEAD requests if your object uses SSE-S3, or you'll get an HTTP status code 400 (Bad Request) error.

### Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use server-side encryption with Amazon S3 managed encryption keys (SSE-S3). This section provides examples of using the AWS SDKs in multiple languages. For information about other SDKs, go to Sample Code and Libraries.

Java

When you use the AWS SDK for Java to upload an object, you can use SSE-S3 to encrypt it. To request server-side encryption, use the `ObjectMetadata` property of the `PutObjectRequest` to set the `x-amz-server-side-encryption` request header. When you call the `putObject()` method of the `AmazonS3Client`, Amazon S3 encrypts and saves the data.

You can also request SSE-S3 encryption when uploading objects with the multipart upload API operation:

- When using the high-level multipart upload API operation, you use the `TransferManager` methods to apply server-side encryption to objects as you upload them. You can use any of the upload methods that take `ObjectMetadata` as a parameter. For more information, see Uploading an object using multipart upload.
- When using the low-level multipart upload API operation, you specify server-side encryption when you initiate the multipart upload. You add the `ObjectMetadata` property by calling

the `InitiateMultipartUploadRequest.setObjectMetadata()` method. For more information, see Using the AWS SDKs (low-level API).

You can't directly change the encryption state of an object (encrypting an unencrypted object or decrypting an encrypted object). To change an object's encryption state, you make a copy of the object, specifying the desired encryption state for the copy, and then delete the original object. Amazon S3 encrypts the copied object only if you explicitly request server-side encryption. To request encryption of the copied object through the Java API, use the `ObjectMetadata` property to specify server-side encryption in the `CopyObjectRequest`.

**Example Example**

The following example shows how to set server-side encryption by using the AWS SDK for Java. It shows how to perform the following tasks:

- Upload a new object by using SSE-S3.
- Change an object's encryption state (in this example, encrypting a previously unencrypted object) by making a copy of the object.
- Check the encryption state of the object.

For more information about server-side encryption, see Using the REST API. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {

    public static void main(String[] args) {
```

```
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
***";
        String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to
be encrypted by copying ***";
        String copiedObjectKeyName = "*** Key name for the encrypted copy of the
unencrypted object ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Upload an object and encrypt it with SSE.
            uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

            // Upload a new unencrypted object, then change its encryption state
            // to encrypted by making a copy.
            changeSSEEncryptionStatusByCopying(s3Client,
                    bucketName,
                    keyNameToCopyAndEncrypt,
                    copiedObjectKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String
bucketName, String keyName) {
        String objectContent = "Test object encrypted with SSE";
        byte[] objectBytes = objectContent.getBytes();

        // Specify server-side encryption.
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setContentLength(objectBytes.length);
```

```
objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
       PutObjectRequest putRequest = new PutObjectRequest(bucketName,
               keyName,
               new ByteArrayInputStream(objectBytes),
               objectMetadata);

       // Upload the object and check its encryption status.
       PutObjectResult putResult = s3Client.putObject(putRequest);
       System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
       printEncryptionStatus(putResult);
   }

   private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
           String bucketName,
           String sourceKey,
           String destKey) {
       // Upload a new, unencrypted object.
       PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey,
"Object example to encrypt by copying");
       System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
       printEncryptionStatus(putResult);

       // Make a copy of the object and use server-side encryption when storing the
       // copy.
       CopyObjectRequest request = new CopyObjectRequest(bucketName,
               sourceKey,
               bucketName,
               destKey);
       ObjectMetadata objectMetadata = new ObjectMetadata();

objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
       request.setNewObjectMetadata(objectMetadata);

       // Perform the copy operation and display the copy's encryption status.
       CopyObjectResult response = s3Client.copyObject(request);
       System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
       printEncryptionStatus(response);

       // Delete the original, unencrypted object, leaving only the encrypted copy
in
       // Amazon S3.
       s3Client.deleteObject(bucketName, sourceKey);
       System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
```

```
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET

When you upload an object, you can direct Amazon S3 to encrypt it. To change the encryption state of an existing object, you make a copy of the object and delete the source object. By default, the copy operation encrypts the target only if you explicitly request server-side encryption of the target object. To specify SSE-S3 in the `CopyObjectRequest`, add the following:

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

For a working sample of how to copy an object, see Using the AWS SDKs.

The following example uploads an object. In the request, the example directs Amazon S3 to encrypt the object. The example then retrieves object metadata and verifies the encryption method that was used. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine("Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

This topic shows how to use classes from version 3 of the AWS SDK for PHP to add SSE-S3 to objects that you upload to Amazon S3. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

To upload an object to Amazon S3, use the Aws\S3\S3Client::putObject() method. To add the `x-amz-server-side-encryption` request header to your upload request, specify the `ServerSideEncryption` parameter with the value `AES256`, as shown in the following code example. For information about server-side encryption requests, see Using the REST API.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
]);
```

In response, Amazon S3 returns the `x-amz-server-side-encryption` header with the value of the encryption algorithm that was used to encrypt your object's data.

When you upload large objects by using the multipart upload API operation, you can specify SSE-S3 for the objects that you are uploading, as follows:

- When you're using the low-level multipart upload API operation, specify server-side encryption when you call the Aws\S3\S3Client::createMultipartUpload() method. To add the `x-amz-server-side-encryption` request header to your request, specify the `array` parameter's `ServerSideEncryption` key with the value `AES256`. For more information about the low-level multipart upload API operation, see Using the AWS SDKs (low-level API).
- When you're using the high-level multipart upload API operation, specify server-side encryption by using the `ServerSideEncryption` parameter of the CreateMultipartUpload API operation. For an example of using the `setOption()` method with the high-level multipart upload API operation, see Uploading an object using multipart upload.

To determine the encryption state of an existing object, retrieve the object metadata by calling the Aws\S3\S3Client::headObject() method as shown in the following PHP code example.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

To change the encryption state of an existing object, make a copy of the object by using the Aws\S3\S3Client::copyObject() method and delete the source object. By default, `copyObject()` does not encrypt the target unless you explicitly request server-side encryption of the destination object by using the `ServerSideEncryption` parameter with the value `AES256`. The following PHP code example makes a copy of an object and adds server-side encryption to the copied object.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Copy an object and add server-side encryption.
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "$sourceBucket/$sourceKeyname",
    'ServerSideEncryption' => 'AES256',
]);
```

For more information, see the following topics:

- AWS SDK for PHP for Amazon S3 Aws\S3\S3Client Class
- AWS SDK for PHP Documentation

### Ruby

When using the AWS SDK for Ruby to upload an object, you can specify that the object be stored encrypted at rest with SSE-S3. When you read the object back, it is automatically decrypted.

The following AWS SDK for Ruby Version 3 example demonstrates how to specify that a file uploaded to Amazon S3 be encrypted at rest.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

The following code example demonstrates how to determine the encryption state of an existing object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

If server-side encryption is not used for the object that is stored in Amazon S3, the method returns `null`.

To change the encryption state of an existing object, make a copy of the object and delete the source object. By default, the copy methods do not encrypt the target unless you explicitly request server-side encryption. You can request the encryption of the target object by specifying the `server_side_encryption` value in the option's hash argument, as shown in the following Ruby code example. The code example demonstrates how to copy an object and encrypt the copy with SSE-S3.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
```

```
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

### Using the AWS CLI

To specify SSE-S3 when you upload an object by using the AWS CLI, use the following example.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

For more information, see put-object in the *AWS CLI reference*. To specify SSE-S3 when you copy an object by using the AWS CLI, see copy-object.

### Using AWS CloudFormation

For examples of setting up encryption using AWS CloudFormation, see Create a bucket with default encryption and the Create a bucket by using AWS KMS server-side encryption with an S3 Bucket Key example in the `AWS::S3::Bucket ServerSideEncryptionRule` topic in the *AWS CloudFormation User Guide*.

## Using server-side encryption with AWS KMS keys (SSE-KMS)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all

new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

Server-side encryption is the encryption of data at its destination by the application or service that receives it.

Amazon S3 automatically enables server-side encryption with Amazon S3 managed keys (SSE-S3) for new object uploads.

Unless you specify otherwise, buckets use SSE-S3 by default to encrypt objects. However, you can choose to configure buckets to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) instead. For more information, see Specifying server-side encryption with AWS KMS (SSE-KMS).

AWS KMS is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Amazon S3 uses server-side encryption with AWS KMS (SSE-KMS) to encrypt your S3 object data. Also, when SSE-KMS is requested for the object, the S3 checksum (as part of the object's metadata) is stored in encrypted form. For more information about checksum, see Checking object integrity.

If you use KMS keys, you can use AWS KMS through the AWS Management Console or the AWS KMS API to do the following:

- Centrally create, view, edit, monitor, enable or disable, rotate, and schedule deletion of KMS keys.
- Define the policies that control how and by whom KMS keys can be used.
- Audit their usage to prove that they are being used correctly. Auditing is supported by the AWS KMS API, but not by the AWS KMSAWS Management Console.

The security controls in AWS KMS can help you meet encryption-related compliance requirements. You can use these KMS keys to protect your data in Amazon S3 buckets. When you use SSE-KMS encryption with an S3 bucket, the AWS KMS keys must be in the same Region as the bucket.

There are additional charges for using AWS KMS keys. For more information, see AWS KMS key concepts in the *AWS Key Management Service Developer Guide* and AWS KMS pricing.

### Permissions

To upload an object encrypted with an AWS KMS key to Amazon S3, you need `kms:GenerateDataKey` permissions on the key. To download an object encrypted with an AWS KMS key, you need `kms:Decrypt` permissions. For information about the AWS KMS permissions that are required for multipart uploads, see Multipart upload API and permissions.

> **Important**
>
> Carefully review the permissions that are granted in your KMS key policies. Always restrict customer-managed KMS key policy permissions only to the IAM principals and AWS services that must access the relevant AWS KMS key action. For more information, see Key policies in AWS KMS.

### Topics

- AWS KMS keys
- Amazon S3 Bucket Keys
- Requiring server-side encryption
- Encryption context
- Sending requests for AWS KMS encrypted objects
- Specifying server-side encryption with AWS KMS (SSE-KMS)
- Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

### AWS KMS keys

When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed key, or you can specify a customer managed key that you have already created. AWS KMS supports *envelope encryption*. S3 uses the AWS KMS features for *envelope encryption* to further protect your data. Envelope encryption is the practice of encrypting your plaintext data with a data key, and then encrypting that data key with a KMS key. For more information about envelope encryption, see Envelope encryption in the *AWS Key Management Service Developer Guide*.

If you don't specify a customer managed key, Amazon S3 automatically creates an AWS managed key in your AWS account the first time that you add an object encrypted with SSE-KMS to a bucket. By default, Amazon S3 uses this KMS key for SSE-KMS.

> **Note**
>
> Objects encrypted using SSE-KMS with AWS managed keys can't be shared cross-account. If you need to share SSE-KMS data cross-account, you must use a customer managed key from AWS KMS.

If you want to use a customer managed key for SSE-KMS, create a symmetric encryption customer managed key before you configure SSE-KMS. Then, when you configure SSE-KMS for your bucket, specify the existing customer managed key. For more information about symmetric encryption key, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

Creating a customer managed key gives you more flexibility and control. For example, you can create, rotate, and disable customer managed keys. You can also define access controls and audit the customer managed key that you use to protect your data. For more information about customer managed and AWS managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> When you use server-side encryption with a customer managed key that's stored in an external key store, unlike standard KMS keys, you are responsible for ensuring the availability and durability of your key material. For more information about external key stores and how they shift the shared responsibility model, see External key stores in the *AWS Key Management Service Developer Guide*.

#### Using SSE-KMS encryption for cross-account operations

When using encryption for cross-account operations, be aware of the following:

- If an AWS KMS key Amazon Resource Name (ARN) or alias is not provided at request time or through the bucket's default encryption configuration, the AWS managed key (`aws/s3`) is used.

- If you're uploading or accessing S3 objects by using AWS Identity and Access Management (IAM) principals that are in the same AWS account as your KMS key, you can use the AWS managed key (`aws/s3`).
- If you want to grant cross-account access to your S3 objects, use a customer managed key. You can configure the policy of a customer managed key to allow access from another account.
- If you're specifying a customer managed KMS key, we recommend using a fully qualified KMS key ARN. If you use a KMS key alias instead, AWS KMS resolves the key within the requester's account. This behavior can result in data that's encrypted with a KMS key that belongs to the requester, and not the bucket owner.
- You must specify a key that you (the requester) have been granted `Encrypt` permission to. For more information, see Allow key users to use a KMS key for cryptographic operations in the *AWS Key Management Service Developer Guide*.

For more information about when to use customer managed keys and AWS managed KMS keys, see Should I use an AWS managed key or a customer managed key to encrypt my objects in Amazon S3?

**SSE-KMS encryption workflow**

If you choose to encrypt your data using an AWS managed key or a customer managed key, AWS KMS and Amazon S3 perform the following envelope encryption actions:

1. Amazon S3 requests a plaintext data key and a copy of the key encrypted under the specified KMS key.
2. AWS KMS generates a data key, encrypts it under the KMS key, and sends both the plaintext data key and the encrypted data key to Amazon S3.
3. Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.
4. Amazon S3 stores the encrypted data key as metadata with the encrypted data.

When you request that your data be decrypted, Amazon S3 and AWS KMS perform the following actions:

1. Amazon S3 sends the encrypted data key to AWS KMS in a `Decrypt` request.
2. AWS KMS decrypts the encrypted data key by using the same KMS key and returns the plaintext data key to Amazon S3.

3. Amazon S3 decrypts the encrypted data, using the plaintext data key, and removes the plaintext data key from memory as soon as possible.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

### Auditing SSE-KMS encryption

To identify requests that specify SSE-KMS, you can use the **All SSE-KMS requests** and **% all SSE-KMS requests** metrics in Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can also use the SSE-KMS enabled bucket count and % SSE-KMS enabled buckets to understand the count of buckets that (SSE-KMS) for default bucket encryption. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

To audit the usage of your AWS KMS keys for your SSE-KMS encrypted data, you can use AWS CloudTrail logs. You can get insight into your cryptographic operations, such as GenerateDataKey and Decrypt. CloudTrail supports numerous attribute values for filtering your search, including event name, user name, and event source.

### Amazon S3 Bucket Keys

When you configure server-side encryption using AWS KMS (SSE-KMS), you can configure your buckets to use S3 Bucket Keys for SSE-KMS. Using a bucket-level key for SSE-KMS can reduce your AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS.

When you configure a bucket to use an S3 Bucket Key for SSE-KMS on new objects, AWS KMS generates a bucket-level key that is used to create unique data keys for objects in the bucket. This S3 Bucket Key is used for a time-limited period within Amazon S3, further reducing the need for Amazon S3 to make requests to AWS KMS to complete encryption operations. For more information about using S3 Bucket Keys, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

### Requiring server-side encryption

To require server-side encryption of all objects in a particular Amazon S3 bucket, you can use a bucket policy. For example, the following bucket policy denies the upload object (`s3:PutObject`) permission to everyone if the request does not include an `x-amz-server-side-encryption-aws-kms-key-id` header that requests server-side encryption with SSE-KMS.

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

To require that a particular AWS KMS key be used to encrypt the objects in a bucket, you can use the `s3:x-amz-server-side-encryption-aws-kms-key-id` condition key. To specify the KMS key, you must use a key Amazon Resource Name (ARN) that is in the `arn:aws:kms:region:acct-id:key/key-id` format. AWS Identity and Access Management does not validate if the string for `s3:x-amz-server-side-encryption-aws-kms-key-id` exists.

> **Note**
>
> When you upload an object, you can specify the KMS key by using the `x-amz-server-side-encryption-aws-kms-key-id` header or rely on your default bucket encryption configuration. If your PutObject request specifies `aws:kms` in the `x-amz-server-side-encryption` header, but does not specify the `x-amz-server-side-encryption-aws-kms-key-id` header, then Amazon S3 assumes that you want to use the AWS managed

key. Regardless, the AWS KMS key ID that Amazon S3 uses for object encryption must match the AWS KMS key ID in the policy, otherwise Amazon S3 denies the request.

For a complete list of Amazon S3 specific condition keys, see Condition keys for Amazon S3 in the *Service Authorization Reference*.

**Encryption context**

An *encryption context* is a set of key-value pairs that contains additional contextual information about the data. The encryption context is not encrypted. When an encryption context is specified for an encryption operation, Amazon S3 must specify the same encryption context for the decryption operation. Otherwise, the decryption fails. AWS KMS uses the encryption context as additional authenticated data (AAD) to support authenticated encryption. For more information about the encryption context, see Encryption context in the *AWS Key Management Service Developer Guide*.

By default, Amazon S3 uses the object or bucket Amazon Resource Name (ARN) as the encryption context pair:

- **If you use SSE-KMS without enabling an S3 Bucket Key**, the object ARN is used as the encryption context.

```
arn:aws:s3:::object_ARN
```

- **If you use SSE-KMS and enable an S3 Bucket Key**, the bucket ARN is used as the encryption context. For more information about S3 Bucket Keys, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

```
arn:aws:s3:::bucket_ARN
```

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header in an s3:PutObject request. However, because the encryption context is not encrypted, make sure it does not include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context. When it processes your PUT request, Amazon S3 appends the default encryption context of `aws:s3:arn` to the one that you provide.

You can use the encryption context to identify and categorize your cryptographic operations. You can also use the default encryption context ARN value to track relevant requests in AWS CloudTrail by viewing which Amazon S3 ARN was used with which encryption key.

In the `requestParameters` field of a CloudTrail log file, the encryption context looks similar to the following one.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1/file_name"
}
```

When you use SSE-KMS with the optional S3 Bucket Keys feature, the encryption context value is the ARN of the bucket.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
}
```

**Sending requests for AWS KMS encrypted objects**

> **Important**
>
> All GET and PUT requests for AWS KMS encrypted objects must be made using Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Requests must also be signed using valid credentials, such as AWS Signature Version 4 (or AWS Signature Version 2).

AWS Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see Authenticating Requests (AWS Signature Version 4) and Signature Version 4 signing process.

> **Important**
>
> If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests. Otherwise, you'll get an HTTP 400 Bad Request error.

**Topics**



## Specifying server-side encryption with AWS KMS (SSE-KMS)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

You can apply encryption when you are either uploading a new object or copying an existing object.

You can specify SSE-KMS by using the Amazon S3 console, REST API operations, AWS SDKs, and the AWS Command Line Interface (AWS CLI). For more information, see the following topics.

**Note**

You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in *AWS Key Management Service Developer Guide*.

**Note**

If you want to use a KMS key that's owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

### Using the S3 console

This topic describes how to set or change the type of encryption of an object to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) by using the Amazon S3 console.

**Note**

- If you change an object's encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).
- If you change the encryption type for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing the encryption type for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, the encryption type for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.

**To add or change encryption for an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object.
4. In the **Objects** list, choose the name of the object that you want to add or change encryption for.

   The object's details page appears, with several sections that display the properties for your object.
5. Choose the **Properties** tab.
6. Scroll down to the **Server-side encryption settings** section and choose **Edit**.

   The **Edit server-side encryption** page opens.
7. Under **Server-side encryption**, for **Encryption settings**, choose **Override default encryption bucket settings**.
8. Under **Encryption type**, choose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**.

   > **Important**
   >
   > If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.
9. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.

- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN.
> Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

10. Choose **Save changes**.

> **Note**
>
> This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

### Using the REST API

When you create an object—that is, when you upload a new object or copy an existing object—you can specify the use of server-side encryption with AWS KMS keys (SSE-KMS) to encrypt your data. To do this, add the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `aws:kms`. Amazon S3 confirms that your object is stored using SSE-KMS by returning the response header `x-amz-server-side-encryption`.

If you specify the `x-amz-server-side-encryption` header with a value of `aws:kms`, you can also use the following request headers:

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`

- `x-amz-server-side-encryption-bucket-key-enabled`

**Topics**



### Amazon S3 REST API operations that support SSE-KMS

The following REST API operations accept the `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, and `x-amz-server-side-encryption-context` request headers.

- PutObject – When you upload data by using the PUT API operation, you can specify these request headers.
- CopyObject – When you copy an object, you have both a source object and a target object. When you pass SSE-KMS headers with the `CopyObject` operation, they're applied only to the target object. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object isn't encrypted unless you explicitly request server-side encryption.
- POST Object – When you use a POST operation to upload an object, instead of the request headers, you provide the same information in the form fields.
- CreateMultipartUpload – When you upload large objects by using the multipart upload API operation, you can specify these headers. You specify these headers in the `CreateMultipartUpload` request.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored by using server-side encryption.

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload

- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - All GET and PUT requests for an object protected by AWS KMS fail if you don't make these requests by using Secure Sockets Layer (SSL), Transport Layer Security (TLS), or Signature Version 4.
> - If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests, or you'll get an HTTP 400 BadRequest error.

#### Encryption context (`x-amz-server-side-encryption-context`)

If you specify `x-amz-server-side-encryption:aws:kms`, the Amazon S3 API supports an encryption context with the `x-amz-server-side-encryption-context` header. An encryption context is a set of key-value pairs that contain additional contextual information about the data.

Amazon S3 automatically uses the object or bucket Amazon Resource Name (ARN) as the encryption context pair. If you use SSE-KMS without enabling an S3 Bucket Key, you use the object ARN as your encryption context; for example, `arn:aws:s3:::`*`object_ARN`*. However, if you use SSE-KMS and enable an S3 Bucket Key, you use the bucket ARN for your encryption context; for example, `arn:aws:s3:::`*`bucket_ARN`*.

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header. However, because the encryption context isn't encrypted, make sure it doesn't include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context.

For information about the encryption context in Amazon S3, see Encryption context. For general information about the encryption context, see AWS Key Management Service Concepts - Encryption context in the *AWS Key Management Service Developer Guide*.

### AWS KMS key ID (x-amz-server-side-encryption-aws-kms-key-id)

You can use the `x-amz-server-side-encryption-aws-kms-key-id` header to specify the ID of the customer managed key that's used to protect the data. If you specify the `x-amz-server-side-encryption:aws:kms` header but don't provide the `x-amz-server-side-encryption-aws-kms-key-id` header, Amazon S3 uses the AWS managed key (`aws/s3`) to protect the data. If you want to use a customer managed key, you must provide the `x-amz-server-side-encryption-aws-kms-key-id` header of the customer managed key.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

### S3 Bucket Keys (x-amz-server-side-encryption-aws-bucket-key-enabled)

You can use the `x-amz-server-side-encryption-aws-bucket-key-enabled` request header to enable or disable an S3 Bucket Key at the object level. S3 Bucket Keys reduce your AWS KMS request costs by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

If you specify the `x-amz-server-side-encryption:aws:kms` header but don't provide the `x-amz-server-side-encryption-aws-bucket-key-enabled` header, your object uses the S3 Bucket Key settings for the destination bucket to encrypt your object. For more information, see Configuring an S3 Bucket Key at the object level .

### Using the AWS CLI

To use the following example AWS CLI commands, replace the *`user input placeholders`* with your own information.

When you upload a new object or copy an existing object, you can specify the use of server-side encryption with AWS KMS keys to encrypt your data. To do this, add the `--server-side-encryption aws:kms` header to the request. Use the `--ssekms-key-id` *`example-key-id`* to add your customer managed AWS KMS key that you created. If you specify `--server-side-encryption aws:kms`, but don't provide an AWS KMS key ID, Amazon S3 will use an AWS managed key.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --
server-side-encryption aws:kms --ssekms-key-id example-key-id --ssekms-encryption-
context example-encryption-context --body filepath
```

You can enable or disable S3 Bucket Keys on your `put-object` or `copy-object` operations by adding `--bucket-key-enabled` or `--no-bucket-key-enabled`. S3 Bucket Keys can reduce your AWS KMS request costs by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with S3 Bucket Keys.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-
side-encryption aws:kms --bucket-key-enabled --body filepath
```

You can copy an object from a source bucket to a new bucket and specify SSE-KMS encryption.

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/example-object-key --
bucket amzn-s3-demo-bucket2 --key example-object-key --server-side-encryption aws:kms
 --sse-kms-key-id example-key-id --ssekms-encryption-context example-encryption-context
```

#### Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use AWS KMS keys for server-side encryption. The following examples show how to use SSE-KMS with the AWS SDKs for Java and .NET. For information about other SDKs, see Sample code and libraries on the AWS Developer Center.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

#### `CopyObject` operation

When copying objects, you add the same request properties (`ServerSideEncryptionMethod` and `ServerSideEncryptionKeyManagementServiceKeyId`) to request Amazon S3 to use an AWS KMS key. For more information about copying objects, see Copying, moving, and renaming objects.

### PUT operation

Java

When uploading an object by using the AWS SDK for Java, you can request Amazon S3 to use an AWS KMS key by adding the `SSEAwsKeyManagementParams` property as shown in the following request:

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

In this case, Amazon S3 uses the AWS managed key (`aws/s3`). For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS). You can optionally create a symmetric encryption KMS key and specify that in the request, as shown in the following example:

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects.
- For multipart uploads that use the high-level or low-level multipart upload API operations, see Uploading an object using multipart upload.

.NET

When uploading an object by using the AWS SDK for .NET, you can request Amazon S3 to use an AWS KMS key by adding the `ServerSideEncryptionMethod` property as shown in the following request:

```
PutObjectRequest putRequest = new PutObjectRequest
 {
```

```
    BucketName = amzn-s3-demo-bucket,
    Key = keyName,
    // other properties
    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
};
```

In this case, Amazon S3 uses the AWS managed key. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS). You can optionally create your own symmetric encryption customer managed key and specify that in the request, as shown in the following example:

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
  BucketName = amzn-s3-demo-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects.
- For multipart uploads that use the high-level or low-level multipart upload API operations, see Uploading an object using multipart upload.

### Presigned URLs

Java

When creating a presigned URL for an object that's encrypted with an AWS KMS key, you must explicitly specify Signature Version 4, as shown in the following example:

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
```

```
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
...
```

For a code example, see Sharing objects with presigned URLs.

.NET

When creating a presigned URL for an object that's encrypted with an AWS KMS key, you must explicitly specify Signature Version 4, as shown in the following example:

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

For a code example, see Sharing objects with presigned URLs.

### Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

Amazon S3 Bucket Keys reduce the cost of Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). Using a bucket-level key for SSE-KMS can reduce AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS. With a few clicks in the AWS Management Console, and without any changes to your client applications, you can configure your bucket to use an S3 Bucket Key for SSE-KMS encryption on new objects.

> **Note**
>
> S3 Bucket Keys aren't supported for dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS).

### S3 Bucket Keys for SSE-KMS

Workloads that access millions or billions of objects encrypted with SSE-KMS can generate large volumes of requests to AWS KMS. When you use SSE-KMS to protect your data without an S3 Bucket Key, Amazon S3 uses an individual AWS KMS data key for every object. In this case, Amazon S3 makes a call to AWS KMS every time a request is made against a KMS-encrypted object. For information about how SSE-KMS works, see Using server-side encryption with AWS KMS keys (SSE-KMS).

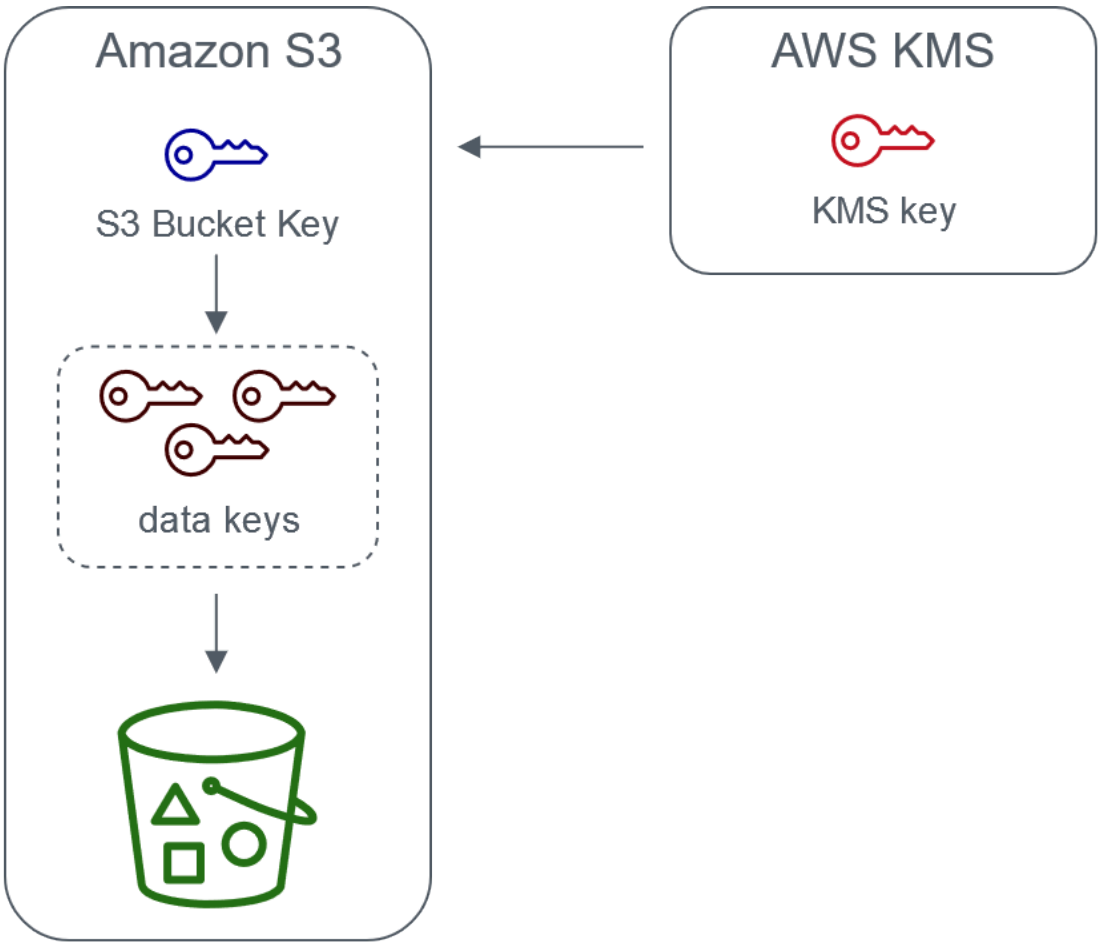
When you configure your bucket to use an S3 Bucket Key for SSE-KMS, AWS generates a short-lived bucket-level key from AWS KMS, then temporarily keeps it in S3. This bucket-level key will create data keys for new objects during its lifecycle. S3 Bucket Keys are used for a limited time period within Amazon S3, reducing the need for S3 to make requests to AWS KMS to complete encryption operations. This reduces traffic from S3 to AWS KMS, allowing you to access AWS KMS-encrypted objects in Amazon S3 at a fraction of the previous cost.

Unique bucket-level keys are fetched at least once per requester to ensure that the requester's access to the key is captured in an AWS KMS CloudTrail event. Amazon S3 treats callers as different requesters when they use different roles or accounts, or the same role with different scoping policies. AWS KMS request savings reflect the number of requesters, request patterns, and relative age of the objects requested. For example, a fewer number of requesters, requesting multiple objects in a limited time window, and encrypted with the same bucket-level key, results in greater savings.

> **Note**
>
> Using S3 Bucket Keys allows you to save on AWS KMS request costs by decreasing your requests to AWS KMS for `Encrypt`, `GenerateDataKey`, and `Decrypt` operations through the use of a bucket-level key. By design, subsequent requests that take advantage of this bucket-level key do not result in AWS KMS API requests or validate access against the AWS KMS key policy.

When you configure an S3 Bucket Key, objects that are already in the bucket do not use the S3 Bucket Key. To configure an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level .

Amazon S3 will only share an S3 Bucket Key for objects encrypted by the same AWS KMS key. S3 Bucket Keys are compatible with KMS keys created by AWS KMS, imported key material, and key material backed by custom key stores.




Server-side encryption with AWS Key Management service using an S3 Bucket Key

**Configuring S3 Bucket Keys**

You can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects through the Amazon S3 console, AWS SDKs, AWS CLI, or REST API. With S3 Bucket Keys enabled on your bucket, objects uploaded with a different specified SSE-KMS key will use their own S3 Bucket Keys. Regardless of your S3 Bucket Key setting, you can include the `x-amz-server-side-encryption-bucket-key-enabled` header with a `true` or `false` value in your request, to override the bucket setting.

Before you configure your bucket to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

### Configuring an S3 Bucket Key using the Amazon S3 console

When you create a new bucket, you can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. You can also configure an existing bucket to use an S3 Bucket Key for SSE-KMS on new objects by updating your bucket properties.

For more information, see Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects.

### REST API, AWS CLI, and AWS SDK support for S3 Bucket Keys

You can use the REST API, AWS CLI, or AWS SDK to configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. You can also enable an S3 Bucket Key at the object level.

For more information, see the following:

- Configuring an S3 Bucket Key at the object level
- Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects

The following API operations support S3 Bucket Keys for SSE-KMS:

- PutBucketEncryption
  - `ServerSideEncryptionRule` accepts the `BucketKeyEnabled` parameter for enabling and disabling an S3 Bucket Key.
- GetBucketEncryption
  - `ServerSideEncryptionRule` returns the settings for `BucketKeyEnabled`.
- PutObject, CopyObject, CreateMultipartUpload, and POST Object
  - The `x-amz-server-side-encryption-bucket-key-enabled` request header enables or disables an S3 Bucket Key at the object level.
- HeadObject, GetObject, UploadPartCopy, UploadPart, and CompleteMultipartUpload
  - The `x-amz-server-side-encryption-bucket-key-enabled` response header indicates if an S3 Bucket Key is enabled or disabled for an object.

### Working with AWS CloudFormation

In AWS CloudFormation, the `AWS::S3::Bucket` resource includes an encryption property called `BucketKeyEnabled` that you can use to enable or disable an S3 Bucket Key.

For more information, see Using AWS CloudFormation.

**Changes to note before enabling an S3 Bucket Key**

Before you enable an S3 Bucket Key, note the following related changes:

**IAM or AWS KMS key policies**

If your existing AWS Identity and Access Management (IAM) policies or AWS KMS key policies use your object Amazon Resource Name (ARN) as the encryption context to refine or limit access to your KMS key, these policies won't work with an S3 Bucket Key. S3 Bucket Keys use the bucket ARN as encryption context. Before you enable an S3 Bucket Key, update your IAM policies or AWS KMS key policies to use your bucket ARN as the encryption context.

For more information about the encryption context and S3 Bucket Keys, see Encryption context.

**CloudTrail events for AWS KMS**

After you enable an S3 Bucket Key, your AWS KMS CloudTrail events log your bucket ARN instead of your object ARN. Additionally, you see fewer KMS CloudTrail events for SSE-KMS objects in your logs. Because key material is time-limited in Amazon S3, fewer requests are made to AWS KMS.

**Using an S3 Bucket Key with replication**

You can use S3 Bucket Keys with Same-Region Replication (SRR) and Cross-Region Replication (CRR).

When Amazon S3 replicates an encrypted object, it generally preserves the encryption settings of the replica object in the destination bucket. However, if the source object is not encrypted and your destination bucket uses default encryption or an S3 Bucket Key, Amazon S3 encrypts the object with the destination bucket's configuration.

The following examples illustrate how an S3 Bucket Key works with replication. For more information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

**Example Example 1 – Source object uses S3 Bucket Keys; destination bucket uses default encryption**

If your source object uses an S3 Bucket Key but your destination bucket uses default encryption with SSE-KMS, the replica object maintains its S3 Bucket Key encryption settings in the destination bucket. The destination bucket still uses default encryption with SSE-KMS.

**Example Example 2 – Source object is not encrypted; destination bucket uses an S3 Bucket Key with SSE-KMS**

If your source object is not encrypted and the destination bucket uses an S3 Bucket Key with SSE-KMS, the replica object is encrypted by using an S3 Bucket Key with SSE-KMS in the destination bucket. This results in the `ETag` of the source object being different from the `ETag` of the replica object. You must update applications that use the `ETag` to accommodate for this difference.

**Working with S3 Bucket Keys**

For more information about enabling and working with S3 Bucket Keys, see the following sections:

- Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects
- Configuring an S3 Bucket Key at the object level
- Viewing the settings for an S3 Bucket Key

**Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects**

When you configure server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. S3 Bucket Keys decrease the request traffic from Amazon S3 to AWS KMS and reduce the cost of SSE-KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

You can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects by using the Amazon S3 console, REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or AWS CloudFormation. If you want to enable or disable an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level and Using S3 Batch Operations to encrypt objects with S3 Bucket Keys.

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket Amazon Resource Name (ARN) and not the object ARN, for example, `arn:aws:s3:::`*`bucket_ARN`*. You need to update your IAM policies to use the bucket ARN for the encryption context. For more information, see S3 Bucket Keys and replication.

The following examples illustrate how an S3 Bucket Key works with replication. For more information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

**Prerequisites**

Before you configure your bucket to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

**Using the S3 console**

In the S3 console, you can enable or disable an S3 Bucket Key for a new or existing bucket. Objects in the S3 console inherit their S3 Bucket Key setting from the bucket configuration. When you enable an S3 Bucket Key for your bucket, new objects that you upload to the bucket use an S3 Bucket Key for SSE-KMS.

**Uploading, copying, or modifying objects in buckets that have an S3 Bucket Key enabled**

If you upload, modify, or copy an object in a bucket that has an S3 Bucket Key enabled, the S3 Bucket Key settings for that object might be updated to align with the bucket configuration.

If an object already has an S3 Bucket Key enabled, the S3 Bucket Key settings for that object don't change when you copy or modify the object. However, if you modify or copy an object that doesn't have an S3 Bucket Key enabled, and the destination bucket has an S3 Bucket Key configuration, the object inherits the destination bucket's S3 Bucket Key settings. For example, if your source object doesn't have an S3 Bucket Key enabled but the destination bucket has S3 Bucket Key enabled, an S3 Bucket Key is enabled for the object.

**To enable an S3 Bucket Key when you create a new bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. Enter your bucket name, and choose your AWS Region.
5. Under **Default encryption**, for **Encryption key type**, choose **AWS Key Management Service key (SSE-KMS)**.
6. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating Keys in the *AWS Key Management Service Developer Guide*.

7. Under **Bucket Key**, choose **Enable**.
8. Choose **Create bucket**.

   Amazon S3 creates your bucket with an S3 Bucket Key enabled. New objects that you upload to the bucket will use an S3 Bucket Key.

   To disable an S3 Bucket Key, follow the previous steps, and choose **Disable**.

**To enable an S3 Bucket Key for an existing bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
4. Choose the **Properties** tab.
5. Under **Default encryption**, choose **Edit**.
6. Under **Default encryption**, for **Encryption key type**, choose **AWS Key Management Service key (SSE-KMS)**.
7. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

For more information about creating an AWS KMS key, see Creating Keys in the *AWS Key Management Service Developer Guide*.

8. Under **Bucket Key**, choose **Enable**.
9. Choose **Save changes**.

Amazon S3 enables an S3 Bucket Key for new objects added to your bucket. Existing objects don't use the S3 Bucket Key. To configure an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level .

To disable an S3 Bucket Key, follow the previous steps, and choose **Disable**.

#### Using the REST API

You can use PutBucketEncryption to enable or disable an S3 Bucket Key for your bucket. To configure an S3 Bucket Key with `PutBucketEncryption`, use the ServerSideEncryptionRule data type, which includes default encryption with SSE-KMS. You can also optionally use a customer managed key by specifying the KMS key ID for the customer managed key.

For more information and example syntax, see PutBucketEncryption.

#### Using the AWS SDK for Java

The following example enables default bucket encryption with SSE-KMS and an S3 Bucket Key by using the AWS SDK for Java.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
```

```
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

### Using the AWS CLI

The following example enables default bucket encryption with SSE-KMS and an S3 Bucket Key by using the AWS CLI. Replace the *`user input placeholders`* with your own information.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

### Using AWS CloudFormation

For more information about configuring an S3 Bucket Key with AWS CloudFormation, see AWS::S3::Bucket ServerSideEncryptionRule in the *AWS CloudFormation User Guide*.

### Configuring an S3 Bucket Key at the object level

When you perform a PUT or COPY operation using the REST API, AWS SDKs, or AWS CLI, you can enable or disable an S3 Bucket Key at the object level by adding the `x-amz-server-side-encryption-bucket-key-enabled` request header with a `true` or `false` value. S3 Bucket Keys reduce the cost of server-side encryption using AWS Key Management Service (AWS KMS) (SSE-KMS) by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

When you configure an S3 Bucket Key for an object using a PUT or COPY operation, Amazon S3 only updates the settings for that object. The S3 Bucket Key settings for the destination bucket do not change. If you submit a PUT or COPY request for a KMS-encrypted object into a bucket with S3 Bucket Keys enabled, your object level operation will automatically use S3 Bucket Keys unless you disable the keys in the request header. If you don't specify an S3 Bucket Key for your object, Amazon S3 applies the S3 Bucket Key settings for the destination bucket to the object.

**Prerequisite:**

Before you configure your object to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

**Topics**

- Amazon S3 Batch Operations
- Using the REST API
- Using the AWS SDK for Java (PutObject)
- Using the AWS CLI (PutObject)

### Amazon S3 Batch Operations

To encrypt your existing Amazon S3 objects, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on, and Batch Operations calls the respective API to perform the specified operation.

You can use the S3 Batch Operations Copy operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing large-scale batch operations on Amazon S3 objects and Encrypting objects with Amazon S3 Batch Operations.

### Using the REST API

When you use SSE-KMS, you can enable an S3 Bucket Key for an object by using the following API operations:

- PutObject – When you upload an object, you can specify the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key at the object level.

- CopyObject – When you copy an object and configure SSE-KMS, you can specify the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key for your object.
- POST Object – When you use a POST operation to upload an object and configure SSE-KMS, you can use the `x-amz-server-side-encryption-bucket-key-enabled` form field to enable or disable an S3 Bucket Key for your object.
- CreateMultipartUpload – When you upload large objects by using the `CreateMultipartUpload` API operation and configure SSE-KMS, you can use the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key for your object.

To enable an S3 Bucket Key at the object level, include the `x-amz-server-side-encryption-bucket-key-enabled` request header. For more information about SSE-KMS and the REST API, see Using the REST API.

**Using the AWS SDK for Java (PutObject)**

You can use the following example to configure an S3 Bucket Key at the object level using the AWS SDK for Java.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "amzn-s3-demo-bucket1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

### Using the AWS CLI (PutObject)

You can use the following AWS CLI example to configure an S3 Bucket Key at the object level as part of a `PutObject` request.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object key name --server-side-
encryption aws:kms --bucket-key-enabled --body filepath
```

### Viewing the settings for an S3 Bucket Key

You can view the settings for an S3 Bucket Key at the bucket or object level by using the Amazon S3 console, REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

S3 Bucket Keys decrease request traffic from Amazon S3 to AWS KMS and reduce the cost of server-side encryption using AWS Key Management Service (SSE-KMS). For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To view the S3 Bucket Key settings for a bucket or an object that has inherited S3 Bucket Key settings from the bucket configuration, you need permission to perform the `s3:GetEncryptionConfiguration` action. For more information, see GetBucketEncryption in the *Amazon Simple Storage Service API Reference*.

### Using the S3 console

In the S3 console, you can view the S3 Bucket Key settings for your bucket or object. S3 Bucket Key settings are inherited from the bucket configuration unless the source objects already has an S3 Bucket Key configured.

Objects and folders in the same bucket can have different S3 Bucket Key settings. For example, if you upload an object using the REST API and enable an S3 Bucket Key for the object, the object retains its S3 Bucket Key setting in the destination bucket, even if S3 Bucket Key is disabled in the destination bucket. As another example, if you enable an S3 Bucket Key for an existing bucket, objects that are already in the bucket do not use an S3 Bucket Key. However, new objects have an S3 Bucket Key enabled.

#### To view the S3 Bucket Key setting for your bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
4. Choose **Properties**.
5. In the **Default encryption** section, under **Bucket Key**, you see the S3 Bucket Key setting for your bucket.

   If you can't see the S3 Bucket Key setting, you might not have permission to perform the `s3:GetEncryptionConfiguration` action. For more information, see GetBucketEncryption in the *Amazon Simple Storage Service API Reference*.

**To view the S3 Bucket Key setting for your object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
3. In the **Objects** list, choose your object name.
4. On the **Details** tab, under **Server-side encryption settings**, choose **Edit**.

   Under **Bucket Key**, you see the S3 Bucket Key setting for your object. You cannot edit this setting.

**Using the AWS CLI**

**To return bucket-level S3 Bucket Key settings**

To use this example, replace each *`user input placeholder`* with your own information.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

For more information, see get-bucket-encryption in the *AWS CLI Command Reference*.

**To return object-level S3 Bucket Key settings**

To use this example, replace each *`user input placeholder`* with your own information.

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key my_images.tar.bz2
```

For more information, see head-object in the *AWS CLI Command Reference*.

#### Using the REST API

#### To return bucket-level S3 Bucket Key settings

To return encryption information for a bucket, including the settings for an S3 Bucket Key, use the `GetBucketEncryption` operation. S3 Bucket Key settings are returned in the response body in the `ServerSideEncryptionConfiguration` element with the `BucketKeyEnabled` setting. For more information, see GetBucketEncryption in the *Amazon S3 API Reference*.

#### To return object-level settings for an S3 Bucket Key

To return the S3 Bucket Key status for an object, use the `HeadObject` operation. `HeadObject` returns the `x-amz-server-side-encryption-bucket-key-enabled` response header to show whether an S3 Bucket Key is enabled or disabled for the object. For more information, see HeadObject in the *Amazon S3 API Reference*.

The following API operations also return the `x-amz-server-side-encryption-bucket-key-enabled` response header if an S3 Bucket Key is configured for an object:

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload
- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

## Using dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

Using dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) applies two layers of encryption to objects when they are uploaded to Amazon S3. DSSE-KMS helps you more easily fulfill compliance standards that require you to apply multilayer encryption to your data and have full control of your encryption keys.

When you use DSSE-KMS with an Amazon S3 bucket, the AWS KMS keys must be in the same Region as the bucket. Also, when DSSE-KMS is requested for the object, the S3 checksum that's part of the object's metadata is stored in encrypted form. For more information about checksums, see Checking object integrity.

There are additional charges for using DSSE-KMS and AWS KMS keys. For more information about DSSE-KMS pricing, see AWS KMS key concepts in the *AWS Key Management Service Developer Guide* and AWS KMS pricing.

> **Note**
>
> S3 Bucket Keys aren't supported for DSSE-KMS.

### Requiring dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

To require dual-layer server-side encryption of all objects in a particular Amazon S3 bucket, you can use a bucket policy. For example, the following bucket policy denies the upload object (`s3:PutObject`) permission to everyone if the request does not include an `x-amz-server-side-encryption` header that requests server-side encryption with DSSE-KMS.

```
{
            "Version":"2012-10-17",
            "Id":"PutObjectPolicy",
            "Statement":[{
                  "Sid":"DenyUnEncryptedObjectUploads",
                  "Effect":"Deny",
                  "Principal":"*",
                  "Action":"s3:PutObject",
                  "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
                  "Condition":{
                     "StringNotEquals":{
                        "s3:x-amz-server-side-encryption":"aws:kms:dsse"
                     }
                  }
               }
            ]
         }
```

### Topics

- Specifying dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

### Specifying dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

You can apply encryption when you are either uploading a new object or copying an existing object.

You can specify DSSE-KMS by using the Amazon S3 console, Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI). For more information, see the following topics.

> **Note**
>
> You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the

multi-Region features of the key. For more information, see Using multi-Region keys in *AWS Key Management Service Developer Guide*.

**Note**

If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

### Using the S3 console

This section describes how to set or change the type of encryption of an object to use dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) by using the Amazon S3 console.

**Note**

- If you change an object's method of encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).
- If you change the encryption type for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing the encryption type for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.

  If the destination bucket policy denies the `s3:GetObjectTagging` action, the encryption type for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.

### To add or change encryption for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want to encrypt.
4. In the **Objects** list, select the check box next to the object that you want to add or change encryption for.

   The object's details page appears, with several sections that display the properties for your object.
5. Choose the **Properties** tab.
6. Scroll down to the **Default encryption** section and choose **Edit**.

   The **Edit default encryption** page opens.
7. Under **Encryption type**, choose **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**.
8. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

     For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key, and then you must enter the KMS key ARN.

Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

9. For **Bucket Key**, choose **Disable**. S3 Bucket Keys aren't supported for DSSE-KMS.
10. Choose **Save changes**.

> **Note**
>
> This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

### Using the REST API

When you create an object—that is, when you upload a new object or copy an existing object—you can specify the use of dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) to encrypt your data. To do this, add the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `aws:kms:dsse`. Amazon S3 confirms that your object is stored with DSSE-KMS encryption by returning the response header `x-amz-server-side-encryption`.

If you specify the `x-amz-server-side-encryption` header with a value of `aws:kms:dsse`, you can also use the following request headers:

- `x-amz-server-side-encryption-aws-kms-key-id: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

### Topics

- Amazon S3 REST API operations that support DSSE-KMS
- Encryption context (x-amz-server-side-encryption-context)
- AWS KMS key ID (x-amz-server-side-encryption-aws-kms-key-id)

### Amazon S3 REST API operations that support DSSE-KMS

The following REST API operations accept the `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, and `x-amz-server-side-encryption-context` request headers.

- PutObject – When you upload data by using the PUT API operation, you can specify these request headers.
- CopyObject – When you copy an object, you have both a source object and a target object. When you pass DSSE-KMS headers with the `CopyObject` operation, they are applied only to the target object. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object is not encrypted unless you explicitly request server-side encryption.
- POST Object – When you use a `POST` operation to upload an object, instead of the request headers, you provide the same information in the form fields.
- CreateMultipartUpload – When you upload large objects by using a multipart upload, you can specify these headers in the `CreateMultipartUpload` request.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored with server-side encryption.

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

**Important**

- All GET and PUT requests for an object that's protected by AWS KMS fail if you don't make them by using Secure Sockets Layer (SSL), Transport Layer Security (TLS), or Signature Version 4.
- If your object uses DSSE-KMS, don't send encryption request headers for GET requests and HEAD requests, or you'll get an HTTP 400 (Bad Request) error.

#### Encryption context (`x-amz-server-side-encryption-context`)

If you specify `x-amz-server-side-encryption:aws:kms:dsse`, the Amazon S3 API supports an encryption context with the `x-amz-server-side-encryption-context` header. An encryption context is a set of key-value pairs that contain additional contextual information about the data.

Amazon S3 automatically uses the object's Amazon Resource Name (ARN) as the encryption context pair; for example, `arn:aws:s3:::`*`object_ARN`*.

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header. However, because the encryption context is not encrypted, make sure it does not include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context.

For information about the encryption context in Amazon S3, see Encryption context. For general information about the encryption context, see AWS Key Management Service Concepts - Encryption context in the *AWS Key Management Service Developer Guide*.

#### AWS KMS key ID (`x-amz-server-side-encryption-aws-kms-key-id`)

You can use the `x-amz-server-side-encryption-aws-kms-key-id` header to specify the ID of the customer managed key that's used to protect the data. If you specify the `x-amz-server-side-encryption:aws:kms:dsse` header but don't provide the `x-amz-server-side-encryption-aws-kms-key-id` header, Amazon S3 uses the AWS managed key (`aws/s3`) to protect the data. If you want to use a customer managed key, you must provide the `x-amz-server-side-encryption-aws-kms-key-id` header of the customer managed key.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

#### Using the AWS CLI

When you upload a new object or copy an existing object, you can specify the use of DSSE-KMS to encrypt your data. To do this, add the `--server-side-encryption aws:kms:dsse` parameter to the request. Use the `--ssekms-key-id` *`example-key-id`* parameter to add your customer managed AWS KMS key that you created. If you specify `--server-side-encryption aws:kms:dsse`, but do not provide an AWS KMS key ID, then Amazon S3 will use the AWS managed key (`aws/s3`).

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --server-
side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

You can encrypt an unencrypted object to use DSSE-KMS by copying the object back in place.

```
aws s3api copy-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --
body filepath --bucket DOC-EXAMPLE-BUCKET --key example-object-key --sse aws:kms:dsse
 --sse-kms-key-id example-key-id --body filepath
```

## Using server-side encryption with customer-provided keys (SSE-C)

Server-side encryption is about protecting data at rest. Server-side encryption encrypts only the object data, not the object metadata. By using server-side encryption with customer-provided keys (SSE-C), you can store your data encrypted with your own encryption keys. With the encryption key that you provide as part of your request, Amazon S3 manages data encryption as it writes to disks and data decryption when you access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing that you need to do is manage the encryption keys that you provide.

When you upload an object, Amazon S3 uses the encryption key that you provide to apply AES-256 encryption to your data. Amazon S3 then removes the encryption key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. Amazon S3

first verifies that the encryption key that you provided matches, and then it decrypts the object before returning the object data to you.

There are no additional charges for using SSE-C. However, requests to configure and use SSE-C incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing.

> **Note**
>
> Amazon S3 does not store the encryption key that you provide. Instead, it stores a randomly salted Hash-based Message Authentication Code (HMAC) value of the encryption key to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means if you lose the encryption key, you lose the object.

S3 Replication supports objects that are encrypted with SSE-C. For more information about replicating encrypted objects, see the section called "Replicating encrypted objects".

For more information about SSE-C, see the following topics.

**Topics**

- SSE-C overview
- Requiring and restricting SSE-C
- Presigned URLs and SSE-C
- Specifying server-side encryption with customer-provided keys (SSE-C)

### SSE-C overview

This section provides an overview of SSE-C. When using SSE-C, keep the following considerations in mind.

- You must use HTTPS.

> **Important**
>
> Amazon S3 rejects any requests made over HTTP when using SSE-C. For security considerations, we recommend that you consider any key that you erroneously send over HTTP to be compromised. Discard the key and rotate as appropriate.

- The entity tag (ETag) in the response is not the MD5 hash of the object data.
- You manage a mapping of which encryption key was used to encrypt which object. Amazon S3 does not store encryption keys. You are responsible for tracking which encryption key you provided for which object.
  - If your bucket is versioning-enabled, each object version that you upload by using this feature can have its own encryption key. You are responsible for tracking which encryption key was used for which object version.
  - Because you manage encryption keys on the client side, you manage any additional safeguards, such as key rotation, on the client side.

> **Warning**
>
> If you lose the encryption key, any GET request for an object without its encryption key fails, and you lose the object.

### Requiring and restricting SSE-C

To require SSE-C for all objects in a particular Amazon S3 bucket, you can use a bucket policy.

For example, the following bucket policy denies upload object (`s3:PutObject`) permissions for all requests that don't include the `x-amz-server-side-encryption-customer-algorithm` header requesting SSE-C.

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
```

```
    ]
}
```

You can also use a policy to restrict server-side encryption of all objects in a particular Amazon S3 bucket. For example, the following bucket policy denies the upload object (`s3:PutObject`) permission to everyone if the request includes the `x-amz-server-side-encryption-customer-algorithm` header requesting SSE-C.

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

**Important**

If you use a bucket policy to require SSE-C on `s3:PutObject`, you must include the `x-amz-server-side-encryption-customer-algorithm` header in all multipart upload requests (CreateMultipartUpload, UploadPart, and CompleteMultipartUpload).

### Presigned URLs and SSE-C

You can generate a presigned URL that can be used for operations such as uploading a new object, retrieving an existing object, or retrieving object metadata. Presigned URLs support SSE-C as follows:

- When creating a presigned URL, you must specify the algorithm by using the `x-amz-server-side-encryption-customer-algorithm` header in the signature calculation.
- When using the presigned URL to upload a new object, retrieve an existing object, or retrieve only object metadata, you must provide all the encryption headers in your client application's request.

> **Note**
>
> For non-SSE-C objects, you can generate a presigned URL and directly paste that URL into a browser to access the data.
> However, you cannot do this for SSE-C objects, because in addition to the presigned URL, you also must include HTTP headers that are specific to SSE-C objects. Therefore, you can use presigned URLs for SSE-C objects only programmatically.

For more information about presigned URLs, see the section called “Working with presigned URLs”.

**Specifying server-side encryption with customer-provided keys (SSE-C)**

At the time of object creation with the REST API, you can specify server-side encryption with customer-provided keys (SSE-C). When you use SSE-C, you must provide encryption key information using the following request headers.

| Name | Description |
| --- | --- |
| `x-amz-server-side-encryption-customer-algorithm` | Use this header to specify the encryption algorithm. The header value must be `AES256`. |
| `x-amz-server-side-encryption-customer-key` | Use this header to provide the 256-bit, base64-encoded encryption key for Amazon S3 to use to encrypt or decrypt your data. |
| `x-amz-server-side-encryption-customer-key-MD5` | Use this header to provide the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure that the encryption key was transmitted without error. |

You can use AWS SDK wrapper libraries to add these headers to your request. If you need to, you can make the Amazon S3 REST API calls directly in your application.

> **Note**
>
> You cannot use the Amazon S3 console to upload an object and request SSE-C. You also cannot use the console to update (for example, change the storage class or add metadata) an existing object stored using SSE-C.

### Using the REST API

#### Amazon S3 rest APIs that support SSE-C

The following Amazon S3 APIs support server-side encryption with customer-provided encryption keys (SSE-C).

- **GET operation** – When retrieving objects using the GET API (see GET Object), you can specify the request headers.
- **HEAD operation** – To retrieve object metadata using the HEAD API (see HEAD Object), you can specify these request headers.
- **PUT operation** – When uploading data using the PUT Object API (see PUT Object), you can specify these request headers.
- **Multipart Upload** – When uploading large objects using the multipart upload API, you can specify these headers. You specify these headers in the initiate request (see Initiate Multipart Upload) and each subsequent part upload request (see Upload Part or Upload Part - Copy). For each part upload request, the encryption information must be the same as what you provided in the initiate multipart upload request.
- **POST operation** – When using a POST operation to upload an object (see POST Object), instead of the request headers, you provide the same information in the form fields.
- **Copy operation** – When you copy an object (see PUT Object - Copy), you have both a source object and a target object:
  - If you want the target object encrypted using server-side encryption with AWS managed keys, you must provide the `x-amz-server-side-encryption` request header.
  - If you want the target object encrypted using SSE-C, you must provide encryption information using the three headers described in the preceding table.

- If the source object is encrypted using SSE-C, you must provide encryption key information using the following headers so that Amazon S3 can decrypt the object for copying.

| Name | Description |
| --- | --- |
| `x-amz-copy-source-server-side-encryption-customer-algorithm` | Include this header to specify the algorithm Amazon S3 should use to decrypt the source object. This value must be AES256. |
| `x-amz-copy-source-server-side-encryption-customer-key` | Include this header to provide the base64-encoded encryption key for Amazon S3 to use to decrypt the source object. This encryption key must be the one that you provided Amazon S3 when you created the source object. Otherwise, Amazon S3 cannot decrypt the object. |
| `x-amz-copy-source-server-side-encryption-customer-key-MD5` | Include this header to provide the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. |

**Using the AWS SDKs to specify SSE-C for PUT, GET, Head, and Copy operations**

The following examples show how to request server-side encryption with customer-provided keys (SSE-C) for objects. The examples perform the following operations. Each operation shows how to specify SSE-C-related headers in the request:

- **Put object** – Uploads an object and requests server-side encryption using a customer-provided encryption key.
- **Get object** – Downloads the object uploaded in the previous step. In the request, you provide the same encryption information you provided when you uploaded the object. Amazon S3 needs this information to decrypt the object so that it can return it to you.
- **Get object metadata** – Retrieves the object's metadata. You provide the same encryption information used when the object was created.

- **Copy object** – Makes a copy of the previously-uploaded object. Because the source object is stored using SSE-C, you must provide its encryption information in your copy request. By default, Amazon S3 encrypts the copy of the object only if you explicitly request it. This example directs Amazon S3 to store an encrypted copy of the object.

Java

> **Note**
>
> This example shows how to upload an object in a single operation. When using the Multipart Upload API to upload large objects, you provide encryption information in the same way shown in this example. For examples of multipart uploads that use the AWS SDK for Java, see Uploading an object using multipart upload.

To add the required encryption information, you include an `SSECustomerKey` in your request. For more information about the `SSECustomerKey` class, see the REST API section.

For information about SSE-C, see Using server-side encryption with customer-provided keys (SSE-C). For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

**Example**

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
```

```
public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
    private static KeyGenerator KEY_GENERATOR;

    public static void main(String[] args) throws IOException,
 NoSuchAlgorithmException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to
 retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }

    private static void uploadObject(String bucketName, String keyName, File file) {
        PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
        S3_CLIENT.putObject(putRequest);
        System.out.println("Object uploaded");
    }

    private static void downloadObject(String bucketName, String keyName) throws
IOException {
        GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
        S3Object object = S3_CLIENT.getObject(getObjectRequest);

        System.out.println("Object content: ");
        displayTextInputStream(object.getObjectContent());
    }

    private static void retrieveObjectMetadata(String bucketName, String keyName) {
        GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
                .withSSECustomerKey(SSE_KEY);
        ObjectMetadata objectMetadata =
S3_CLIENT.getObjectMetadata(getMetadataRequest);
        System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
    }

    private static void copyObject(String bucketName, String keyName, String
targetKeyName)
            throws NoSuchAlgorithmException {
        // Create a new encryption key for target so that the target is saved using
        // SSE-C.
        SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

        CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
bucketName, targetKeyName)
                .withSourceSSECustomerKey(SSE_KEY)
                .withDestinationSSECustomerKey(newSSEKey);

        S3_CLIENT.copyObject(copyRequest);
```

```
        System.out.println("Object copied");
    }

    private static void displayTextInputStream(S3ObjectInputStream input) throws
 IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET

> **Note**
>
> For examples of uploading large objects using the multipart upload API, see Uploading an object using multipart upload and Using the AWS SDKs (low-level API).

For information about SSE-C, see Using server-side encryption with customer-provided keys (SSE-C). For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

**Example**

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
```

```
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what
you uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side
encryption with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        private static async Task<PutObjectRequest> UploadObjectAsync(string
base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
        private static async Task DownloadObjectAsync(string base64Key,
PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (String.Compare(putObjectRequest.ContentBody, content) == 0)
                    Console.WriteLine("Object content is same as we uploaded");
                else
```

```
                    Console.WriteLine("Error...Object content is not same.");

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                else
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");

                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
        private static async Task CopyObjectAsync(Aes aesEncryption, string
base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
```

```
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
}
```

### Using the AWS SDKs to specify SSE-C for multipart uploads

The example in the preceding section shows how to request server-side encryption with customer-provided key (SSE-C) in the PUT, GET, Head, and Copy operations. This section describes other Amazon S3 APIs that support SSE-C.

Java

To upload large objects, you can use multipart upload API (see Uploading and copying objects using multipart upload). You can use either high-level or low-level APIs to upload large objects. These APIs support encryption-related headers in the request.

- When using the high-level `TransferManager` API, you provide the encryption-specific headers in the `PutObjectRequest` (see Uploading an object using multipart upload).
- When using the low-level API, you provide encryption-related information in the `InitiateMultipartUploadRequest`, followed by identical encryption information in each `UploadPartRequest`. You do not need to provide any encryption-specific headers in your `CompleteMultipartUploadRequest`. For examples, see Using the AWS SDKs (low-level API).

The following example uses `TransferManager` to create objects and shows how to provide SSE-C related information. The example does the following:

- Creates an object using the `TransferManager.upload()` method. In the `PutObjectRequest` instance, you provide encryption key information to request. Amazon S3 encrypts the object using the customer-provided key.
- Makes a copy of the object by calling the `TransferManager.copy()` method. The example directs Amazon S3 to encrypt the object copy using a new `SSECustomerKey`. Because the source object is encrypted using SSE-C, the `CopyObjectRequest` also provides the encryption key of the source object so that Amazon S3 can decrypt the object before copying it.

**Example**

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";
```

```
        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
keyName, new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this
call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
keyName, bucketName, targetKeyName);
            SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);

copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call
returns
            // immediately.
            Copy copy = tm.copy(copyObjectRequest);
```

```
            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
            System.out.println("Copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

To upload large objects, you can use multipart upload API (see Uploading and copying objects using multipart upload). AWS SDK for .NET provides both high-level or low-level APIs to upload large objects. These APIs support encryption-related headers in the request.

- When using high-level `Transfer-Utility` API, you provide the encryption-specific headers in the `TransferUtilityUploadRequest` as shown. For code examples, see Uploading an object using multipart upload.

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- When using the low-level API, you provide encryption-related information in the initiate multipart upload request, followed by identical encryption information in the subsequent upload part requests. You do not need to provide any encryption-specific headers in your complete multipart upload request. For examples, see Using the AWS SDKs (low-level API).

The following is a low-level multipart upload example that makes a copy of an existing large object. In the example, the object to be copied is stored in Amazon S3 using SSE-C, and you want to save the target object also using SSE-C. In the example, you do the following:

- Initiate a multipart upload request by providing an encryption key and related information.
- Provide source and target object encryption keys and related information in the `CopyPartRequest`.
- Obtain the size of the source object to be copied by retrieving the object metadata.
- Upload the objects in 5 MB parts.

**Example**

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name
 ***";
        private const string targetKeyName      = "*** key name for the target
 object ***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }
```

```
        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key,
s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string
base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored
encrypted with
                // customer provided key you need to provide encryption
information in your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
```

```
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " *
**source object encryption key ***"
                };

                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
                for (int i = 1; filePosition <
getObjectMetadataResponse.ContentLength; i++)
                {
                    CopyPartRequest copyPartRequest = new CopyPartRequest
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
information.
                        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size
then use the remaining size.
                        LastByte = lastByte >
getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 :
lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
```

```
                    };
                    uploadResponses.Add(await
s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task
CreateSampleObjUsingClientEncryptionKeyAsync(string base64Key, IAmazonS3
s3Client)
        {
            // List to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // 1. Initialize.
```

```
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
```

```
                CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
}
```

## Protecting data by using client-side encryption

*Client-side encryption* is the act of encrypting your data locally to help ensure its security in transit and at rest. To encrypt your objects before you send them to Amazon S3, use the Amazon S3 Encryption Client. When your objects are encrypted in this manner, your objects aren't exposed to any third party, including AWS. Amazon S3 receives your objects already encrypted; Amazon S3 does not play a role in encrypting or decrypting your objects. You can use both the Amazon S3 Encryption Client and server-side encryption to encrypt your data. When you send encrypted objects to Amazon S3, Amazon S3 doesn't recognize the objects as being encrypted, it only detects typical objects.

The Amazon S3 Encryption Client works as an intermediary between you and Amazon S3. After you instantiate the Amazon S3 Encryption Client, your objects are automatically encrypted and decrypted as part of your Amazon S3 `PutObject` and `GetObject` requests. Your objects are all encrypted with a unique data key. The Amazon S3 Encryption Client does not use or interact with bucket keys, even if you specify a KMS key as your wrapping key.

The *Amazon S3 Encryption Client Developer Guide* focuses on versions 3.0 and later of the Amazon S3 Encryption Client. For more information, see What is the Amazon S3 Encryption Client? in the *Amazon S3 Encryption Client Developer Guide*.

For more information about previous versions of the Amazon S3 Encryption client, see the AWS SDK Developer Guide for your programming language.

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby
- AWS SDK for C++

# Internetwork traffic privacy

This topic describes how Amazon S3 secures connections from the service to other locations.

## Traffic between service and on-premises clients and applications

The following connections can be combined with AWS PrivateLink to provide connectivity between your private network and AWS:

- An AWS Site-to-Site VPN connection. For more information, see What is AWS Site-to-Site VPN?
- An AWS Direct Connect connection. For more information, see What is AWS Direct Connect?

Access to Amazon S3 via the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend TLS 1.3. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Most modern systems such as Java 7 and later support these

modes. Additionally, you must sign requests using an access key ID and a secret access key that are associated with an IAM principal, or you can use the AWS Security Token Service (STS) to generate temporary security credentials to sign requests.

### Traffic between AWS resources in the same Region

A virtual private cloud (VPC) endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. The VPC routes requests to Amazon S3 and routes responses back to the VPC. For more information, see VPC Endpoints in the *VPC User Guide*. For example bucket policies that you can use to control S3 bucket access from VPC endpoints, see Controlling access from VPC endpoints with bucket policies.

## AWS PrivateLink for Amazon S3

With AWS PrivateLink for Amazon S3, you can provision *interface VPC endpoints* (interface endpoints) in your virtual private cloud (VPC). These endpoints are directly accessible from applications that are on premises over VPN and AWS Direct Connect, or in a different AWS Region over VPC peering.

Interface endpoints are represented by one or more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests to Amazon S3 over interface endpoints stay on the Amazon network. You can also access interface endpoints in your VPC from on-premises applications through AWS Direct Connect or AWS Virtual Private Network (AWS VPN). For more information about how to connect your VPC with your on-premises network, see the AWS Direct Connect User Guide and the AWS Site-to-Site VPN User Guide.

For general information about interface endpoints, see Interface VPC endpoints (AWS PrivateLink) in the *AWS PrivateLink Guide*.

**Topics**

- Types of VPC endpoints for Amazon S3
- Restrictions and limitations of AWS PrivateLink for Amazon S3
- Creating a VPC endpoint
- Accessing Amazon S3 interface endpoints
- Private DNS
- Accessing buckets, access points, and Amazon S3 Control API operations from S3 interface endpoints

- Updating an on-premises DNS configuration
- Creating a VPC endpoint policy for Amazon S3

## Types of VPC endpoints for Amazon S3

You can use two types of VPC endpoints to access Amazon S3: *gateway endpoints* and *interface endpoints* (by using AWS PrivateLink). A *gateway endpoint* is a gateway that you specify in your route table to access Amazon S3 from your VPC over the AWS network. *Interface endpoints* extend the functionality of gateway endpoints by using private IP addresses to route requests to Amazon S3 from within your VPC, on premises, or from a VPC in another AWS Region by using VPC peering or AWS Transit Gateway. For more information, see What is VPC peering? and Transit Gateway vs VPC peering.

Interface endpoints are compatible with gateway endpoints. If you have an existing gateway endpoint in the VPC, you can use both types of endpoints in the same VPC.

| Gateway endpoints for Amazon S3 | Interface endpoints for Amazon S3 |
|---|---|
| In both cases, your network traffic remains on the AWS network. | |
| Use Amazon S3 public IP addresses | Use private IP addresses from your VPC to access Amazon S3 |
| Use the same Amazon S3 DNS names | Require endpoint-specific Amazon S3 DNS names |
| Do not allow access from on premises | Allow access from on premises |
| Do not allow access from another AWS Region | Allow access from a VPC in another AWS Region by using VPC peering or AWS Transit Gateway |
| Not billed | Billed |

For more information about gateway endpoints, see Gateway VPC endpoints in the *AWS PrivateLink Guide*.

## Restrictions and limitations of AWS PrivateLink for Amazon S3

VPC limitations apply to AWS PrivateLink for Amazon S3. For more information, see Interface endpoint considerations and AWS PrivateLink quotas in the *AWS PrivateLink Guide*. In addition, the following restrictions apply.

AWS PrivateLink for Amazon S3 does not support the following:

- Federal Information Processing Standard (FIPS) endpoints
- Website endpoints
- Legacy global endpoints
- S3 dash Region endpoints
- Amazon S3 dual-stack endpoints
- Using CopyObject or UploadPartCopy between buckets in different AWS Regions
- Transport Layer Security (TLS) 1.1

## Creating a VPC endpoint

To create a VPC interface endpoint, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

## Accessing Amazon S3 interface endpoints

When you create an interface endpoint, Amazon S3 generates two types of endpoint-specific, S3 DNS names: *Regional* and *zonal*.

- A *Regional* DNS name includes a unique VPC endpoint ID, a service identifier, the AWS Region, and `vpce.amazonaws.com` in its name. For example, for VPC endpoint ID *`vpce-1a2b3c4d`*, the DNS name generated might be similar to *`vpce-1a2b3c4d-5e6f`*`.s3.us-east-1.vpce.amazonaws.com`.
- A *Zonal* DNS name includes the Availability Zone—for example, *`vpce-1a2b3c4d-5e6f`***`-us-east-1a.`**`s3.us-east-1.vpce.amazonaws.com`. You might use this option if your architecture isolates Availability Zones. For example, you could use it for fault containment or to reduce Regional data transfer costs.

Endpoint-specific S3 DNS names can be resolved from the S3 public DNS domain.

# Private DNS

Private DNS options for VPC interface endpoints simplify routing S3 traffic over VPC endpoints and help you take advantage of the lowest-cost network path available to your application. You can use private DNS options to route Regional S3 traffic without updating your S3 clients to use the endpoint-specific DNS names of your interface endpoints, or managing DNS infrastructure. With private DNS names enabled, Regional S3 DNS queries resolve to the private IP addresses of AWS PrivateLink for the following endpoints:

- Regional bucket endpoints (for example, `s3.us-east-1.amazonaws.com`)
- Control endpoints (for example, `s3-control.us-east-1.amazonaws.com`)
- Access point endpoints (for example, `s3-accesspoint.us-east-1.amazonaws.com`)

If you have a gateway endpoint in your VPC, you can automatically route in-VPC requests over your existing S3 gateway endpoint and on-premises requests over your interface endpoint. This approach allows you to optimize your networking costs by using gateway endpoints, which are not billed, for your in-VPC traffic. Your on-premises applications can use AWS PrivateLink with the help of the inbound Resolver endpoint. Amazon provides a DNS server, called the Route 53 Resolver, for your VPC. An inbound Resolver endpoint forwards DNS queries from the on-premises network to Route 53 Resolver.

**Important**

To take advantage of the lowest cost network path when using **Enable private DNS only for inbound endpoints**, a gateway endpoint must be present in your VPC. The presence of a gateway endpoint helps ensure that in-VPC traffic always routes over the AWS private network when the **Enable private DNS only for inbound endpoints** option is selected. You must maintain this gateway endpoint while you have the **Enable private DNS only for inbound endpoints** option selected. If you want to delete your gateway endpoint you must first clear **Enable private DNS only for inbound endpoints**.

If you want to update an existing interface endpoint to **Enable private DNS only for inbound endpoints**, first confirm that your VPC has an S3 gateway endpoint. For more information about gateway endpoints and managing private DNS names, see Gateway VPC endpoints and Manage DNS names respectively in the *AWS PrivateLink Guide*.

The **Enable private DNS only for inbound endpoints** option is available only for services that support gateway endpoints.

For more information about creating a VPC endpoint that uses **Enable private DNS only for inbound endpoints**, see Create an interface endpoint in the *AWS PrivateLink Guide*.

**Using the VPC console**

In the console you have two options: **Enable DNS name** and **Enable private DNS only for inbound endpoints**. **Enable DNS name** is an option supported by AWS PrivateLink. By using the **Enable DNS name** option, you can use Amazon's private connectivity to Amazon S3, while making requests to the default public endpoint DNS names. When this option is enabled, customers can take advantage of the lowest cost network path available to their application.

When you enable private DNS names on an existing or new VPC interface endpoint for Amazon S3, the **Enable private DNS only for inbound endpoints** option is selected by default. If this option is selected, your applications use only interface endpoints for your on-premises traffic. This in-VPC traffic automatically uses the lower-cost gateway endpoints. Alternatively, you can clear **Enable private DNS only for inbound endpoints** to route all S3 requests over your interface endpoint.

**Using the AWS CLI**

If you don't specify a value for `PrivateDnsOnlyForInboundResolverEndpoint`, it will default to `true`. However, before your VPC applies your settings, it performs a check to make sure that you have a gateway endpoint present in the VPC. If a gateway endpoint is present in the VPC, the call succeeds. If not, you will see the following error message:

To set PrivateDnsOnlyForInboundResolverEndpoint to true, the VPC *`vpce_id`* must have a gateway endpoint for the service.

**For a new VPC Interface endpoint**

Use the `private-dns-enabled` and `dns-options` attributes to enable private DNS through the command line. The `PrivateDnsOnlyForInboundResolverEndpoint` option in the `dns-options` attribute must be set to `true`. Replace the *`user input placeholders`* with your own information.

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
```

```
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

**For an existing VPC endpoint**

If you want to use private DNS for an existing VPC endpoint, use the following example command and replace the *user input placeholders* with your own information.

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

If you want to update an existing VPC endpoint to enable private DNS only for the Inbound Resolver, use the following example and replace the sample values with your own.

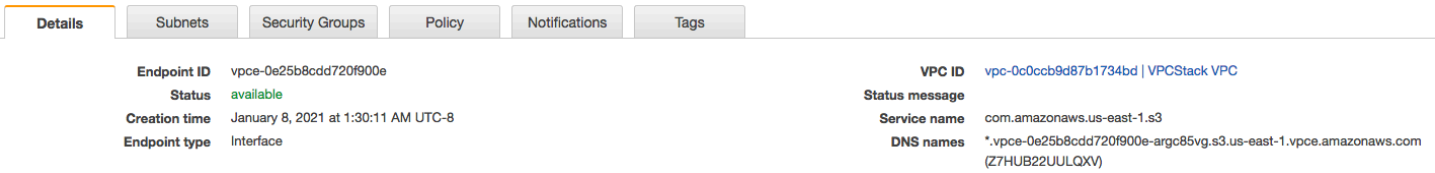
```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

## Accessing buckets, access points, and Amazon S3 Control API operations from S3 interface endpoints

You can use the AWS CLI or AWS SDKs to access buckets, S3 access points, and Amazon S3 Control API operations through S3 interface endpoints.

The following image shows the VPC console **Details** tab, where you can find the DNS name of a VPC endpoint. In this example, the *VPC endpoint ID (vpce-id)* is

vpce-0e25b8cdd720f900e and the *DNS name* is *.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com.




When using the DNS name to access a resource, replace * with the appropriate value. The appropriate values to use in place of * are as follows:

- bucket
- accesspoint
- control

For example, to access a bucket, use a *DNS name* like this:

bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com

For examples of how to use DNS names to access buckets, access points, and Amazon S3 Control API operations, see the following sections of AWS CLI examples and AWS SDK examples.

For more information about how to view your endpoint-specific DNS names, see Viewing endpoint service private DNS name configuration in the *VPC User Guide*.

## AWS CLI examples

To access S3 buckets, S3 access points, or Amazon S3 Control API operations through S3 interface endpoints in AWS CLI commands, use the --region and --endpoint-url parameters.

**Example: Use an endpoint URL to list objects in your bucket**

In the following example, replace the bucket name *my-bucket*, Region *us-east-1*, and the DNS name of the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

**Example: Use an endpoint URL to list objects from an access point**

- **Method 1** – Using the Amazon Resource Name (ARN) of the access point with the access point endpoint

  Replace the ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*, the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
accesspointexamplename --region us-east-1 --endpoint-url
 https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

  If you can't run the command successfully, update your AWS CLI to the latest version and try again. For more information on the update instructions, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

- **Method 2** – Using the alias of the access point with the regional bucket endpoint

  In the following example, replace the access point alias *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*, the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
 --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

- **Method 3** – Using the alias of the access point with the access point endpoint

  First, to construct an S3 endpoint with the bucket included as part of the hostname, set the addressing style to `virtual` for `aws s3api` to use. For more information about `AWS configure`, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

```
aws configure set default.s3.addressing_style virtual
```

  Then, in the following example, replace the access point alias *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*,

the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information. For more information about access point alias, see Using a bucket-style alias for your S3 bucket access point.

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

**Example: Use an endpoint URL to list jobs with an S3 control API operation**

In the following example, replace the Region *us-east-1*, the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*, and the account ID *12345678* with your own information.

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

## AWS SDK examples

To access S3 buckets, S3 access points, or Amazon S3 Control API operations through S3 interface endpoints when using the AWS SDKs, update your SDKs to the latest version. Then configure your clients to use an endpoint URL for accessing a bucket, access point, or Amazon S3 Control API operations through S3 interface endpoints.

SDK for Python (Boto3)

**Example: Use an endpoint URL to access an S3 bucket**

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

**Example: Use an endpoint URL to access an S3 access point**

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

**Example: Use an endpoint URL to access the Amazon S3 Control API**

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

SDK for Java 1.x

**Example: Use an endpoint URL to access an S3 bucket**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

**Example: Use an endpoint URL to access an S3 access point**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and ARN *us-east-1:123456789012:accesspoint/prod* with your own information.

```
// accesspoint client
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-
east-1:123456789012:accesspoint/prod");
```

**Example: Use an endpoint URL to access an Amazon S3 Control API operation**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
// control client
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

**Example: Use an endpoint URL to access an S3 bucket**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
```

```
                    .build()
```

**Example: Use an endpoint URL to access an S3 access point**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                    .build()
```

**Example: Use an endpoint URL to access the Amazon S3 Control API**

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

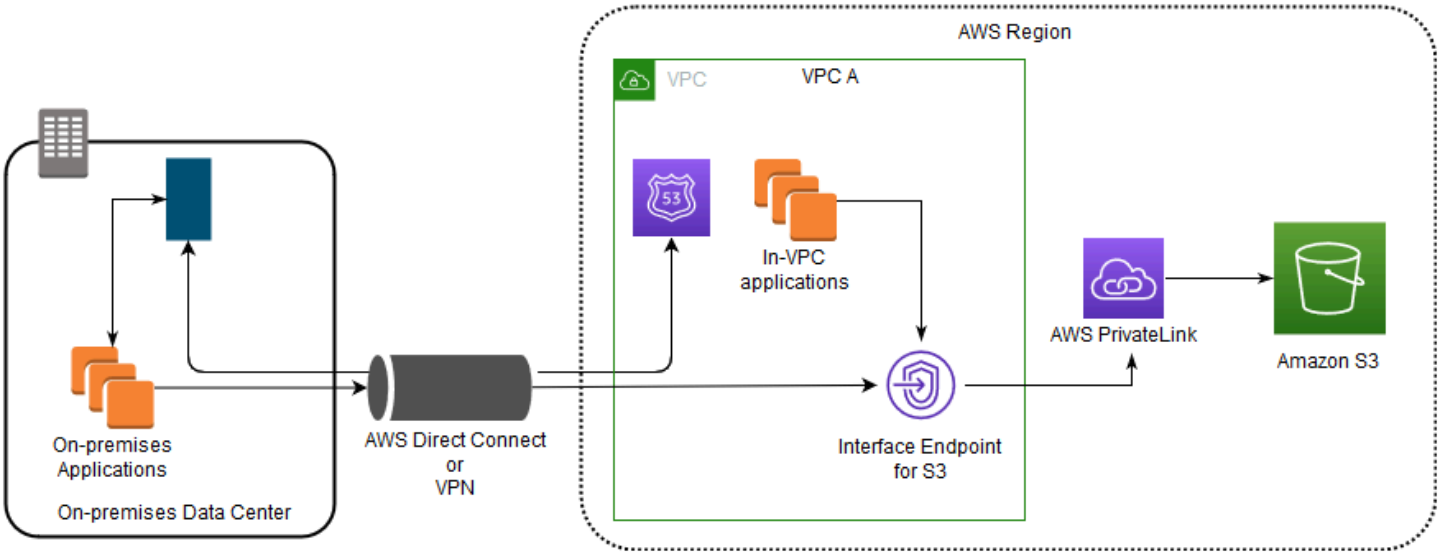
```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                                  .build()
```

## Updating an on-premises DNS configuration

When using endpoint-specific DNS names to access the interface endpoints for Amazon S3, you don't have to update your on-premises DNS resolver. You can resolve the endpoint-specific DNS name with the private IP address of the interface endpoint from the public Amazon S3 DNS domain.

## Using interface endpoints to access Amazon S3 without a gateway endpoint or an internet gateway in the VPC

Interface endpoints in your VPC can route both in-VPC applications and on-premises applications to Amazon S3 over the Amazon network, as illustrated in the following diagram.



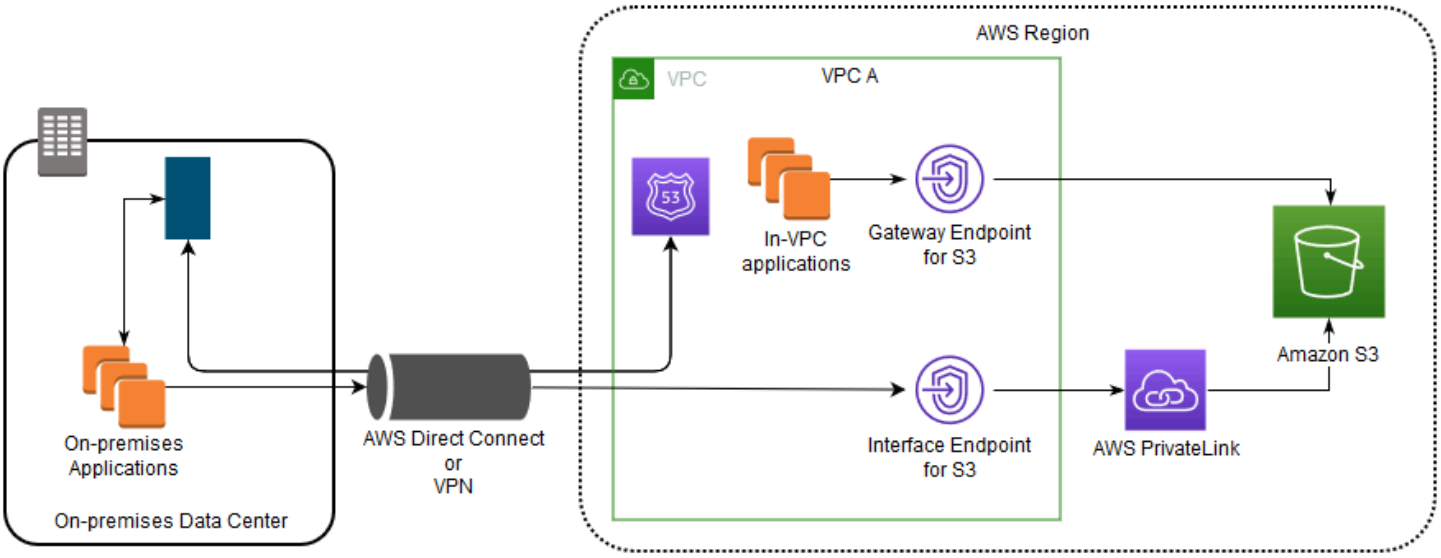

The diagram illustrates the following:

- Your on-premises network uses AWS Direct Connect or AWS VPN to connect to VPC A.
- Your applications on-premises and in VPC A use endpoint-specific DNS names to access Amazon S3 through the S3 interface endpoint.
- On-premises applications send data to the interface endpoint in the VPC through AWS Direct Connect (or AWS VPN). AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.
- In-VPC applications also send traffic to the interface endpoint. AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.

## Using gateway endpoints and interface endpoints together in the same VPC to access Amazon S3

You can create interface endpoints and retain the existing gateway endpoint in the same VPC, as the following diagram shows. By taking this approach, you allow in-VPC applications to continue accessing Amazon S3 through the gateway endpoint, which is not billed. Then, only your on-premises applications would use interface endpoints to access Amazon S3. To access Amazon S3

this way, you must update your on-premises applications to use endpoint-specific DNS names for Amazon S3.




The diagram illustrates the following:

- On-premises applications use endpoint-specific DNS names to send data to the interface endpoint within the VPC through AWS Direct Connect (or AWS VPN). AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.
- Using default Regional Amazon S3 names, in-VPC applications send data to the gateway endpoint that connects to Amazon S3 over the AWS network.

For more information about gateway endpoints, see Gateway VPC endpoints in the *VPC User Guide*.

## Creating a VPC endpoint policy for Amazon S3

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon S3. The policy specifies the following information:

- The AWS Identity and Access Management (IAM) principal that can perform actions
- The actions that can be performed
- The resources on which actions can be performed

You can also use Amazon S3 bucket policies to restrict access to specific buckets from a specific VPC endpoint by using the `aws:sourceVpce` condition in your bucket policy. The following examples show policies that restrict access to a bucket or to an endpoint.

**Topics**



## Example: Restricting access to a specific bucket from a VPC endpoint

You can create an endpoint policy that restricts access to only specific Amazon S3 buckets. This type of policy is useful if you have other AWS services in your VPC that use buckets. The following bucket policy restricts access to only the *amzn-s3-demo-bucket1*. To use this endpoint policy, replace *amzn-s3-demo-bucket1* with the name of your bucket.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket1",
                   "arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }
  ]
}
```

## Example: Restricting access to buckets in a specific account from a VPC endpoint

You can create an endpoint policy that restricts access to only the S3 buckets in a specific AWS account. To prevent clients within your VPC from accessing buckets that you don't own, use the following statement in your endpoint policy. The following example statement creates a policy that restricts access to resources owned by a single AWS account ID, *111122223333*.

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
```

```
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

**Note**

To specify the AWS account ID of the resource being accessed, you can use either the `aws:ResourceAccount` or the `s3:ResourceAccount` key in your IAM policy. However, be aware that some AWS services rely on access to AWS managed buckets. Therefore, using the `aws:ResourceAccount` or `s3:ResourceAccount` key in your IAM policy might also affect access to these resources.

## Example: Restricting access to a specific VPC endpoint in the S3 bucket policy

### Example: Restricting access to a specific VPC endpoint in the S3 bucket policy

The following Amazon S3 bucket policy allows access to a specific bucket, *amzn-s3-demo-bucket2*, from only the VPC endpoint *vpce-1a2b3c4d*. The policy denies all access to the bucket if the specified endpoint is not being used. The `aws:sourceVpce` condition specifies the endpoint and doesn't require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the endpoint ID. To use this bucket policy, replace *amzn-s3-demo-bucket2* and *vpce-1a2b3c4d* with your bucket name and endpoint.

**Important**

- When applying the following Amazon S3 bucket policy to restrict access to only certain VPC endpoints, you might block your access to the bucket without intending to do

so. Bucket policies that are intended to specifically limit bucket access to connections originating from your VPC endpoint can block all connections to the bucket. For information about how to fix this issue, see My bucket policy has the wrong VPC or VPC endpoint ID. How can I fix the policy so that I can access the bucket? in the *AWS Support Knowledge Center*.

- Before using the following example policy, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
- This policy disables *console* access to the specified bucket, because console requests don't originate from the specified VPC endpoint.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket2",
                   "arn:aws:s3:::amzn-s3-demo-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
    }
  ]
}
```

For more policy examples, see Endpoints for Amazon S3 in the *VPC User Guide*.

For more information about VPC connectivity, see Network-to-VPC connectivity options in the AWS whitepaper Amazon Virtual Private Cloud Connectivity Options.

## Access Management

In AWS, a resource is an entity that you can work with. In Amazon Simple Storage Service (S3), *buckets* and *objects* are the original Amazon S3 resources. Every S3 customer likely has buckets with objects in them. As new features were added to S3, additional resources were also added, but not every customer uses these feature-specific resources. For more information about Amazon S3 resources, see S3 resources.

By default, all Amazon S3 resources are private. By default, the root user of the AWS account that created the resource (resource owner) and IAM users within that account with the necessary permissions can access a resource that they created. The resource owner decides who else can access the resource and the actions that others are allowed to perform on the resource. S3 has various access management tools that you can use to grant others access to your S3 resources.

The following sections provide you with an overview of S3 resources, the S3 access management tools available, and the best use cases for each access management tool. The lists in these sections aim to be comprehensive and include all S3 resources, access management tools, and common access management use cases. At the same time, these sections are designed to be directories that lead you to the technical details you want. If you have a good understanding of some of the following topics, you can skip to the section that applies to you.

**Topics**



## S3 resources

The original Amazon S3 resources are buckets and the objects that they contain. As new features are added to S3, new resources are also added. The following is a complete list of S3 resources and their respective features.

| Resource type | Amazon S3 feature | Description |
| --- | --- | --- |
| `bucket` | Core features | A bucket is a container for objects. To store an object in S3, create a bucket and then upload one or more objects to the bucket. For more information, see Creating, configuring, and working with Amazon S3 buckets. |

| Resource type | Amazon S3 feature | Description |
| --- | --- | --- |
| `object` | | An object can be a file and any metadata that describes that file. When an object is in the bucket, you can open it, download it, and move it. For more information, see Uploading, downloading, and working with objects in Amazon S3. |
| `accesspoint` | Access Points | Access Points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 **object** operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions, network controls, and a customized *access point policy* that works in conjunction with the bucket policy that is attached to the underlying bucket. You can configure any access point to accept requests only from a virtual private cloud (VPC) or configure custom block public access settings for each access point. For more information, see Managing data access with Amazon S3 access points. |
| `objectlambdaaccesspoint` | | An Object Lambda Access Point is an access point for a bucket that is also associated with a Lambda function. With Object Lambda Access Point, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it's returned to an application. For more information, see Creating Object Lambda Access Points. |

| Resource type | Amazon S3 feature | Description |
| --- | --- | --- |
| `multiregionaccesspoint` | | Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from Amazon S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, application requests made to a Multi-Region Access Point global endpoint automatically route through the AWS global network to the closest proximity Amazon S3 bucket. For more information, see Multi-Region Access Points in Amazon S3. |
| `job` | S3 Batch Operations | A job is a resource of the S3 Batch Operations feature. You can use S3 Batch Operations to perform large-scale batch operations on lists of Amazon S3 objects that you specify. Amazon S3 tracks the progress of the batch operation job, sends notifications, and stores a detailed completion report of all actions, providing you with a fully managed, auditable, and serverless experience. For more information, see Performing large-scale batch operations on Amazon S3 objects. |
| `storagelensconfiguration` | S3 Storage Lens | An S3 Storage Lens configuration collects organization-wide storage metrics and user data across accounts. S3 Storage Lens provides admins with a single view of object storage usage and activity across hundreds, or even thousands, of accounts in an organization, with details to generate insights at multiple aggregation levels. For more information, see Assessing your storage activity and usage with Amazon S3 Storage Lens. |

| Resource type | Amazon S3 feature | Description |
| --- | --- | --- |
| `storagelensgroup` | | An S3 Storage Lens group aggregates metrics by using custom filters based on object metadata. S3 Storage Lens groups help you investigate characteristics of your data, such as distribution of objects by age, your most common file types, and more. For more information, see Working with S3 Storage Lens groups. |
| `accessgrantsinstance` | S3 Access Grants | An S3 Access Grants instance is a container for the S3 grants that you create. With S3 Access Grants, you can create grants to your Amazon S3 data for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants. |
| `accessgrantslocation` | | An Access Grants Location is a bucket, prefix within a bucket, or an object that you register in your S3 Access Grants instance. You must register locations within the S3 Access Grants instance before you can create a grant to that location. Then, with S3 Access Grants, you can grant access to the bucket, prefix, or object for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants |

| Resource type | Amazon S3 feature | Description |
| --- | --- | --- |
| `accessgrant` | | An Access Grant is an individual grant to your Amazon S3 data. With S3 Access Grants, you can create grants to your Amazon S3 data for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants |

### Buckets

There are two types of Amazon S3 buckets: *general purpose buckets* and *directory buckets*.

- **General purpose buckets** are the original S3 bucket type and are recommended for most use cases and access patterns. General purpose buckets also allow objects that are stored across all storage classes, except S3 Express One Zone. For more information about S3 storage classes, see Using Amazon S3 storage classes.
- **Directory buckets** use the S3 Express One Zone storage class, which is recommended if your application is performance-sensitive and benefits from single-digit millisecond PUT and GET latencies. For more information, see Directory buckets, What is S3 Express One Zone?, and AWS Identity and Access Management (IAM) for S3 Express One Zone.

### Categorizing S3 resources

Amazon S3 provides features to categorize and organize your S3 resources. Categorizing your resources is not only useful for organizing them, but you can also set access management rules based on the resource categories. In particular, prefixes and tagging are two storage organization features that you can use when setting access management permissions.

> **Note**
>
> The following information applies to general purpose buckets. Directory buckets do not support tagging, and they have prefix limitations. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

- **Prefixes** — A prefix in Amazon S3 is a string of characters at the beginning of an object key name that's used to organize the objects that are stored in your S3 buckets. You can use a delimiter character, such as a forward slash (/), to indicate the end of the prefix within the object key name. For example, you might have object key names that start with the `engineering/` prefix or object key names that start with the `marketing/campaigns/` prefix. Using a delimeter at the end of your prefix, such as as a forward slash character / emulates folder and file naming conventions. However, in S3, the prefix is part of the object key name. In general purpose S3 buckets, there is no actual folder hierarchy.

  Amazon S3 supports organizing and grouping objects by using their prefixes. You can also manage access to objects by their prefixes. For example, you can limit access to only the objects with names that start with a specific prefix.

  For more information, see Organizing objects using prefixes. S3 Console uses the concept of *folders*, which, in general purpose buckets, are essentially prefixes that are pre-pended to the object key name. For more information, see Organizing objects in the Amazon S3 console by using folders.

- **Tags** — Each tag is a key-value pair that you assign to resources. For example, you can tag some resources with the tag `topicCategory=engineering`. You can use tagging to help with cost allocation, categorizing and organizing, and access control. Bucket tagging is only used for cost allocation. You can tag objects, S3 Storage Lens, jobs, and S3 Access Grants for the purposes of organizing or for access control. In S3 Access Grants, you can also use tagging for cost-allocation. As an example of controlling access to resources by using their tags, you can share only the objects that have a specific tag or a combination of tags.

  For more information, see Controlling access to AWS resources by using resource tags in the *IAM User Guide*.

## Identities

In Amazon S3, the resource owner is the identity that created the resource, such as a bucket or an object. By default, only the root user of the account that created the resource and IAM identities within the account that have the required permission can access the S3 resource. Resource owners can give other identities access to their S3 resources.

Identities that don't own a resource can request access to that resource. Requests to a resource are either authenticated or unauthenticated. Authenticated requests must include a signature value

that authenticates the request sender, but unauthenticated requests do not require a signature. We recommend that you grant access only to authenticated users. For more information about request authentication, see Making requests.

> **Important**
>
> We recommend that you don't use the AWS account root user credentials to make authenticated requests. Instead, create an IAM role and grant that role full access. We refer to users with this role as *administrator users*. You can use credentials assigned to the administrator role, instead of AWS account root user credentials, to interact with AWS and perform tasks, such as create a bucket, create users, and grant permissions. For more information, see AWS account root user credentials and IAM user credentials in the *AWS General Reference*, and see Security best practices in IAM in the *IAM User Guide*.

Identities accessing your data in Amazon S3 can be one of the following:

**AWS account owner**

The AWS account that created the resource. For example, the account that created the bucket. This account owns the resource. For more information, see AWS account root user.

**IAM identities in the same account of the AWS account owner**

When setting up accounts for new team members who require S3 access, the AWS account owner can use AWS Identity and Access Management (IAM) to create users, groups, and roles. The AWS account owner can then share resources with these IAM identities. The account owner can also specify the permissions to give the IAM identities, which allow or deny the actions that can be performed on the shared resources.

IAM identities provide increased capabilities, including the ability to require users to enter login credentials before accessing shared resources. By using IAM identities, you can implement a form of IAM multi-factor authentication (MFA) to support a strong identity foundation. An IAM best practice is to create roles for access management instead of granting permissions to each individual user. You assign individual users to the appropriate role. For more information, see Security best practices in IAM.

**Other AWS account owners and their IAM identities (cross-account access)**

The AWS account owner can also give other AWS account owners, or IAM identities that belong to another AWS account, access to resources.

> **Note**
>
> **Permission delegation** — If an AWS account owns a resource, it can grant those permissions to another AWS account. That account can then delegate those permissions, or a subset of them, to users in the same account. This is referred to as permission delegation. But an account that receives permissions from another account cannot delegate those permissions "cross-account" to another AWS account.

#### Anonymous users (public access)

The AWS account owner can make resources public. Making a resource public technically shares the resource with *the anonymous user*. Buckets created since April 2023 block all public access by default, unless you change this setting. We recommend that you set your buckets to block public access, and that you only grant access to authenticated users. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

#### AWS services

The resource owner can grant another AWS service access to an Amazon S3 resource. For example, you can grant the AWS CloudTrail service `s3:PutObject` permission to write log files to your bucket. For more information, see Providing access to an AWS service.

#### Corporate directory identities

The resource owner can grant users or roles from your corporate directory access to an S3 resource by using S3 Access Grants. For more information about adding your corporate directory to AWS IAM Identity Center, see What is IAM Identity Center?.

### Bucket or resource owners

The AWS account that you use to create buckets and upload objects owns those resources. A bucket owner can grant cross-account permissions to another AWS account (or users in another account) to upload objects.

When a bucket owner permits another account to upload objects to a bucket, the bucket owner, by default, owns all objects uploaded to their bucket. However, if both the *Bucket owner enforced* and

*Bucket owner preferred* bucket settings are turned off, the AWS account that uploads the objects owns those objects, and the bucket owner does not have permissions on the objects owned by another account, with the following exceptions:

- The bucket owner pays the bills. The bucket owner can deny access to any objects, or delete any objects in the bucket, regardless of who owns them.
- The bucket owner can archive any objects or restore archived objects, regardless of who owns them. Archival refers to the storage class used to store the objects. For more information, see Managing your storage lifecycle.

## Access management tools

Amazon S3 provides a variety of security features and tools. The following is a comprehensive list of these features and tools. You do not need all of these access management tools, but you must use one or more to grant access to your Amazon S3 resources. Proper application of these tools can help make sure that your resources are accessible only to the intended users.

The most commonly used access management tool is an *access policy*. An access policy can be a *resource-based policy* that is attached to an AWS resource, such as a bucket policy for a bucket. An access policy can also be an *identity-based policy* that is attached to an AWS Identity and Access Management (IAM) identity, such as an IAM user, group, or role. Write an access policy to grant AWS accounts and IAM users, groups, and roles permission to perform operations on a resource. For example, you can grant `PUT Object` permission to another AWS account so that the other account can upload objects to your bucket.

An access policy describes who has access to what things. When Amazon S3 receives a request, it must evaluate all of the access policies to determine whether to authorize or deny the request. For more information about how Amazon S3 evaluates these policies, see How Amazon S3 authorizes a request.

The following are the access management tools available in Amazon S3.

**Bucket policy**

An Amazon S3 bucket policy is a JSON-formatted AWS Identity and Access Management (IAM) resource-based policy that is attached to a particular bucket. Use bucket policies to grant other AWS accounts or IAM identities permissions for the bucket and the objects in it. Many S3 access management use cases can be met by using a bucket policy. With bucket policies, you can personalize bucket access to help make sure that only the identities that you have approved can

access resources and perform actions within them. For more information, see Bucket policies for Amazon S3.

The following is an example bucket policy. You express the bucket policy by using a JSON file. This example policy grants an IAM role read permission to all objects in the bucket. It contains one statement named `BucketLevelReadPermissions`, which allows the `s3:GetObject` action (read permission) on objects in a bucket named `amzn-s3-demo-bucket1`. By specifying an IAM role as the `Principal`, this policy grants access to any IAM user with this role. To use this example policy, replace the *`user input placeholders`* with your own information.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }]
  }
```

**Note**

When creating policies, avoid the use of wildcard characters (*) in the `Principal` element because using a wildcard character allows anyone to access your Amazon S3 resources. Instead, explicitly list users or groups that are allowed to access the bucket, or list conditions that must be met by using a condition clause in the policy. Also, rather than including a wildcard character for the actions of your users or groups, grant them specific permissions when applicable.

### Identity-based policy

An identity-based or IAM user policy is a type of AWS Identity and Access Management (IAM) policy. An identity-based policy is a JSON-formatted policy that is attached to IAM users, groups, or roles in your AWS account. You can use identity-based policies to grant an IAM identity access to your buckets or objects. You can create IAM users, groups, and roles in your account and

attach access policies to them. You can then grant access to AWS resources, including Amazon S3 resources. For more information, see Identity-based policies for Amazon S3.

The following is an example of an identity-based policy. The example policy allows the associated IAM role to perform six different Amazon S3 actions (permissions) on a bucket and the objects in it. If you attach this policy to an IAM role in your account and assign the role to some of your IAM users, the users with this role will be able to perform these actions on the resources (buckets) specified in your policy. To use this example policy, replace the *user input placeholders* with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
   "s3:ListBucket",
   "s3:DeleteObject",
   "s3:GetBucketLocation"
    ],
    "Resource": [
   "arn:aws:s3:::amzn-s3-demo-bucket1/*",
   "arn:aws:s3:::amzn-s3-demo-bucket1"
    ]
    },
  {
    "Sid": "AssignARoleActions2",
    "Effect": "Allow",
    "Action": "s3:ListAllMyBuckets",
    "Resource": "*"
  }
  ]
}
```

### S3 Access Grants

Use S3 Access Grants to create access grants to your Amazon S3 data for both identities in corporate identity directories, such as Active Directory, and to AWS Identity and Access Management (IAM) identities. S3 Access Grants helps you manage data permissions at scale.

Additionally, S3 Access Grants logs end-user identity and the application used to access the S3 data in AWS CloudTrail. This provides a detailed audit history down to the end-user identity for all access to the data in your S3 buckets. For more information, see Managing access with S3 Access Grants.

**Access Points**

Amazon S3 Access Points simplifies managing data access at scale for applications that use shared datasets on S3. Access Points are named network endpoints that are attached to a bucket. You can use access points to perform S3 object operations at scale, such as uploading and retrieving objects. A bucket can have up to 10,000 access points attached, and for each access point, you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. S3 Access Points can be associated with buckets in the same account or in another trusted account. Access Points policies are resource-based policies that are evaluated in conjunction with the underlying bucket policy. For more information, see Managing data access with Amazon S3 access points.

**Access control list (ACL)**

An ACL is a list of grants identifying the grantee and the permission granted. ACLs grant basic read or write permissions to other AWS accounts. ACLs use an Amazon S3–specific XML schema. An ACL is a type of AWS Identity and Access Management (IAM) policy. An object ACL is used to manage access to an object, and a bucket ACL is used to manage access to a bucket. With bucket policies, there is a single policy for the entire bucket, but object ACLs are specified for each object. We recommend that you keep ACLs turned off, except in unusual circumstances where you must individually control access for each object. For more information about using ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Warning**
>
> The majority of modern use cases in Amazon S3 do not require the use of ACLs.

The following is an example bucket ACL. The grant in the ACL shows a bucket owner that has full control permission.

```
<?xml version="1.0" encoding="UTF-8"?>
 <AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
```

```
   <ID>Owner-Canonical-User-ID</ID>
   <DisplayName>owner-display-name</DisplayName>
  </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Canonical
 User">
    <ID>Owner-Canonical-User-ID</ID>
    <DisplayName>display-name</DisplayName>
   </Grantee>
   <Permission>FULL_CONTROL</Permission>
  </Grant>
 </AccessControlList>
</AccessControlPolicy>
```

### Object Ownership

To manage access to your objects, you must be the owner of the object. You can use the Object Ownership bucket-level setting to control ownership of objects uploaded to your bucket. Also, use Object Ownership to turn on ACLs. By default, Object Ownership is set to the *Bucket owner enforced setting* and all ACLs are turned off. When ACLs are turned off, the bucket owner owns all of the objects in the bucket and exclusively manages access to data. To manage access, the bucket owner uses policies or another access management tool, excluding ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Object Ownership has three settings that you can use both to control ownership of objects that are uploaded to your bucket and to turn on ACLs:

#### ACLs turned off

- **Bucket owner enforced (default)** – ACLs are turned off, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs do not affect permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

#### ACLs turned on

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

### Additional best practices

Consider using the following bucket settings and tools to help protect data in transit and at rest, both of which are crucial in maintaining the integrity and accessibility of your data:

- **Block Public Access** — Do not turn off the default bucket-level setting *Block Public Access*. This setting blocks public access to your data by default. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
- **S3 Versioning** — For data integrity, you can implement the S3 Versioning bucket setting, which versions your objects as you make updates, instead of overwriting them. You can use S3 Versioning to preserve, retrieve, and restore a previous version, if needed. For information about S3 Versioning, see Using versioning in S3 buckets.
- **S3 Object Lock** — S3 Object Lock is another setting that you can implement for achieving data integrity. This feature can implement a write-once-read-many (WORM) model to store objects immutably. For information about Object Lock, see Using S3 Object Lock.
- **Object encryption** — Amazon S3 offers several object encryption options that protect data in transit and at rest. *Server-side encryption* encrypts your object before saving it on disks in its data centers and then decrypts it when you download the objects. If you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects. For more information, see Protecting data with server-side encryption. S3 encrypts newly uploaded objects by default. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets. *Client-side encryption* is the act of encrypting data before sending it to Amazon S3. For more information, see Protecting data by using client-side encryption.
- **Signing methods** — Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see Authenticating Requests (AWS Signature Version 4) and Signature Version 4 signing process.

## Actions

For a complete list of S3 permissions and condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

### Actions

The AWS Identity and Access Management (IAM) actions for Amazon S3 are the possible actions that can be performed on an S3 bucket or object. You grant these actions to identities so they can act on your S3 resources. Examples of S3 actions are `s3:GetObject` to read objects in a bucket, and `s3:PutObject` to write objects to a bucket.

**Condition keys**

In addition to actions, IAM condition keys are limited to granting access to only when a condition is met. Condition keys are optional.

> **Note**
>
> In a resource-based access policy, such as a bucket policy, or in an identity-based policy, you can specify the following:
>
> - An action or an array of actions in the `Action` element of the policy statement.
> - In the `Effect` element of the policy statement, you can specify `Allow` to grant the actions listed, or you can specify `Deny` to block the listed actions. To further maintain the practice of least privileges, `Deny` statements in the `Effect` element of the access policy should be as broad as possible, and `Allow` statements should be as narrow as possible. `Deny` effects paired with the `s3:*` action are another good way to implement opt-in best practices for the identities that are included in policy condition statements.
> - A condition key in the `Condition` element of a policy statement.

## Access management use cases

Amazon S3 provides resource owners with a variety of tools for granting access. The S3 access management tool that you use depends on the S3 resources that you want to share, the identities that you are granting access to, and the actions that you want to allow or deny. You might want to use one or a combination of S3 access management tools to manage access to your S3 resources.

In most cases, you can use an access policy to manage permissions. An access policy can be a resource-based policy, which is attached to a resource, such as a bucket, or another Amazon S3 resource (S3 resources). An access policy can also be an identity-based policy, which is attached to an AWS Identity and Access Management (IAM) user, group, or role in your account. You might find that a bucket policy works better for your use case. For more information, see Bucket policies for

Amazon S3. Alternatively, with AWS Identity and Access Management (IAM), you can create IAM users, groups, and roles within your AWS account and manage their access to buckets and objects through identity-based policies. For more information, see Identity-based policies for Amazon S3.

To help you navigate these access management options, the following are common Amazon S3 customer use cases and recommendations for each of the S3 access management tools.

**The AWS account owner wants to share buckets only with users within the same account**

All access management tools can fulfill this basic use case. We recommend the following access management tools for this use case:

- **Bucket policy** – If you want to grant access to one bucket or a small number of buckets, or if your bucket access permissions are similar from bucket to bucket, use a bucket policy. With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.
- **Identity-based policy** – If you have a very large number of buckets with different access permissions for each bucket, and only a few user roles to manage, you can use an IAM policy for users, groups, or roles. IAM policies are also a good option if you are managing user access to other AWS resources, as well as Amazon S3 resources. For more information, see Example 1: Bucket owner granting its users bucket permissions.
- **S3 Access Grants** – You can use S3 Access Grants to grant access to your S3 buckets, prefixes, or objects. S3 Access Grants allows you to specify varying object-level permissions at scale; whereas, bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use Access Points, which are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing data access with Amazon S3 access points.

**The AWS account owner wants to share buckets or objects with users from another AWS account (cross-account)**

To grant permission to another AWS account, you must use a bucket policy or one of the following recommended access management tools. You cannot use an identity-based access policy for this use case. For more information about granting cross-account access, see How do I provide cross-account access to objects that are in Amazon S3 buckets?

We recommend the following access management tools for this use case:

- **Bucket policy** – With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.
- **S3 Access Grants** – You can use S3 Access Grants to grant cross-account permissions to your S3 buckets, prefixes, or objects. You can use S3 Access Grants to specify varying object-level permissions at scale; whereas, bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use Access Points, which are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing data access with Amazon S3 access points.

**The AWS account owner or bucket owner must grant permissions at the object-level or prefix-level, and these permissions vary from object to object or prefix to prefix**

In a bucket policy, for example, you can grant access to the objects within a bucket that share a specific key name prefix or have a specific tag. You can grant read permission on objects starting with the key name prefix `logs/`. However, if your access permissions vary by object, granting permissions to individual objects by using a bucket policy might not be practical, especially since bucket policies are limited to 20 KB in size.

We recommend the following access management tools for this use case:

- **S3 Access Grants** – You can use S3 Access Grants to manage object-level or prefix-level permissions. Unlike bucket policies, you can use S3 Access Grants to specify varying object-level permissions at scale. Bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use access points to manage object-level or prefix-level permissions. Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing data access with Amazon S3 access points.
- **ACLs** – We do not recommend using Access Control Lists (ACLs), especially because ACLs are limited to 100 grants per object. However, if you choose to turn on ACLs, in your Bucket Settings, set *Object Ownership* to *Bucket owner preferred* and *ACLs enabled*. With this setting, new objects

that are written with the `bucket-owner-full-control` canned ACL are automatically owned by the bucket owner rather than the object writer. You can then use object ACLs, which is an XML-formatted access policy, to grant other users access to the object. For more information, see Access control list (ACL) overview.

**The AWS account owner or bucket owner wants to limit bucket access only to specific account IDs**

We recommend the following access management tools for this use case:

- **Bucket policy** – With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.
- **Access Points** – Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing data access with Amazon S3 access points.

**The AWS account owner or bucket owner wants distinct endpoints for every user or application that accesses their data**

We recommend the following access management tool for this use case:

- **Access Points** – Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Managing data access with Amazon S3 access points.

**The AWS account owner or bucket owner must manage access from Virtual Private Cloud (VPC) endpoints for S3**

Virtual Private Cloud (VPC) endpoints for Amazon S3 are logical entities within a VPC that allow connectivity only to S3. We recommend the following access management tools for this use case:

- **Buckets in a VPC setting** – You can use a bucket policy to control who is allowed to access your buckets and which VPC endpoints they can access. For more information, see Controlling access from VPC endpoints with bucket policies.

- **Access Points** – If you choose to set up access points, you can use an access point policy. You can configure any access point to accept requests only from a virtual private cloud (VPC) to restrict Amazon S3 data access to a private network. You can also configure custom block public access settings for each access point. For more information, see Managing data access with Amazon S3 access points.

**The AWS account owner or bucket owner must make a static website publicly available**

With S3, you can host a static website and allow anyone to view the content of the website, which is hosted from an S3 bucket.

We recommend the following access management tools for this use case:

- **Amazon CloudFront** – This solution allows you to host an Amazon S3 static website to the public while also continuing to block all public access to a bucket's content. If you want to keep all four S3 Block Public Access settings enabled and host an S3 static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Also, Amazon S3 static websites that do not use this solution can only support HTTP endpoints. CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS. HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks.

  For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.
- **Making your Amazon S3 bucket publicly accessible** – You can configure a bucket to be used as a publicly accessed static website.

  > **Warning**
  >
  > We do not recommend this method. Instead, we recommend you use Amazon S3 static websites as a part of Amazon CloudFront. For more information, see the previous option, or see Getting started with a secure static website.

  To create an Amazon S3 static website, without Amazon CloudFront, first, you must turn off all Block Public Access settings. When writing the bucket policy for your static website, make sure that you allow only `s3:GetObject` actions, not `ListObject` or `PutObject` permissions. This

helps make sure that users cannot view all the objects in your bucket or add their own content. For more information, see Setting permissions for website access.

### The AWS account owner or bucket owner wants to make the content of a bucket publicly available

When creating a new Amazon S3 bucket, the *Block Public Access* setting is enabled by default. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

We do not recommend allowing public access to your bucket. However, if you must do so for a particular use case, we recommend the following access management tool for this use case:

- **Disable Block Public Access setting** – A bucket owner can allow unauthenticated requests to the bucket. For example, unauthenticated PUT Object requests are allowed when a bucket has a public bucket policy, or when a bucket ACL grants public access. All unauthenticated requests are made by other arbitrary AWS users, or even unauthenticated, anonymous users. This user is represented in ACLs by the specific canonical user ID `65a011a29cdf8ec533ec3d1ccaae921c`. If an object is uploaded to a `WRITE` or `FULL_CONTROL`, then this specifically grants access to the All Users group or the anonymous user. For more information about public bucket policies and public access control lists (ACLs), see The meaning of "public".

### The AWS account owner or bucket owner has exceeded access policy size limits

Both bucket policies and identity-based policies have a 20 KB size limit. If your access permission requirements are complex, you might exceed this size limit.

We recommended the following access management tools for this use case:

- **Access Points** – Use access points if this works with your use case. With access points, each bucket has multiple named network endpoints, each with its own access point policy that works with the underlying bucket policy. However, access points can only act on objects, not buckets, and does not support cross-Region replication. For more information, see Managing data access with Amazon S3 access points.
- **S3 Access Grants** – Use S3 Access Grants, which supports a very large number of grants that give access to buckets, prefixes, or objects. For more information, see Getting started with S3 Access Grants.

**The AWS account owner or admin role wants to grant bucket, prefix, or object access directly to users or groups in a corporate directory**

Instead of managing users, groups, and roles through AWS Identity and Access Management (IAM), you can add your corporate directory to AWS IAM Identity Center. For more information, see What is IAM Identity Center?.

After you add your corporate directory to AWS IAM Identity Center, we recommend that you use the following access management tool to grant corporate directory identities access to your S3 resources:

- **S3 Access Grants** – Use S3 Access Grants, which supports granting access to users or roles in your corporate directory. For more information, see Getting started with S3 Access Grants.

**The AWS account owner or bucket owner wants to give the AWS CloudFront service access to write CloudFront logs to an S3 bucket**

We recommended the following access management tool for this use case:

- **Bucket ACL** – The only recommended use case for bucket ACLs is to grant permissions to certain AWS services, such as the Amazon CloudFront `awslogsdelivery` account. When you create or update a distribution and turn on CloudFront logging, CloudFront updates the bucket ACL to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the *Bucket owner enforced* setting for S3 Object Ownership to turn off ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**You, as the bucket owner, want to maintain full control of objects that are added to the bucket by other users**

You can grant other accounts access to upload objects to your bucket by using a bucket policy, access point, or S3 Access Grants. If you have granted cross-account access to your bucket, you can make sure that any objects uploaded to your bucket remain under your full control.

We recommended the following access management tool for this use case:

- **Object Ownership** – Keep the bucket-level setting *Object Ownership* at the default *Bucket owner enforced* setting.

## Access management troubleshooting

The following resources can help you troubleshoot any issues with S3 access management:

**Troubleshooting Access Denied (403 Forbidden) errors**

If you encounter access denial issues, check the account-level and bucket-level settings. Also, check the access management feature that you are using to grant access to make sure that the policy, setting, or configuration is correct. For more information about common causes of Access Denied (403 Forbidden) errors in Amazon S3, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

**IAM Access Analyzer for S3**

If you do not want to make any of your resources publicly available, or if you want to limit public access to your resources, you can use IAM Access Analyzer for S3. On the Amazon S3 console, use IAM Access Analyzer for S3 to review all buckets that have bucket access control lists (ACLs), bucket policies, or access point policies that grant public or shared access. IAM Access Analyzer for S3 alerts you to buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings that report the source and level of public or shared access.

In IAM Access Analyzer for S3, you can block all public access to a bucket with a single action. We recommend that you block all public access to your buckets, unless you require public access to support a specific use case. Before you block all public access, make sure that your applications will continue to work correctly without public access. For more information, see Blocking public access to your Amazon S3 storage.

You can also review your bucket-level permission settings to configure detailed levels of access. For specific and verified use cases that require public or shared access, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket. You can revisit and modify these bucket configurations at any time. You can also download your findings as a CSV report for auditing purposes.

IAM Access Analyzer for S3 is available at no extra cost on the Amazon S3 console. IAM Access Analyzer for S3 is powered by AWS Identity and Access Management (IAM) IAM Access Analyzer.

To use IAM Access Analyzer for S3 on the Amazon S3 console, you must visit the IAM Console and create an account-level analyzer in IAM Access Analyzer for each individual Region.

For more information about IAM Access Analyzer for S3, see Reviewing bucket access using IAM Access Analyzer for S3.

**Logging and monitoring**

Monitoring is an important part of maintaining the reliability, availability, and performance of your Amazon S3 solutions so that you can more easily debug an access failure. Logging can provide insight into any errors users are receiving, and when and what requests are made. AWS provides several tools for monitoring your Amazon S3 resources, such as the following:

- AWS CloudTrail
- Amazon S3 Access Logs
- AWS Trusted Advisor
- Amazon CloudWatch

For more information, see Logging and monitoring in Amazon S3.

**Topics**



## Identity and Access Management for Amazon S3

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon S3 resources. IAM is an AWS service that you can use with no additional charge.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon S3.

**Service user** – If you use the Amazon S3 service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon S3 features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon S3, see Troubleshooting Amazon S3 identity and access.

**Service administrator** – If you're in charge of Amazon S3 resources at your company, you probably have full access to Amazon S3. It's your job to determine which Amazon S3 features and resources your service users should access. You must then submit requests to your IAM administrator to

change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon S3, see How Amazon S3 works with IAM.

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon S3. To view example Amazon S3 identity-based policies that you can use in IAM, see Identity-based policies for Amazon S3.

## Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

### AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and

is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

**Federated identity**

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

**IAM users and groups**

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

### IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

## Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A

user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

**Identity-based policies**

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

**Resource-based policies**

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

**Access control lists (ACLs)**

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

### Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

### Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

## How Amazon S3 works with IAM

Before you use IAM to manage access to Amazon S3, learn what IAM features are available to use with Amazon S3.

**IAM features you can use with Amazon S3**

| IAM feature | Amazon S3 support |
| --- | --- |
| Identity-based policies | Yes |
| Resource-based policies | Yes |
| Policy actions | Yes |
| Policy resources | Yes |
| Policy condition keys (service-specific) | Yes |
| ACLs | Yes |
| ABAC (tags in policies) | Partial |
| Temporary credentials | Yes |
| Forward access sessions (FAS) | Yes |
| Service roles | Yes |
| Service-linked roles | Partial |

To get a high-level view of how Amazon S3 and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

**Identity-based policies for Amazon S3**

**Supports identity-based policies:** Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all

of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

**Identity-based policy examples for Amazon S3**

To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.

**Resource-based policies within Amazon S3**

**Supports resource-based policies:** Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

The Amazon S3 service supports *bucket policies*, *access points policies*, and *access grants*:

- Bucket policies are resource-based policies that are attached to an Amazon S3 bucket. A bucket policy defines which principals can perform actions on the bucket.
- Access point policies are resource-based polices that are evaluated in conjunction with the underlying bucket policy.
- Access grants are a simplified model for defining access permissions to data in Amazon S3 by prefix, bucket, or object. For information about S3 Access Grants, see Managing access with S3 Access Grants.

### Principals for bucket policies

The `Principal` element specifies the user, account, service, or other entity that is either allowed or denied access to a resource. The following are examples of specifying `Principal`. For more information, see Principal in the *IAM User Guide*.

#### Grant permissions to an AWS account

To grant permissions to an AWS account, identify the account using the following format.

```
"AWS":"account-ARN"
```

The following are examples.

```
"Principal":{"AWS":"arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

#### Grant permissions to an IAM user

To grant permission to an IAM user within your account, you must provide an `"AWS":"user-ARN"` name-value pair.

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

For detailed examples that provide step-by-step instructions, see Example 1: Bucket owner granting its users bucket permissions and Example 3: Bucket owner granting permissions to objects it does not own.

> **Note**
>
> If an IAM identity is deleted after you update your bucket policy, the bucket policy will show a unique identifier in the principal element instead of an ARN. These unique IDs are never reused, so you can safely remove principals with unique identifiers from all of your policy statements. For more information about unique identifiers, see IAM identifiers in the *IAM User Guide*.

### Grant anonymous permissions

> **Warning**
>
> Use caution when granting anonymous access to your Amazon S3 bucket. When you grant anonymous access, anyone in the world can access your bucket. We highly recommend that you never grant any kind of anonymous write access to your S3 bucket.

To grant permission to everyone, also referred as anonymous access, you set the wildcard (`"*"`) as the `Principal` value. For example, if you configure your bucket as a website, you want all the objects in the bucket to be publicly accessible.

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

Using `"Principal": "*"` with an `Allow` effect in a resource-based policy allows anyone, even if they're not signed in to AWS, to access your resource.

Using `"Principal" : { "AWS" : "*" }` with an `Allow` effect in a resource-based policy allows any root user, IAM user, assumed-role session, or federated user in any account in the same partition to access your resource.

For anonymous users, these two methods are equivalent. For more information, see All principals in the *IAM User Guide*.

You cannot use a wildcard to match part of a principal name or ARN.

> **Important**
>
> Because anyone can create an AWS account, the **security level** of these two methods is equivalent, even though they function differently.

### Restrict resource permissions

You can also use resource policy to restrict access to resources that would otherwise be available to IAM principals. Use a `Deny` statement to prevent access.

The following example blocks access if a secure transport protocol isn't used:

```
{"Effect": "Deny",
 "Principal": "*",
 "Action": "s3:*",
 "Resource": <bucket ARN>,
 "Condition": {
   "Boolean": { "aws:SecureTransport" : "false"}
 }
}
```

Using `"Principal": "*"` so that this restriction applies to everyone is a best practice for this policy, instead of attempting to deny access only to specific accounts or principals using this method.

**Require access through CloudFront URLs**

You can require that your users access your Amazon S3 content only by using CloudFront URLs instead of Amazon S3 URLs. To do this, create a CloudFront origin access control (OAC). Then, change the permissions on your S3 data. In your bucket policy, you can set CloudFront as the Principal as follows:

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

Use a `Condition` element in the policy to allow CloudFront to access the bucket only when the request is on behalf of the CloudFront distribution that contains the S3 origin.

```
       "Condition": {
          "StringEquals": {
             "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
          }
       }
```

For more information about requiring S3 access through CloudFront URLs, see Restricting access to an Amazon Simple Storage Service origin in the *Amazon CloudFront Developer Guide*. For more information about the security and privacy benefits of using Amazon CloudFront, see Configuring secure access and restricting access to content.

### Resource-based policy examples for Amazon S3

- To view policy examples for Amazon S3 buckets, see Bucket policies for Amazon S3.
- To view policy examples for access points, see Configuring IAM policies for using access points.

### Policy actions for Amazon S3

**Supports policy actions:** Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

The following shows different types of mapping relationship between S3 API operations and the required policy actions.

- One-to-one mapping with the same name. For example, to use the `PutBucketPolicy` API operation, the `s3:PutBucketPolicy` policy action is required.
- One-to-one mapping with different names. For example, to use the `ListObjectsV2` API operation, the `s3:ListBucket` policy action is required.
- One-to-many mapping. For example, to use the `HeadObject` API operation, the `s3:GetObject` is required. Also, when you use S3 Object Lock and want to get an object's Legal Hold status or retention settings, the corresponding `s3:GetObjectLegalHold` or `s3:GetObjectRetention` policy actions are also required before you can use the `HeadObject` API operation.
- Many-to-one mapping. For example, to use the `ListObjectsV2` or `HeadBucket` API operations, the `s3:ListBucket` policy action is required.

To see a list of Amazon S3 actions for use in policies, see Actions defined by Amazon S3 in the *Service Authorization Reference*. For a complete list of Amazon S3 API operations, see Amazon S3 API Actions in the *Amazon Simple Storage Service API Reference*.

Policy actions in Amazon S3 use the following prefix before the action:

```
s3
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

### Bucket operations

Bucket operations are S3 API operations that operate on the bucket resource type. For example, `CreateBucket`, `ListObjectsV2`, and `PutBucketPolicy`. S3 policy actions for bucket operations require the `Resource` element in bucket policies or IAM identity-based policies to be the S3 bucket type Amazon Resource Name (ARN) identifier in the following example format.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
```

The following bucket policy grants the user *Akua* with account *12345678901* the `s3:ListBucket` permission to perform the ListObjectsV2 API operation and list objects in an S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        }
    ]
```

```
}
```

### Bucket operations in access point policies

Permissions granted in an access point policy are effective only if the underlying bucket allows the same permissions. When you use S3 Access Points, you must delegate access control from the bucket to the access point or add the same permissions in the access point policies to the underlying bucket's policy. For more information, see Configuring IAM policies for using access points. In access point policies, S3 policy actions for bucket operations require you to use the `accesspoint` ARN for the `Resource` element in the following format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:ListBucket` permission to perform the ListObjectsV2 API operation through the S3 access point *DOC-EXAMPLE-ACCESS-POINT* to list objects in the access point's associated bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-
ACCESS-POINT"
        }
    ]
}
```

> **Note**
>
> Not all bucket operations are supported by S3 Access Point. For more information, see Access point compatibility with S3 operations.

### Object operations

Object operations are S3 API operations that act upon the object resource type. For example, `GetObject`, `PutObject`, and `DeleteObject`. S3 policy actions for object operations require the `Resource` element in policies to be the S3 object ARN in the following example formats.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/*"
```

> **Note**
>
> The object ARN must contain a forward slash after the bucket name, as seen in the previous examples.

The following bucket policy grants the user *Akua* with account *12345678901* the `s3:PutObject` permission to perform the PutObject API operation to upload objects to an S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

### Object operations in access point policies

When you use S3 Access Points to control access to object operations, you can use access point policies. When you use access point policies, S3 policy actions for object operations

require you to use the `accesspoint` ARN for the `Resource` element in the following format: `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`. For object operations that use access point, you must include the `/object/` value after the whole access point ARN in the `Resource` element. Here are some examples.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/
object/*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/
object/prefix/*"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:GetObject` permission to perform the GetObject API operation through the access point *DOC-EXAMPLE-ACCESS-POINT* on all objects in the access point's associated bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-
ACCESS-POINT/object/*"
        }
    ]
}
```

**Note**

Not all object operations are supported by S3 Access Point. For more information, see Access point compatibility with S3 operations.

### Access point operations

Access point operations are S3 API operations that operate on the `accesspoint` resource type. For example, `CreateAccessPoint`, `DeleteAccessPoint`, and `GetAccessPointPolicy`. S3 policy actions for access point operations can only be used in IAM identity-based policies, not in bucket policies or access point policies. Access points operations require the `Resource` element to be the `accesspoint` ARN in the following example format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

The following IAM identity-based policy grants the `s3:GetAccessPointPolicy` permission to perform the GetAccessPointPolicy API operation on S3 access point *DOC-EXAMPLE-ACCESS-POINT*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point DOC-EXAMPLE-ACCESS-POINT",
            "Effect": "Allow",
            "Action": [
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:access point/DOC-EXAMPLE-ACCESS-
POINT"
        }
    ]
}
```

When you use Access Points, to control access to bucket operations, see Bucket operations in access point policies; to control access to object operations, see Object operations in access point policies. For more information about how to configure access point policies, see Configuring IAM policies for using access points.

### Object Lambda Access Point operations

With Amazon S3 Object Lambda, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it is returned to an application. You can make requests through an Object Lambda Access Point, which works the same as making requests through other access points. For more information, see Transforming objects with S3 Object Lambda.

For more information about how to configure policies for Object Lambda Access Point operations, see Configuring IAM policies for Object Lambda Access Points.

**Multi-Region Access Point operations**

A Multi-Region Access Point provides a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Region. You can use a Multi-Region Access Point to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. For more information, see Multi-Region Access Points in Amazon S3.

For more information about how to configure policies for Multi-Region Access Point operations, see Multi-Region Access Point policy examples.

**Batch job operations**

(Batch Operations) job operations are S3 API operations that operate on the job resource type. For example, `DescribeJob` and `CreateJob`. S3 policy actions for job operations can only be used in IAM identity-based policies, not in bucket policies. Also, job operations require the `Resource` element in IAM identity-based policies to be the `job` ARN in the following example format.

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

The following IAM identity-based policy grants the `s3:DescribeJob` permission to perform the DescribeJob API operation on S3 Batch Operations Job *DOC-EXAMPLE-JOB*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow describing the Batch operation job DOC-EXAMPLE-JOB",
            "Effect": "Allow",
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:123456789012:job/DOC-EXAMPLE-JOB"
        }
    ]
}
```

### S3 Storage Lens configuration operations

For more information about how to configure S3 Storage Lens configuration operations, see Amazon S3 Storage Lens permissions.

### Account operations

Account operations are S3 API operations that operate on the account level. For example, `GetPublicAccessBlock` (for account). Account isn't a resource type defined by Amazon S3. S3 policy actions for account operations can only be used in IAM identity-based policies, not in bucket policies. Also, account operations require the `Resource` element in IAM identity-based policies to be `"*"`.

The following IAM identity-based policy grants the `s3:GetAccountPublicAccessBlock` permission to perform the account-level GetPublicAccessBlock API operation and retrieve the account-level Public Access Block settings.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Allow retrieving the account-level Public Access Block settings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

### Policy examples for Amazon S3

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

### Policy resources for Amazon S3

**Supports policy resources:** Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

Some Amazon S3 API actions support multiple resources. For example, `s3:GetObject` accesses EXAMPLE-RESOURCE-1 and EXAMPLE-RESOURCE-2, so a principal must have permissions to access both resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Resources in Amazon S3 are buckets, objects, access points, or jobs. In a policy, use the Amazon Resource Name (ARN) of the bucket, object, access point, or job to identify the resource.

To see a complete list of Amazon S3 resource types and their ARNs, see Resources defined by Amazon S3 in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon S3.

### Wildcards for resource ARNs

You can use wildcards as part of the resource ARN. You can use wildcard characters (* and ?) within any ARN segment (the parts separated by colons). An asterisk (*) represents any combination of zero or more characters, and a question mark (?) represents any single character. You can use multiple * or ? characters in each segment, but a wildcard cannot span segments.

- The following ARN uses the wildcard * in the relative-ID part of the ARN to identify all objects in the examplebucket bucket.

```
arn:aws:s3:::examplebucket/*
```

- The following ARN uses * to indicate all S3 buckets and objects.

```
arn:aws:s3:::*
```

- The following ARN uses both wildcards, * and ?, in the relative-ID part. It identifies all objects in buckets such as example1bucket, example2bucket, example3bucket, and so on.

```
arn:aws:s3:::example?bucket/*
```

### Policy variables for resource ARNs

You can use policy variables in Amazon S3 ARNs. At policy evaluation time, these predefined variables are replaced by their corresponding values. Suppose that you organize your bucket as a collection of folders, one folder for each of your users. The folder name is the same as the user name. To grant users permission to their folders, you can specify a policy variable in the resource ARN:

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

At runtime, when the policy is evaluated, the variable ${aws:username} in the resource ARN is substituted with the username of the person who is making the request.

### Policy examples for Amazon S3

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

### Policy condition keys for Amazon S3

**Supports service-specific policy condition keys:** Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

Each Amazon S3 condition key maps to the same name request header allowed by the API on which the condition can be set. Amazon S3-specific condition keys dictate the behavior of the same name request headers. For example, the condition key `s3:VersionId` used to grant conditional permission for the
`s3:GetObjectVersion`
permission defines behavior of the `versionId` query parameter that you set in a GET Object request.

To see a list of Amazon S3 condition keys, see Condition keys for Amazon S3 in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon S3.

**Example: Restricting object uploads to objects with a specific storage class**

Suppose that Account A, represented by account ID 123456789012, owns a bucket. The Account A administrator wants to restrict Dave, a user in Account A, so that Dave can only upload objects to the bucket that's stored with the `STANDARD_IA` storage class. To restrict object uploads to a specific storage class, the Account A administrator can use the `s3:x-amz-storage-class` condition key, as shown in the following example bucket policy.

```
{
                "Version": "2012-10-17",
                "Statement": [
                  {
                    "Sid": "statement1",
                    "Effect": "Allow",
                    "Principal": {
                      "AWS": "arn:aws:iam::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
                ]
            }
```

In the example, the `Condition` block specifies the `StringEquals` condition that is applied to the specified key-value pair, `"s3:x-amz-acl":["public-read"]`. There is a set of predefined keys that you can use in expressing a condition. The example uses the `s3:x-amz-acl` condition key. This condition requires the user to include the `x-amz-acl` header with value `public-read` in every PUT object request.

**Policy examples for Amazon S3**

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

**ACLs in Amazon S3**

**Supports ACLs:** Yes

In Amazon S3, access control lists (ACLs) control which AWS accounts have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

> **Important**
>
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs.

For information about using ACLs to control access in Amazon S3, see Managing access with ACLs.

**ABAC with Amazon S3**

**Supports ABAC (tags in policies):** Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

To view example identity-based policies for limiting access to S3 Batch Operations jobs based on tags, see Controlling permissions for S3 Batch Operations using job tags.

**ABAC and object tags**

In ABAC policies, objects use `s3:` tags instead of `aws:` tags. To control access to objects based on object tags, you provide tag information in the condition element of a policy using the following tags:

- `s3:ExistingObjectTag/tag-key`
- `s3:s3:RequestObjectTagKeys`
- `s3:RequestObjectTag/tag-key`

For information about using object tags to control access, including example permission policies, see Tagging and access control policies.

**Using temporary credentials with Amazon S3**

**Supports temporary credentials:** Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

**Forward access sessions for Amazon S3**

**Supports forward access sessions (FAS):** Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- FAS is used by Amazon S3 to make calls to AWS KMS to decrypt an object when SSE-KMS was used to encrypt it. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).
- S3 Access Grants also uses FAS. After you create an access grant to your S3 data for a particular identity, the grantee requests a temporary credential from S3 Access Grants. S3 Access Grants obtains a temporary credential for the requester from AWS STS and vends the credential to the requester. For more information, see Request access to Amazon S3 data through S3 Access Grants.

**Service roles for Amazon S3**

**Supports service roles:** Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon S3 functionality. Edit service roles only when Amazon S3 provides guidance to do so.

**Service-linked roles for Amazon S3**

**Supports service-linked roles:** Partial

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Amazon S3 supports service-linked roles for Amazon S3 Storage Lens. For details about creating or managing Amazon S3 service-linked roles, see Using service-linked roles for Amazon S3 Storage Lens.

**Amazon S3 Service as a Principal**

| Service name in the policy | S3 feature | More information |
|---|---|---|
| `s3.amazonaws.com` | S3 Replication | Setting up live replication |
| `s3.amazonaws.com` | S3 event notifications | Amazon S3 Event Notifications |
| `s3.amazonaws.com` | S3 Inventory | Amazon S3 Inventory |
| `access-grants.s3.amazonaws.com` | S3 Access Grants | Register a location |
| `batchoperations.s3.amazonaws.com` | S3 Batch Operations | Granting permissions for Amazon S3 Batch Operations |
| `logging.s3.amazonaws.com` | S3 Server Access Logging | Enabling Amazon S3 server access logging |
| `storage-lens.s3.amazonaws.com` | S3 Storage Lens | Viewing Amazon S3 Storage Lens metrics using a data export |

## Policies and permissions in Amazon S3

This page provides an overview of bucket and user policies in Amazon S3 and describes the basic elements of an AWS Identity and Access Management (IAM) policy. Each listed element links to more details about that element and examples of how to use it.

For a complete list of Amazon S3 actions, resources, and conditions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

In its most basic sense, a policy contains the following elements:

- Resource – The Amazon S3 bucket, object, access point, or job that the policy applies to. Use the Amazon Resource Name (ARN) of the bucket, object, access point, or job to identify the resource.

  An example for bucket-level operations:

  `"Resource": "arn:aws:s3:::bucket_name"`

  Examples for object-level operations:

  - `"Resource": "arn:aws:s3:::bucket_name/*"` for all objects in the bucket.
  - `"Resource": "arn:aws:s3:::bucket_name/prefix/*"` for objects under a certain prefix in the bucket.

  For more information, see Policy resources for Amazon S3.

- Actions – For each resource, Amazon S3 supports a set of operations. You identify resource operations that you will allow (or deny) by using action keywords.

  For example, the `s3:ListBucket` permission allows the user to use the Amazon S3 ListObjectsV2 operation. (The `s3:ListBucket` permission is a case where the action name doesn't map directly to the operation name.) For more information about using Amazon S3 actions, see Policy actions for Amazon S3. For a complete list of Amazon S3 actions, see Actions in the *Amazon Simple Storage Service API Reference*.

- Effect – What the effect will be when the user requests the specific action—this can be either `Allow` or `Deny`.

  If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user can't access the resource, even if a different policy grants access. For more information, see IAM JSON Policy Elements: Effect in the *IAM User Guide*.

- Principal – The account or user who is allowed access to the actions and resources in the statement. In a bucket policy, the principal is the user, account, service, or other entity that is the recipient of this permission. For more information, see Principals for bucket policies.

- Condition – Conditions for when a policy is in effect. You can use AWS-wide keys and Amazon S3-specific keys to specify conditions in an Amazon S3 access policy. For more information, see Bucket policy examples using condition keys.

The following example bucket policy shows the `Effect`, `Principal`, `Action`, and `Resource` elements. This policy allows *Akua*, a user in account *123456789012*, `s3:GetObject`,

`s3:GetBucketLocation`, and `s3:ListBucket` Amazon S3 permissions on the *amzn-s3-demo-bucket1* bucket.

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        }
    ]
}
```

For complete policy language information, see Policies and permissions in IAM and IAM JSON policy reference in the *IAM User Guide*.

**Permission delegation**

If an AWS account owns a resource, it can grant those permissions to another AWS account. That account can then delegate those permissions, or a subset of them, to users in the account. This is referred to as *permission delegation*. But an account that receives permissions from another account can't delegate permission cross-account to another AWS account.

**Amazon S3 bucket and object ownership**

Buckets and objects are Amazon S3 resources. By default, only the resource owner can access these resources. The resource owner refers to the AWS account that creates the resource. For example:

- The AWS account that you use to create buckets and upload objects owns those resources.

- If you upload an object using AWS Identity and Access Management (IAM) user or role credentials, the AWS account that the user or role belongs to owns the object.
- A bucket owner can grant cross-account permissions to another AWS account (or users in another account) to upload objects. In this case, the AWS account that uploads objects owns those objects. The bucket owner doesn't have permissions on the objects that other accounts own, with the following exceptions:
  - The bucket owner pays the bills. The bucket owner can deny access to any objects, or delete any objects in the bucket, regardless of who owns them.
  - The bucket owner can archive any objects or restore archived objects regardless of who owns them. Archival refers to the storage class used to store the objects. For more information, see Managing your storage lifecycle.

**Ownership and request authentication**

All requests to a bucket are either authenticated or unauthenticated. Authenticated requests must include a signature value that authenticates the request sender, and unauthenticated requests do not. For more information about request authentication, see Making requests.

A bucket owner can allow unauthenticated requests. For example, unauthenticated PutObject requests are allowed when a bucket has a public bucket policy, or when a bucket ACL grants `WRITE` or `FULL_CONTROL` access to the `All Users` group or the anonymous user specifically. For more information about public bucket policies and public access control lists (ACLs), see The meaning of "public".

All unauthenticated requests are made by the anonymous user. This user is represented in ACLs by the specific canonical user ID `65a011a29cdf8ec533ec3d1ccaae921c`. If an object is uploaded to a bucket through an unauthenticated request, the anonymous user owns the object. The default object ACL grants `FULL_CONTROL` to the anonymous user as the object's owner. Therefore, Amazon S3 allows unauthenticated requests to retrieve the object or modify its ACL.

To prevent objects from being modified by the anonymous user, we recommend that you do not implement bucket policies that allow anonymous public writes to your bucket or use ACLs that allow the anonymous user write access to your bucket. You can enforce this recommended behavior by using Amazon S3 Block Public Access.

For more information about blocking public access, see Blocking public access to your Amazon S3 storage. For more information about ACLs, see Access control list (ACL) overview.

**Important**

We recommend that you don't use the AWS account root user credentials to make authenticated requests. Instead, create an IAM role and grant that role full access. We refer to users with this role as *administrator users*. You can use credentials assigned to the administrator role, instead of AWS account root user credentials, to interact with AWS and perform tasks, such as create a bucket, create users, and grant permissions. For more information, see AWS security credentials and Security best practices in IAM in the *IAM User Guide*.

## Bucket policies for Amazon S3

A bucket policy is a resource-based policy that you can use to grant access permissions to your Amazon S3 bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. These permissions don't apply to objects that are owned by other AWS accounts.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting and all ACLs are disabled. The bucket owner owns all the objects in the bucket and manages access to data exclusively using policies.

Bucket policies use JSON-based AWS Identity and Access Management (IAM) policy language. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies can allow or deny requests based on the elements in the policy. These elements include the requester, S3 actions, resources, and aspects or conditions of the request (such as the IP address that's used to make the request).

For example, you can create a bucket policy that does the following:

- Grants other accounts cross-account permissions to upload objects to your S3 bucket
- Makes sure that you, the bucket owner, has full control of the uploaded objects

For more information, see Examples of Amazon S3 bucket policies.

**Important**

You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

The topics in this section provide examples and show you how to add a bucket policy in the S3 console. For information about identity-based policies, see Identity-based policies for Amazon S3. For information about bucket policy language, see Policies and permissions in Amazon S3.

**Topics**

- Adding a bucket policy by using the Amazon S3 console
- Controlling access from VPC endpoints with bucket policies
- Examples of Amazon S3 bucket policies
- Bucket policy examples using condition keys

### Adding a bucket policy by using the Amazon S3 console

You can use the AWS Policy Generator and the Amazon S3 console to add a new bucket policy or edit an existing bucket policy. A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy. You add a bucket policy to a bucket to grant other AWS accounts or IAM users access permissions for the bucket and the objects in it. Object permissions apply only to the objects that the bucket owner creates. For more information about bucket policies, see Identity and Access Management for Amazon S3.

Make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that are functional and conform to security best practices. To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

For guidance on troubleshooting errors with a policy, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

**To create or edit a bucket policy**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to create a bucket policy for or whose bucket policy you want to edit.
4. Choose the **Permissions** tab.
5. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.
6. On the **Edit bucket policy** page, do one of the following:

   - To see examples of bucket policies, choose **Policy examples**. Or see Examples of Amazon S3 bucket policies in the *Amazon S3 User Guide*.
   - To generate a policy automatically, or edit the JSON in the **Policy** section, choose **Policy generator**.

   If you choose **Policy generator**, the AWS Policy Generator opens in a new window.

   a. On the **AWS Policy Generator** page, for **Select Type of Policy**, choose **S3 Bucket Policy**.

   b. Add a statement by entering the information in the provided fields, and then choose **Add Statement**. Repeat this step for as many statements as you would like to add. For more information about these fields, see the IAM JSON policy elements reference in the *IAM User Guide*.

   > **Note**
   >
   > For your convenience, the **Edit bucket policy** page displays the **Bucket ARN** (Amazon Resource Name) of the current bucket above the **Policy** text field. You can copy this ARN for use in the statements on the **AWS Policy Generator** page.

   c. After you finish adding statements, choose **Generate Policy**.

   d. Copy the generated policy text, choose **Close**, and return to the **Edit bucket policy** page in the Amazon S3 console.
7. In the **Policy** box, edit the existing policy or paste the bucket policy from the AWS Policy Generator. Make sure to resolve security warnings, errors, general warnings, and suggestions before you save your policy.

> **Note**
>
> Bucket policies are limited to 20 KB in size.

8. (Optional) Choose **Preview external access** in the lower-right corner to preview how your new policy affects public and cross-account access to your resource. Before you save your policy, you can check whether it introduces new IAM Access Analyzer findings or resolves existing findings. If you don't see an active analyzer, choose **Go to Access Analyzer** to create an account analyzer in IAM Access Analyzer. For more information, see Preview access in the *IAM User Guide*.
9. Choose **Save changes**, which returns you to the **Permissions** tab.

### Controlling access from VPC endpoints with bucket policies

You can use Amazon S3 bucket policies to control access to buckets from specific virtual private cloud (VPC) endpoints or specific VPCs. This section contains example bucket policies that you can use to control Amazon S3 bucket access from VPC endpoints. To learn how to set up VPC endpoints, see VPC Endpoints in the *VPC User Guide*.

A VPC enables you to launch AWS resources into a virtual network that you define. A VPC endpoint enables you to create a private connection between your VPC and another AWS service. This private connection doesn't require access over the internet, through a virtual private network (VPN) connection, through a NAT instance, or through AWS Direct Connect.

A VPC endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. The VPC endpoint routes requests to Amazon S3 and routes responses back to the VPC. VPC endpoints change only how requests are routed. Amazon S3 public endpoints and DNS names will continue to work with VPC endpoints. For important information about using VPC endpoints with Amazon S3, see Gateway endpoints and Gateway endpoints for Amazon S3 in the *VPC User Guide*.

VPC endpoints for Amazon S3 provide two ways to control access to your Amazon S3 data:

- You can control the requests, users, or groups that are allowed through a specific VPC endpoint. For information about this type of access control, see Controlling access to VPC endpoints using endpoint policies in the *VPC User Guide*.

- You can control which VPCs or VPC endpoints have access to your buckets by using Amazon S3 bucket policies. For examples of this type of bucket policy access control, see the following topics on restricting access.

### Topics

- Restricting access to a specific VPC endpoint
- Restricting access to a specific VPC

> **Important**
>
> When applying the Amazon S3 bucket policies for VPC endpoints described in this section, you might block your access to the bucket unintentionally. Bucket permissions that are intended to specifically limit bucket access to connections originating from your VPC endpoint can block all connections to the bucket. For information about how to fix this issue, see How do I fix my bucket policy when it has the wrong VPC or VPC endpoint ID? in the *AWS Support Knowledge Center*.

### Restricting access to a specific VPC endpoint

The following is an example of an Amazon S3 bucket policy that restricts access to a specific bucket, `awsexamplebucket1`, only from the VPC endpoint with the ID `vpce-1a2b3c4d`. If the specified endpoint is not used, the policy denies all access to the bucket. The `aws:SourceVpce` condition specifies the endpoint. The `aws:SourceVpce` condition doesn't require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the VPC endpoint ID. For more information about using conditions in a policy, see Bucket policy examples using condition keys.

> **Important**
>
> - Before using the following example policy, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
> - This policy disables console access to the specified bucket because console requests don't originate from the specified VPC endpoint.

```
{
```

```
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-1a2b3c4d"
         }
       }
     }
   ]
}
```

### Restricting access to a specific VPC

You can create a bucket policy that restricts access to a specific VPC by using the `aws:SourceVpc` condition. This is useful if you have multiple VPC endpoints configured in the same VPC, and you want to manage access to your Amazon S3 buckets for all of your endpoints. The following is an example of a policy that denies access to `awsexamplebucket1` and its objects from anyone outside VPC `vpc-111bbb22`. If the specified VPC isn't used, the policy denies all access to the bucket. This statement doesn't grant access to the bucket. To grant access, you must add a separate `Allow` statement. The `vpc-111bbb22` condition key doesn't require an ARN for the VPC resource, only the VPC ID.

> **Important**
>
> - Before using the following example policy, replace the VPC ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
> - This policy disables console access to the specified bucket because console requests don't originate from the specified VPC.

```
{
```

```
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpc": "vpc-111bbb22"
         }
       }
     }
   ]
}
```

### Examples of Amazon S3 bucket policies

With Amazon S3 bucket policies, you can secure access to objects in your buckets, so that only users with the appropriate permissions can access them. You can even prevent authenticated users without the appropriate permissions from accessing your Amazon S3 resources.

This section presents examples of typical use cases for bucket policies. These sample policies use *amzn-s3-demo-bucket* as the resource value. To test these policies, replace the *user input placeholders* with your own information (such as your bucket name).

To grant or deny permissions to a set of objects, you can use wildcard characters (*) in Amazon Resource Names (ARNs) and other values. For example, you can control access to groups of objects that begin with a common prefix or end with a specific extension, such as `.html`.

For more information about AWS Identity and Access Management (IAM) policy language, see Policies and permissions in Amazon S3.

> **Note**
>
> When testing permissions by using the Amazon S3 console, you must grant additional permissions that the console requires—`s3:ListAllMyBuckets`, `s3:GetBucketLocation`, and `s3:ListBucket`. For an example walkthrough that grants

permissions to users and tests those permissions by using the console, see Controlling access to a bucket with user policies.

Additional resources for creating bucket policies include the following:

- For a list of the IAM policy actions, resources, and condition keys that you can use when creating a bucket policy, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.
- For guidance on creating your S3 policy, see Adding a bucket policy by using the Amazon S3 console.
- To troubleshoot errors with a policy, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

**Topics**



### Granting read-only permission to a public anonymous user

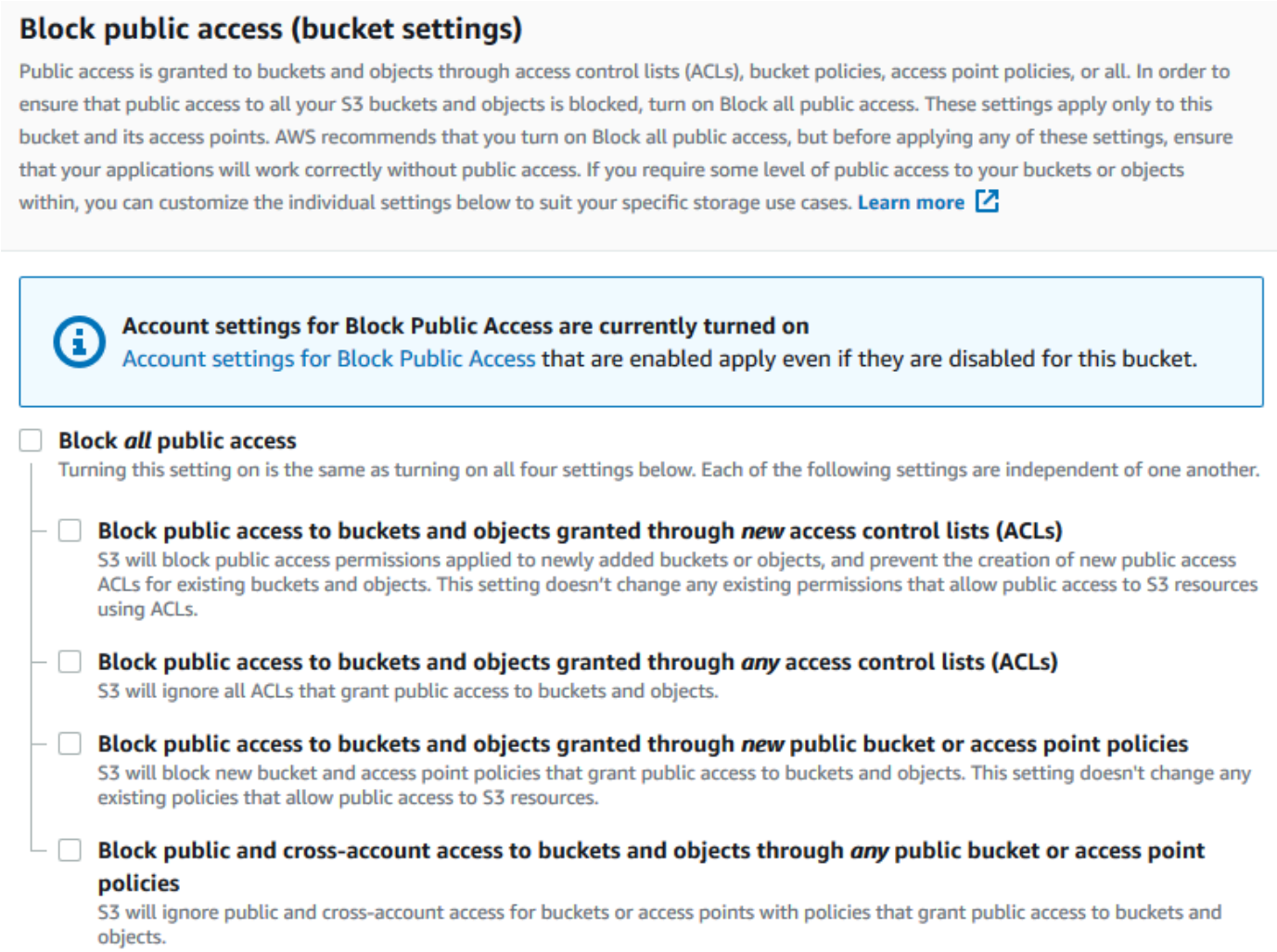
You can use your policy settings to grant access to public anonymous users, which is useful if you're configuring your bucket as a static website. Granting access to public anonymous users requires

you to disable the Block Public Access settings for your bucket. For more information about how to do this, and the policy required, see Setting permissions for website access. To learn how to set up more restrictive policies for the same purpose, see How can I grant public read access to some objects in my Amazon S3 bucket? in the AWS Knowledge Center.

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.




Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

### Requiring encryption

You can require server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), as shown in the following examples.

### Require SSE-KMS for all objects written to a bucket

The following example policy requires every object that is written to the bucket to be encrypted with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). If the object isn't encrypted with SSE-KMS, the request is denied.

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

**Require SSE-KMS with a specific AWS KMS key for all objects written to a bucket**

The following example policy denies any objects from being written to the bucket if they aren't encrypted with SSE-KMS by using a specific KMS key ID. Even if the objects are encrypted with SSE-KMS by using a per-request header or bucket default encryption, the objects can't be written to the bucket if they haven't been encrypted with the specified KMS key. Make sure to replace the KMS key ARN that's used in this example with your own KMS key ARN.

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-2:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
    }
  }
}]
}
```

### Managing buckets using canned ACLs

#### Granting permissions to multiple accounts to upload objects or set object ACLs for public access

The following example policy grants the `s3:PutObject` and `s3:PutObjectAcl` permissions to multiple AWS accounts. Also, the example policy requires that any requests for these operations must include the `public-read` canned access control list (ACL). For more information, see Policy actions for Amazon S3 and Policy condition keys for Amazon S3.

> **Warning**
>
> The `public-read` canned ACL allows anyone in the world to view the objects in your bucket. Use caution when granting anonymous access to your Amazon S3 bucket or disabling block public access settings. When you grant anonymous access, anyone in the world can access your bucket. We recommend that you never grant anonymous access to your Amazon S3 bucket unless you specifically need to, such as with static website hosting. If you want to enable block public access settings for static website hosting, see Tutorial: Configuring a static website on Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
```

```
                    ]
                }
            }
        }
    ]
}
```

### Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control

The following example shows how to allow another AWS account to upload objects to your bucket while ensuring that you have full control of the uploaded objects. This policy grants a specific AWS account (*111122223333*) the ability to upload objects only if that account includes the `bucket-owner-full-control` canned ACL on upload. The `StringEquals` condition in the policy specifies the `s3:x-amz-acl` condition key to express the canned ACL requirement. For more information, see Policy condition keys for Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
       }
     }
   ]
}
```

### Managing object access with object tagging

### Allow a user to read only objects that have a specific tag key and value

The following permissions policy limits a user to only reading objects that have the `environment: production` tag key and value. This policy uses the `s3:ExistingObjectTag` condition key to specify the tag key and value.

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

**Restrict which object tag keys that users can add**

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition uses the `s3:RequestObjectTagKeys` condition key to specify the allowed tag keys, such as `Owner` or `CreationDate`. For more information, see Creating a condition that tests multiple key values in the *IAM User Guide*.

The policy ensures that every tag key specified in the request is an authorized tag key. The `ForAnyValue` qualifier in the condition ensures that at least one of the specified keys must be present in the request.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
```

```
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

### Require a specific tag key and value when allowing users to add object tags

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition requires the user to include a specific tag key (such as *`Project`*) with the value set to *`X`*.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
        ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

### Allow a user to only add objects with a specific object tag key and value

The following example policy grants a user permission to perform the `s3:PutObject` action so that they can add objects to a bucket. However, the `Condition` statement restricts the tag keys and values that are allowed on the uploaded objects. In this example, the user can only add objects that have the specific tag key (*`Department`*) with the value set to *`Finance`* to the bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

### Managing object access by using global condition keys

Global condition keys are condition context keys with an `aws` prefix. AWS services can support global condition keys or service-specific keys that include the service prefix. You can use the `Condition` element of a JSON policy to compare the keys in a request with the key values that you specify in your policy.

### Restrict access to only Amazon S3 server access log deliveries

In the following example bucket policy, the aws:SourceArn global condition key is used to compare the Amazon Resource Name (ARN) of the resource, making a service-to-service request with the ARN that is specified in the policy. The `aws:SourceArn` global condition key is used to prevent the

Amazon S3 service from being used as a confused deputy during transactions between services. Only the Amazon S3 service is allowed to add objects to the Amazon S3 bucket.

This example bucket policy grants `s3:PutObject` permissions to only the logging service principal (`logging.s3.amazonaws.com`).

> **Note**
>
> The NotPrincipal element can't be used with AWS service principals in Amazon S3 resource-based policies, such as bucket policies. Instead, we recommend using the `aws:PrincipalServiceName` condition key, as shown in the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::EXAMPLE-SOURCE-BUCKET"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:PrincipalServiceName": "logging.s3.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

### Allow access to only your organization

If you want to require all IAM principals accessing a resource to be from an AWS account in your organization (including the AWS Organizations management account), you can use the `aws:PrincipalOrgID` global condition key.

To grant or restrict this type of access, define the `aws:PrincipalOrgID` condition and set the value to your organization ID in the bucket policy. The organization ID is used to control access to the bucket. When you use the `aws:PrincipalOrgID` condition, the permissions from the bucket policy are also applied to all new accounts that are added to the organization.

Here's an example of a resource-based bucket policy that you can use to grant specific IAM principals in your organization direct access to your bucket. By adding the `aws:PrincipalOrgID` global condition key to your bucket policy, the principal account is now required to be in your organization to obtain access to the resource. Even if you accidentally specify an incorrect account when granting access, the aws:PrincipalOrgID global condition key acts as an additional safeguard. When this global key is used in a policy, it prevents all principals from outside of the specified organization from accessing the S3 bucket. Only principals from accounts in the listed organization are able to obtain access to the resource.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
        }
```

```
    }]
}
```

### Managing access based on HTTP or HTTPS requests

#### Restrict access to only HTTPS requests

If you want to prevent potential attackers from manipulating network traffic, you can use HTTPS (TLS) to only allow encrypted connections while restricting HTTP requests from accessing your bucket. To determine whether the request is HTTP or HTTPS, use the aws:SecureTransport global condition key in your S3 bucket policy. The `aws:SecureTransport` condition key checks whether a request was sent by using HTTP.

If a request returns `true`, then the request was sent through HTTPS. If the request returns `false`, then the request was sent through HTTP. You can then allow or deny access to your bucket based on the desired request scheme.

In the following example, the bucket policy explicitly denies HTTP requests.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

#### Restrict access to a specific HTTP referer

Suppose that you have a website with the domain name *www.example.com* or *example.com* with links to photos and videos stored in your bucket named *amzn-s3-demo-bucket*. By default, all

Amazon S3 resources are private, so only the AWS account that created the resources can access them.

To allow read access to these objects from your website, you can add a bucket policy that allows the `s3:GetObject` permission with a condition that the `GET` request must originate from specific webpages. The following policy restricts requests by using the `StringLike` condition with the `aws:Referer` condition key.

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
      "Sid":"Allow only GET requests originating from www.example.com and
 example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/
*"]}
      }
    }
  ]
}
```

Make sure that the browsers that you use include the HTTP `referer` header in the request.

> **Warning**
>
> We recommend that you use caution when using the `aws:Referer` condition key. It is dangerous to include a publicly known HTTP referer header value. Unauthorized parties can use modified or custom browsers to provide any `aws:Referer` value that they choose. Therefore, do not use `aws:Referer` to prevent unauthorized parties from making direct AWS requests.
>
> The `aws:Referer` condition key is offered only to allow customers to protect their digital content, such as content stored in Amazon S3, from being referenced on unauthorized third-party sites. For more information, see aws:Referer in the *IAM User Guide*.

### Managing user access to specific folders

#### Grant users access to specific folders

Suppose that you're trying to grant users access to a specific folder. If the IAM user and the S3 bucket belong to the same AWS account, then you can use an IAM policy to grant the user access to a specific bucket folder. With this approach, you don't need to update your bucket policy to grant access. You can add the IAM policy to an IAM role that multiple users can switch to.

If the IAM identity and the S3 bucket belong to different AWS accounts, then you must grant cross-account access in both the IAM policy and the bucket policy. For more information about granting cross-account access, see Bucket owner granting cross-account bucket permissions.

The following example bucket policy grants *JohnDoe* full console access to only his folder (home/*JohnDoe*/). By creating a home folder and granting the appropriate permissions to your users, you can have multiple users share a single bucket. This policy consists of three `Allow` statements:

- *AllowRootAndHomeListingOfCompanyBucket*: Allows the user (*JohnDoe*) to list objects at the root level of the *DOC-EXAMPLE-BUCKET* bucket and in the home folder. This statement also allows the user to search on the prefix home/ by using the console.
- *AllowListingOfUserFolder*: Allows the user (*JohnDoe*) to list all objects in the home/*JohnDoe*/ folder and any subfolders.
- *AllowAllS3ActionsInUserFolder*: Allows the user to perform all Amazon S3 actions by granting `Read`, `Write`, and `Delete` permissions. Permissions are limited to the bucket owner's home folder.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
```

```
            "Condition": {
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/home/JohnDoe/*"]
        }
    ]
}
```

### Managing access for access logs

#### Grant access to Application Load Balancer for enabling access logs

When you enable access logs for Application Load Balancer, you must specify the name of the S3 bucket where the load balancer will store the logs. The bucket must have an attached policy that grants Elastic Load Balancing permission to write to the bucket.

In the following example, the bucket policy grants Elastic Load Balancing (ELB) permission to write the access logs to the bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

> **Note**
>
> Make sure to replace *elb-account-id* with the AWS account ID for Elastic Load Balancing for your AWS Region. For the list of Elastic Load Balancing Regions, see Attach a policy to your Amazon S3 bucket in the *Elastic Load Balancing User Guide*.

If your AWS Region does not appear in the supported Elastic Load Balancing Regions list, use the following policy, which grants permissions to the specified log delivery service.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Principal": {
```

```
          "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
           },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

Then, make sure to configure your Elastic Load Balancing access logs by enabling them. You can verify your bucket permissions by creating a test file.

**Managing access to an Amazon CloudFront OAI**

**Grant permission to an Amazon CloudFront OAI**

The following example bucket policy grants a CloudFront origin access identity (OAI) permission to get (read) all objects in your S3 bucket. You can use a CloudFront OAI to allow users to access objects in your bucket through CloudFront but not directly through Amazon S3. For more information, see Restricting access to Amazon S3 content by using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

The following policy uses the OAI's ID as the policy's `Principal`. For more information about using S3 bucket policies to grant access to a CloudFront OAI, see Migrating from origin access identity (OAI) to origin access control (OAC) in the *Amazon CloudFront Developer Guide*.

To use this example:

- Replace *EH1HDMB1FH2TC* with the OAI's ID. To find the OAI's ID, see the Origin Access Identity page on the CloudFront console, or use ListCloudFrontOriginAccessIdentities in the CloudFront API.
- Replace *amzn-s3-demo-bucket* with the name of your bucket.

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

### Managing access for Amazon S3 Storage Lens

### Grant permissions for Amazon S3 Storage Lens

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

S3 Storage Lens can export your aggregated storage usage metrics to an Amazon S3 bucket for further analysis. The bucket where S3 Storage Lens places its metrics exports is known as the *destination bucket*. When setting up your S3 Storage Lens metrics export, you must have a bucket policy for the destination bucket. For more information, see Assessing your storage activity and usage with Amazon S3 Storage Lens.

The following example bucket policy grants Amazon S3 permission to write objects (PUT requests) to a destination bucket. You use a bucket policy like this on the destination bucket when setting up an S3 Storage Lens metrics export.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-code:111122223333:storage-
lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
    ]
}
```

When you're setting up an S3 Storage Lens organization-level metrics export, use the following modification to the previous bucket policy's `Resource` statement.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/your-organization-id/*",
```

### Managing permissions for S3 Inventory, S3 analytics, and S3 Inventory reports

#### Grant permissions for S3 Inventory and S3 analytics

S3 Inventory creates lists of the objects in a bucket, and S3 analytics Storage Class Analysis export creates output files of the data used in the analysis. The bucket that the inventory lists the objects for is called the *source bucket*. The bucket where the inventory file or the analytics export file is written to is called a *destination bucket*. When setting up an inventory or an analytics export, you must create a bucket policy for the destination bucket. For more information, see Amazon S3 Inventory and Amazon S3 analytics – Storage Class Analysis.

The following example bucket policy grants Amazon S3 permission to write objects (PUT requests) from the account for the source bucket to the destination bucket. You use a bucket policy like this on the destination bucket when setting up S3 Inventory and S3 analytics export.

```
{
      "Version": "2012-10-17",
```

```
      "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

**Control S3 Inventory report configuration creation**

Amazon S3 Inventory creates lists of the objects in an S3 bucket and the metadata for each object. The `s3:PutInventoryConfiguration` permission allows a user to create an inventory configuration that includes all object metadata fields that are available by default and to specify the destination bucket to store the inventory. A user with read access to objects in the destination bucket can access all object metadata fields that are available in the inventory report. For more information about the metadata fields that are available in S3 Inventory, see Amazon S3 Inventory list.

To restrict a user from configuring an S3 Inventory report, remove the `s3:PutInventoryConfiguration` permission from the user.

Some object metadata fields in S3 Inventory report configurations are optional, meaning that they're available by default but they can be restricted when you grant a user the `s3:PutInventoryConfiguration` permission. You can control whether users can include these optional metadata fields in their reports by using the `s3:InventoryAccessibleOptionalFields` condition key. For a list of the optional metadata

fields available in S3 Inventory, see OptionalFields in the *Amazon Simple Storage Service API Reference*.

To grant a user permission to create an inventory configuration with specific optional metadata fields, use the `s3:InventoryAccessibleOptionalFields` condition key to refine the conditions in your bucket policy.

The following example policy grants a user (*Ana*) permission to create an inventory configuration conditionally. The `ForAllValues:StringEquals` condition in the policy uses the `s3:InventoryAccessibleOptionalFields` condition key to specify the two allowed optional metadata fields, namely `Size` and `StorageClass`. So, when *Ana* is creating an inventory configuration, the only optional metadata fields that she can include are `Size` and `StorageClass`.

```
{
 "Id": "InventoryConfigPolicy",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

To restrict a user from configuring an S3 Inventory report that includes specific optional metadata fields, add an explicit Deny statement to the bucket policy for the source bucket. The following

example bucket policy denies the user *Ana* from creating an inventory configuration in the source bucket *DOC-EXAMPLE-SOURCE-BUCKET* that includes the optional `ObjectAccessControlList` or `ObjectOwner` metadata fields. The user *Ana* can still create an inventory configuration with other optional metadata fields.

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",

  },
  {
   "Sid": "DenyCertainInventoryFieldCreation",
   "Effect": "Deny",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
     "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "ObjectOwner",
        "ObjectAccessControlList"
        ]
      }
    }
   }
 ]
}
```

**Note**

The use of the `s3:InventoryAccessibleOptionalFields` condition key in bucket policies doesn't affect the delivery of inventory reports based on the existing inventory configurations.

**Important**

We recommend that you use `ForAllValues` with an `Allow` effect or `ForAnyValue` with a `Deny` effect, as shown in the prior examples.
Don't use `ForAllValues` with a `Deny` effect nor `ForAnyValue` with an `Allow` effect, because these combinations can be overly restrictive and block inventory configuration deletion.
To learn more about the `ForAllValues` and `ForAnyValue` condition set operators, see Multivalued context keys in the *IAM User Guide*.

### Requiring MFA

Amazon S3 supports MFA-protected API access, a feature that can enforce multi-factor authentication (MFA) for access to your Amazon S3 resources. Multi-factor authentication provides an extra level of security that you can apply to your AWS environment. MFA is a security feature that requires users to prove physical possession of an MFA device by providing a valid MFA code. For more information, see AWS Multi-Factor Authentication. You can require MFA for any requests to access your Amazon S3 resources.

To enforce the MFA requirement, use the `aws:MultiFactorAuthAge` condition key in a bucket policy. IAM users can access Amazon S3 resources by using temporary credentials issued by the AWS Security Token Service (AWS STS). You provide the MFA code at the time of the AWS STS request.

When Amazon S3 receives a request with multi-factor authentication, the `aws:MultiFactorAuthAge` condition key provides a numeric value that indicates how long ago (in seconds) the temporary credential was created. If the temporary credential provided in the request was not created by using an MFA device, this key value is null (absent). In a bucket policy, you can add a condition to check this value, as shown in the following example.

This example policy denies any Amazon S3 operation on the */taxdocuments* folder in the *amzn-s3-demo-bucket* bucket if the request is not authenticated by using MFA. To learn more about MFA, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

The `Null` condition in the `Condition` block evaluates to `true` if the `aws:MultiFactorAuthAge` condition key value is null, indicating that the temporary security credentials in the request were created without an MFA device.

The following bucket policy is an extension of the preceding bucket policy. The following policy includes two policy statements. One statement allows the `s3:GetObject` permission on a bucket (*amzn-s3-demo-bucket*) to everyone. Another statement further restricts access to the *amzn-s3-demo-bucket/taxdocuments* folder in the bucket by requiring MFA.

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "",
```

```
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
 }
```

You can optionally use a numeric condition to limit the duration for which the `aws:MultiFactorAuthAge` key is valid. The duration that you specify with the `aws:MultiFactorAuthAge` key is independent of the lifetime of the temporary security credential that's used in authenticating the request.

For example, the following bucket policy, in addition to requiring MFA authentication, also checks how long ago the temporary session was created. The policy denies any operation if the `aws:MultiFactorAuthAge` key value indicates that the temporary session was created more than an hour ago (3,600 seconds).

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": "*",
```

```
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
   ]
}
```

### Preventing users from deleting objects

By default, users have no permissions. But as you create policies, you might grant users permissions that you didn't intend to grant. To avoid such permission loopholes, you can write a stricter access policy by adding an explicit deny.

To explicitly block users or accounts from deleting objects, you must add the following actions to a bucket policy: `s3:DeleteObject`, `s3:DeleteObjectVersion`, and `s3:PutLifecycleConfiguration` permissions. All three actions are required because you can delete objects either by explicitly calling the `DELETE Object` API operations or by configuring their lifecycle (see Managing your storage lifecycle) so that Amazon S3 can remove the objects when their lifetime expires.

In the following policy example, you explicitly deny `DELETE Object` permissions to the user *`MaryMajor`*. An explicit `Deny` statement always supersedes any other permission granted.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
   "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Sid": "statement2",
```

```
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
     "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

### Bucket policy examples using condition keys

You can use access policy language to specify conditions when you grant permissions. You can use the optional `Condition` element, or `Condition` block to specify conditions for when a policy is in effect.

For policies that use Amazon S3 condition keys for object and bucket operations, see the following examples. For more information about condition keys, see Policy condition keys for Amazon S3. For a complete list of Amazon S3 actions, condition keys, and resources that you can specify in policies, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

### Examples — Amazon S3 condition keys for object operations

This section provides examples that show you how you can use Amazon S3-specific condition keys for object operations. For a complete list of Amazon S3 actions, condition keys, and resources that you can specify in policies, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

Several of the example policies show how you can use conditions keys with PUT Object operations. PUT Object operations allow access control list (ACL)–specific headers that you can use to grant ACL-based permissions. Using these keys, the bucket owner can set a condition to require specific access permissions when the user uploads an object. You can also grant ACL–based permissions with the PutObjectAcl operation. For more information, see PutObjectAcl in the *Amazon S3*

*Amazon Simple Storage Service API Reference*. For more information about ACLs, see Access control list (ACL) overview.

**Topics**



### Example 1: Granting s3:PutObject permission requiring objects stored using server-side encryption

Suppose that Account A owns a bucket. The account administrator wants to grant Jane, a user in Account A, permission to upload objects with a condition that Jane always request server-side encryption so that Amazon S3 saves objects encrypted. The Account A administrator can accomplish using the `s3:x-amz-server-side-encryption` condition key as shown. The key-value pair in the `Condition` block specifies the `s3:x-amz-server-side-encryption` key.

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }}
```

When testing the permission using the AWS CLI, you must add the required parameter using the `--server-side-encryption` parameter.

```
aws s3api put-object --bucket example1bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountBadmin
```

### Example 2: Granting s3:PutObject permission to copy objects with a restriction on the copy source

In the PUT Object request, when you specify a source object, it is a copy operation (see PUT Object - Copy). Accordingly, the bucket owner can grant a user permission to copy objects with restrictions on the source, for example:

- Allow copying objects only from the `sourcebucket` bucket.
- Allow copying objects from the source bucket and only the objects whose key name prefix starts with public/ f (for example, sourcebucket/public/*).
- Allow copying only a specific object from the sourcebucket (for example, sourcebucket/example.jpg).

The following bucket policy grants user (Dave) `s3:PutObject` permission. It allows him to copy objects only with a condition that the request include the `s3:x-amz-copy-source` header and the header value specify the `/awsexamplebucket1/public/*` key name prefix.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "awsexamplebucket1/public/*"
```

```
                }
            }
        }
    ]
}
```

### Test the policy with the AWS CLI

You can test the permission using the AWS CLI `copy-object` command. You specify the source by adding the `--copy-source` parameter; the key name prefix must match the prefix allowed in the policy. You need to provide the user Dave credentials using the `--profile` parameter. For more information about setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

```
aws s3api copy-object --bucket awsexamplebucket1 --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

### Give permission to copy only a specific object

The preceding policy uses the `StringNotLike` condition. To grant permission to copy only a specific object, you must change the condition from `StringNotLike` to `StringNotEquals` and then specify the exact object key as shown.

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "awsexamplebucket1/public/PublicHappyFace1.jpg"
       }
}
```

### Example 3: Granting access to a specific version of an object

Suppose that Account A owns a versioning-enabled bucket. The bucket has several versions of the `HappyFace.jpg` object. The account administrator now wants to grant its user Dave permission to get only a specific version of the object. The account administrator can accomplish this by granting Dave `s3:GetObjectVersion` permission conditionally as shown below. The key-value pair in the `Condition` block specifies the `s3:VersionId` condition key. In this case, Dave needs to know the exact object version ID to retrieve the object.

For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

### Test the policy with the AWS CLI

You can test the permissions using the AWS CLI `get-object` command with the `--version-id` parameter identifying the specific object version. The command retrieves the object and saves it to the `OutputFile.jpg` file.

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
 OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

### Example 4: Granting permissions based on object tags

For examples on how to use object tagging condition keys with Amazon S3 operations, see Tagging and access control policies.

### Example 5: Restricting access by the AWS account ID of the bucket owner

You can use either the `aws:ResourceAccount` or `s3:ResourceAccount` key to write IAM or virtual private cloud (VPC) endpoint policies that restrict user, role, or application access to the Amazon S3 buckets that are owned by a specific AWS account ID. You can use this condition key to restrict clients within your VPC from accessing buckets that you do not own.

However, be aware that some AWS services rely on access to AWS managed buckets. Therefore, using the `aws:ResourceAccount` or `s3:ResourceAccount` key in your IAM policy might also affect access to these resources.

For more information and examples, see the following resources:

- Restrict access to buckets in a specified AWS account in the *AWS PrivateLink Guide*
- Restrict access to buckets that Amazon ECR uses in the *Amazon ECR Guide*
- Provide required access to Systems Manager for AWS managed Amazon S3 buckets in the *AWS Systems Manager Guide*
- Limit access to Amazon S3 buckets owned by specific AWS accounts in the *AWS Storage Blog*

### Example 6: Requiring a minimum TLS version

You can use the s3:TlsVersion condition key to write IAM, Virtual Private Cloud Endpoint (VPCE), or bucket policies that restrict user or application access to Amazon S3 buckets based on the TLS version used by the client. You can use this condition key to write policies that require a minimum TLS version.

#### Example

This example bucket policy *denies* PutObject requests by clients that have a TLS version lower than 1.2, for example, 1.1 or 1.0.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

### Example

This example bucket policy *allows* PutObject requests by clients that have a TLS version higher than 1.1, for example, 1.2, 1.3 or higher.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
    ]
}
```

### Examples — Amazon S3 condition keys for bucket operations

This section provides example policies that show you how you can use Amazon S3-specific condition keys for bucket operations.

#### Topics



#### Example 1: Granting s3:GetObject permission with a condition on an IP address

You can give authenticated users permission to use the `s3:GetObject` action if the request originates from a specific range of IP addresses (192.0.2.*), unless the IP address is 192.0.2.188. In the condition block, the `IpAddress` and the `NotIpAddress` are conditions, and each condition is provided a key-value pair for evaluation. Both the key-value pairs in this example use the `aws:SourceIp` AWS-wide key.

> **Note**
>
> The `IPAddress` and `NotIpAddress` key values specified in the condition uses CIDR notation as described in RFC 4632. For more information, see http://www.rfc-editor.org/rfc/rfc4632.txt.

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action":"s3:GetObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
```

```
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

You can also use other AWS-wide condition keys in Amazon S3 policies. For example, you can specify the `aws:SourceVpce` and `aws:SourceVpc` condition keys in bucket policies for VPC endpoints. For specific examples, see Controlling access from VPC endpoints with bucket policies.

> **Note**
>
> For some AWS global condition keys, only certain resource types are supported. Therefore, check whether Amazon S3 supports the global condition key and resource type that you want to use, or if you'll need to use an Amazon S3-specific condition key instead. For a complete list of supported resource types and condition keys for Amazon S3, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

**Example 2: Getting a list of objects in a bucket with a specific prefix**

You can use the `s3:prefix` condition key to limit the response of the GET Bucket (ListObjects) API to key names with a specific prefix. If you are the bucket owner, you can restrict a user to list the contents of a specific prefix in the bucket. This condition key is useful if objects in the bucket are organized by key name prefixes. The Amazon S3 console uses key name prefixes to show a folder concept. Only the console supports the concept of folders; the Amazon S3 API supports only buckets and objects. For more information about using prefixes and delimiters to filter access permissions, see Controlling access to a bucket with user policies.

For example, if you have two objects with key names `public/object1.jpg` and `public/object2.jpg`, the console shows the objects under the `public` folder. In the Amazon S3 API, these are objects with prefixes, not objects in folders. However, in the Amazon S3 API, if you organize your object keys using such prefixes, you can grant `s3:ListBucket` permission with the `s3:prefix` condition that will allow the user to get a list of key names with those specific prefixes.

In this example, the bucket owner and the parent account to which the user belongs are the same. So the bucket owner can use either a bucket policy or a user policy. For more information about other condition keys that you can use with the GET Bucket (ListObjects) API, see ListObjects.

### User policy

The following user policy grants the `s3:ListBucket` permission (see GET Bucket (List Objects)) with a condition that requires the user to specify the `prefix` in the request with the value `projects`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

The condition restricts the user to listing object keys with the `projects` prefix. The added explicit deny denies the user request for listing keys with any other prefix no matter what other permissions the user might have. For example, it is possible that the user gets permission to list object keys without any restriction, either by updates to the preceding user policy or via a bucket policy. Because explicit deny always supersedes, the user request to list keys other than the `projects` prefix is denied.

### Bucket policy

If you add the `Principal` element to the above user policy, identifying the user, you now have a bucket policy as shown.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

### Test the policy with the AWS CLI

You can test the policy using the following `list-object` AWS CLI command. In the command, you provide user credentials using the `--profile` parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

```
aws s3api list-objects --bucket awsexamplebucket1 --prefix examplefolder --profile
 AccountADave
```

If the bucket is versioning-enabled, to list the objects in the bucket, you must grant the `s3:ListBucketVersions` permission in the preceding policy, instead of `s3:ListBucket` permission. This permission also supports the `s3:prefix` condition key.

**Example 3: Setting the maximum number of keys**

You can use the `s3:max-keys` condition key to set the maximum number of keys that requester can return in a GET Bucket (ListObjects) or ListObjectVersions request. By default, the API returns up to 1,000 keys. For a list of numeric condition operators that you can use with `s3:max-keys` and accompanying examples, see Numeric Condition Operators in the *IAM User Guide*.

## Identity-based policies for Amazon S3

By default, users and roles don't have permission to create or modify Amazon S3 resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Amazon S3, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

**Topics**

- Policy best practices
- Identity-based policy examples for Amazon S3
- Controlling access to a bucket with user policies

### Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon S3 resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

### Identity-based policy examples for Amazon S3

This section shows several example AWS Identity and Access Management (IAM) identity-based policies for controlling access to Amazon S3. For example *bucket policies* (resource-based policies), see Bucket policies for Amazon S3. For information about IAM policy language, see Policies and permissions in Amazon S3.

The following example policies will work if you use them programmatically. However, to use them with the Amazon S3 console, you must grant additional permissions that are required by the console. For information about using policies such as these with the Amazon S3 console, see Controlling access to a bucket with user policies.

**Topics**



### Allowing an IAM user access to one of your buckets

In this example, you want to grant an IAM user in your AWS account access to one of your buckets, *amzn-s3-demo-bucket1*, and allow the user to add, update, and delete objects.

In addition to granting the `s3:PutObject`, `s3:GetObject`, and `s3:DeleteObject` permissions to the user, the policy also grants the `s3:ListAllMyBuckets`, `s3:GetBucketLocation`, and `s3:ListBucket` permissions. These are the additional permissions required by the console. Also, the `s3:PutObjectAcl` and the `s3:GetObjectAcl` actions are required to be able to copy, cut, and paste objects in the console. For an example walkthrough that grants permissions to users and tests them using the console, see Controlling access to a bucket with user policies.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": "s3:ListAllMyBuckets",
```

```
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":["s3:ListBucket","s3:GetBucketLocation"],
            "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {
            "Effect":"Allow",
            "Action":[
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

### Allowing each IAM user access to a folder in a bucket

In this example, you want two IAM users, Mary and Carlos, to have access to your bucket, *amzn-s3-demo-bucket1*, so that they can add, update, and delete objects. However, you want to restrict each user's access to a single prefix (folder) in the bucket. You might create folders with names that match their usernames.

```
amzn-s3-demo-bucket1
   Mary/
   Carlos/
```

To grant each user access only to their folder, you can write a policy for each user and attach it individually. For example, you can attach the following policy to the user Mary to allow her specific Amazon S3 permissions on the *amzn-s3-demo-bucket1*/*Mary* folder.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
```

```
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/Mary/*"
      }
   ]
}
```

You can then attach a similar policy to the user Carlos, specifying the folder *Carlos* in the `Resource` value.

Instead of attaching policies to individual users, you can write a single policy that uses a policy variable and then attach the policy to a group. First, you must create a group and add both Mary and Carlos to the group. The following example policy allows a set of Amazon S3 permissions in the *amzn-s3-demo-bucket1*`/${aws:username}` folder. When the policy is evaluated, the policy variable `${aws:username}` is replaced by the requester's username. For example, if Mary sends a request to put an object, the operation is allowed only if Mary is uploading the object to the *amzn-s3-demo-bucket1*`/Mary` folder.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
   ]
}
```

> **Note**
>
> When using policy variables, you must explicitly specify version 2012-10-17 in the policy. The default version of the IAM policy language, 2008-10-17, does not support policy variables.

If you want to test the preceding policy on the Amazon S3 console, the console requires additional permissions, as shown in the following policy. For information about how the console uses these permissions, see Controlling access to a bucket with user policies.

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [
       "s3:ListAllMyBuckets",
       "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition":{
            "StringEquals":{
                   "s3:prefix":[""], "s3:delimiter":["/"]
                         }
               }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
```

```
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
  ]
}
```

**Note**

In the 2012-10-17 version of the policy, policy variables start with $. This change in syntax can potentially create a conflict if your object key (object name) includes a $.
To avoid this conflict, specify the $ character by using ${$}. For example, to include the object key my$file in a policy, specify it as my${$}file.

Although IAM user names are friendly, human-readable identifiers, they aren't required to be globally unique. For example, if the user Carlos leaves the organization and another Carlos joins, then the new Carlos could access the old Carlos's information.

Instead of using usernames, you could create folders based on IAM user IDs. Each IAM user ID is unique. In this case, you must modify the preceding policy to use the ${aws:userid} policy variable. For more information about user identifiers, see IAM Identifiers in the *IAM User Guide*.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
```

```
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/home/${aws:userid}/*"
      }
   ]
}
```

**Allowing non-IAM users (mobile app users) access to folders in a bucket**

Suppose that you want to develop a mobile app, a game that stores users' data in an S3 bucket. For each app user, you want to create a folder in your bucket. You also want to limit each user's access to their own folder. But you can't create folders before someone downloads your app and starts playing the game, because you don't have their user ID.

In this case, you can require users to sign in to your app by using public identity providers such as Login with Amazon, Facebook, or Google. After users have signed in to your app through one of these providers, they have a user ID that you can use to create user-specific folders at runtime.

You can then use web identity federation in AWS Security Token Service to integrate information from the identity provider with your app and to get temporary security credentials for each user. You can then create IAM policies that allow the app to access your bucket and perform such operations as creating user-specific folders and uploading data. For more information about web identity federation, see About web identity Federation in the *IAM User Guide*.

**Allowing a group to have a shared folder in Amazon S3**

Attaching the following policy to the group grants everybody in the group access to the following folder in Amazon S3: *amzn-s3-demo-bucket1*/share/marketing. Group members are allowed to access only the specific Amazon S3 permissions shown in the policy and only for objects in the specified folder.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
```

```
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/share/marketing/*"
      }
   ]
}
```

### Allowing all your users to read objects in a portion of a bucket

In this example, you create a group named *AllUsers*, which contains all the IAM users that are owned by the AWS account. You then attach a policy that gives the group access to `GetObject` and `GetObjectVersion`, but only for objects in the *amzn-s3-demo-bucket1*`/readonly` folder.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/readonly/*"
      }
   ]
}
```

### Allowing a partner to drop files into a specific portion of a bucket

In this example, you create a group called *AnyCompany* that represents a partner company. You create an IAM user for the specific person or application at the partner company that needs access, and then you put the user in the group.

You then attach a policy that gives the group `PutObject` access to the following folder in a bucket:

*amzn-s3-demo-bucket1*`/uploads/anycompany`

You want to prevent the *AnyCompany* group from doing anything else with the bucket, so you add a statement that explicitly denies permission to any Amazon S3 actions except `PutObject` on any Amazon S3 resource in the AWS account.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      }
   ]
}
```

### Restricting access to Amazon S3 buckets within a specific AWS account

If you want to ensure that your Amazon S3 principals are accessing only the resources that are inside of a trusted AWS account, you can restrict access. For example, this identity-based IAM policy uses a Deny effect to block access to Amazon S3 actions, unless the Amazon S3 resource that's being accessed is in account *222222222222*. To prevent an IAM principal in an AWS account from accessing Amazon S3 objects outside of the account, attach the following IAM policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
```

```
  ]
}
```

> **Note**
>
> This policy doesn't replace your existing IAM access controls, because it doesn't grant any access. Instead, this policy acts as an additional guardrail for your other IAM permissions, regardless of the permissions granted through other IAM policies.

Make sure to replace account ID *222222222222* in the policy with your own AWS account. To apply a policy to multiple accounts while still maintaining this restriction, replace the account ID with the `aws:PrincipalAccount` condition key. This condition requires that the principal and the resource must be in the same account.

**Restricting access to Amazon S3 buckets within your organizational unit**

If you have an organizational unit (OU) set up in AWS Organizations, you might want to restrict Amazon S3 bucket access to a specific part of your organization. In this example, we'll use the `aws:ResourceOrgPaths` key to restrict Amazon S3 bucket access to an OU in your organization. For this example, the OU ID is *ou-acroot-exampleou*. Make sure to replace this value in your own policy with your own OU IDs.

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowS3AccessOutsideMyBoundary",
       "Effect": "Allow",
       "Action": [
         "s3:*"
       ],
       "Resource": "*",
       "Condition": {
         "ForAllValues:StringNotLike": {
           "aws:ResourceOrgPaths": [
             "o-acorg/r-acroot/ou-acroot-exampleou/"
           ]
         }
       }
```

```
    }
  ]
}
```

> **Note**
>
> This policy doesn't grant any access. Instead, this policy acts as a backstop for your other IAM permissions, preventing your principals from accessing Amazon S3 objects outside of an OU-defined boundary.

The policy denies access to Amazon S3 actions unless the Amazon S3 object that's being accessed is in the *ou-acroot-exampleou* OU in your organization. The IAM policy condition requires `aws:ResourceOrgPaths`, a multivalued condition key, to contain any of the listed OU paths. The policy uses the `ForAllValues:StringNotLike` operator to compare the values of `aws:ResourceOrgPaths` to the listed OUs without case-sensitive matching.

**Restricting access to Amazon S3 buckets within your organization**

To restrict access to Amazon S3 objects within your organization, attach an IAM policy to the root of the organization, applying it to all accounts in your organization. To require your IAM principals to follow this rule, use a service-control policy (SCP). If you choose to use an SCP, make sure to thoroughly test the SCP before attaching the policy to the root of the organization.

In the following example policy, access is denied to Amazon S3 actions unless the Amazon S3 object that's being accessed is in the same organization as the IAM principal that is accessing it:

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "arn:aws:s3:::*/*",
       "Condition": {
         "StringNotEquals": {
           "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
```

```
        }
      }
    }
  ]
}
```

> **Note**
>
> This policy doesn't grant any access. Instead, this policy acts as a backstop for your other IAM permissions, preventing your principals from accessing any Amazon S3 objects outside of your organization. This policy also applies to Amazon S3 resources that are created after the policy is put into effect.

The IAM policy condition in this example requires `aws:ResourceOrgID` and `aws:PrincipalOrgID` to be equal to each other. With this requirement, the principal making the request and the resource being accessed must be in the same organization.

**Granting permission to retrieve the PublicAccessBlock configuration for an AWS account**

The following example identity-based policy grants the `s3:GetAccountPublicAccessBlock` permission to a user. For these permissions, you set the `Resource` value to `"*"`. For information about resource ARNs, see Policy resources for Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

### Restricting bucket creation to one Region

Suppose that an AWS account administrator wants to grant its user (Dave) permission to create a bucket in the South America (São Paulo) Region only. The account administrator can attach the following user policy granting the `s3:CreateBucket` permission with a condition as shown. The key-value pair in the `Condition` block specifies the `s3:LocationConstraint` key and the `sa-east-1` Region as its value.

> **Note**
>
> In this example, the bucket owner is granting permission to one of its users, so either a bucket policy or a user policy can be used. This example shows a user policy.

For a list of Amazon S3 Regions, see Regions and Endpoints in the *AWS General Reference*.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

### Add explicit deny

The preceding policy restricts the user from creating a bucket in any other Region except `sa-east-1`. However, some other policy might grant this user permission to create buckets in another Region. For example, if the user belongs to a group, the group might have a policy attached to it that allows all users in the group permission to create buckets in another Region. To ensure that the user doesn't get permission to create buckets in any other Region, you can add an explicit deny statement in the above policy.

The Deny statement uses the StringNotLike condition. That is, a create bucket request is denied if the location constraint is not sa-east-1. The explicit deny doesn't allow the user to create a bucket in any other Region, no matter what other permission the user gets. The following policy includes an explicit deny statement.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringNotLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

### Test the policy with the AWS CLI

You can test the policy using the following create-bucket AWS CLI command. This example uses the bucketconfig.txt file to specify the location constraint. Note the Windows file path. You need to update the bucket name and path as appropriate. You must provide user credentials using the --profile parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-
configuration file://c:/Users/someUser/bucketconfig.txt
```

The `bucketconfig.txt` file specifies the configuration as follows.

```
{"LocationConstraint": "sa-east-1"}
```

### Controlling access to a bucket with user policies

This walkthrough explains how user permissions work with Amazon S3. In this example, you create a bucket with folders. You then create AWS Identity and Access Management IAM users in your AWS account and grant those users incremental permissions on your Amazon S3 bucket and the folders in it.

#### Topics



### Basics of buckets and folders

The Amazon S3 data model is a flat structure: You create a bucket, and the bucket stores objects. There is no hierarchy of subbuckets or subfolders, but you can emulate a folder hierarchy. Tools like the Amazon S3 console can present a view of these logical folders and subfolders in your bucket.

The console shows that a bucket named companybucket has three folders, Private, Development, and Finance, and an object, s3-dg.pdf. The console uses the object names (keys) to create a logical hierarchy with folders and subfolders. Consider the following examples:

- When you create the Development folder, the console creates an object with the key Development/. Note the trailing slash (/) delimiter.
- When you upload an object named Projects1.xls in the Development folder, the console uploads the object and gives it the key Development/Projects1.xls.

  In the key, Development is the prefix and / is the delimiter. The Amazon S3 API supports prefixes and delimiters in its operations. For example, you can get a list of all objects from a bucket with a specific prefix and delimiter. On the console, when you open the Development folder, the console lists the objects in that folder. In the following example, the Development folder contains one object.

  When the console lists the Development folder in the companybucket bucket, it sends a request to Amazon S3 in which it specifies a prefix of Development and a delimiter of / in the request. The console's response looks just like a folder list in your computer's file system. The preceding example shows that the bucket companybucket has an object with the key Development/Projects1.xls.

The console is using object keys to infer a logical hierarchy. Amazon S3 has no physical hierarchy. Amazon S3 only has buckets that contain objects in a flat file structure. When you create objects using the Amazon S3 API, you can use object keys that imply a logical hierarchy. When you create a logical hierarchy of objects, you can manage access to individual folders, as this walkthrough demonstrates.

Before you start, be sure that you are familiar with the concept of the *root-level* bucket content. Suppose that your companybucket bucket has the following objects:

- Private/privDoc1.txt
- Private/privDoc2.zip
- Development/project1.xls
- Development/project2.xls
- Finance/Tax2011/document1.pdf
- Finance/Tax2011/document2.pdf

- `s3-dg.pdf`

These object keys create a logical hierarchy with `Private`, `Development`, and the `Finance` as root-level folders and `s3-dg.pdf` as a root-level object. When you choose the bucket name on the Amazon S3 console, the root-level items appear. The console shows the top-level prefixes (`Private/`, `Development/`, and `Finance/`) as root-level folders. The object key `s3-dg.pdf` has no prefix, and so it appears as a root-level item.

### Walkthrough summary

In this walkthrough, you create a bucket with three folders (`Private`, `Development`, and `Finance`) in it.

You have two users, Alice and Bob. You want Alice to access only the `Development` folder, and you want Bob to access only the `Finance` folder. You want to keep the `Private` folder content private. In the walkthrough, you manage access by creating IAM users (the example uses the usernames Alice and Bob) and granting them the necessary permissions.

IAM also supports creating user groups and granting group-level permissions that apply to all users in the group. This helps you better manage permissions. For this exercise, both Alice and Bob need some common permissions. So you also create a group named `Consultants` and then add both Alice and Bob to the group. You first grant permissions by attaching a group policy to the group. Then you add user-specific permissions by attaching policies to specific users.

> **Note**
>
> The walkthrough uses `companybucket` as the bucket name, Alice and Bob as the IAM users, and `Consultants` as the group name. Because Amazon S3 requires that bucket names be globally unique, you must replace the bucket name with a name that you create.

### Preparing for the walkthrough

In this example, you use your AWS account credentials to create IAM users. Initially, these users have no permissions. You incrementally grant these users permissions to perform specific Amazon S3 actions. To test these permissions, you sign in to the console with each user's credentials. As you incrementally grant permissions as an AWS account owner and test permissions as an IAM user, you need to sign in and out, each time using different credentials. You can do this testing with one

browser, but the process will go faster if you can use two different browsers. Use one browser to connect to the AWS Management Console with your AWS account credentials and another browser to connect with the IAM user credentials.

To sign in to the AWS Management Console with your AWS account credentials, go to https://console.aws.amazon.com/. An IAM user can't sign in using the same link. An IAM user must use an IAM-enabled sign-in page. As the account owner, you can provide this link to your users.

For more information about IAM, see The AWS Management Console Sign-in Page in the *IAM User Guide*.

**To provide a sign-in link for IAM users**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the **Navigation** pane, choose **IAM Dashboard** .
3. Note the URL under **IAM users sign in link:**. You will give this link to IAM users to sign in to the console with their IAM user name and password.

### Step 1: Create a bucket

In this step, you sign in to the Amazon S3 console with your AWS account credentials, create a bucket, add folders to the bucket, and upload one or two sample documents in each folder.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket.

   For step-by-step instructions, see Creating a bucket.
3. Upload one document to the bucket.

   This exercise assumes that you have the `s3-dg.pdf` document at the root level of this bucket. If you upload a different document, substitute its file name for `s3-dg.pdf`.
4. Add three folders named `Private`, `Finance`, and `Development` to the bucket.

   For step-by-step instructions to create a folder, see Organizing objects in the Amazon S3 console by using folders> in the *Amazon Simple Storage Service User Guide*.
5. Upload one or two documents to each folder.

For this exercise, assume that you have uploaded a couple of documents in each folder, resulting in the bucket having objects with the following keys:

- `Private/privDoc1.txt`
- `Private/privDoc2.zip`
- `Development/project1.xls`
- `Development/project2.xls`
- `Finance/Tax2011/document1.pdf`
- `Finance/Tax2011/document2.pdf`
- `s3-dg.pdf`

For step-by-step instructions, see Uploading objects.

### Step 2: Create IAM users and a group

Now use the IAM Console to add two IAM users, Alice and Bob, to your AWS account. For step-by-step instructions, see Creating an IAM user in your AWS account in the *IAM User Guide*.

Also create an administrative group named `Consultants`. Then add both users to the group. For step-by-step instructions, see Creating IAM user groups.

> **Warning**
>
> When you add users and a group, do not attach any policies that grant permissions to these users. At first, these users don't have any permissions. In the following sections, you grant permissions incrementally. First you must ensure that you have assigned passwords to these IAM users. You use these user credentials to test Amazon S3 actions and verify that the permissions work as expected.

For step-by-step instructions for creating a new IAM user, see Creating an IAM user in your AWS account in the *IAM User Guide*. When you create the users for this walkthrough, select **AWS Management Console access** and clear programmatic access.

For step-by-step instructions for creating an administrative group, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.

### Step 3: Verify that IAM users have no permissions

If you are using two browsers, you can now use the second browser to sign in to the console using one of the IAM user credentials.

1. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console using either of the IAM user credentials.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Verify the console message telling you that access is denied.

Now, you can begin granting incremental permissions to the users. First, you attach a group policy that grants permissions that both users must have.

### Step 4: Grant group-level permissions

You want the users to be able to do the following:

- List all buckets owned by the parent account. To do so, Bob and Alice must have permission for the `s3:ListAllMyBuckets` action.
- List root-level items, folders, and objects in the `companybucket` bucket. To do so, Bob and Alice must have permission for the `s3:ListBucket` action on the `companybucket` bucket.

First, you create a policy that grants these permissions, and then you attach it to the `Consultants` group.

### Step 4.1: Grant permission to list all buckets

In this step, you create a managed policy that grants the users minimum permissions to enable them to list all buckets owned by the parent account. Then you attach the policy to the `Consultants` group. When you attach the managed policy to a user or a group, you grant the user or group permission to obtain a list of buckets owned by the parent AWS account.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

> **Note**
>
> Because you are granting user permissions, sign in using your AWS account credentials, not as an IAM user.

2. Create the managed policy.

   a. In the navigation pane on the left, choose **Policies**, and then choose **Create Policy**.

   b. Choose the **JSON** tab.

   c. Copy the following access policy and paste it into the policy text field.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    }
  ]
}
```

   A policy is a JSON document. In the document, a `Statement` is an array of objects, each describing a permission using a collection of name-value pairs. The preceding policy describes one specific permission. The `Action` specifies the type of access. In the policy, the `s3:ListAllMyBuckets` is a predefined Amazon S3 action. This action covers the Amazon S3 GET Service operation, which returns a list of all buckets owned by the authenticated sender. The `Effect` element value determines whether specific permission is allowed or denied.

   d. Choose **Review Policy**. On the next page, enter `AllowGroupToSeeBucketListInTheConsole` in the **Name** field, and then choose **Create policy**.

> **Note**
>
> The **Summary** entry displays a message stating that the policy does not grant any permissions. For this walkthrough, you can safely ignore this message.

3. Attach the `AllowGroupToSeeBucketListInTheConsole` managed policy that you created to the `Consultants` group.

   For step-by-step instructions for attaching a managed policy, see Adding and removing IAM identity permissions in the *IAM User Guide*.

   You attach policy documents to IAM users and groups in the IAM console. Because you want both users to be able to list the buckets, you attach the policy to the group.

4. Test the permission.

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the console using any one of IAM user credentials.

   b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

      The console should now list all the buckets but not the objects in any of the buckets.

### Step 4.2: Enable users to list root-level content of a bucket

Next, you allow all users in the `Consultants` group to list the root-level `companybucket` bucket items. When a user chooses the company bucket on the Amazon S3 console, the user can see the root-level items in the bucket.

> **Note**
>
> This example uses `companybucket` for illustration. You must use the name of the bucket that you created.

To understand the request that the console sends to Amazon S3 when you choose a bucket name, the response that Amazon S3 returns, and how the console interprets the response, examine the flow a little more closely.

When you choose a bucket name, the console sends the GET Bucket (List Objects) request to Amazon S3. This request includes the following parameters:

- The `prefix` parameter with an empty string as its value.
- The `delimiter` parameter with `/` as its value.

The following is an example request.

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 returns a response that includes the following `<ListBucketResult/>` element.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

The key `s3-dg.pdf` object does not contain the slash (/) delimiter, and Amazon S3 returns the key in the `<Contents>` element. However, all other keys in the example bucket contain the `/` delimiter. Amazon S3 groups these keys and returns a `<CommonPrefixes>` element for each of the distinct prefix values `Development/`, `Finance/`, and `Private/` that is a substring from the beginning of these keys to the first occurrence of the specified `/` delimiter.

The console interprets this result and displays the root-level items as three folders and one object key.

If Bob or Alice opens the **Development** folder, the console sends the GET Bucket (List Objects) request to Amazon S3 with the `prefix` and the `delimiter` parameters set to the following values:

- The `prefix` parameter with the value `Development/`.
- The `delimiter` parameter with the "`/`" value.

In response, Amazon S3 returns the object keys that start with the specified prefix.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

The console shows the object keys.

Now, return to granting users permission to list the root-level bucket items. To list bucket content, users need permission to call the `s3:ListBucket` action, as shown in the following policy statement. To ensure that they see only the root-level content, you add a condition that users must specify an empty `prefix` in the request—that is, they are not allowed to double-click any of the root-level folders. Finally, you add a condition to require folder-style access by requiring user requests to include the `delimiter` parameter with the value "`/`".

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
```

```
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                       }
              }
}
```

When you choose a bucket on the Amazon S3 console, the console first sends the GET Bucket location request to find the AWS Region where the bucket is deployed. Then the console uses the Region-specific endpoint for the bucket to send the GET Bucket (List Objects) request. As a result, if users are going to use the console, you must grant permission for the `s3:GetBucketLocation` action as shown in the following policy statement.

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

**To enable users to list root-level bucket content**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Replace the existing `AllowGroupToSeeBucketListInTheConsole` managed policy that is attached to the `Consultants` group with the following policy, which also allows the `s3:ListBucket` action. Remember to replace *`companybucket`* in the policy `Resource` with the name of your bucket.

   For step-by-step instructions, see Editing IAM policies in the *IAM User Guide*. When following the step-by-step instructions, be sure to follow the steps for applying your changes to all principal entities that the policy is attached to.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
        {
   ```

```
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::*"  ]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
            "StringEquals":{
                  "s3:prefix":[""], "s3:delimiter":["/"]
                        }
                  }
    }
  ]
}
```

3. Test the updated permissions.

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console.

      Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. Choose the bucket that you created, and the console shows the root-level bucket items. If you choose any folders in the bucket, you won't be able to see the folder content because you haven't yet granted those permissions.

This test succeeds when users use the Amazon S3 console. When you choose a bucket on the console, the console implementation sends a request that includes the `prefix` parameter with an empty string as its value and the `delimiter` parameter with "/" as its value.

**Step 4.3: Summary of the group policy**

The net effect of the group policy that you added is to grant the IAM users Alice and Bob the following minimum permissions:

- List all buckets owned by the parent account.
- See root-level items in the `companybucket` bucket.

However, the users still can't do much. Next, you grant user-specific permissions, as follows:

- Allow Alice to get and put objects in the `Development` folder.
- Allow Bob to get and put objects in the `Finance` folder.

For user-specific permissions, you attach a policy to the specific user, not to the group. In the following section, you grant Alice permission to work in the `Development` folder. You can repeat the steps to grant similar permission to Bob to work in the `Finance` folder.

**Step 5: Grant IAM user Alice specific permissions**

Now you grant additional permissions to Alice so that she can see the content of the `Development` folder and get and put objects in that folder.

**Step 5.1: Grant IAM user Alice permission to list the development folder content**

For Alice to list the `Development` folder content, you must apply a policy to the user Alice that grants permission for the `s3:ListBucket` action on the `companybucket` bucket, provided the request includes the prefix `Development/`. You want this policy to be applied only to the user Alice, so you use an inline policy. For more information about inline policies, see Managed policies and inline policies in the *IAM User Guide*.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.
2. Create an inline policy to grant the user Alice permission to list the `Development` folder content.

   a. In the navigation pane on the left, choose **Users**.
   b. Choose the username **Alice**.
   c. On the user details page, choose the **Permissions** tab and then choose **Add inline policy**.
   d. Choose the **JSON** tab.
   e. Copy the following policy, and paste it into the policy text field.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
```

```
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{  "StringLike":{"s3:prefix":["Development/*"] }
       }
    }
  ]
}
```

f. Choose **Review Policy**. On the next page, enter a name in the **Name** field, and then choose **Create policy**.

3. Test the change to Alice's permissions:

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console.

   b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   c. On the Amazon S3 console, verify that Alice can see the list of objects in the `Development/` folder in the bucket.

      When the user chooses the `/Development` folder to see the list of objects in it, the Amazon S3 console sends the `ListObjects` request to Amazon S3 with the prefix `/Development`. Because the user is granted permission to see the object list with the prefix `Development` and delimiter `/`, Amazon S3 returns the list of objects with the key prefix `Development/`, and the console displays the list.

### Step 5.2: Grant IAM user Alice permissions to get and put objects in the development folder

For Alice to get and put objects in the `Development` folder, she needs permission to call the `s3:GetObject` and `s3:PutObject` actions. The following policy statements grant these permissions, provided that the request includes the `prefix` parameter with a value of `Development/`.

```
{
    "Sid":"AllowUserToReadWriteObjectData",
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
 }
```

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Edit the inline policy that you created in the previous step.

   a. In the navigation pane on the left, choose **Users**.

   b. Choose the user name Alice.

   c. On the user details page, choose the **Permissions** tab and expand the **Inline Policies** section.

   d. Next to the name of the policy that you created in the previous step, choose **Edit Policy**.

   e. Copy the following policy and paste it into the policy text field, replacing the existing policy.

```
{
     "Version": "2012-10-17",
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. Test the updated policy:

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign into the AWS Management Console.

b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

c. On the Amazon S3 console, verify that Alice can now add an object and download an object in the `Development` folder.

**Step 5.3: Explicitly deny IAM user Alice permissions to any other folders in the bucket**

User Alice can now list the root-level content in the `companybucket` bucket. She can also get and put objects in the `Development` folder. If you really want to tighten the access permissions, you could explicitly deny Alice access to any other folders in the bucket. If there is any other policy (bucket policy or ACL) that grants Alice access to any other folders in the bucket, this explicit deny overrides those permissions.

You can add the following statement to the user Alice policy that requires all requests that Alice sends to Amazon S3 to include the `prefix` parameter, whose value can be either `Development/*` or an empty string.

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
    }
}
```

There are two conditional expressions in the `Condition` block. The result of these conditional expressions is combined by using the logical `AND`. If both conditions are true, the result of the combined condition is true. Because the `Effect` in this policy is `Deny`, when the `Condition` evaluates to true, users can't perform the specified `Action`.

- The `Null` conditional expression ensures that requests from Alice include the `prefix` parameter.

  The `prefix` parameter requires folder-like access. If you send a request without the `prefix` parameter, Amazon S3 returns all the object keys.

  If the request includes the `prefix` parameter with a null value, the expression evaluates to true, and so the entire `Condition` evaluates to true. You must allow an empty string as value of the

`prefix` parameter. From the preceding discussion, recall that allowing the null string allows Alice to retrieve root-level bucket items as the console does in the preceding discussion. For more information, see Step 4.2: Enable users to list root-level content of a bucket.

- The `StringNotLike` conditional expression ensures that if the value of the `prefix` parameter is specified and is not `Development/*`, the request fails.

Follow the steps in the preceding section and again update the inline policy that you created for user Alice.

Copy the following policy and paste it into the policy text field, replacing the existing policy.

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

### Step 6: Grant IAM user Bob specific permissions

Now you want to grant Bob permission to the `Finance` folder. Follow the steps that you used earlier to grant permissions to Alice, but replace the `Development` folder with the `Finance` folder. For step-by-step instructions, see Step 5: Grant IAM user Alice specific permissions.

### Step 7: Secure the private folder

In this example, you have only two users. You granted all the minimum required permissions at the group level and granted user-level permissions only when you really need to permissions at the individual user level. This approach helps minimize the effort of managing permissions. As the number of users increases, managing permissions can become cumbersome. For example, you don't want any of the users in this example to access the content of the `Private` folder. How do you ensure that you don't accidentally grant a user permission to the `Private` folder? You add a policy that explicitly denies access to the folder. An explicit deny overrides any other permissions.

To ensure that the `Private` folder remains private, you can add the following two deny statements to the group policy:

- Add the following statement to explicitly deny any action on resources in the `Private` folder (`companybucket/Private/*`).

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
}
```

- You also deny permission for the list objects action when the request specifies the `Private/` prefix. On the console, if Bob or Alice opens the `Private` folder, this policy causes Amazon S3 to return an error response.

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
```

```
}
```

Replace the `Consultants` group policy with an updated policy that includes the preceding deny statements. After the updated policy is applied, none of the users in the group can access the `Private` folder in your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Replace the existing `AllowGroupToSeeBucketListInTheConsole` managed policy that is attached to the `Consultants` group with the following policy. Remember to replace *`companybucket`* in the policy with the name of your bucket.

   For instructions, see Editing customer managed policies in the *IAM User Guide*. When following the instructions, make sure to follow the directions for applying your changes to all principal entities that the policy is attached to.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
```

```
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

### Step 8: Clean up

To clean up, open the IAM Console and remove the users Alice and Bob. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

To ensure that you aren't charged further for storage, you should also delete the objects and the bucket that you created for this exercise.

### Related resources

- Managing IAM policies in the *IAM User Guide*

## Walkthroughs that use policies to manage access to your Amazon S3 resources

This topic provides the following introductory walkthrough examples for granting access to Amazon S3 resources. These examples use the AWS Management Console to create resources (buckets, objects, users) and grant them permissions. The examples then show you how to verify permissions using the command line tools, so you don't have to write any code. We provide commands using both the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell.

- Example 1: Bucket owner granting its users bucket permissions

  The IAM users you create in your account have no permissions by default. In this exercise, you grant a user permission to perform bucket and object operations.

- Example 2: Bucket owner granting cross-account bucket permissions

  In this exercise, a bucket owner, Account A, grants cross-account permissions to another AWS account, Account B. Account B then delegates those permissions to users in its account.

- **Managing object permissions when the object and bucket owners are not the same**

  The example scenarios in this case are about a bucket owner granting object permissions to others, but not all objects in the bucket are owned by the bucket owner. What permissions does the bucket owner need, and how can it delegate those permissions?

  The AWS account that creates a bucket is called the *bucket owner*. The owner can grant other AWS accounts permission to upload objects, and the AWS accounts that create objects own them. The bucket owner has no permissions on those objects created by other AWS accounts. If the bucket owner writes a bucket policy granting access to objects, the policy doesn't apply to objects that are owned by other accounts.

  In this case, the object owner must first grant permissions to the bucket owner using an object ACL. The bucket owner can then delegate those object permissions to others, to users in its own account, or to another AWS account, as illustrated by the following examples.

  - Example 3: Bucket owner granting permissions to objects it does not own

    In this exercise, the bucket owner first gets permissions from the object owner. The bucket owner then delegates those permissions to users in its own account.

  - Example 4 - Bucket owner granting cross-account permission to objects it does not own

After receiving permissions from the object owner, the bucket owner can't delegate permission to other AWS accounts because cross-account delegation isn't supported (see Permission delegation). Instead, the bucket owner can create an IAM role with permissions to perform specific operations (such as get object) and allow another AWS account to assume that role. Anyone who assumes the role can then access objects. This example shows how a bucket owner can use an IAM role to enable this cross-account delegation.

### Before you try the example walkthroughs

These examples use the AWS Management Console to create resources and grant permissions. To test permissions, the examples use the command line tools, AWS CLI, and AWS Tools for Windows PowerShell, so you don't need to write any code. To test permissions, you must set up one of these tools. For more information, see Setting up the tools for the walkthroughs.

In addition, when creating resources, these examples don't use root user credentials of an AWS account. Instead, you create an administrator user in these accounts to perform these tasks.

### About using an administrator user to create resources and grant permissions

AWS Identity and Access Management (IAM) recommends not using the root user credentials of your AWS account to make requests. Instead, create an IAM user or role, grant them full access, and then use their credentials to make requests. We refer to this as an administrative user or role. For more information, go to AWS account root user credentials and IAM identities in the *AWS General Reference* and IAM Best Practices in the *IAM User Guide*.

All example walkthroughs in this section use the administrator user credentials. If you have not created an administrator user for your AWS account, the topics show you how.

To sign in to the AWS Management Console using the user credentials, you must use the IAM user sign-In URL. The IAM Console provides this URL for your AWS account. The topics show you how to get the URL.

### Setting up the tools for the walkthroughs

The introductory examples (see Walkthroughs that use policies to manage access to your Amazon S3 resources) use the AWS Management Console to create resources and grant permissions. To test permissions, the examples use the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code. To test permissions, you must set up one of these tools.

**To set up the AWS CLI**

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Install or update to the latest version of the AWS Command Line Interface

   Get started with the AWS Command Line Interface

2. Set the default profile.

   You store user credentials in the AWS CLI config file. Create a default profile in the config file using your AWS account credentials. For instructions on finding and editing your AWS CLI config file, see Configuration and credential file settings.

   ```
   [default]
   aws_access_key_id = access key ID
   aws_secret_access_key = secret access key
   region = us-west-2
   ```

3. Verify the setup by entering the following command at the command prompt. Both these commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try the `help` command.

     ```
     aws help
     ```

   - To get a list of buckets on the configured account, use the `aws s3 ls` command.

     ```
     aws s3 ls
     ```

As you go through the walkthroughs, you will create users, and you will save user credentials in the config files by creating profiles, as the following example shows. These profiles have the names of `AccountAadmin` and `AccountBadmin`.

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2
```

```
[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

To run a command using these user credentials, you add the `--profile` parameter specifying the profile name. The following AWS CLI command retrieves a listing of objects in *examplebucket* and specifies the `AccountBadmin` profile.

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

Alternatively, you can configure one set of user credentials as the default profile by changing the `AWS_DEFAULT_PROFILE` environment variable from the command prompt. After you've done this, whenever you perform AWS CLI commands without the `--profile` parameter, the AWS CLI uses the profile you set in the environment variable as the default profile.

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

**To set up AWS Tools for Windows PowerShell**

1. Download and configure the AWS Tools for Windows PowerShell. For instructions, go to Installing the AWS Tools for Windows PowerShell in the *AWS Tools for Windows PowerShell User Guide*.

   > **Note**
   >
   > To load the AWS Tools for Windows PowerShell module, you must enable PowerShell script execution. For more information, see Enable Script Execution in the *AWS Tools for Windows PowerShell User Guide*.

2. For these walkthroughs, you specify AWS credentials per session using the `Set-AWSCredentials` command. The command saves the credentials to a persistent store (`-StoreAs` parameter).

   ```
   Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -
   storeas string
   ```

3. Verify the setup.

- To retrieve a list of available commands that you can use for Amazon S3 operations, run the `Get-Command` command.

```
Get-Command -module awspowershell -noun s3* -StoredCredentials string
```

- To retrieve a list of objects in a bucket, run the `Get-S3Object` command.

```
Get-S3Object -BucketName bucketname -StoredCredentials string
```

For a list of commands, see AWS Tools for PowerShell Cmdlet Reference.

Now you're ready to try the walkthroughs. Follow the links provided at the beginning of each section.

### Example 1: Bucket owner granting its users bucket permissions

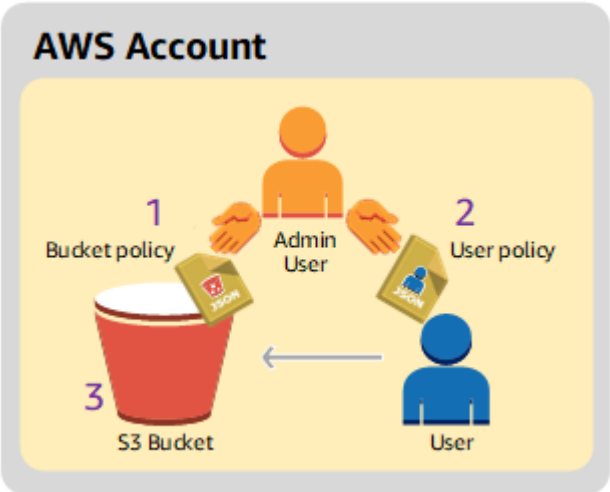
> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users.For more information about how to grant permissions to IAM roles, see Understanding cross-account permissions and using IAM roles.

**Topics**



In this walkthrough, an AWS account owns a bucket, and the account includes an IAM user By default, the user has no permissions. For the user to perform any tasks, the parent account must grant them permissions. The bucket owner and parent account are the same. Therefore, to grant the user permissions on the bucket, the AWS account can use a bucket policy, a user policy, or both. The account owner will grant some permissions using a bucket policy and other permissions using a user policy.

The following steps summarize the walkthrough:




1. Account administrator creates a bucket policy granting a set of permissions to the user.
2. Account administrator attaches a user policy to the user granting additional permissions.
3. User then tries permissions granted via both the bucket policy and the user policy.

For this example, you will need an AWS account. Instead of using the root user credentials of the account, you will create an administrator user (see About using an administrator user to create resources and grant permissions). We refer to the AWS account and the administrator user as shown in the following table.

| Account ID | Account referred to as | Administrator user in the account |
|---|---|---|
| *1111-1111-1111* | Account A | AccountAadmin |

> **Note**
>
> The administrator user in this example is **AccountAadmin**, which refers to Account A, and not **AccountAdmin**.

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

**Preparing for the walkthrough**

1. Make sure you have an AWS account and that it has a user with administrator privileges.

   a. Sign up for an AWS account, if needed. We refer to this account as Account A.

      i. Go to https://aws.amazon.com/s3 and choose **Create an AWS account**.

      ii. Follow the on-screen instructions.

      AWS will notify you by email when your account is active and available for you to use.

   b. In Account A, create an administrator user **`AccountAadmin`**. Using Account A credentials, sign in to the IAM console and do the following:

      i. Create user **`AccountAadmin`** and note the user security credentials.

      For instructions, see Creating an IAM user in your AWS account in the *IAM User Guide*.

      ii. Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access.

For instructions, see Managing IAM policies in the *IAM User Guide*.

iii. Note the **IAM user Sign-In URL** for **AccountAadmin**. You will need to use this URL when signing in to the AWS Management Console. For more information about where to find the sign-in URL, see Sign in to the AWS Management Console as an IAM user in *IAM User Guide*. Note the URL for each of the accounts.

2. Set up either the AWS CLI or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

   - If using the AWS CLI, create a profile, `AccountAadmin`, in the config file.
   - If using the AWS Tools for Windows PowerShell, make sure you store credentials for the session as `AccountAadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

**Step 1: Create resources in Account A and grant permissions**

Using the credentials of user `AccountAadmin` in Account A, and the special IAM user sign-in URL, sign in to the AWS Management Console and do the following:

1. Create resources of a bucket and an IAM user

   a. In the Amazon S3 console, create a bucket. Note the AWS Region in which you created the bucket. For instructions, see Creating a bucket.

   b. In the IAM Console, do the following:

      i. Create a user named Dave.

      For step-by-step instructions, see Creating IAM users (console) in the *IAM User Guide*.

      ii. Note the `UserDave` credentials.

      iii. Note the Amazon Resource Name (ARN) for user Dave. In the IAM Console, select the user, and the **Summary** tab provides the user ARN.

2. Grant permissions.

   Because the bucket owner and the parent account to which the user belongs are the same, the AWS account can grant user permissions using a bucket policy, a user policy, or both. In this

example, you do both. If the object is also owned by the same account, the bucket owner can grant object permissions in the bucket policy (or an IAM policy).

a. In the Amazon S3 console, attach the following bucket policy to *awsexamplebucket1*.

The policy has two statements.

- The first statement grants Dave the bucket operation permissions `s3:GetBucketLocation` and `s3:ListBucket`.
- The second statement grants the `s3:GetObject` permission. Because Account A also owns the object, the account administrator is able to grant the `s3:GetObject` permission.

In the `Principal` statement, Dave is identified by his user ARN. For more information about policy elements, see Policies and permissions in Amazon S3.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1"
         ]
      },
      {
         "Sid": "statement2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
             "s3:GetObject"
```

```
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

b. Create an inline policy for the user Dave by using the following policy. The policy grants Dave the `s3:PutObject` permission. You need to update the policy by providing your bucket name.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

For instructions, see Managing IAMpolicies in the *IAM User Guide*. Note you need to sign in to the console using Account A credentials.

### Step 2: Test permissions

Using Dave's credentials, verify that the permissions work. You can use either of the following two procedures.

#### Test permissions using the AWS CLI

1. Update the AWS CLI config file by adding the following `UserDaveAccountA` profile. For more information, see Setting up the tools for the walkthroughs.

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. Verify that Dave can perform the operations as granted in the user policy. Upload a sample object using the following AWS CLI `put-object` command.

   The `--body` parameter in the command identifies the source file to upload. For example, if the file is in the root of the C: drive on a Windows machine, you specify `c:\HappyFace.jpg`. The `--key` parameter provides the key name for the object.

   ```
   aws s3api put-object --bucket awsexamplebucket1 --key HappyFace.jpg --
   body HappyFace.jpg --profile UserDaveAccountA
   ```

   Run the following AWS CLI command to get the object.

   ```
   aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
    --profile UserDaveAccountA
   ```

**Test permissions using the AWS Tools for Windows PowerShell**

1. Store Dave's credentials as `AccountADave`. You then use these credentials to PUT and GET an object.

   ```
   set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
    AccountADave
   ```

2. Upload a sample object using the AWS Tools for Windows PowerShell `Write-S3Object` command using user Dave's stored credentials.

   ```
   Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
    -StoredCredentials AccountADave
   ```

   Download the previously uploaded object.

```
Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

### Example 2: Bucket owner granting cross-account bucket permissions

> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users. To learn how to do this, see Understanding cross-account permissions and using IAM roles.

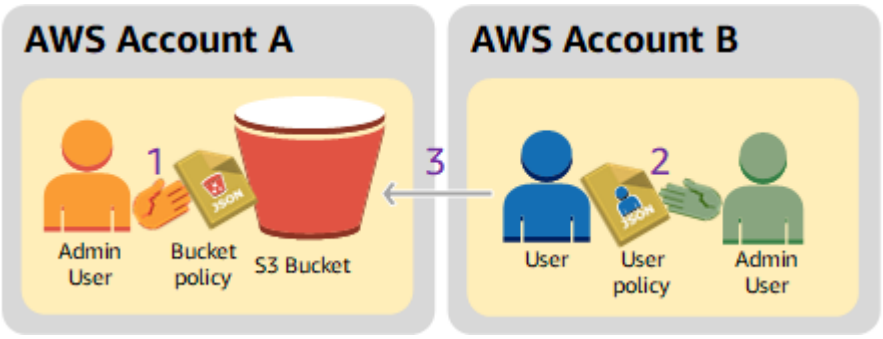
**Topics**



An AWS account—for example, Account A—can grant another AWS account, Account B, permission to access its resources such as buckets and objects. Account B can then delegate those permissions to users in its account. In this example scenario, a bucket owner grants cross-account permission to another account to perform specific bucket operations.

> **Note**
>
> Account A can also directly grant a user in Account B permissions using a bucket policy. However, the user will still need permission from the parent account, Account B, to which the user belongs, even if Account B doesn't have permissions from Account A. As long as the user has permission from both the resource owner and the parent account, the user will be able to access the resource.

The following is a summary of the walkthrough steps:




1. Account A administrator user attaches a bucket policy granting cross-account permissions to Account B to perform specific bucket operations.

   Note that administrator user in Account B will automatically inherit the permissions.
2. Account B administrator user attaches user policy to the user delegating the permissions it received from Account A.
3. User in Account B then verifies permissions by accessing an object in the bucket owned by Account A.

For this example, you need two accounts. The following table shows how we refer to these accounts and the administrator users in them. In accordance with the IAM guidelines (see About using an administrator user to create resources and grant permissions), we don't use the root user credentials in this walkthrough. Instead, you create an administrator user in each account and use those credentials when creating resources and granting them permissions.

| AWS account ID | Account referred to as | Administrator user in the account |
|---|---|---|
| *1111-1111-1111* | Account A | AccountAadmin |
| *2222-2222-2222* | Account B | AccountBadmin |

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

### Preparing for the walkthrough

1. Make sure you have two AWS accounts and that each account has one administrator user as shown in the table in the preceding section.

   a. Sign up for an AWS account, if needed.

   b. Using Account A credentials, sign in to the IAM console to create the administrator user:

      i. Create user **`AccountAadmin`** and note the security credentials. For instructions, see Creating an IAM user in Your AWS account in the *IAM User Guide*.

      ii. Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access. For instructions, see Working with Policies in the *IAM User Guide*.

   c. While you are in the IAM console, note the **IAM user Sign-In URL** on the **Dashboard**. All users in the account must use this URL when signing in to the AWS Management Console.

      For more information, see How Users Sign in to Your Account in *IAM User Guide*.

   d. Repeat the preceding step using Account B credentials and create administrator user **`AccountBadmin`**.

2. Set up either the AWS Command Line Interface (AWS CLI) or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

   - If using the AWS CLI, create two profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
   - If using the AWS Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

3. Save the administrator user credentials, also referred to as profiles. You can use the profile name instead of specifying credentials for each command you enter. For more information, see Setting up the tools for the walkthroughs.

   a. Add profiles in the AWS CLI credentials file for each of the administrator users, `AccountAadmin` and `AccountBadmin`, in the two accounts.

```
[AccountAadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
```

```
region = us-east-1

[AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

b. If you're using the AWS Tools for Windows PowerShell, run the following command.

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-
access-key -storeas AccountAadmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-
access-key -storeas AccountBadmin
```

**Step 1: Do the Account A tasks**

**Step 1.1: Sign in to the AWS Management Console**

Using the IAM user sign-in URL for Account A, first sign in to the AWS Management Console as **AccountAadmin** user. This user will create a bucket and attach a policy to it.

**Step 1.2: Create a bucket**

1. In the Amazon S3 console, create a bucket. This exercise assumes the bucket is created in the US East (N. Virginia) AWS Region and is named *amzn-s3-demo-bucket*.

   For instructions, see Creating a bucket.
2. Upload a sample object to the bucket.

   For instructions, go to Step 2: Upload an object to your bucket.

**Step 1.3: Attach a bucket policy to grant cross-account permissions to Account B**

The bucket policy grants the `s3:GetLifecycleConfiguration` and `s3:ListBucket` permissions to Account B. It's assumed that you're still signed in to the console using **AccountAadmin** user credentials.

1. Attach the following bucket policy to *amzn-s3-demo-bucket*. The policy grants Account B permission for the `s3:GetLifecycleConfiguration` and `s3:ListBucket` actions.

   For instructions, see Adding a bucket policy by using the Amazon S3 console.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. Verify that Account B (and thus its administrator user) can perform the operations.

   - Verify using the AWS CLI

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
     profile AccountBadmin
     ```

   - Verify using the AWS Tools for Windows PowerShell

     ```
     get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBadmin
     get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
     StoredCredentials AccountBadmin
     ```

### Step 2: Do the Account B tasks

Now the Account B administrator creates a user, Dave, and delegates the permissions received from Account A.

### Step 2.1: Sign in to the AWS Management Console

Using the IAM user sign-in URL for Account B, first sign in to the AWS Management Console as **AccountBadmin** user.

### Step 2.2: Create user Dave in Account B

In the IAM Console, create a user, **Dave**.

For instructions, see Creating IAM users (console) in the *IAM User Guide*.

### Step 2.3: Delegate permissions to user Dave

Create an inline policy for the user Dave by using the following policy. You will need to update the policy by providing your bucket name.

It's assumed that you're signed in to the console using **AccountBadmin** user credentials.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

For instructions, see Managing IAM policies in the *IAM User Guide*.

### Step 2.4: Test permissions

Now Dave in Account B can list the contents of *amzn-s3-demo-bucket* owned by Account A. You can verify the permissions using either of the following procedures.

### Test permissions using the AWS CLI

1. Add the `UserDave` profile to the AWS CLI config file. For more information about the config file, see Setting up the tools for the walkthroughs.

```
[profile UserDave]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. At the command prompt, enter the following AWS CLI command to verify Dave can now get an object list from the *amzn-s3-demo-bucket* owned by Account A. Note the command specifies the `UserDave` profile.

```
aws s3 ls s3://amzn-s3-demo-bucket --profile UserDave
```

Dave doesn't have any other permissions. So, if he tries any other operation—for example, the following `get-bucket-lifecycle` configuration—Amazon S3 returns permission denied.

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --profile
 UserDave
```

### Test permissions using AWS Tools for Windows PowerShell

1. Store Dave's credentials as `AccountBDave`.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. Try the List Bucket command.

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
```

Dave doesn't have any other permissions. So, if he tries any other operation—for example, the following `get-s3bucketlifecycleconfiguration`—Amazon S3 returns permission denied.

```
get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
StoredCredentials AccountBDave
```

### Step 3: (Optional) Try explicit deny

You can have permissions granted by using an access control list (ACL), a bucket policy, or a user policy. But if there is an explicit deny set by either a bucket policy or a user policy, the explicit deny takes precedence over any other permissions. For testing, update the bucket policy and explicitly deny Account B the `s3:ListBucket` permission. The policy also grants `s3:ListBucket` permission. However, explicit deny takes precedence, and Account B or users in Account B will not be able to list objects in *amzn-s3-demo-bucket*.

1. Using credentials of user `AccountAadmin` in Account A, replace the bucket policy by the following.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      },
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
```

```
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
         }
      ]
}
```

2. Now if you try to get a bucket list using `AccountBadmin` credentials, access is denied.

   - Using the AWS CLI, run the following command:

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     ```

   - Using the AWS Tools for Windows PowerShell, run the following command:

     ```
     get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
     ```

**Step 4: Clean up**

1. After you're done testing, you can do the following to clean up:

   - Sign in to the AWS Management Console (AWS Management Console) using Account A credentials, and do the following:

     - In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket*. In the bucket **Properties**, delete the policy in the **Permissions** section.
     - If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
     - In the IAM Console, remove the `AccountAadmin` user.

2. Sign in to the IAM Console using Account B credentials. Delete user `AccountBadmin`. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

### Example 3: Bucket owner granting permissions to objects it does not own

> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users. To learn how to do this, see Understanding cross-account permissions and using IAM roles.

**Topics**



The scenario for this example is that a bucket owner wants to grant permission to access objects, but the bucket owner doesn't own all objects in the bucket. For this example, the bucket owner is trying to grant permission to users in its own account.

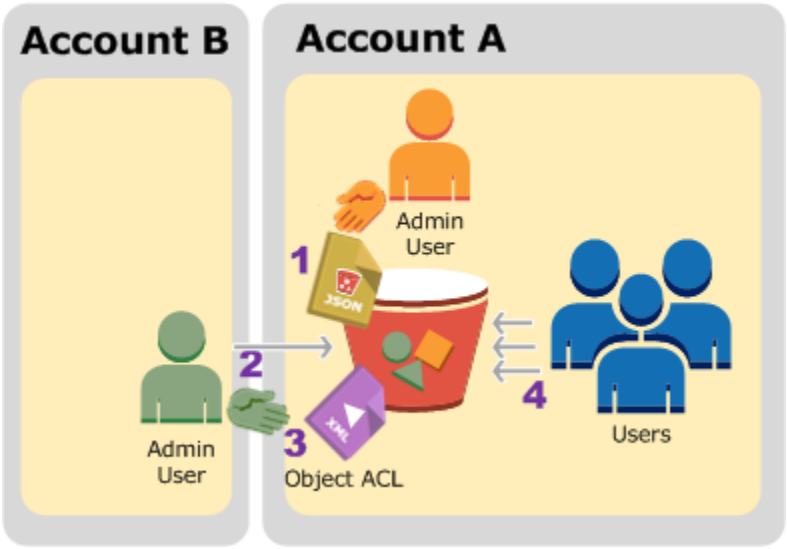
A bucket owner can enable other AWS accounts to upload objects. By default, the bucket owner doesn't own objects written to a bucket by another AWS account. Objects are owned by the accounts that write them to an S3 bucket. If the bucket owner doesn't own objects in the bucket, the object owner must first grant permission to the bucket owner using an object access control list (ACL). Then, the bucket owner can grant permissions to an object that they don't own. For more information, see Amazon S3 bucket and object ownership.

If the bucket owner applies the bucket owner enforced setting for S3 Object Ownership for the bucket, the bucket owner will own all objects in the bucket, including objects written by another AWS account. This approach resolves the issue that objects are not owned by the bucket owner. Then, you can delegate permission to users in your own account or to other AWS accounts.

> **Note**
>
> S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are

disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.
A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

In this example, we assume the bucket owner has not applied the bucket owner enforced setting for Object Ownership. The bucket owner delegates permission to users in its own account. The following is a summary of the walkthrough steps:




1. Account A administrator user attaches a bucket policy with two statements.
   - Allow cross-account permission to Account B to upload objects.
   - Allow a user in its own account to access objects in the bucket.
2. Account B administrator user uploads objects to the bucket owned by Account A.
3. Account B administrator updates the object ACL adding grant that gives the bucket owner full-control permission on the object.
4. User in Account A verifies by accessing objects in the bucket, regardless of who owns them.

For this example, you need two accounts. The following table shows how we refer to these accounts and the administrator users in these accounts. In this walkthrough, you don't use

the account root user credentials, according to the recommended IAM guidelines. For more information, see About using an administrator user to create resources and grant permissions. Instead, you create an administrator in each account and use those credentials in creating resources and granting them permissions.

| AWS account ID | Account referred to as | Administrator in the account |
|---|---|---|
| *1111-1111-1111* | Account A | AccountAadmin |
| *2222-2222-2222* | Account B | AccountBadmin |

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

**Step 0: Preparing for the walkthrough**

1. Make sure that you have two AWS accounts and each account has one administrator as shown in the table in the preceding section.

   a. Sign up for an AWS account, if needed.

   b. Using Account A credentials, sign in to the IAM Console and do the following to create an administrator user:

      - Create user **`AccountAadmin`** and note the user's security credentials. For more information about adding users, see Creating an IAM user in your AWS account in the *IAM User Guide*.
      - Grant administrator permissions to **AccountAadmin** by attaching a user policy that gives full access. For instructions, see Managing IAM policies in the *IAM User Guide*.
      - In the IAM Console **Dashboard**, note the **IAM User Sign-In URL**. Users in this account must use this URL when signing in to the AWS Management Console. For more information, see How users sign in to your account in *IAM User Guide*.

   c. Repeat the preceding step using Account B credentials and create administrator user **`AccountBadmin`**.

2. Set up either the AWS CLI or the Tools for Windows PowerShell. Make sure that you save the administrator credentials as follows:

- If using the AWS CLI, create two profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
- If using the Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

For instructions, see Setting up the tools for the walkthroughs.

### Step 1: Do the Account A tasks

Perform the following steps for Account A:

#### Step 1.1: Sign in to the console

Using the IAM user sign-in URL for Account A, sign in to the AWS Management Console as **`AccountAadmin`** user. This user will create a bucket and attach a policy to it.

#### Step 1.2: Create a bucket and user, and add a bucket policy to grant user permissions

1. In the Amazon S3 console, create a bucket. This exercise assumes that the bucket is created in the US East (N. Virginia) AWS Region, and the name is *`amzn-s3-demo-bucket1`*.

   For instructions, see Creating a bucket.
2. In the IAM Console, create a user **`Dave`**.

   For step-by-step instructions, see Creating IAM users (console) in the *IAM User Guide*.
3. Note the user Dave credentials.
4. In the Amazon S3 console, attach the following bucket policy to *`amzn-s3-demo-bucket1`* bucket. For instructions, see Adding a bucket policy by using the Amazon S3 console. Follow the steps to add a bucket policy. For information about how to find account IDs, see Finding your AWS account ID.

   The policy grants Account B the `s3:PutObject` and `s3:ListBucket` permissions. The policy also grants user `Dave` the `s3:GetObject` permission.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```

```
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountB-ID:root"
            },
            "Action": [
               "s3:PutObject",
               "s3:ListBucket"
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket1/*",
               "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
         },
         {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
         }
      ]
}
```

### Step 2: Do the Account B tasks

Now that Account B has permissions to perform operations on Account A's bucket, the Account B administrator does the following:

- Uploads an object to Account A's bucket
- Adds a grant in the object ACL to allow Account A, the bucket owner, full control

### Using the AWS CLI

1. Using the `put-object` AWS CLI command, upload an object. The `--body` parameter in the command identifies the source file to upload. For example, if the file is on the `C:` drive of a Windows machine, specify `c:\HappyFace.jpg`. The `--key` parameter provides the key name for the object.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --body
 HappyFace.jpg --profile AccountBadmin
```

2. Add a grant to the object ACL to allow the bucket owner full control of the object. For information about how to find a canonical user ID, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --grant-
full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
```

### Using the Tools for Windows PowerShell

1. Using the `Write-S3Object` command, upload an object.

```
Write-S3Object -BucketName amzn-s3-demo-bucket1 -key HappyFace.jpg -file
 HappyFace.jpg -StoredCredentials AccountBadmin
```

2. Add a grant to the object ACL to allow the bucket owner full control of the object.

```
Set-S3ACL -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -CannedACLName
 "bucket-owner-full-control" -StoredCreden
```

### Step 3: Test permissions

Now verify that user Dave in Account A can access the object owned by Account B.

### Using the AWS CLI

1. Add user Dave credentials to the AWS CLI config file and create a new profile, `UserDaveAccountA`. For more information, see Setting up the tools for the walkthroughs.

```
[profile UserDaveAccountA]
```

```
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. Run the `get-object` CLI command to download `HappyFace.jpg` and save it locally. You provide user Dave credentials by adding the `--profile` parameter.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key
 HappyFace.jpg Outputfile.jpg --profile UserDaveAccountA
```

**Using the Tools for Windows PowerShell**

1. Store user Dave AWS credentials, as `UserDaveAccountA`, to persistent store.

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-
SecretAccessKey -storeas UserDaveAccountA
```

2. Run the `Read-S3Object` command to download the `HappyFace.jpg` object and save it locally. You provide user Dave credentials by adding the `-StoredCredentials` parameter.

```
Read-S3Object -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -file
 HappyFace.jpg  -StoredCredentials UserDaveAccountA
```

**Step 4: Clean up**

1. After you're done testing, you can do the following to clean up:

   - Sign in to the AWS Management Console using Account A credentials, and do the following:

     - In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket1*. In the bucket **Properties**, delete the policy in the **Permissions** section.
     - If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
     - In the IAM Console, remove the **AccountAadmin** user. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

2. Sign in to the AWS Management Console using Account B credentials. In the IAM Console, delete the user **AccountBadmin**.

### Example 4 - Bucket owner granting cross-account permission to objects it does not own

**Topics**



In this example scenario, you own a bucket and you have enabled other AWS accounts to upload objects. If you have applied the bucket owner enforced setting for S3 Object Ownership for the bucket, you will own all objects in the bucket, including objects written by another AWS account. This approach resolves the issue that objects are not owned by you, the bucket owner. Then, you can delegate permission to users in your own account or to other AWS accounts. Suppose the bucket owner enforced setting for S3 Object Ownership is not enabled. That is, your bucket can have objects that other AWS accounts own.

Now, suppose as a bucket owner, you need to grant cross-account permission on objects, regardless of who the owner is, to a user in another account. For example, that user could be a billing application that needs to access object metadata. There are two core issues:

- The bucket owner has no permissions on those objects created by other AWS accounts. For the bucket owner to grant permissions on objects it doesn't own, the object owner must first grant permission to the bucket owner. The object owner is the AWS account that created the objects. The bucket owner can then delegate those permissions.
- The bucket owner account can delegate permissions to users in its own account (see Example 3: Bucket owner granting permissions to objects it does not own). However, the bucket owner account can't delegate permissions to other AWS accounts because cross-account delegation isn't supported.

In this scenario, the bucket owner can create an AWS Identity and Access Management (IAM) role with permission to access objects Then, the bucket owner can grant another AWS account permission to assume the role, temporarily enabling it to access objects in the bucket.

> **Note**
>
> S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**Understanding cross-account permissions and using IAM roles**

IAM roles enable several scenarios to delegate access to your resources, and cross-account access is one of the key scenarios. In this example, the bucket owner, Account A, uses an IAM role to temporarily delegate object access cross-account to users in another AWS account, Account C. Each IAM role that you create has the following two policies attached to it:

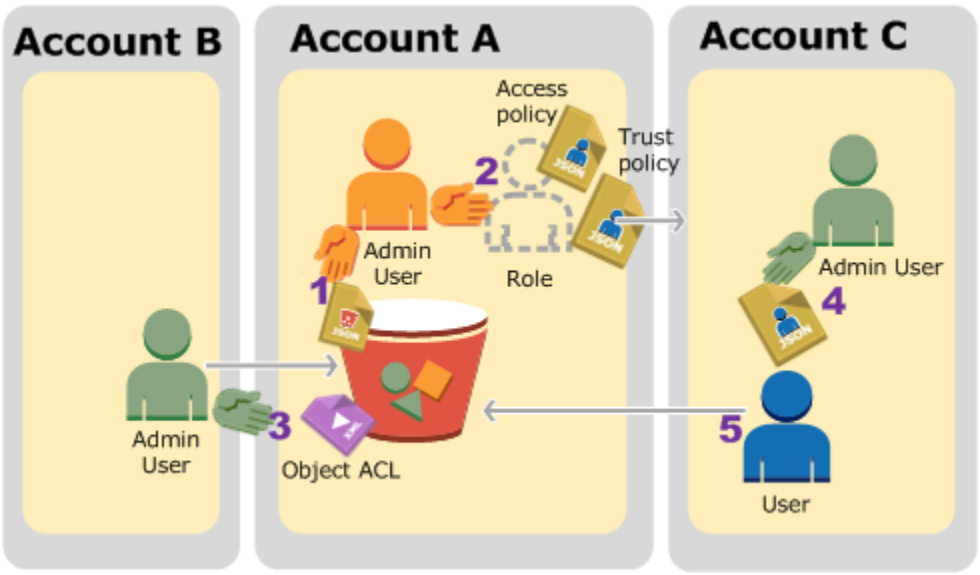
- A trust policy identifying another AWS account that can assume the role.
- An access policy defining what permissions—for example, `s3:GetObject`—are allowed when someone assumes the role. For a list of permissions you can specify in a policy, see Policy actions for Amazon S3.

The AWS account identified in the trust policy then grants its user permission to assume the role. The user can then do the following to access objects:

- Assume the role and, in response, get temporary security credentials.
- Using the temporary security credentials, access the objects in the bucket.

For more information about IAM roles, see IAM Roles in the *IAM User Guide*.

The following is a summary of the walkthrough steps:




1. Account A administrator user attaches a bucket policy granting Account B conditional permission to upload objects.
2. Account A administrator creates an IAM role, establishing trust with Account C, so users in that account can access Account A. The access policy attached to the role limits what user in Account C can do when the user accesses Account A.
3. Account B administrator uploads an object to the bucket owned by Account A, granting full-control permission to the bucket owner.
4. Account C administrator creates a user and attaches a user policy that allows the user to assume the role.
5. User in Account C first assumes the role, which returns the user temporary security credentials. Using those temporary credentials, the user then accesses objects in the bucket.

For this example, you need three accounts. The following table shows how we refer to these accounts and the administrator users in these accounts. In accordance with the IAM guidelines (see About using an administrator user to create resources and grant permissions), we don't use the AWS account root user credentials in this walkthrough. Instead, you create an administrator user in each account and use those credentials when creating resources and granting them permissions.

| AWS account ID | Account referred to as | Administrator user in the account |
|---|---|---|
| *1111-1111-1111* | Account A | AccountAadmin |

| AWS account ID | Account referred to as | Administrator user in the account |
|---|---|---|
| *2222-2222-2222* | Account B | AccountBadmin |
| *3333-3333-3333* | Account C | AccountCadmin |

### Step 0: Preparing for the walkthrough

> **Note**
>
> You might want to open a text editor, and write down some of the information as you go through the steps. In particular, you will need account IDs, canonical user IDs, IAM user Sign-in URLs for each account to connect to the console, and Amazon Resource Names (ARNs) of the IAM users, and roles.

1. Make sure that you have three AWS accounts and each account has one administrator user as shown in the table in the preceding section.

   a. Sign up for AWS accounts, as needed. We refer to these accounts as Account A, Account B, and Account C.

   b. Using Account A credentials, sign in to the IAM console and do the following to create an administrator user:

      - Create user **`AccountAadmin`** and note its security credentials. For more information about adding users, see Creating an IAM user in your AWS account in the *IAM User Guide*.
      - Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access. For instructions, see Managing IAM policies in the *IAM User Guide*.
      - In the IAM Console **Dashboard**, note the **IAM User Sign-In URL**. Users in this account must use this URL when signing in to the AWS Management Console. For more information, see Sign in to the AWS Management Console as an IAM user in the *IAM User Guide*.

   c. Repeat the preceding step to create administrator users in Account B and Account C.

2. For Account C, note the canonical user ID.

When you create an IAM role in Account A, the trust policy grants Account C permission to assume the role by specifying the account ID. You can find account information as follows:

a. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the Amazon S3 console.

b. Choose the name of an Amazon S3 bucket to view the details about that bucket.

c. Choose the **Permissions** tab and then choose **Access Control List**.

d. In the **Access for your AWS account** section, in the **Account** column is a long identifier, such as `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`. This is your canonical user ID.

3. When creating a bucket policy, you will need the following information. Note these values:

- **Canonical user ID of Account A** – When the Account A administrator grants conditional upload object permission to the Account B administrator, the condition specifies the canonical user ID of the Account A user that must get full-control of the objects.

> **Note**
>
> The canonical user ID is the Amazon S3–only concept. It is a 64-character obfuscated version of the account ID.

- **User ARN for Account B administrator** – You can find the user ARN in the IAM Console.You must select the user and find the user's ARN in the **Summary** tab.

  In the bucket policy, you grant `AccountBadmin` permission to upload objects and you specify the user using the ARN. Here's an example ARN value:

```
arn:aws:iam::AccountB-ID:user/AccountBadmin
```

4. Set up either the AWS Command Line Interface (CLI) or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

- If using the AWS CLI, create profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
- If using the AWS Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

For instructions, see Setting up the tools for the walkthroughs.

### Step 1: Do the account A tasks

In this example, Account A is the bucket owner. So user AccountAadmin in Account A will do the following:

- Create a bucket.
- Attach a bucket policy that grants the Account B administrator permission to upload objects.
- Create an IAM role that grants Account C permission to assume the role so it can access objects in the bucket.

### Step 1.1: Sign in to the AWS Management Console

Using the IAM user Sign-in URL for Account A, first sign in to the AWS Management Console as **`AccountAadmin`** user. This user will create a bucket and attach a policy to it.

### Step 1.2: Create a bucket and attach a bucket policy

In the Amazon S3 console, do the following:

1. Create a bucket. This exercise assumes the bucket name is *`amzn-s3-demo-bucket1`*.

   For instructions, see Creating a bucket.
2. Attach the following bucket policy. The policy grants conditional permission to the Account B administrator permission to upload objects.

   Update the policy by providing your own values for *`amzn-s3-demo-bucket1`*, *`AccountB-ID`*, and the *`CanonicalUserId-of-AWSaccountA-BucketOwner`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
```

```
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
    ]
}
```

### Step 1.3: Create an IAM role to allow Account C cross-account access in Account A

In the IAM Console, create an IAM role (**`examplerole`**) that grants Account C permission to assume the role. Make sure that you are still signed in as the Account A administrator because the role must be created in Account A.

1. Before creating the role, prepare the managed policy that defines the permissions that the role requires. You attach this policy to the role in a later step.

   a. In the navigation pane on the left, choose **Policies** and then choose **Create Policy**.

   b. Next to **Create Your Own Policy**, choose **Select**.

   c. Enter **`access-accountA-bucket`** in the **Policy Name** field.

   d. Copy the following access policy and paste it into the **Policy Document** field. The access policy grants the role `s3:GetObject` permission so, when the Account C user assumes the role, it can only perform the `s3:GetObject` operation.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
    }
  ]
}
```

e. Choose **Create Policy**.

The new policy appears in the list of managed policies.

2. In the navigation pane on the left, choose **Roles** and then choose **Create New Role**.
3. Under **Select Role Type**, select **Role for Cross-Account Access**, and then choose the **Select** button next to **Provide access between AWS accounts you own**.
4. Enter the Account C account ID.

   For this walkthrough, you don't need to require users to have multi-factor authentication (MFA) to assume the role, so leave that option unselected.
5. Choose **Next Step** to set the permissions that will be associated with the role.
6. Select the checkbox next to the **access-accountA-bucket** policy that you created, and then choose **Next Step**.

   The Review page appears so you can confirm the settings for the role before it's created. One very important item to note on this page is the link that you can send to your users who need to use this role. Users who use the link go straight to the **Switch Role** page with the Account ID and Role Name fields already filled in. You can also see this link later on the **Role Summary** page for any cross-account role.
7. Enter `examplerole` for the role name, and then choose **Next Step**.
8. After reviewing the role, choose **Create Role**.

   The `examplerole` role is displayed in the list of roles.
9. Choose the role name `examplerole`.
10. Select the **Trust Relationships** tab.
11. Choose **Show policy document** and verify the trust policy shown matches the following policy.

The following trust policy establishes trust with Account C, by allowing it the `sts:AssumeRole` action. For more information, see AssumeRole in the *AWS Security Token Service API Reference*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountC-ID:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

12. Note the Amazon Resource Name (ARN) of the `examplerole` role that you created.

    Later in the following steps, you attach a user policy to allow an IAM user to assume this role, and you identify the role by the ARN value.

### Step 2: Do the Account B tasks

The example bucket owned by Account A needs objects owned by other accounts. In this step, the Account B administrator uploads an object using the command line tools.

- Using the `put-object` AWS CLI command, upload an object to *amzn-s3-demo-bucket1*.

  ```
  aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
  body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
  profile AccountBadmin
  ```

  Note the following:

  - The `--Profile` parameter specifies the `AccountBadmin` profile, so the object is owned by Account B.
  - The parameter `grant-full-control` grants the bucket owner full-control permission on the object as required by the bucket policy.

- The `--body` parameter identifies the source file to upload. For example, if the file is on the C: drive of a Windows computer, you specify `c:\HappyFace.jpg`.

### Step 3: Do the Account C tasks

In the preceding steps, Account A has already created a role, `examplerole`, establishing trust with Account C. This role allows users in Account C to access Account A. In this step, the Account C administrator creates a user (Dave) and delegates him the `sts:AssumeRole` permission it received from Account A. This approach allows Dave to assume the `examplerole` and temporarily gain access to Account A. The access policy that Account A attached to the role limits what Dave can do when he accesses Account A—specifically, get objects in *`amzn-s3-demo-bucket1`*.

#### Step 3.1: Create a user in Account C and delegate permission to assume examplerole

1. Using the IAM user sign-in URL for Account C, first sign in to the AWS Management Console as **`AccountCadmin`** user.

2. In the IAM Console, create a user, Dave.

   For step-by-step instructions, see Creating IAM users (AWS Management Console) in the *IAM User Guide*.
3. Note the Dave credentials. Dave will need these credentials to assume the `examplerole` role.
4. Create an inline policy for the Dave IAM user to delegate the `sts:AssumeRole` permission to Dave on the `examplerole` role in Account A.

   a. In the navigation pane on the left, choose **Users**.
   b. Choose the user name **Dave**.
   c. On the user details page, select the **Permissions** tab and then expand the **Inline Policies** section.
   d. Choose **click here** (or **Create User Policy**).
   e. Choose **Custom Policy**, and then choose **Select**.
   f. Enter a name for the policy in the **Policy Name** field.
   g. Copy the following policy into the **Policy Document** field.

      You must update the policy by providing the *`AccountA-ID`*.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
    }
  ]
}
```

h. Choose **Apply Policy**.

5. Save Dave's credentials to the config file of the AWS CLI by adding another profile, `AccountCDave`.

```
[profile AccountCDave]
aws_access_key_id = UserDaveAccessKeyID
aws_secret_access_key = UserDaveSecretAccessKey
region = us-west-2
```

### Step 3.2: Assume role (examplerole) and access objects

Now Dave can access objects in the bucket owned by Account A as follows:

- Dave first assumes the `examplerole` using his own credentials. This will return temporary credentials.
- Using the temporary credentials, Dave will then access objects in Account A's bucket.

1. At the command prompt, run the following AWS CLI `assume-role` command using the `AccountCDave` profile.

   You must update the ARN value in the command by providing the *AccountA-ID* where `examplerole` is defined.

```
aws sts assume-role --role-arn arn:aws:iam::AccountA-ID:role/examplerole --profile
 AccountCDave --role-session-name test
```

   In response, AWS Security Token Service (AWS STS) returns temporary security credentials (access key ID, secret access key, and a session token).

2. Save the temporary security credentials in the AWS CLI config file under the `TempCred` profile.

```
[profile TempCred]
aws_access_key_id = temp-access-key-ID
aws_secret_access_key = temp-secret-access-key
aws_session_token = session-token
region = us-west-2
```

3. At the command prompt, run the following AWS CLI command to access objects using the temporary credentials. For example, the command specifies the head-object API to retrieve object metadata for the `HappyFace.jpg` object.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --
key HappyFace.jpg SaveFileAs.jpg --profile TempCred
```

Because the access policy attached to `examplerole` allows the actions, Amazon S3 processes the request. You can try any other action on any other object in the bucket.

If you try any other action—for example, `get-object-acl`—you will get permission denied because the role isn't allowed that action.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
profile TempCred
```

We used user Dave to assume the role and access the object using temporary credentials. It could also be an application in Account C that accesses objects in *amzn-s3-demo-bucket1*. The application can obtain temporary security credentials, and Account C can delegate the application permission to assume `examplerole`.

### Step 4: Clean up

1. After you're done testing, you can do the following to clean up:

   - Sign in to the AWS Management Console using Account A credentials, and do the following:

     - In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket1*. In the bucket **Properties**, delete the policy in the **Permissions** section.

- If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
- In the IAM Console, remove the `examplerole` you created in Account A. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.
- In the IAM Console, remove the **AccountAadmin** user.

2. Sign in to the IAM Console by using Account B credentials. Delete the user **AccountBadmin**.
3. Sign in to the IAM Console by using Account C credentials. Delete **AccountCadmin** and the user Dave.

### Related resources

For more information that's related to this walkthrough, see the following resources in the *IAM User Guide*:

- Creating a role to delegate permissions to an IAM user
- Tutorial: Delegate Access Across AWS accounts Using IAM Roles
- Managing IAM policies

## How Amazon S3 authorizes a request

When Amazon S3 receives a request—for example, a bucket or an object operation—it first verifies that the requester has the necessary permissions. Amazon S3 evaluates all the relevant access policies, user policies, and resource-based policies (bucket policy, bucket access control list (ACL), and object ACL) in deciding whether to authorize the request.

> **Note**
>
> If the Amazon S3 permission check fails to find valid permissions, an Access Denied (403 Forbidden)permission denied error is returned. For more information, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

To determine whether the requester has permission to perform the specific operation, Amazon S3 does the following, in order, when it receives a request:

1. Converts all the relevant access policies (user policy, bucket policy, and ACLs) at run time into a set of policies for evaluation.
2. Evaluates the resulting set of policies in the following steps. In each step, Amazon S3 evaluates a subset of policies in a specific context, based on the context authority.
   a. **User context** – In the user context, the parent account to which the user belongs is the context authority.

      Amazon S3 evaluates a subset of policies owned by the parent account. This subset includes the user policy that the parent attaches to the user. If the parent also owns the resource in the request (bucket or object), Amazon S3 also evaluates the corresponding resource policies (bucket policy, bucket ACL, and object ACL) at the same time.

      A user must have permission from the parent account to perform the operation.

      This step applies only if the request is made by a user in an AWS account. If the request is made by using the root user credentials of an AWS account, Amazon S3 skips this step.

   b. **Bucket context** – In the bucket context, Amazon S3 evaluates policies owned by the AWS account that owns the bucket.

      If the request is for a bucket operation, the requester must have permission from the bucket owner. If the request is for an object, Amazon S3 evaluates all the policies owned by the

bucket owner to check if the bucket owner has not explicitly denied access to the object. If there is an explicit deny set, Amazon S3 does not authorize the request.

c. **Object context** – If the request is for an object, Amazon S3 evaluates the subset of policies owned by the object owner.

Following are some example scenarios that illustrate how Amazon S3 authorizes a request.

**Example – Requester is an IAM principal**

If the requester is an IAM principal, Amazon S3 must determine if the parent AWS account to which the principal belongs has granted the principal necessary permission to perform the operation. In addition, if the request is for a bucket operation, such as a request to list the bucket content, Amazon S3 must verify that the bucket owner has granted permission for the requester to perform the operation. To perform a specific operation on a resource, an IAM principal needs permission from both the parent AWS account to which it belongs and the AWS account that owns the resource.

**Example – Requester is an IAM principal – If the request is for an operation on an object that the bucket owner doesn't own**

If the request is for an operation on an object that the bucket owner doesn't own, in addition to making sure the requester has permissions from the object owner, Amazon S3 must also check the bucket policy to ensure the bucket owner has not set explicit deny on the object. A bucket owner (who pays the bill) can explicitly deny access to objects in the bucket regardless of who owns it. The bucket owner can also delete any object in the bucket.

By default, when another AWS account uploads an object to your S3 bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through access control lists (ACLs). You can use Object Ownership to change this default behavior so that ACLs are disabled and you, as the bucket owner, automatically own every object in your bucket. As a result, access control for your data is based on policies, such as IAM user policies, S3 bucket policies, virtual private cloud (VPC) endpoint policies, and AWS Organizations service control policies (SCPs). For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

For more information about how Amazon S3 evaluates access policies to authorize or deny requests for bucket operations and object operations, see the following topics:

**Topics**



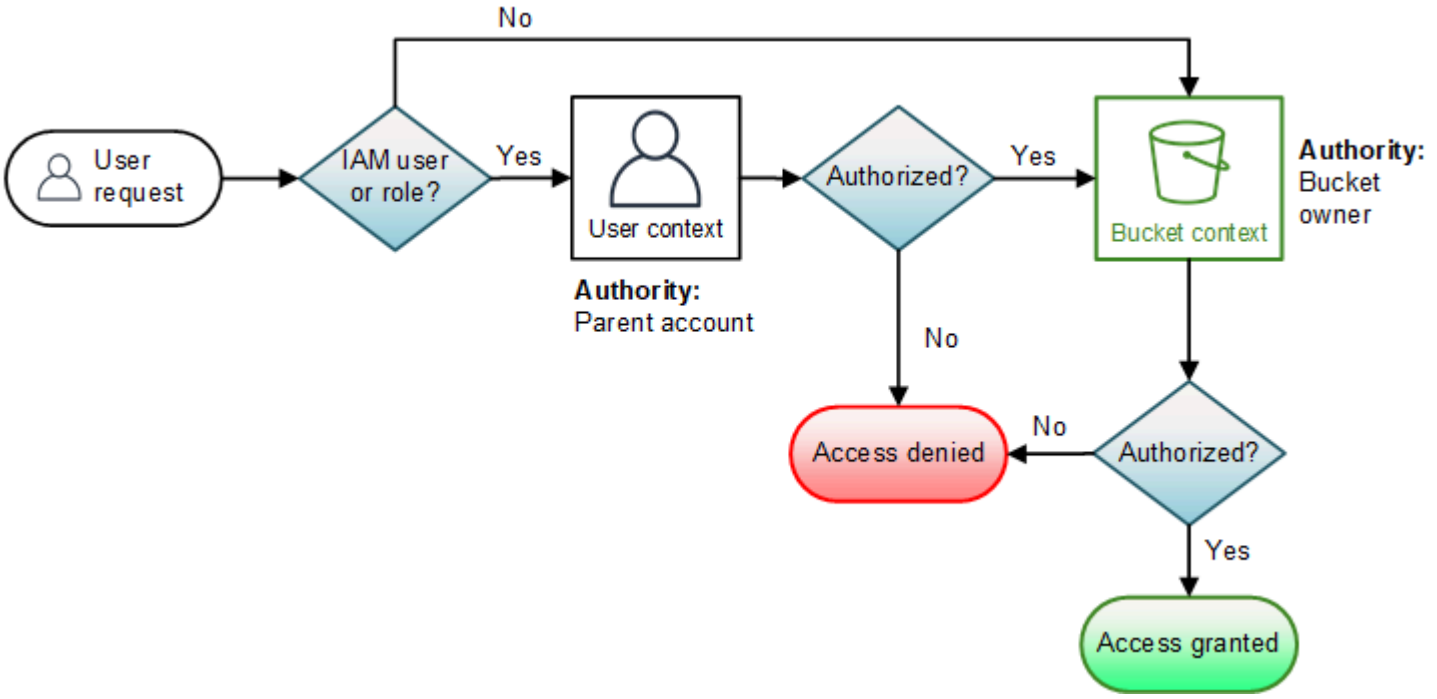
### How Amazon S3 authorizes a request for a bucket operation

When Amazon S3 receives a request for a bucket operation, Amazon S3 converts all the relevant permissions into a set of policies to evaluate at run time. Relevant permissions include resource-based permissions (for example, bucket policies and bucket access control lists) and user policies if the request is from an IAM principal. Amazon S3 then evaluates the resulting set of policies in a series of steps according to a specific context—user context or bucket context:

1. **User context** – If the requester is an IAM principal, the principal must have permission from the parent AWS account to which it belongs. In this step, Amazon S3 evaluates a subset of policies owned by the parent account (also referred to as the context authority). This subset of policies includes the user policy that the parent account attaches to the principal. If the parent also owns the resource in the request (in this case, the bucket), Amazon S3 also evaluates the corresponding resource policies (bucket policy and bucket ACL) at the same time. Whenever a request for a bucket operation is made, the server access logs record the canonical ID of the requester. For more information, see Logging requests with server access logging.
2. **Bucket context** – The requester must have permissions from the bucket owner to perform a specific bucket operation. In this step, Amazon S3 evaluates a subset of policies owned by the AWS account that owns the bucket.

   The bucket owner can grant permission by using a bucket policy or bucket ACL. If the AWS account that owns the bucket is also the parent account of an IAM principal, then it can configure bucket permissions in a user policy.

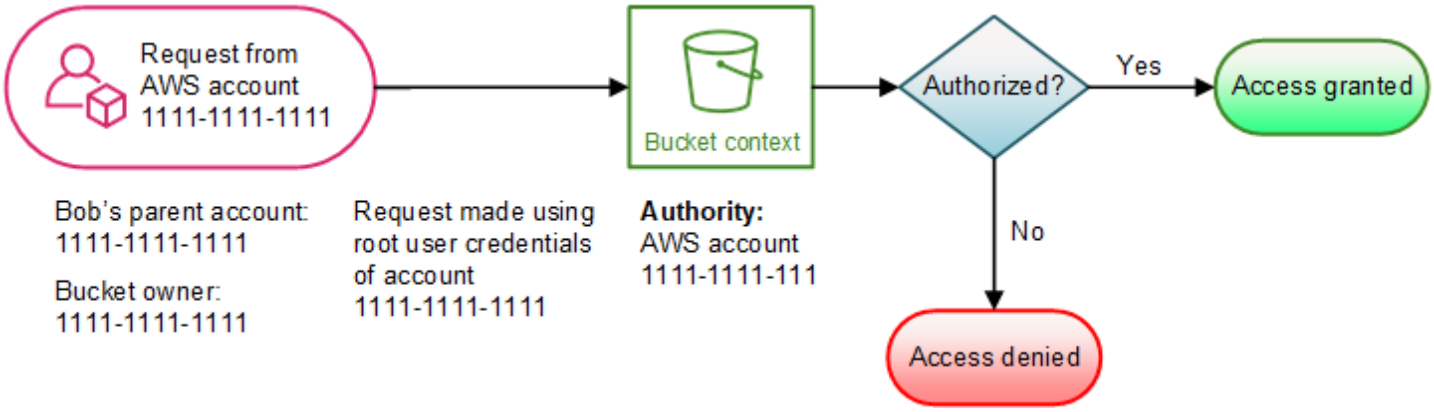
The following is a graphical illustration of the context-based evaluation for bucket operation.




The following examples illustrate the evaluation logic.

**Example 1: Bucket operation requested by bucket owner**

In this example, the bucket owner sends a request for a bucket operation by using the root credentials of the AWS account.

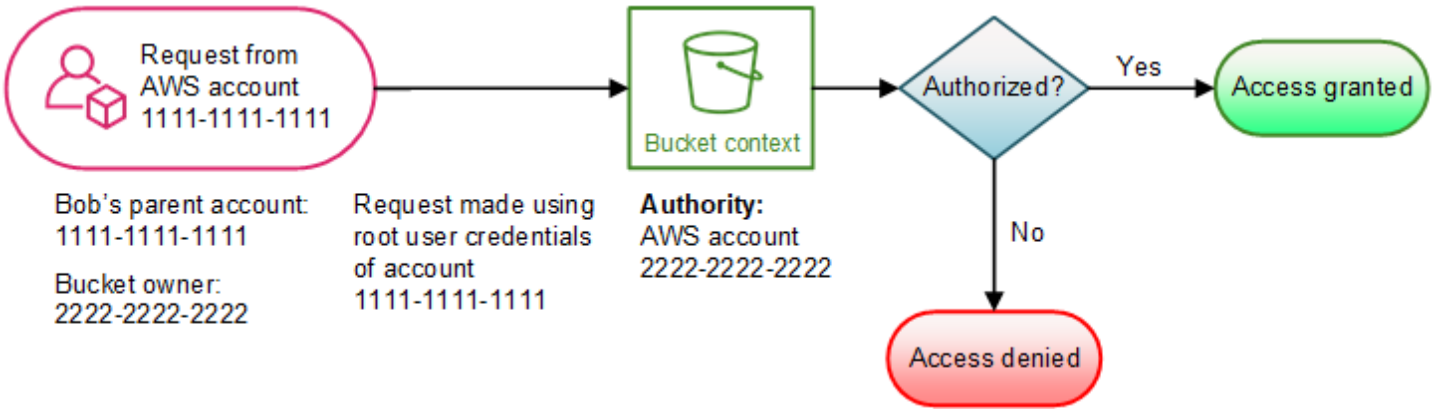



Amazon S3 performs the context evaluation as follows:

1. Because the request is made by using the root user credentials of an AWS account, the user context is not evaluated.
2. In the bucket context, Amazon S3 reviews the bucket policy to determine if the requester has permission to perform the operation. Amazon S3 authorizes the request.

### Example 2: Bucket operation requested by an AWS account that is not the bucket owner

In this example, a request is made by using the root user credentials of AWS account 1111-1111-1111 for a bucket operation owned by AWS account 2222-2222-2222. No IAM users are involved in this request.

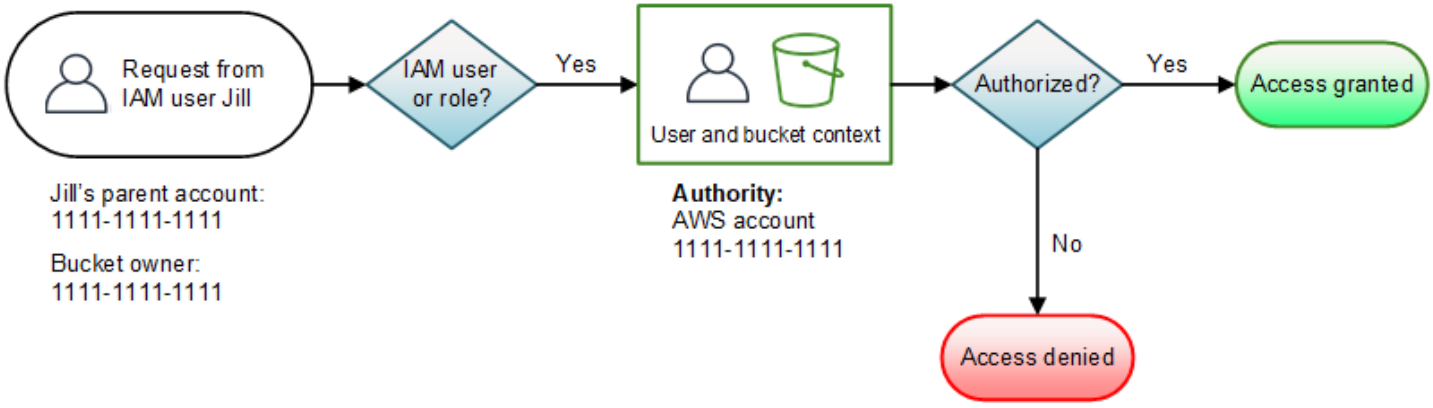



In this example, Amazon S3 evaluates the context as follows:

1. Because the request is made by using the root user credentials of an AWS account, the user context is not evaluated.
2. In the bucket context, Amazon S3 examines the bucket policy. If the bucket owner (AWS account 2222-2222-2222) has not authorized AWS account 1111-1111-1111 to perform the requested operation, Amazon S3 denies the request. Otherwise, Amazon S3 grants the request and performs the operation.

### Example 3: Bucket operation requested by an IAM principal whose parent AWS account is also the bucket owner

In the example, the request is sent by Jill, an IAM user in AWS account 1111-1111-1111, which also owns the bucket.



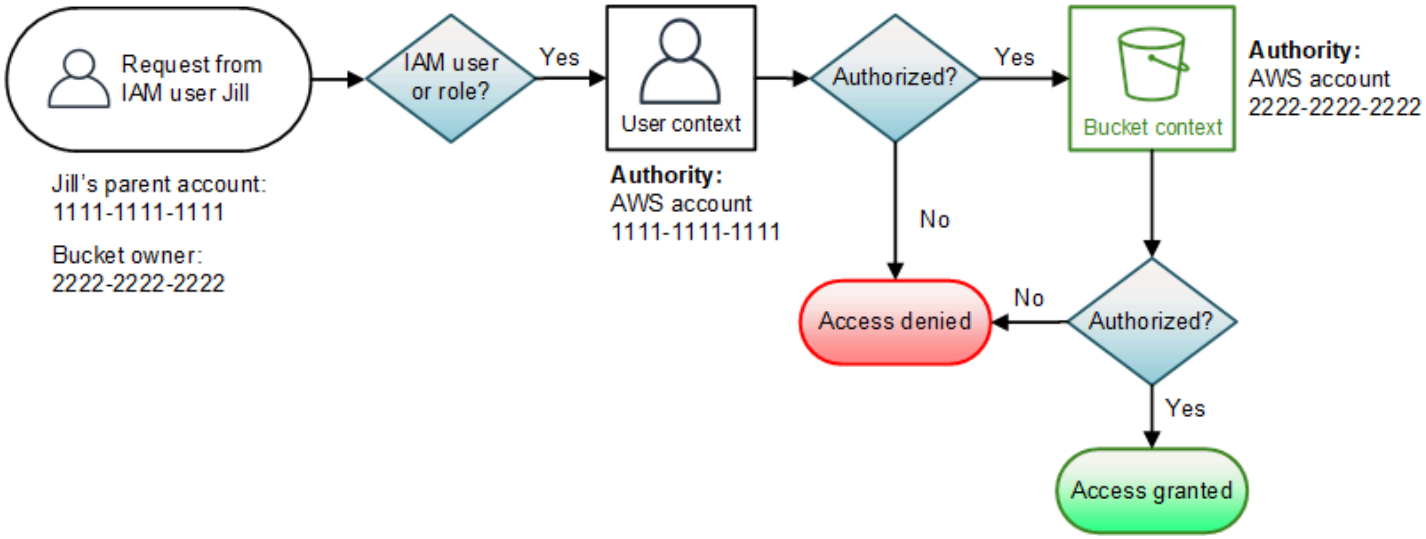
Amazon S3 performs the following context evaluation:

1. Because the request is from an IAM principal, in the user context, Amazon S3 evaluates all policies that belong to the parent AWS account to determine if Jill has permission to perform the operation.

   In this example, parent AWS account 1111-1111-1111, to which the principal belongs, is also the bucket owner. As a result, in addition to the user policy, Amazon S3 also evaluates the bucket policy and bucket ACL in the same context because they belong to the same account.
2. Because Amazon S3 evaluated the bucket policy and bucket ACL as part of the user context, it does not evaluate the bucket context.

**Example 4: Bucket operation requested by an IAM principal whose parent AWS account is not the bucket owner**

In this example, the request is sent by Jill, an IAM user whose parent AWS account is 1111-1111-1111, but the bucket is owned by another AWS account, 2222-2222-2222.




Jill will need permissions from both the parent AWS account and the bucket owner. Amazon S3 evaluates the context as follows:

1. Because the request is from an IAM principal, Amazon S3 evaluates the user context by reviewing the policies authored by the account to verify that Jill has the necessary permissions. If Jill has permission, then Amazon S3 moves on to evaluate the bucket context. If Jill doesn't have permission, it denies the request.

2. In the bucket context, Amazon S3 verifies that bucket owner 2222-2222-2222 has granted Jill (or her parent AWS account) permission to perform the requested operation. If she has that permission, Amazon S3 grants the request and performs the operation. Otherwise, Amazon S3 denies the request.

### How Amazon S3 authorizes a request for an object operation

When Amazon S3 receives a request for an object operation, it converts all the relevant permissions — resource-based permissions (object access control list (ACL), bucket policy, bucket ACL) and IAM user policies—into a set of policies to be evaluated at run time. It then evaluates the resulting set of policies in a series of steps. In each step, it evaluates a subset of policies in three specific contexts—user context, bucket context, and object context:

1. **User context** – If the requester is an IAM principal, the principal must have permission from the parent AWS account to which it belongs. In this step, Amazon S3 evaluates a subset of policies owned by the parent account (also referred as the context authority). This subset of policies includes the user policy that the parent attaches to the principal. If the parent also owns the resource in the request (bucket or object), Amazon S3 evaluates the corresponding resource policies (bucket policy, bucket ACL, and object ACL) at the same time.

   > **Note**
   >
   > If the parent AWS account owns the resource (bucket or object), it can grant resource permissions to its IAM principal by using either the user policy or the resource policy.

2. **Bucket context** – In this context, Amazon S3 evaluates policies owned by the AWS account that owns the bucket.

   If the AWS account that owns the object in the request is not same as the bucket owner, Amazon S3 checks the policies if the bucket owner has explicitly denied access to the object. If there is an explicit deny set on the object, Amazon S3 does not authorize the request.

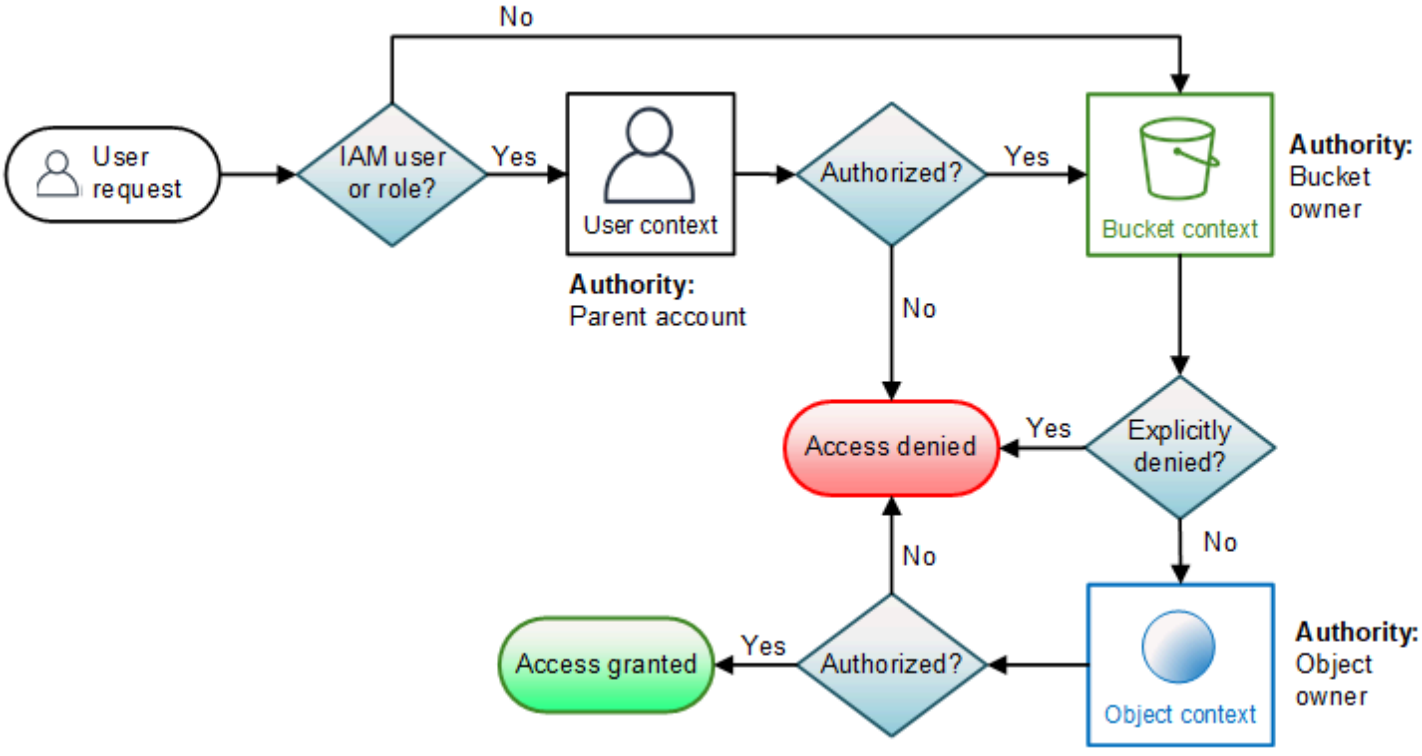
3. **Object context** – The requester must have permissions from the object owner to perform a specific object operation. In this step, Amazon S3 evaluates the object ACL.

   > **Note**
   >
   > If bucket and object owners are the same, access to the object can be granted in the bucket policy, which is evaluated at the bucket context. If the owners are different, the

object owners must use an object ACL to grant permissions. If the AWS account that owns the object is also the parent account to which the IAM principal belongs, it can configure object permissions in a user policy, which is evaluated at the user context. For more information about using these access policy alternatives, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

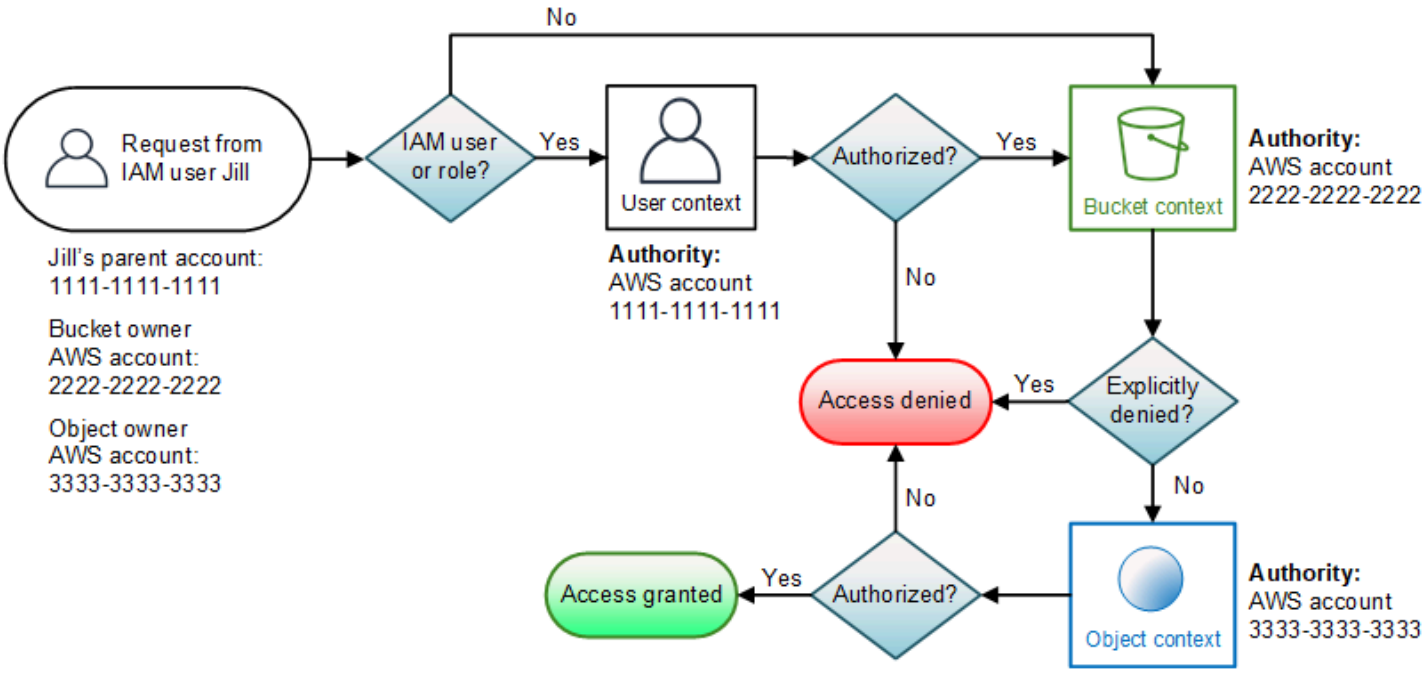
If you as the bucket owner want to own all the objects in your bucket and use bucket policies or policies based on IAMto manage access to these objects, you can apply the bucket owner enforced setting for Object Ownership. With this setting, you as the bucket owner automatically own and have full control over every object in your bucket. Bucket and object ACLs can't be edited and are no longer considered for access. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

The following is an illustration of the context-based evaluation for an object operation.




**Example of an object operation request**

In this example, IAM user Jill, whose parent AWS account is 1111-1111-1111, sends an object operation request (for example, `GetObject`) for an object owned by AWS account 3333-3333-3333 in a bucket owned by AWS account 2222-2222-2222.




Jill will need permission from the parent AWS account, the bucket owner, and the object owner. Amazon S3 evaluates the context as follows:

1. Because the request is from an IAM principal, Amazon S3 evaluates the user context to verify that the parent AWS account 1111-1111-1111 has given Jill permission to perform the requested operation. If she has that permission, Amazon S3 evaluates the bucket context. Otherwise, Amazon S3 denies the request.
2. In the bucket context, the bucket owner, AWS account 2222-2222-2222, is the context authority. Amazon S3 evaluates the bucket policy to determine if the bucket owner has explicitly denied Jill access to the object.
3. In the object context, the context authority is AWS account 3333-3333-3333, the object owner. Amazon S3 evaluates the object ACL to determine if Jill has permission to access the object. If she does, Amazon S3 authorizes the request.

## AWS managed policies for Amazon S3

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you

reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

**AWS managed policy: AmazonS3FullAccess**

You can attach the `AmazonS3FullAccess` policy to your IAM identities. This policy grants permissions that allow full access to Amazon S3.

To view the permissions for this policy, see AmazonS3FullAccess in the AWS Management Console.

**AWS managed policy: AmazonS3ReadOnlyAccess**

You can attach the `AmazonS3ReadOnlyAccess` policy to your IAM identities. This policy grants permissions that allow read-only access to Amazon S3.

To view the permissions for this policy, see AmazonS3ReadOnlyAccess in the AWS Management Console.

**AWS managed policy: AmazonS3ObjectLambdaExecutionRolePolicy**

Provides AWS Lambda functions the required permissions to send data to S3 Object Lambda when requests are made to an S3 Object Lambda access point. Also grants Lambda permissions to write to Amazon CloudWatch logs.

To view the permissions for this policy, see AmazonS3ObjectLambdaExecutionRolePolicy in the AWS Management Console.

**Amazon S3 updates to AWS managed policies**

View details about updates to AWS managed policies for Amazon S3 since this service began tracking these changes.

| Change | Description | Date |
| --- | --- | --- |
| Amazon S3 added Describe permissions to AmazonS3ReadOnlyAccess | Amazon S3 added s3:Describe* permissions to AmazonS3ReadOnlyAccess . | August 11, 2023 |
| Amazon S3 added S3 Object Lambda permissions to AmazonS3FullAccess and AmazonS3ReadOnlyAccess | Amazon S3 updated the AmazonS3FullAccess and AmazonS3ReadOnlyAccess policies to include permissions for S3 Object Lambda. | September 27, 2021 |
| Amazon S3 added AmazonS3ObjectLambdaExecutionRolePolicy | Amazon S3 added a new AWS-managed policy called AmazonS3ObjectLambdaExecutionRolePolicy that provides Lambda functions permissions to interact with S3 Object Lambda and write to CloudWatch logs. | August 18, 2021 |
| Amazon S3 started tracking changes | Amazon S3 started tracking changes for its AWS managed policies. | August 18, 2021 |

## Using service-linked roles for Amazon S3 Storage Lens

To use Amazon S3 Storage Lens to collect and aggregate metrics across all your accounts in AWS Organizations, you must first ensure that S3 Storage Lens has trusted access enabled by the management account in your organization. S3 Storage Lens creates a service-linked role (SLR) to allow it to get the list of AWS accounts belonging to your organization. This list of accounts is used by S3 Storage Lens to collect metrics for S3 resources in all the member accounts when the S3 Storage Lens dashboard or configurations are created or updated.

Amazon S3 Storage Lens uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to S3 Storage Lens. Service-linked roles are predefined by S3 Storage Lens and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up S3 Storage Lens easier because you don't have to add the necessary permissions manually. S3 Storage Lens defines the permissions of its service-linked roles, and unless defined otherwise, only S3 Storage Lens can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

You can delete this service-linked role only after first deleting the related resources. This protects your S3 Storage Lens resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

**Service-linked role permissions for Amazon S3 Storage Lens**

S3 Storage Lens uses the service-linked role named **AWSServiceRoleForS3StorageLens** – This enables access to AWS services and resources used or managed by S3 Storage Lens. This allows S3 Storage Lens to access AWS Organizations resources on your behalf.

The S3 Storage Lens service-linked role trusts the following service on your organization's storage:

- `storage-lens.s3.amazonaws.com`

The role permissions policy allows S3 Storage Lens to complete the following actions:

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

  `organizations:ListAWSServiceAccessForOrganization`

  `organizations:ListDelegatedAdministrators`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

### Creating a service-linked role for S3 Storage Lens

You don't need to manually create a service-linked role. When you complete one of the following tasks while signed into the AWS Organizations management or the delegate administrator accounts, S3 Storage Lens creates the service-linked role for you:

- Create an S3 Storage Lens dashboard configuration for your organization in the Amazon S3 console.
- PUT an S3 Storage Lens configuration for your organization using the REST API, AWS CLI and SDKs.

> **Note**
>
> S3 Storage Lens will support a maximum of five delegated administrators per organization.

If you delete this service-linked role, the preceding actions will re-create it as needed.

### Example policy for S3 Storage Lens service-linked role

### Example Permissions policy for the S3 Storage Lens service-linked role

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

### Editing a service-linked role for Amazon S3 Storage Lens

S3 Storage Lens doesn't allow you to edit the AWSServiceRoleForS3StorageLens service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

### Deleting a service-linked role for Amazon S3 Storage Lens

If you no longer need to use the service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the Amazon S3 Storage Lens service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete the AWSServiceRoleForS3StorageLens you must delete all the organization level S3 Storage Lens configurations present in all AWS Regions using the AWS Organizations management or the delegate administrator accounts.

The resources are organization-level S3 Storage Lens configurations. Use S3 Storage Lens to clean up the resources and then use the IAM Console, CLI, REST API, or AWS SDK to delete the role.

In the REST API, AWS CLI, and SDKs, S3 Storage Lens configurations can be discovered using `ListStorageLensConfigurations` in all the Regions where your organization has created S3 Storage Lens configurations. Use the action `DeleteStorageLensConfiguration` to delete these configurations so that you can then delete the role.

> **Note**
>
> To delete the service-linked role, you must delete all the organization-level S3 Storage Lens configurations in all the Regions where they exist.

**To delete Amazon S3 Storage Lens resources used by the AWSServiceRoleForS3StorageLens SLR**

1. To get a list of your organization level configurations, you must use the `ListStorageLensConfigurations` in every Region that you have S3 Storage Lens configurations. This list can also be obtained from the Amazon S3 console.
2. Delete these configurations from the appropriate Regional endpoints by invoking the `DeleteStorageLensConfiguration` API call or by using the Amazon S3 console.

**To manually delete the service-linked role using IAM**

After you have deleted the configurations, delete the AWSServiceRoleForS3StorageLens SLR from the IAM Console or by invoking the IAM API `DeleteServiceLinkedRole`, or using the AWS CLI or AWS SDK. For more information, see Deleting a service-linked role in the *IAM User Guide*.

**Supported Regions for S3 Storage Lens service-linked roles**

S3 Storage Lens supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see Amazon S3 Regions and Endpoints.

## Troubleshooting Amazon S3 identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon S3 and IAM.

**Topics**

- I received an access denied error
- I am not authorized to perform an action in Amazon S3
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon S3 resources

**I received an access denied error**

Verify that there is not an explicit Deny statement against the requester you are trying to grant permissions to in either the bucket policy or the identity-based policy.

For detailed information about troubleshooting access denied errors, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

### I am not authorized to perform an action in Amazon S3

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `s3:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `s3:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

### I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon S3.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon S3. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

**I want to allow people outside of my AWS account to access my Amazon S3 resources**

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon S3 supports these features, see How Amazon S3 works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

## Managing access with S3 Access Grants

To adhere to the principle of least privilege, you define granular access to your Amazon S3 data based on applications, personas, groups, or organizational units. You can use various approaches to achieve granular access to your data in Amazon S3, depending on the scale and complexity of the access patterns.

The simplest approach for managing access to small-to-medium numbers of datasets in Amazon S3 by AWS Identity and Access Management (IAM) principals is to define IAM permission policies and S3 bucket policies. This strategy works, so long as the necessary policies fit within the policy size limits of S3 bucket policies (20 KB) and IAM policies (5 KB), and within the number of IAM principals allowed per account.

As your number of datasets and use cases scales, you might require more policy space. An approach that offers significantly more space for policy statements is to use S3 Access Points as additional endpoints for S3 buckets, because each access point can have its own policy. You can define quite granular access control patterns, because you can have thousands of access points per AWS Region per account, with a policy up to 20 KB in size for each access point. Although S3 Access Points

increases the amount of policy space available, it requires a mechanism for clients to discover the right access point for the right dataset.

A third approach is to implement an IAM session broker pattern, in which you implement access-decision logic and dynamically generate short-term IAM session credentials for each access session. While the IAM session broker approach supports arbitrarily dynamic permissions patterns and scales effectively, you must build the access-pattern logic.

Instead of using these approaches, you can use S3 Access Grants to manage access to your Amazon S3 data. S3 Access Grants provides a simplified model for defining access permissions to data in Amazon S3 by prefix, bucket, or object. In addition, you can use S3 Access Grants to grant access to both IAM principals and directly to users or groups from your corporate directory.

You commonly define permissions to data in Amazon S3 by mapping users and groups to datasets. You can use S3 Access Grants to define direct access mappings of S3 prefixes to users and roles within Amazon S3 buckets and objects. With the simplified access scheme in S3 Access Grants, you can grant read-only, write-only, or read-write access on a per-S3-prefix basis to both IAM principals and directly to users or groups from a corporate directory. With these S3 Access Grants capabilities, applications can request data from Amazon S3 on behalf of the application's current authenticated user.

When you integrate S3 Access Grants with the trusted identity propagation feature of AWS IAM Identity Center, your applications can make requests to AWS services (including S3 Access Grants) directly on behalf of an authenticated corporate directory user. Your applications no longer need to first map the user to an IAM principal. Furthermore, because end-user identities are propagated all the way to Amazon S3, auditing which user accessed which S3 object is simplified. You no longer need to reconstruct the relationship between different users and IAM sessions. When you're using S3 Access Grants with IAM Identity Center trusted identity propagation, each AWS CloudTrail data event for Amazon S3 contains a direct reference to the end user on whose behalf the data was accessed.

For more information about S3 Access Grants, see the following topics.

**Topics**

- S3 Access Grants concepts
- S3 Access Grants and corporate directory identities
- Getting started with S3 Access Grants
- Create an S3 Access Grants instance

- Register a location
- Create grants
- Request access to Amazon S3 data through S3 Access Grants
- Access S3 data through an access grant
- S3 Access Grants cross-account access
- Using AWS tags with S3 Access Grants
- S3 Access Grants limitations
- S3 Access Grants integrations

## S3 Access Grants concepts

### S3 Access Grants workflow

The S3 Access Grants workflow is:

1. Create an S3 Access Grants instance. See Create an S3 Access Grants instance.
2. Within your S3 Access Grants instance, register locations in your Amazon S3 data, and map these locations to AWS Identity and Access Management (IAM) roles. See Register a location.
3. Create grants for grantees, which give grantees access to your S3 resources. See Create grants.
4. The grantee requests temporary credentials from S3 Access Grants. See Request access to Amazon S3 data through S3 Access Grants.
5. The grantee accesses the S3 data using those temporary credentials. See Access S3 data through an access grant.

For more information, see Getting started with S3 Access Grants.

### S3 Access Grants instances

An *S3 Access Grants instance* is a logical container for individual *grants*. When you create an S3 Access Grants instance, you must specify an AWS Region. Each AWS Region in your AWS account can have one S3 Access Grants instance. For more information, see Create an S3 Access Grants instance.

If you want to use S3 Access Grants to grant access to user and group identities from your corporate directory, you must also associate your S3 Access Grants instance with an AWS IAM

Identity Center instance. For more information, see S3 Access Grants and corporate directory identities.

A newly created S3 Access Grants instance is empty. You must register a location in the instance, which can be the S3 default path (`s3://`), a bucket, or a prefix within a bucket. After you register at least one location, you can create access grants that give access to data in this registered location.

**Locations**

An S3 Access Grants *location* maps buckets or prefixes to an AWS Identity and Access Management (IAM) role. S3 Access Grants assumes this IAM role to vend temporary credentials to the grantee that's accessing that particular location. You must first register at least one location in your S3 Access Grants instance before you can create an access grant.

We recommend that you register the default location (`s3://`) and map it to an IAM role. The location at the default S3 path (`s3://`) covers access to all of your S3 buckets in the AWS Region of your account. When you create an access grant, you can narrow the grant scope to a bucket, a prefix, or an object within the default location.

More complex access-management use cases might require you to register more than the default location. Some examples of such use cases are:

- Suppose that the *`amzn-s3-demo-bucket`* is a registered location in your S3 Access Grants instance with an IAM role mapped to it, but this IAM role is denied access to a particular prefix within the bucket. In this case, you can register the prefix that the IAM role does not have access to as a separate location and map that location to a different IAM role with the necessary access.
- Suppose that you want to create grants that restrict access to only the users within a virtual private cloud (VPC) endpoint. In this case, you can register a location for a bucket in which the IAM role restricts access to the VPC endpoint. Later, when a grantee asks S3 Access Grants for credentials, S3 Access Grants assumes the location's IAM role to vend the temporary credentials. This credential will deny access to the specific bucket unless the caller is within the VPC endpoint. This deny permission is applied in addition to the regular READ, WRITE, or READWRITE permission specified in the grant.

If your use case requires you to register multiple locations in your S3 Access Grants instance, you can register any of the following:

- The default S3 location (`s3://`)

- A bucket (for example, *amzn-s3-demo-bucket*) or multiple buckets
- A bucket and a prefix (for example, *amzn-s3-demo-bucket*/prefix*) or multiple prefixes

For the maximum number of locations that you can register in your S3 Access Grants instance, see S3 Access Grants limitations. For more information about registering an S3 Access Grants location, see Register a location.

After you register the first location in your S3 Access Grants instance, your instance still does not have any individual access grants in it. So, no access has been granted yet to any of your S3 data. You can now create access grants to give access. For more information about creating grants, see Create grants.

**Grants**

An individual *grant* in an S3 Access Grants instance allows a specific identity—an IAM principal, or a user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance.

When you create a grant, you don't have to grant access to the entire registered location. You can narrow the grant's scope of access within a location. If the registered location is the default S3 path (s3://), you are required to narrow the scope of the grant to a bucket, a prefix within a bucket, or a specific object. If the registered location of the grant is a bucket or a prefix, then you can give access to the entire bucket or prefix, or you can optionally narrow the scope of the grant to a prefix, subprefix, or an object.

In the grant, you also set the access level of the grant to READ, WRITE, or READWRITE. Suppose you have a grant that gives the corporate directory group 01234567-89ab-cdef-0123-456789abcdef READ access to the bucket s3://*amzn-s3-demo-bucket*/projects/items/*. Users in this group can have READ access to every object that has an object key name which starts with the prefix projects/items/ in the bucket named *amzn-s3-demo-bucket*.

For the maximum number of grants that you can create in your S3 Access Grants instance, see S3 Access Grants limitations. For more information about creating grants, see Create grants.

**S3 Access Grants temporary credentials**

After you create a grant, an authorized application that utilizes the identity specified in the grant can request *just-in-time access credentials*. To do this, the application calls the GetDataAccess S3 API operation. Grantees can use this API operation to request access to the S3 data you have shared with them.

The S3 Access Grants instance evaluates the `GetDataAccess` request against the grants that it has. If there is a matching grant for the requestor, S3 Access Grants assumes the IAM role that's associated with the registered location of the matching grant. S3 Access Grants scopes the permissions of the temporary credentials to access only the S3 bucket, prefix, or object that's specified by the grant's scope.

The expiration time of the temporary access credentials defaults to 1 hour, but you can set it to any value from 15 minutes to 12 hours. See the maximum duration session in the AssumeRole API reference.

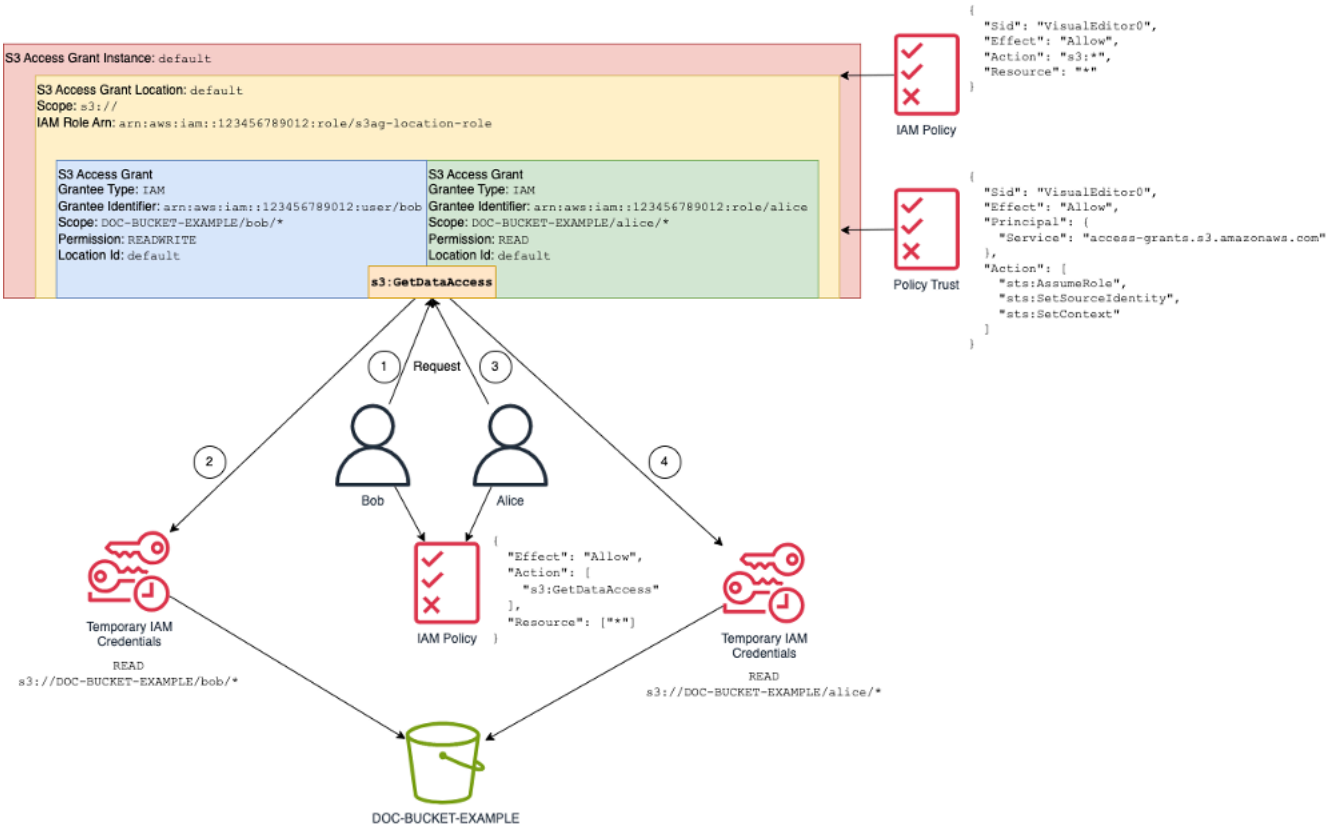
**How it works**

In the following diagram, a default Amazon S3 location with the scope `s3://` is registered with the IAM role `s3ag-location-role`. This IAM role has permissions to perform Amazon S3 actions within the account when its credentials are obtained through S3 Access Grants.

Within this location, two individual access grants are created for two IAM users. The IAM user Bob is granted both `READ` and `WRITE` access on the `bob/` prefix in the `DOC-BUCKET-EXAMPLE` bucket. Another IAM role, Alice, is granted only `READ` access on the `alice/` prefix in the `DOC-BUCKET-EXAMPLE` bucket. A grant, colored in blue, is defined for Bob to access the prefix `bob/` in the `DOC-BUCKET-EXAMPLE` bucket. A grant, colored in green, is defined for Alice to access the prefix `alice/` in the `DOC-BUCKET-EXAMPLE` bucket.

When it's time for Bob to `READ` data, the IAM role that's associated with the location that his grant is in calls the S3 Access Grants GetDataAccess API operation. If Bob tries to `READ` any S3 prefix or object that starts with `s3://DOC-BUCKET-EXAMPLE/bob/*`, the `GetDataAccess` request returns a set of temporary IAM session credentials with permission to `s3://DOC-BUCKET-EXAMPLE/bob/*`. Similarly, Bob can `WRITE` to any S3 prefix or object that starts with `s3://DOC-BUCKET-EXAMPLE/bob/*`, because the grant also allows that.

Similarly, Alice can `READ` anything that starts with `s3://DOC-BUCKET-EXAMPLE/alice/`. However, if she tries to `WRITE` anything to any bucket, prefix, or object in `s3://`, she will get an Access Denied (403 Forbidden) error, because there is no grant that gives her `WRITE` access to any data. In addition, if Alice requests any level of access (`READ` or `WRITE`) to data outside of `s3://DOC-BUCKET-EXAMPLE/alice/`, she will again receive an Access Denied error.




This pattern scales to a high number of users and buckets and simplifies management of those permissions. Rather than editing potentially large S3 bucket policies every time you want to add or remove an individual user-prefix access relationship, you can add and remove individual, discrete grants.

## S3 Access Grants and corporate directory identities

You can use Amazon S3 Access Grants to grant access to AWS Identity and Access Management (IAM) principals (users or roles), both in the same AWS account and in others. However, in many cases, the entity accessing the data is an end user from your corporate directory. Instead of granting access to IAM principals, you can use S3 Access Grants to grant access directly to your corporate users and groups. With S3 Access Grants, you no longer need to map your corporate identities to intermediate IAM principals in order to access your S3 data through your corporate applications.

This new functionality—support for using end-user identities access to data—is provided by associating your S3 Access Grants instance with an AWS IAM Identity Center instance. IAM Identity Center supports standards-based identity providers and is the hub in AWS for any services or features, including S3 Access Grants, that support end-user identities. IAM Identity Center provides

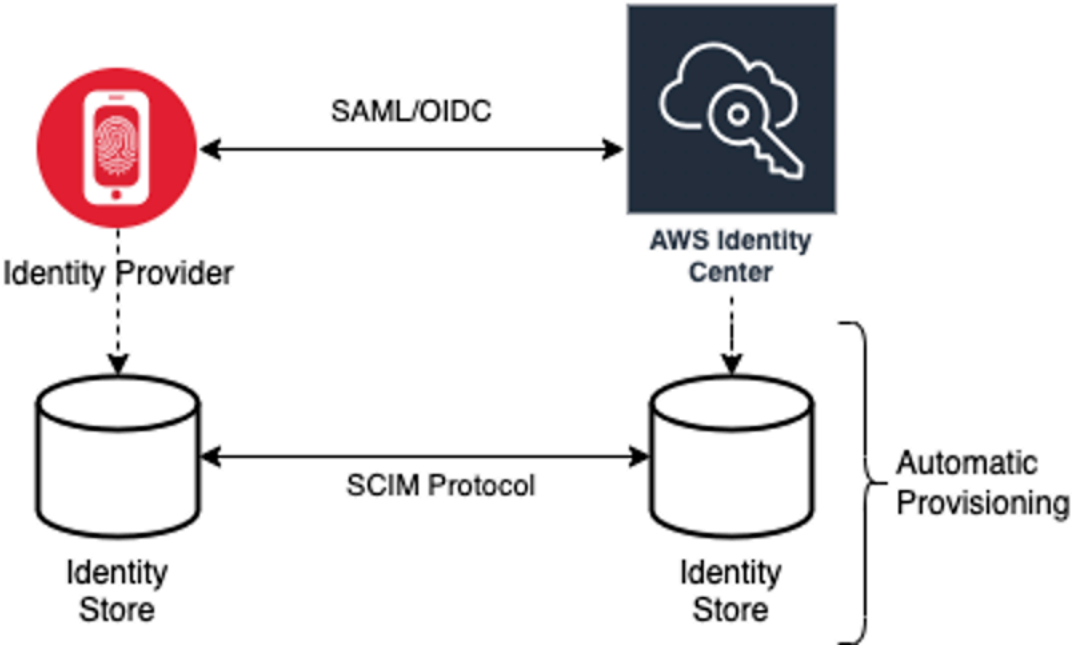
authentication support for corporate identities through its trusted identity propagation feature. For more information, see Trusted identity propagation across applications.

To get started with workforce identity support in S3 Access Grants, as a prerequisite, you start in IAM Identity Center by configuring identity provisioning between your corporate identity provider and IAM Identity Center. IAM Identity Center supports corporate identity providers such as Okta, Microsoft Entra ID (formerly Azure Active Directory), or any other external identity provider (IdP) that supports the System for Cross-domain Identity Management (SCIM) protocol. When you connect IAM Identity Center to your IdP and enable automatic provisioning, the users and groups from your IdP are synchronized into the identity store in IAM Identity Center. After this step, IAM Identity Center has its own view of your users and groups, so that you can refer to them by using other AWS services and features, such as S3 Access Grants. For more information about configuring IAM Identity Center automatic provisioning, see Automatic provisioning in the *AWS IAM Identity Center User Guide*.

IAM Identity Center is integrated with AWS Organizations so that you can centrally manage permissions across multiple AWS accounts without configuring each of your accounts manually. In a typical organization, your identity administrator configures one IAM Identity Center instance for the entire organization, as a single point of identity synchronization. This IAM Identity Center instance typically runs in a dedicated AWS account in your organization. In this common configuration, you can refer to user and group identities in S3 Access Grants from any AWS account in the organization.

However, if your AWS Organizations administrator hasn't yet configured a central IAM Identity Center instance, you can create a local one in the same account as your S3 Access Grants instance. Such a configuration is more common for proof-of-concept or local development use cases. In all cases, the IAM Identity Center instance must be in the same AWS Region as the S3 Access Grants instance to which it will be associated.

In the following diagram of an IAM Identity Center configuration with an external IdP, the IdP is configured with SCIM to synchronize the identity store from the IdP to the identity store in IAM Identity Center.




To use your corporate directory identities with S3 Access Grants, do the following:

- Set up Automatic provisioning in IAM Identity Center to synchronize user and group information from your IdP into IAM Identity Center.
- Configure your external identity source within IAM Identity Center as a trusted token issuer. For more information, see Trusted identity propagation across applications in the *AWS IAM Identity Center User Guide*.
- Associate your S3 Access Grants instance with your IAM Identity Center instance. You can do this when you create your S3 Access Grants instance. If you've already created your S3 Access Grants instance, see Associate or disassociate your IAM Identity Center instance.

**How directory identities can access S3 data**

Suppose that you have corporate directory users who need to access your S3 data through a corporate application, for example, a document-viewer application, that is integrated with your external IdP (for example, Okta) to authenticate users. Authentication of the user in these applications is typically done through redirects in the user's web browser. Because users in the directory are not IAM principals, your application needs IAM credentials with which it can call the S3 Access Grants `GetDataAccess` API operation to get access credentials to S3 data on the users' behalf. Unlike IAM users and roles who get credentials themselves, your application needs a way to represent a directory user, who isn't mapped to an IAM role, so that the user can get data access through S3 Access Grants.

This transition, from authenticated directory user to an IAM caller that can make requests to S3 Access Grants on behalf of the directory user, is done by the application through the trusted token issuer feature of IAM Identity Center. The application, after authenticating the directory user, has an identity token from the IdP (for example, Okta) that represents the directory user according to Okta. The trusted token issuer configuration in IAM Identity Center enables the application to exchange this Okta token (the Okta tenant is configured as the "trusted issuer") for a different identity token from IAM Identity Center that will securely represent the directory user within AWS services. The data application will then assume an IAM role, providing the directory user's token from IAM Identity Center as additional context. The application can use the resulting IAM session to call S3 Access Grants. The token represents both the identity of the application (the IAM principal itself) as well as the directory user's identity.

The main step of this transition is the token exchange. The application performs this token exchange by calling the `CreateTokenWithIAM` API operation in IAM Identity Center. Of course, that too is an AWS API call and requires an IAM principal to sign it. The IAM principal that makes this request is typically an IAM role that's associated with the application. For example, if the application runs on Amazon EC2, the `CreateTokenWithIAM` request is typically performed by the IAM role that's associated with the EC2 instance on which the application runs. The result of a successful `CreateTokenWithIAM` call is a new identity token, which will be recognized within AWS services.

The next step, before the application can call `GetDataAccess` on the directory user's behalf, is for the application to obtain an IAM session that includes the directory user's identity. The application does this with an AWS Security Token Service (AWS STS) `AssumeRole` request that also includes the IAM Identity Center token for the directory user as additional identity context. This additional context is what enables IAM Identity Center to propagate the directory user's identity to the next step. The IAM role that the application assumes is the role that will need IAM permissions to call the `GetDataAccess` operation.

Having assumed the identity bearer IAM role with the IAM Identity Center token for the directory user as additional context, the application now has everything it needs to make a signed request to `GetDataAccess` on behalf of the authenticated directory user.

Token propagation is based on the following steps:

**Create an IAM Identity Center application**

First, create a new application in IAM Identity Center. This application will use a template that allows IAM Identity Center to identify which type of application settings that you can use. The

command to create the application requires you to provide the IAM Identity Center instance Amazon Resource Name (ARN), an application name, and the application provider ARN. The application provider is the SAML or OAuth application provider that the application will use to make calls to IAM Identity Center.

To use the following example command, replace the *user input placeholders* with your own information:

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

Response:

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

**Create a trusted token issuer**

Now that you have your IAM Identity Center application, the next step is to configure a trusted token issuer that will be used to exchange your `IdToken` values from your IdP with IAM Identity Center tokens. In this step you need to provide the following items:

- The identity provider issuer URL
- The trusted token issuer name
- The claim attribute path
- The identity store attribute path
- The JSON Web Key Set (JWKS) retrieval option

The claim attribute path is the identity provider attribute that will be used to map to the identity store attribute. Normally, the claim attribute path is the email address of the user, but you can use other attributes to perform the mapping.

Create a file called `oidc-configuration.json` with the following information. To use this file, replace the *user input placeholders* with your own information.

```
{
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

To create the trusted token issuer, run the following command. To use this example command, replace the *user input placeholders* with your own information.

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

Response

```
{
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

**Connect the IAM Identity Center application with the trusted token issuer**

The trusted token issuer requires a few more configuration settings to work. Set the audience that the trusted token issuer will trust. The audience is the value inside the `IdToken` that's identified by the key and can be found in the identity provider settings. For example:

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

Create a file named `grant.json` that contains the following content. To use this file, change the audience to match your identity provider settings and provide the trusted token issuer ARN that was returned by the previous command.

```
{
```

```
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
         [
           {
             "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
               "AuthorizedAudiences":
                 [
                   "1234973b-abcd-1234-abcd-345c5a9c1234"
                 ]
            }
         ]
     }
 }
```

Run the following example command. To use this command, replace the *`user input placeholders`* with your own information.

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

This command sets the trusted token issuer with configuration settings to trust the audience in the `grant.json` file and link this audience with the application created in the first step for exchanging tokens of the type `jwt-bearer`. The string `urn:ietf:params:oauth:grant-type:jwt-bearer` is not an arbitrary string. It is a registered namespace in OAuth JSON Web Token (JWT) assertion profiles. You can find more information about this namespace in RFC 7523.

Next, use the following command to set up which scopes the trusted token issuer will include when exchanging `IdToken` values from your identity provider. For S3 Access Grants, the value for the `--scope` parameter is `s3:access_grants:read_write`.

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

The last step is to attach a resource policy to the IAM Identity Center application. This policy will allow your application IAM role to make requests to the API operation `sso-oauth:CreateTokenWithIAM` and receive the `IdToken` values from IAM Identity Center.

Create a file named `authentication-method.json` that contains the following content. Replace *123456789012* with your account ID.

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
        }
```

To attach the policy to the IAM Identity Center application, run the following command:

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

This completes the configuration settings for using S3 Access Grants with directory users through a web application. You can test this setup directly in the application or you can call the `CreateTokenWithIAM` API operation by using the following command from an allowed IAM role in the IAM Identity Center application policy:

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
   --assertion IdToken
```

The response will be similar to this:

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

If you decode the `IdToken` value that is encoded with base64, you can see the key-value pairs in JSON format. The key `sts:identity_context` contains the value that your application needs to send in the `sts:AssumeRole` request to include the identity information of the directory user. Here is an example of the `IdToken` decoded:

```
{
    "aws:identity_store_id": "d-996773e796",
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
```

```
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
}
```

You can get the value from `sts:identity_context` and pass this information in an `sts:AssumeRole` call. The following is a CLI example of the syntax. The role to be assumed is a temporary role with permissions to invoke `s3:GetDataAccess`.

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

You can now use the credentials received from this call to invoke the `s3:GetDataAccess` API operation and receive the final credentials with access to your S3 resources.

## Getting started with S3 Access Grants

Amazon S3 Access Grants is an Amazon S3 feature that provides a scalable access control solution for your S3 data. S3 Access Grants is an S3 credential vendor, meaning that you register with S3 Access Grants your list of grants and at what level. Thereafter, when users or clients need access to your S3 data, they first ask S3 Access Grants for credentials. If there is a corresponding grant that authorizes access, S3 Access Grants vends temporary, least-privilege access credentials. The users or clients can then use S3 Access Grants vended credentials to access your S3 data. With that in mind, if your S3 data requirements mandate a complex or large permission configuration, you can use S3 Access Grants to scale S3 data permissions for users, groups, roles, and applications.

For most use cases, you can manage access control for your S3 data by using AWS Identity and Access Management (IAM) with bucket policies or IAM identity-based policies.

However, if you have complex S3 access control requirements, such as the following, you could benefit greatly from using S3 Access Grants:

- You are running into the bucket policy size limit of 20 KB.

- You grant human identities, for example, Microsoft Entra ID (formerly Azure Active Directory), Okta, or Ping users and groups, access to S3 data for analytics and big data.
- You must provide cross-account access without making frequent updates to IAM policies.
- Your data is unstructured and object-level rather than structured, in row and column format.

The S3 Access Grants workflow is as follows:

| Steps | Description |
| --- | --- |
| 1 | Create an S3 Access Grants instance<br><br>To get started, initiate an S3 Access Grants instance that will contain your individual access grants. |
| 2 | Register a location<br><br>Second, register an S3 data location (such as the default, `s3://`) and then specify a default IAM role that S3 Access Grants assumes when providing access to the S3 data location. You can also add custom locations to specific buckets or prefixes and map those to custom IAM roles. |
| 3 | Create grants<br><br>Create individual permission grants. Specify in these permissio n grants the registered S3 location, the scope of data access within the location, the identity of the grantee, and their access level (`READ`, `WRITE`, or `READWRITE` ). |
| 4 | Request access to S3 data<br><br>When users, applications, and AWS services want to access S3 data, they first make an access request. S3 Access Grants determines if the request should be authorized. If there is a corresponding grant that authorizes access, S3 Access Grants uses the registered location's IAM role that's associated with that grant to vend temporary credentials back to the request er. |

| Steps | Description |
| --- | --- |
| 5 | Access S3 data<br><br>Applications use the temporary credentials vended by S3 Access Grants to access S3 data. |

## Create an S3 Access Grants instance

To get started with using AmazonS3 Access Grants, you first create an S3 Access Grants instance. You can create only one S3 Access Grants instance per AWS Region per account. The S3 Access Grants instance serves as the container for your S3 Access Grants resources, which include registered locations and grants.

With S3 Access Grants, you can create permission grants to your S3 data for AWS Identity and Access Management (IAM) users and roles. If you've added your corporate identity directory to AWS IAM Identity Center, you can associate this IAM Identity Center instance of your corporate directory with your S3 Access Grants instance. After you've done so, you can create access grants for your corporate users and groups. If you haven't yet added your corporate directory to IAM Identity Center, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

You can create an S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

### Using the S3 console

Before you can grant access to your S3 data with S3 Access Grants, you must first create an S3 Access Grants instance in the same AWS Region as your S3 data.

#### Prerequisites

If you want to grant access to your S3 data by using identities from your corporate directory, add your corporate identity directory to AWS IAM Identity Center. If you're not yet ready to do so, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

#### To create an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Access Grants**.
4. On the **S3 Access Grants** page, choose **Create S3 Access Grants instance**.

   a. In **Step 1** of the **Set up Access Grants instance** wizard, verify that you want to create the instance in the current AWS Region. Make sure that this is the same AWS Region where your S3 data is located. You can create one S3 Access Grants instance per AWS Region per account.

   b. (Optional) If you've added your corporate identity directory to AWS IAM Identity Center, you can associate this IAM Identity Center instance of your corporate directory with your S3 Access Grants instance.

      To do so, select **Add IAM Identity Center instance in *region***. Then enter the IAM Identity Center instance Amazon Resource Name (ARN).

      If you haven't yet added your corporate directory to IAM Identity Center, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

   c. To create the S3 Access Grants instance, choose **Next**. To register a location, see Step 2 - register a location.
5. If **Next** or **Create S3 Access Grants instance** is disabled:

   **Cannot create instance**

   - You might already have an S3 Access Grants instance in the same AWS Region. In the left navigation pane, choose **Access Grants**. On the **S3 Access Grants** page, scroll down to the **S3 Access Grants instance in your account** section o determine if an instance already exists.
   - You might not have the `s3:CreateAccessGrantsInstance` permission which is required to create an S3 Access Grants instance. Contact your account administrator. For additional permissions that are required if you are associating an IAM Identity Center instance, with your S3 Access Grants instance, see CreateAccessGrantsInstance .

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *user input placeholders* with your own information.

**Example Create an S3 Access Grants instance**

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

Response:

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

### Using the REST API

You can use the Amazon S3 REST API to create an S3 Access Grants instance. For information on the REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance
- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

### Using the AWS SDKs

This section provides an example of how to create an S3 Access Grants instance by using the AWS SDKs.

Java

This example creates the S3 Access Grants instance, which serves as a container for your individual access grants. You can have one S3 Access Grants instance per AWS Region in your account. The response includes the instance ID `default` and an Amazon Resource Name (ARN) that's generated for your S3 Access Grants instance.

**Example Create an S3 Access Grants instance request**

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
}
```

Response:

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

**Topics**

- View the details of an S3 Access Grants instance
- Associate or disassociate your IAM Identity Center instance
- Delete an S3 Access Grants instance

### View the details of an S3 Access Grants instance

You can view the details of your Amazon S3 Access Grants instance in a particular AWS Region. You can also list your S3 Access Grants instances, including the instances that have been shared with you through AWS Resource Access Manager (AWS RAM).

You can view the details of your S3 Access Grants instance or list your S3 Access Grants instances by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

### Using the S3 console

#### To view an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. The **S3 Access Grants** page lists your S3 Access Grants instances and any cross-account instances that have been shared with your account. To view the details of an instance, choose **View details**.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

#### Example – Get the details of an S3 Access Grants instance

```
aws s3control get-access-grants-instance \
 --account-id 111122223333 \
 --region us-east-2
```

Response:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

#### Example – List all S3 Access Grants instances for an account

This action lists the S3 Access Grants instances for an account. You can only have one S3 Access Grants instance per AWS Region. This action also lists other cross-account S3 Access Grants instances that your account has access to.

```
aws s3control list-access-grants-instances \
 --account-id 111122223333 \
 --region us-east-2
```

Response:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

### Using the REST API

For information about the Amazon S3 REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- ListAccessGrantsInstances

### Using the AWS SDKs

This section provides examples of how to get the details of an S3 Access Grants instance by using the AWS SDKs.

To use the following examples, replace the *user input placeholders* with your own information.

Java

#### Example – Get an S3 Access Grants instance

```
public void getAccessGrantsInstance() {
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
```

```
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

Response:

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

**Example – List all S3 Access Grants instances for an account**

This action lists the S3 Access Grants instances for an account. You can only have one S3 Access Grants instance per Region. This action can also list other cross-account S3 Access Grants instances that your account has access to.

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

### Associate or disassociate your IAM Identity Center instance

In Amazon S3 Access Grants, you can associate the AWS IAM Identity Center instance of your corporate identity directory with an S3 Access Grants instance. After you do so, you can create

access grants for your corporate directory users and groups, in addition to AWS Identity and Access Management (IAM) users and roles.

If you no longer want to create access grants for your corporate directory users and groups, you can disassociate your IAM Identity Center instance from your S3 Access Grants instance.

You can associate or disassociate an IAM Identity Center instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

**Using the S3 console**

Before you associate your IAM Identity Center instance with your S3 Access Grants instance, you must add your corporate identity directory to IAM Identity Center. For more information, see the section called "S3 Access Grants and corporate directory identities".

**To associate an IAM Identity Center instance with an S3 Access Grants instance**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, in the **IAM Identity Center** section, choose to either **Add** an IAM Identity Center instance or **Deregister** an already associated IAM Identity Center instance.

**Using the AWS CLI**

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

**Example – Associate an IAM Identity Center instance with an S3 Access Grants instance**

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
```

```
 --region eu-central-1

// No response body
```

**Example – Disassociate an IAM Identity Center instance from an S3 Access Grants instance**

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

**Using the REST API**

For information about the Amazon S3 REST API support for managing the association between an IAM Identity Center instance and an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

### Delete an S3 Access Grants instance

You can delete an Amazon S3 Access Grants instance from an AWS Region in your account. However, before you can delete an S3 Access Grants instance, you must first do the following:

- Delete all resources within the S3 Access Grants instance, including all grants and locations. For more information, see Delete a grant and Delete a location.
- If you've associated an AWS IAM Identity Center instance with your S3 Access Grants instance, you must disassociate the IAM Identity Center instance. For more information, see Associate or disassociate your IAM Identity Center instance.

> **Important**
>
> If you delete an S3 Access Grants instance, the deletion is permanent and can't be undone. All grantees that were given access through the grants in this S3 Access Grants instance will lose access to your S3 data.

You can delete an S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

### Using the S3 console

#### To delete an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the instance details page, choose **Delete instance** in the upper-right corner.
6. In the dialog box that appears, choose **Delete**. This action can't be undone.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

> **Note**
>
> Before you can delete an S3 Access Grants instance, you must first delete all grants and locations created within the S3 Access Grants instance. If you have associated an IAM Identity Center center instance with your S3 Access Grants instance, you must disassociate it first.

#### Example – Delete an S3 Access Grants instance

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
--profile access-grants-profile \
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \
```

```
// No response body
```

### Using the REST API

For information about the Amazon S3 REST API support for deleting an S3 Access Grants instance, see DeleteAccessGrantsInstance in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

This section provides examples of how to delete an S3 Access Grants instance by using the AWS SDKs.

To use the following example, replace the *`user input placeholders`* with your own information.

Java

> **Note**
>
> Before you can delete an S3 Access Grants instance, you must first delete all grants and locations created within the S3 Access Grants instance. If you have associated an IAM Identity Center center instance with your S3 Access Grants instance, you must disassociate it first.

#### Example – Delete an S3 Access Grants instance

```
public void deleteAccessGrantsInstance() {
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

## Register a location

After you create an Amazon S3 Access Grants instance in an AWS Region in your account, you register an S3 location in that instance. An S3 Access Grants location maps the default S3 location

(`s3://`), a bucket, or a prefix to an AWS Identity and Access Management (IAM) role. S3 Access Grants assumes this IAM role to vend temporary credentials to the grantee that is accessing that particular location. You must first register at least one location in your S3 Access Grants instance before you can create an access grant.

**Recommended use case**

We recommend that you register the default location (`s3://`) and map it to an IAM role. The location at the default S3 path (`s3://`) covers access to all of your S3 buckets in that AWS Region of your account. When you create an access grant, you can narrow the grant scope to a bucket, a prefix, or an object within the default location.

**Complex access-management use cases**

More complex access-management use cases might require you to register more than the default location. Some examples of such use cases are:

- Suppose that the *amzn-s3-demo-bucket* is a registered location in your S3 Access Grants instance with an IAM role mapped to it, but this IAM role is denied access to a particular prefix within the bucket. In this case, you can register the prefix that the IAM role does not have access to as a separate location and map that location to a different IAM role with the necessary access.
- Suppose that you want to create grants that restrict access to only the users within a virtual private cloud (VPC) endpoint. In this case, you can register a location for a bucket in which the IAM role restricts access to the VPC endpoint. Later, when a grantee asks S3 Access Grants for credentials, S3 Access Grants assumes the location's IAM role to vend the temporary credentials. This credential will deny access to the specific bucket unless the caller is within the VPC endpoint. This deny permission is applied in addition to the regular READ, WRITE, or READWRITE permission specified in the grant.

When you register a location, you must also specify the IAM role that S3 Access Grants assumes to vend temporary credentials and to scope the permissions for a specific grant.

If your use case requires you to register multiple locations in your S3 Access Grants instance, you can register any of the following:

| S3 URI | IAM role | Description |
| --- | --- | --- |
| s3:// | *Default-IAM-role* | The default location, s3://, includes all buckets in the AWS Region. |
| s3://*amzn-s3-demo-bucket1* / | *IAM-role-For-bucket* | This location includes all objects in the specified bucket. |
| s3://*amzn-s3-demo-bucket1* /*prefix-name* | *IAM-role-For-prefix* | This location includes all objects in the bucket with an object key name that starts with this prefix. |

Before you can register a specific bucket or prefix, make sure that you do the following:

- Create one or more buckets that contain the data that you want to grant access to. These buckets must be located in the same AWS Region as your S3 Access Grants instance. For more information, see Creating a bucket.

  Adding a prefix is an optional step. Prefixes are strings at the beginning of an object key name. You can use them to organize objects in your bucket as well as for access management. To add a prefix to a bucket, see Creating object key names.
- Create an IAM role that has permission to access your S3 data in the AWS Region. For more information, see Creating IAM roles in the *AWS IAM Identity Center user guide*.
- In the IAM role trust policy, give the S3 Access Grants service (access-grants.s3.amazonaws.com) principal access to the IAM role that you created. To do so, you can create a JSON file that contains the following statements. To add the trust policy to your account, see Create a role using custom trust policies.

  *TestRolePolicy.json*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity", "sts:SetContext"],
      "Effect": "Allow",
```

```
      "Principal": {"Service":"access-grants.s3.amazonaws.com"},
   "Condition":{
  "StringEquals":{
   "aws:SourceAccount":"accountId",
   "aws:SourceArn":"arn:aws:s3:region:accountId:access-grants/default"
  }
  //Optionally, for an IAM Identity Center use case, add:
  "ForAnyValue:StringEquals": {
   "aws:RequestContextProvider":"arn:aws:iam::aws:contextProvider/IdentityCenter"
  }
 }
    }
  ]
}
```

- Create an IAM policy to attach Amazon S3 permissions to the IAM role that you created. See the following example `iam-policy.json` file and replace the *`user input placeholders`* with your own information.

  **Note**

  - If you use server-side encryption with AWS Key Management Service (AWS KMS) keys to encrypt your data, the following example includes the necessary AWS KMS permissions for the IAM role in the policy. If you do not use this feature, you can remove these permissions from your IAM policy.
  - You can restrict the IAM role to access S3 data only if the credentials are vended by S3 Access Grants. This example shows you how to add a `Condition` statement for a specific S3 Access Grants instance. To use this `Condition`, replace the S3 Access Grants instance ARN in the `Condition` statement with your S3 Access Grants instance ARN, which has the format: `arn:aws:s3:`*`region`*`:`*`accountId`*`:access-grants/default`

  *iam-policy.json*

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
```

```
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS
 Region:accountId:access-grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
```

```
            "Effect":"Allow",
            "Action":[
               "s3:ListBucket"
            ],
            "Resource":[
               "arn:aws:s3:::*"
            ],
            "Condition":{
               "StringEquals": { "aws:ResourceAccount": "accountId" },
               "ArnEquals": {
                   "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS
 Region:accountId:access-grants/default"]
               }
            }
         },
     //Optionally add the following section if you use SSE-KMS encryption
         {
            "Sid": "KMSPermissions",
            "Effect":"Allow",
            "Action":[
               "kms:Decrypt",
               "kms:GenerateDataKey"
            ],
            "Resource":[
               "*"
            ]
         }
      ]
}
```

You can register a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, or the AWS SDKs.

> **Note**
>
> After you register the first location in your S3 Access Grants instance, your instance still does not have any individual access grants in it. To create an access grant, see Create grants.

### Using the S3 console

Before you can grant access to your S3 data with S3 Access Grants, you must have at least one registered location.

**To register a location in your S3 Access Grants instance**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.

   If you're using S3 Access Grants instance for the first time, make sure that you have completed Step 1 - create an S3 Access Grants instance and navigated to **Step 2** of the **Set up Access Grants instance** wizard. If you already have an S3 Access Grants instance, choose **View details**, and then from the **Locations** tab, choose **Register location**.

   a. For the **Location scope**, choose **Browse S3** or enter the S3 URI path to the location that you want to register. For S3 URI formats, see the location formats table. After you enter a URI, you can choose **View** to browse the location.

   b. For the **IAM role**, choose one of the following:

      - **Choose from existing IAM roles**

        Choose an IAM role from the dropdown list. After you choose a role, choose **View** to make sure that this role has the necessary permissions to manage the location that you're registering. Specifically, make sure that this role grants S3 Access Grants the permissions `sts:AssumeRole` and `sts:SetSourceIdentity`.

      - **Enter IAM role ARN**

        Navigate to the IAM Console. Copy the Amazon Resource Name (ARN) of the IAM role and paste it in this box.

   c. To finish, choose **Next** or **Register location**.

4. Troubleshooting:

   **Cannot register location**

   - The location might already be registered.

You might not have the `s3:CreateAccessGrantsLocation` permission to register locations. Contact your account administrator.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can register the default location, `s3://`, or a custom location in your S3 Access Grants instance. Make sure that you first create an IAM role with principal access to the location, and then make sure that you grant S3 Access Grants permission to assume this role.

To use the following example commands, replace the *`user input placeholders`* with your own information.

#### Example Create a resource policy

Create a policy that allows S3 Access Grants to assume the IAM role. To do so, you can create a JSON file that contains the following statements. To add the resource policy to your account, see Create and attach your first customer managed policy.

*TestRolePolicy.json*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

#### Example Create the role

Run the following IAM command to create the role.

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
```

```
 --assume-role-policy-document file://TestRolePolicy.json
```

Running the `create-role` command returns the policy:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

**Example**

Create an IAM policy to attach Amazon S3 permissions to the IAM role. See the following example `iam-policy.json` file and replace the *`user input placeholders`* with your own information.

> **Note**
>
> If you use server-side encryption with AWS Key Management Service (AWS KMS) keys to encrypt your data, the following example adds the necessary AWS KMS permissions for the IAM role in the policy. If you do not use this feature, you can remove these permissions from your IAM policy.

To make sure that the IAM role can only be used to access data in S3 if the credentials are vended out by S3 Access Grants, this example shows you how to add a `Condition` statement that specifies the S3 Access Grants instance (`s3:AccessGrantsInstance:` *`InstanceArn`*) in your IAM policy. When using following example policy, replace the *`user input placeholders`* with your own information.

*iam-policy.json*

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
```

```
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS Region:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS Region:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
```

```
}
```

### Example

Run the following command:

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

### Example Register the default location

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

Response:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://”
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

### Example Register a custom location

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

Response:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
```

```
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

### Using the REST API

For information about Amazon S3 REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations
- UpdateAccessGrantsLocation

### Using the AWS SDKs

This section provides examples of how to register locations by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

You can register the default location, `s3://`, or a custom location in your S3 Access Grants instance. Make sure that you first create an IAM role with principal access to the location, and then make sure that you grant S3 Access Grants permission to assume this role.

To use the following example commands, replace the *`user input placeholders`* with your own information.

**Example Register a default location**

Request:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
```

```
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

Response:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

**Example Register a custom location**

Request:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

Response:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

**Topics**



### View the details of a registered location

You can get the details of a location that's registered in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

#### Using the S3 console

**To view the locations registered in your S3 Access Grants instance**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page for the instance, choose the **Locations** tab.
6. Find the registered location that you want to view. To filter the list of registered locations, use the search box.

#### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

**Example – Get the details of a registered location**

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
```

```
--access-grants-location-id default
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

**Example – List all of the locations that are registered in an S3 Access Grants instance**

To restrict the results to an S3 prefix or bucket, you can optionally use the `--location-scope s3://bucket-and-or-prefix` parameter.

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

Response:

```
{"AccessGrantsLocationsList": [
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

### Using the REST API

For information about the Amazon S3 REST API support for getting the details of a registered location or listing all of the locations that are registered with an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrantsLocation
- ListAccessGrantsLocations

### Using the AWS SDKs

This section provides examples of how to get the details of a registered location or list all of the registered locations in an S3 Access Grants instance by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

#### Example – Get the details of a registered location

```
public void getAccessGrantsLocation() {
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

Response:

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
```

```
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

### Example – List all registered locations in an S3 Access Grants instance

To restrict the results to an S3 prefix or bucket, you can optionally pass an S3 URI, such as `s3://bucket-and-or-prefix`, in the `LocationScope` parameter.

```
public void listAccessGrantsLocations() {

ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

### Update a registered location

You can update the AWS Identity and Access Management (IAM) role of a location that's registered in your Amazon S3 Access Grants instance. For each new IAM role that you use to register a location in S3 Access Grants, be sure to give the S3 Access Grants service principal (`access-grants.s3.amazonaws.com`) access to this role. To do this, add an entry for the new IAM role in the same trust policy JSON file that you used when you first registered the location.

You can update a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

#### Using the S3 console

#### To update the IAM role of a location registered with your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page for the instance, choose the **Locations** tab.
6. Find the location that you want to update. To filter the list of locations, use the search box.
7. Choose the options button next to the registered location that you want to update.
8. Update the IAM role, and then choose **Save changes**.

#### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

#### Example – Update the IAM role of a registered location

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
```

```
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

### Using the REST API

For information on the Amazon S3 REST API support for updating a location in an S3 Access Grants instance, see UpdateAccessGrantsLocation in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

This section provides examples of how to update the IAM role of a registered location by using the AWS SDKs.

To use the following example, replace the *`user input placeholders`* with your own information.

Java

#### Example – Update the IAM role of a registered location

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

Response:

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

### Delete a registered location

You can delete a location registration from an Amazon S3 Access Grants instance. Deleting the location deregisters it from the S3 Access Grants instance.

Before you can remove a location registration from an S3 Access Grants instance, you must delete all of the grants that are associated with this location. For information about how to delete grants, see Delete a grant.

You can delete a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

#### Using the S3 console

#### To delete a location registration from your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page for the instance, choose the **Locations** tab.
6. Find the location that you want to update. To filter the list of locations, use the search box.
7. Choose the option button next to the registered location that you want to delete.
8. Choose **Deregister**.
9. A dialog box appears that warns you that this action can't be undone. To delete the location, choose **Deregister**.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *user input placeholders* with your own information.

#### Example – Delete a location registration

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

### Using the REST API

For information about the Amazon S3 REST API support for deleting a location from an S3 Access Grants instance, see DeleteAccessGrantsLocation in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

This section provides an example of how to delete a location by using the AWS SDKs.

To use the following example, replace the *user input placeholders* with your own information.

Java

#### Example – Delete a location registration

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

Response:

```
DeleteAccessGrantsLocationResponse()
```

## Create grants

An individual access *grant* in an S3 Access Grants instance allows a specific identity—an AWS Identity and Access Management (IAM) principal, or a user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance. A location maps buckets or prefixes to an IAM role. S3 Access Grants assumes this IAM role to vend temporary credentials to grantees.

After you register at least one location in your S3 Access Grants instance, you can create an access grant.

The grantee can be an IAM user or role or a directory user or group. A directory user is a user from your corporate directory or external identity source that you associated with your S3 Access Grants instance. For more information, see S3 Access Grants and corporate directory identities. To create a grant for a specific directory user or group from IAM Identity Center, find the GUID that IAM Identity Center uses to identify that user in IAM Identity Center, for example, `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`. For more information about how to use IAM Identity Center to view user information, see View user and group assignments in the *AWS IAM Identity Center user guide*.

You can grant access to a bucket, a prefix, or an object. A prefix in Amazon S3 is a string of characters in the beginning of an object key name that is used to organize objects within a bucket. This can be any string of allowed characters, for example, object key names in your bucket that start with the `engineering/` prefix.

### Subprefix

When granting access to a registered location, you can use the `Subprefix` field to narrow the scope of access to a subset of the location scope. If the registered location that you choose for the grant is the default S3 path (`s3://`), you must narrow the grant scope. You cannot create an access grant for the default location (`s3://`), which would give the grantee access to every bucket in an AWS Region. Instead, you must narrow the grant scope to one of the following:

- A bucket: `s3://`*`bucket`*`/*`
- A prefix within a bucket: `s3://`*`bucket`*`/`*`prefix`*`*`
- A prefix within a prefix: `s3://`*`bucket`*`/`*`prefixA`*`/`*`prefixB`*`*`

- An object: `s3://bucket/object-key-name`

If you are creating an access grant where the registered location is a bucket, you can pass one of the following in the `Subprefix` field to narrow the grant scope:

- A prefix within the bucket: *prefix**
- A prefix within a prefix: *prefixA/prefixB**
- An object: */object-key-name*

After you create the grant, the grant scope that's displayed in the Amazon S3 console or the `GrantScope` that is returned in the API or AWS Command Line Interface (AWS CLI) response is the result of concatenating the location path with the `Subprefix`. Make sure that this concatenated path maps correctly to the S3 bucket, prefix, or object to which you want to grant access.

> **Note**
>
> - If you need to create an access grant that grants access to only one object, you must specify that the grant type is for an object. To do this in an API call or a CLI command, pass the `s3PrefixType` parameter with the value `Object`. In the Amazon S3 console, when you create the grant, after you select a location, under **Grant Scope**, select the **Grant scope is an object** checkbox.
> - You cannot create a grant to a bucket if the bucket does not yet exist. However, you can create a grant to a prefix that does not yet exisit.
> - For the maximum number of grants that you can create in your S3 Access Grants instance, see S3 Access Grants limitations.

You can create an access grant by using the Amazon S3 console, AWS CLI, the Amazon S3 REST API, and AWS SDKs.

**Using the S3 console**

**To create an access grant**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.

   If you're using the S3 Access Grants instance for the first time, make sure that you have completed Step 2 - register a location and navigated to **Step 3** of the **Set up Access Grants instance** wizard. If you already have an S3 Access Grants instance, choose **View details**, and then from the **Grants** tab, choose **Create grant**.

   a. In the **Grant scope** section, select or enter a registered location.

      If you selected the default `s3://` location, use the **Subprefix** box to can narrow the scope of the access grant. For more information, see Subprefix. If you're granting access only to an object, select **Grant scope is an object**.

   b. Under **Permissions and access**, select the **Permission** level, either **Read**, **Write**, or both.

      Then choose the **Grantee type**. If you have added your corporate directory to IAM Identity Center and associated this IAM Identity Center instance with your S3 Access Grants instance, you can choose **Directory identity from IAM Identity Center**. If you choose this option, get the ID of the user or group from IAM Identity Center and enter it in this section.

      If the **Grantee type** is an IAM user or role, choose **IAM principal**. Under **IAM principal type**, choose **User** or **Role**. Then, under **IAM principal user**, either choose from the list or enter the identity's ID.

   c. To create the S3 Access Grants grant, choose **Next** or **Create grant**.
4. If **Next** or **Create grant** is disabled:

   **Cannot create grant**

   - You might need to register a location first in your S3 Access Grants instance.
   - You might not have the `s3:CreateAccessGrant` permission to create an access grant. Contact your account administrator.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following examples show how to create an access grant request for an IAM principal and how to create an access grant request for a corporate directory user or group.

To use the following example commands, replace the *user input placeholders* with your own information.

> **Note**
>
> If you're creating an access grant that grants access to only one object, include the required parameter `--s3-prefix-type Object`.

### Example Create an access grant request for an IAM principal

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

### Example Create an access grant response

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-BUCKET-EXAMPLE/prefix*",
    "Permission": "READ"
}
```

### Create an access grant request for a directory user or group

To create an access grant request for a directory user or group, you must first get the GUID for the directory user or group by running one of the following commands.

**Example Get a GUID for a directory user or group**

You can find the GUID of an IAM Identity Center user through the IAM Identity Center console or by using the AWS CLI or AWS SDKs. The following command lists the users in he specified IAM Identity Center instance, with their names and identifiers.

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

This command lists the groups in the specified IAM Identity Center instance.

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

**Example Create an access grant for a directory user or group**

This command is similar to creating a grant for IAM users or roles, except the grantee type is `DIRECTORY_USER` or `DIRECTORY_GROUP`, and the grantee identifier is the GUID for the directory user or group.

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="DOC-EXAMPLE-BUCKET/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

**Using the REST API**

For information about the Amazon S3 REST API support for managing access grants, see the following sections in the *Amazon Simple Storage Service API Reference*:

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

### Using the AWS SDKs

This section provides examples of how to create an access grant by using the AWS SDKs.

Java

To use the following example, replace the *user input placeholders* with your own information:

> **Note**
>
> If you are creating an access grant that grants access to only one object, include the required parameter `.s3PrefixType(S3PrefixType.Object)`.

#### Example Create an access grant request

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

#### Example Create an access grant response

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
```

```
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-BUCKET-EXAMPLE/prefixB,
Permission=READ
)
```

**Topics**

- View a grant
- Delete a grant

### View a grant

You can view the details of an access grant in your Amazon S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

#### Using the S3 console

**To view the details of an access grant**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, choose the **Grants** tab.
6. In the **Grants** section, find the access grant that you want to view. To filter the list of grants, use the search box.

#### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example commands, replace the *user input placeholders* with your own information.

**Example – Get the details of an access grant**

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://amzn-s3-demo-bucket/"
}
```

**Example – List all of the access grants in an S3 Access Grants instance**

You can optionally use the following parameters to restrict the results to an S3 prefix or AWS Identity and Access Management (IAM) identity:

- **Subprefix** – `--grant-scope s3://bucket-name/prefix*`
- **IAM identity** – `--grantee-type IAM` and `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

Response:

```
{
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixB*"
        },

    ]
}
```

### Using the REST API

You can use Amazon S3 API operations to view the details of an access grant and list all access grants in an S3 Access Grants instance. For information about the REST API support for managing access grants, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrant
- ListAccessGrants

### Using the AWS SDKs

This section provides examples of how to get the details of an access grant by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

#### Example – Get the details of an access grant

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

Response:

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://amzn-s3-demo-bucket/
)
```

**Example – List all of the access grants in an S3 Access Grants instance**

You can optionally use these parameters to restrict the results to an S3 prefix or IAM identity:

- **Scope** – `GrantScope=s3://bucket-name/prefix*`
- **Grantee** – `GranteeType=IAM` and `GranteeIdentifier= arn:aws:iam::111122223333:role/accessGrantsConsumerRole`

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://amzn-s3-demo-bucket/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
```

```
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://amzn-s3-demo-bucket/prefixB*
)
]
)
```

### Delete a grant

You can delete access grants from your Amazon S3 Access Grants instance. You can't undo an access grant deletion. After you delete an access grant, the grantee will no longer have access to your Amazon S3 data.

You can delete an access grant by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

#### Using the S3 console

**To delete an access grant**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, choose the **Grants** tab.
6. Search for the grant that you want to delete. When you locate the grant, choose the radio button next to it.
7. Choose **Delete**. A dialog box appears with a warning that your action can't be undone. Choose **Delete** again to delete the grant.

#### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

#### Example – Delete an access grant

```
aws s3control delete-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

### Using the REST API

For information about the Amazon S3 REST API support for managing access grants, see DeleteAccessGrant in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

This section provides examples of how to delete an access grant by using the AWS SDKs. To use the following example, replace the *user input placeholders* with your own information.

Java

#### Example – Delete an access grant

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

Response:

```
DeleteAccessGrantResponse()
```

## Request access to Amazon S3 data through S3 Access Grants

After you use Amazon S3 Access Grants to create an access grant that gives AWS Identity and Access Management (IAM) principals, your corporate directory identities, or authorized applications access to your S3 data, your grantees can request credentials to access this data.

When an application or AWS service uses the `GetDataAccess` API operation to ask S3 Access Grants for access to your S3 data on behalf of a grantee, S3 Access Grants first verifies that you have granted this identity access to the data. Then, S3 Access Grants uses the AssumeRole API operation to obtain a temporary credential token and vends it to the requester. This temporary credential token is an AWS Security Token Service (AWS STS) token.

The `GetDataAccess` request must include the `target` parameter, which specifies the scope of the S3 data that the temporary credentials apply to. This `target` scope can be the same as the scope of the grant or a subset of that scope, but the `target` scope must be within the scope of the grant that was given to the requester. The request must also specify the `permission` parameter to indicate the permission level for the temporary credentials, whether `READ`, `WRITE`, or `READWRITE`.

The requester can specify the privilege level of the temporary token in their credential request. Using the `privilege` parameter, the requester can reduce or increase the temporary credentials' scope of access, within the boundaries of the grant scope. The default value of the `privilege` parameter is `Default`, which means that the target scope of the credential returned is the original grant scope. The other possible value for `privilege` is `Minimal`. If the `target` scope is reduced from the original grant scope, then the temporary credential is de-scoped to match the `target` scope, as long as the `target` scope is within the grant scope.

The following table details the effect of the `privilege` parameter on two grants. One grant has the scope `S3://amzn-s3-demo-bucket1/bob/*`, which includes the entire bob/ prefix in the *amzn-s3-demo-bucket1* bucket. The other grant has the scope `S3://amzn-s3-demo-bucket1/bob/reports/*`, which includes only the bob/reports/ prefix in the *amzn-s3-demo-bucket1* bucket.

| Grant scope | Requested scope | Privilege | Returned scope | Effect |
|---|---|---|---|---|
| S3://*amzn-s3-demo-bucket1* /bob/* | *amzn-s3-demo-bucket1* /bob/* | Default | *amzn-s3-demo-bucket1* /bob/* | The requester has access to all objects that have key names that start with the prefix *bob/* in the *amzn-s3-demo-bucket1* bucket. |
| S3://*amzn-s3-d* | *amzn-s3-demo-* | Minimal | *amzn-s3-demo-bucket1* /bob/ | Without a wild card * character after the |

| Grant scope | Requested scope | Privilege | Returned scope | Effect |
| --- | --- | --- | --- | --- |
| *emo-bucket1* /bob/* | *bucket1* /bob/ | | | prefix name bob/, the requester has access to only the object named bob/ in the *amzn-s3-demo-bucket1* bucket. It's not common to have such an object. The requester doesn't have access to any other objects, including those that have key names that start with the bob/ prefix. |
| S3://*amzn-s3-demo-bucket1* /bob/* | *amzn-s3-demo-bucket1* /bob/images/* | Minimal | *amzn-s3-demo-bucket1* /bob/images/* | The requester has access to all objects that have key names that start with the prefix *bob/images/** in the *amzn-s3-demo-bucket1* bucket. |
| S3://*amzn-s3-demo-bucket1* /bob/reports/* | *amzn-s3-demo-bucket1* /bob/reports/file.txt | Default | *amzn-s3-demo-bucket1* /bob/reports/* | The requester has access to all objects that have key names that start with the bob/reports prefix in the *amzn-s3-demo-bucket1* bucket, which is the scope of the matching grant. |

| Grant scope | Requested scope | Privilege | Returned scope | Effect |
| --- | --- | --- | --- | --- |
| S3://*amzn-s3-demo-bucket1* /bob/reports/* | *amzn-s3-demo-bucket1* /bob/reports/file.txt | Minimal | *amzn-s3-demo-bucket1* /bob/reports/file.txt | The requester has access only to the object with the key name bob/reports/file.txt in the *amzn-s3-demo-bucket1* bucket. The requester has no access to any other object. |

The `durationSeconds` parameter sets the temporary credential's duration, in seconds. The default value is `3600` seconds (1 hour), but the requester (the grantee) can specify a range from `900` seconds (15 minutes) up to `43200` seconds (12 hours). If the grantee requests a value higher than this maximum, the request fails.

> **Note**
>
> In your request for a temporary token, if the location is an object, set the value of the `targetType` parameter in your request to `Object`. This parameter is required only if the location is an object and the privilege level is `Minimal`. If the location is a bucket or a prefix, you don't need to specify this parameter.

For more information, see GetDataAccess in the *Amazon Simple Storage Service API Reference*.

You can request temporary credentials by using AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

**Using the AWS CLI**

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

**Example Request temporary credentials**

Request:

```
aws s3control get-data-access \
--account-id 111122223333 \
--target s3://amzn-s3-demo-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

Response:

```
{
"Credentials": {
 "AccessKeyId": "Example-key-id",
 "SecretAccessKey": "Example-access-key",
 "SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
    "MatchedGrantTarget": "s3://amzn-s3-demo-bucket/prefixA**"
}
```

**Using the REST API**

For information about the Amazon S3 REST API support for requesting temporary credentials from S3 Access Grants, see GetDataAccess in the *Amazon Simple Storage Service API Reference*.

**Using the AWS SDKs**

This section provides an example of how grantees request temporary credentials from S3 Access Grants by using the AWS SDKs.

Java

The following code example returns the temporary credentials that the grantee uses to access your S3 data. To use this code example, replace the *`user input placeholders`* with your own information.

**Example Get temporary credentials**

Request:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://amzn-s3-demo-bucket/prefixA*")
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

Response:

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

## Access S3 data through an access grant

After a grantee obtains temporary credentials through their access grant, they can use these temporary credentials to call Amazon S3 API operations to access your data.

Grantees can access S3 data by using the AWS Command Line Interface (AWS CLI), the AWS SDKs, and the Amazon S3 REST API.

### Using the AWS CLI

After the grantee obtains their temporary credentials from S3 Access Grants, they can set up a profile with these credentials to retrieve the data.

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example commands, replace the *`user input placeholders`* with your own information.

**Example – Set up a profile**

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

To use the following example command, replace the *user input placeholders* with your own information.

**Example – Get the S3 data**

The grantee can use the get-object AWS CLI command to access the data. The grantee can also use put-object, ls, and other S3 AWS CLI commands.

```
aws s3api get-object \
--bucket amzn-s3-demo-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

**Using the AWS SDKs**

This section provides examples of how grantees can access your S3 data by using the AWS SDKs.

Java

> For examples of how to get S3 data by using temporary credentials, see how to get an object by using the AWS SDKs and Amazon S3 code examples for the AWS SDK for Java 2.x.

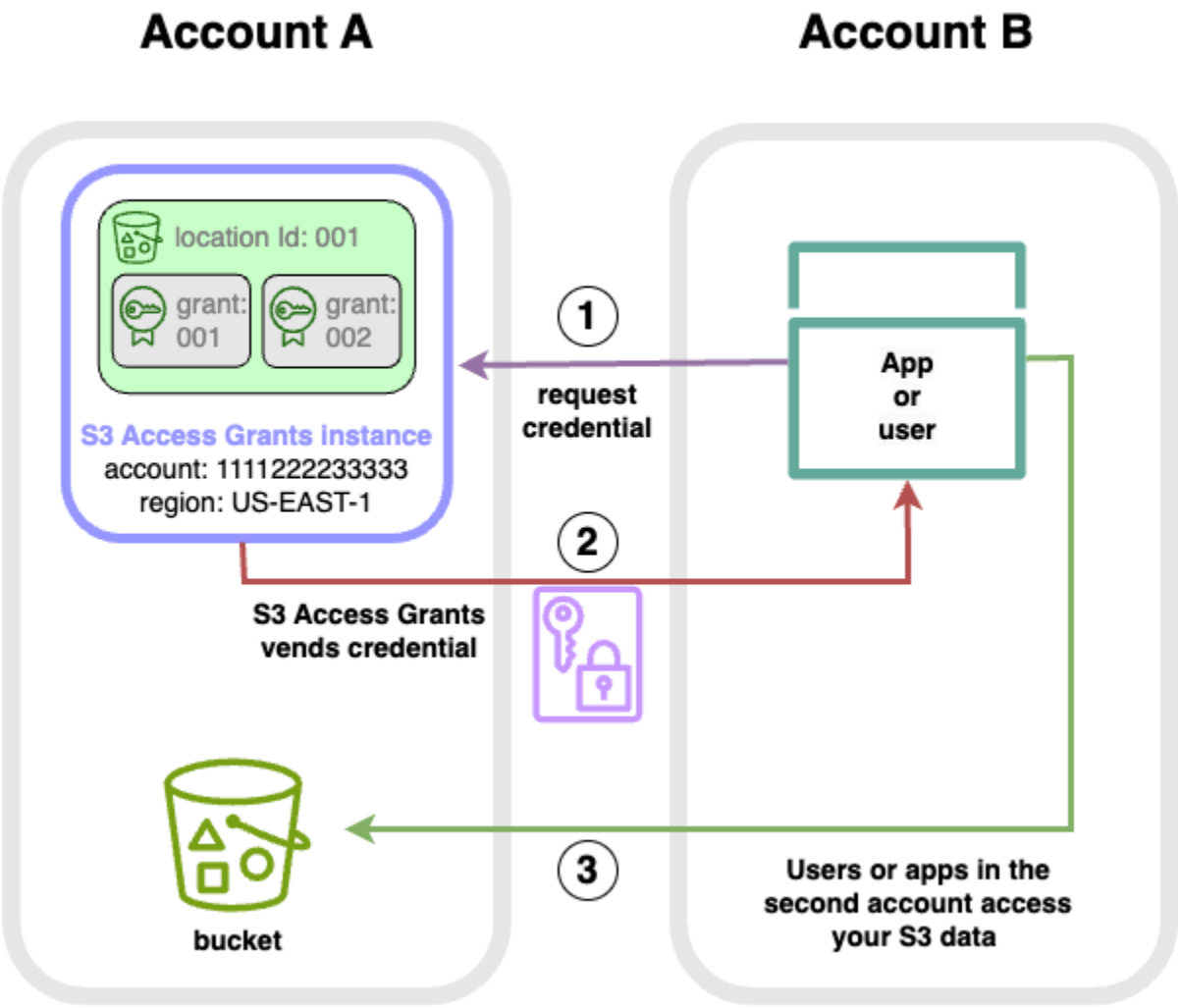
## S3 Access Grants cross-account access

With S3 Access Grants, you can grant Amazon S3 data access to the following:

- AWS Identity and Access Management (IAM) identities within your account
- IAM identities in other AWS accounts
- Directory users or groups in your AWS IAM Identity Center instance

First, configure cross-account access for the other account. This includes granting access to your S3 Access Grants instance by using a resource policy. Then, grant access to your S3 data (buckets, prefixes, or objects) by using grants.

After you configure cross-account access, the other account can request temporary access credentials to your Amazon S3 data from S3 Access Grants. The following image shows the user flow for cross-account S3 access through S3 Access Grants:




1. Users or applications in a second account (B) request credentials from the S3 Access Grants instance in your account (A), where the Amazon S3 data is stored. For more information, see Request access to Amazon S3 data through S3 Access Grants.
2. The S3 Access Grants instance in your account (A) returns temporary credentials if there is a grant that gives the second account access to your Amazon S3 data. For more information, see the section called “Create grants”.

3. Users or applications in the second account (B) use the S3 Access Grants-vended credentials to access the S3 data in your account (A).

### Configuring S3 Access Grants cross-account access

To grant cross-account S3 access through S3 Access Grants, follow these steps:

- **Step 1:** Configure an S3 Access Grants instance in your account, for example, account ID 111122223333, where the S3 data is stored.
- **Step 2:** Configure the resource policy for the S3 Access Grants instance in your account 111122223333 to give access to the second account, for example, account ID 444455556666.
- **Step 3:** Configure the IAM permissions for the IAM Principal in the second account 444455556666 to request credentials from the S3 Access Grants instance in your account 111122223333.
- **Step 4:** Create a grant in your account 111122223333 that gives the IAM Principal in the second account 444455556666 access to some of the S3 data in your account 111122223333.

### Step 1: Configure an S3 Access Grants instance in your account

First, you must have an S3 Access Grants instance in your account 111122223333 to manage access to your Amazon S3 data. You must create an S3 Access Grants instance in each AWS Region where the S3 data that you want to share is stored. If you are sharing data in more than one AWS Region, then repeat each of these configuration steps for each AWS Region. If you already have an S3 Access Grants instance in the AWS Region where your S3 data is stored, proceed to the next step. If you haven't configured an S3 Access Grants instance, see Create an S3 Access Grants instance to complete this step.

### Step 2: Configure the resource policy for your S3 Access Grants instance to grant cross-account access

After you create an S3 Access Grants instance in your account 111122223333 for cross-account access, configure the resource-based policy for the S3 Access Grants instance in your account 111122223333 to grant cross-account access. The S3 Access Grants instance itself supports resource-based policies. With the correct resource-based policy in place, you can grant access for AWS Identity and Access Management (IAM) users or roles from other AWS accounts to your S3 Access Grants instance. Cross-account access only grants these permissions (actions):

- `s3:GetAccessGrantsInstanceForPrefix` — the user, role, or app can retrieve the S3 Access Grants instance that contains a particular prefix.
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:GetDataAccess` — the user, role, or app can request temporary credentials based on the access you were granted through S3 Access Grants. Use these credentials to access the S3 data to which you have been granted access.

You can choose which of these permissions to include in the resource policy. This resource policy on the S3 Access Grants instance is a normal resource-based policy and supports everything that the IAM policy language supports. In the same policy, you can grant access to specific IAM identities in your account `111122223333`, for example, by using the `aws:PrincipalArn` condition, but you don't have to do that with S3 Access Grants. Instead, within your S3 Access Grants instance, you can create grants for individual IAM identities from your account, as well as for the other account. By managing each access grant through S3 Access Grants, you can scale your permissions.

If you already use AWS Resource Access Manager (AWS RAM), you can use it to share your `s3:AccessGrants` resources with other accounts or within your organization. See Working with shared AWS resources for more information. If you don't use AWS RAM, you can also add the resource policy by using the S3 Access Grants API operations or the AWS Command Line Interface (AWS CLI).

**Using the S3 console**

We recommend that you use the AWS Resource Access Manager (AWS RAM) Console to share your `s3:AccessGrants` resources with other accounts or within your organization. To share S3 Access Grants cross-account, do the following:

**To configure the S3 Access Grants instance resource policy:**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the AWS Region from the AWS Region selector.
3. From the left navigation pane, select **Access Grants**.
4. On the Access Grants instance page, in the **Instance in this account** section, select **Share instance**. This will redirect you to the AWS RAM Console.
5. Select **Create resource share**.

6. Follow the AWS RAM steps to create the resource share. For more information, see Creating a resource share in AWS RAM.

### Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can add the resource policy by using the `put-access-grants-instance-resource-policy` CLI command.

If you want to grant cross-account access for the S3 Access Grants instance is in your account `111122223333` to the second account `444455556666`, the resource policy for the S3 Access Grants instance in your account `111122223333` should give the second account `444455556666` permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account `444455556666` as the `Principal`. To use the following example, replace the *`user input placeholders`* with your own information.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
```

```
}
```

To add or update the S3 Access Grants instance resource policy, use the following command. When you use the following example command, replace the *user input placeholders* with your own information.

**Example Add or update the S3 Access Grants instance resource policy**

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
--policy file://resourcePolicy.json \
--region us-east-2
{
 "Policy": "{\n
   \"Version\": \"2012-10-17\",\n
   \"Statement\": [{\n
  \"Effect\": \"Allow\",\n
  \"Principal\": {\n
    \"AWS\": \"444455556666\"\n
  },\n
  \"Action\": [\n
    \"s3:ListAccessGrants\",\n
    \"s3:ListAccessGrantsLocations\",\n
    \"s3:GetDataAccess\",\n
    \"s3:GetAccessGrantsInstanceForPrefix\"\n
  ],\n
  \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
    }\n
   ]\n
   }\n",
 "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

**Example Get an S3 Access Grants resource policy**

You can also use the CLI to get or delete a resource policy for an S3 Access Grants instance.

To get an S3 Access Grants resource policy, use the following example command. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
```

```
--region us-east-2

{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":[\"s3:ListAccessGrants\",
\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\"],\"Resource\":\"arn:aws:s3:us-
east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

**Example Delete an S3 Access Grants resource policy**

To delete an S3 Access Grants resource policy, use the following example command. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

// No response body
```

**Using the REST API**

You can add the resource policy by using the PutAccessGrantsInstanceResourcePolicy API.

If you want to grant cross-account access for the S3 Access Grants instance is in your account 111122223333 to the second account 444455556666, the resource policy for the S3 Access Grants instance in your account 111122223333 should give the second account 444455556666 permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account 444455556666 as the `Principal`. To use the following example, replace the *user input placeholders* with your own information.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

You can then use the PutAccessGrantsInstanceResourcePolicy API to configure the policy.

For information on the REST API support to update, get, or delete a resource policy for an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

### Using the AWS SDKs

This section provides you with the AWS SDK examples of how to configure your S3 Access Grants resource policy to grant a second AWS account access to some of your S3 data.

Java

Add, update, get, or delete a resource policy to manage cross-account access to your S3 Access Grants instance.

**Example Add or update an S3 Access Grants instance resource policy**

If you want to grant cross-account access for the S3 Access Grants instance is in your account 111122223333 to the second account 444455556666, the resource policy for the S3

Access Grants instance in your account 111122223333 should give the second account 444455556666 permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account 444455556666 as the `Principal`. To use the following example, replace the *`user input placeholders`* with your own information.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

To add or update an S3 Access Grants instance resource policy, use the following code example:

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
```

```
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

Response:

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

**Example Get an S3 Access Grants resource policy**

To get an S3 Access Grants resource policy, use the following code example. To use the following example command, replace the *user input placeholders* with your own information.

```
public void getAccessGrantsInstanceResourcePolicy() {
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

Response:

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess"],"Resource":"arn:aw
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

**Example Delete an S3 Access Grants resource policy**

To delete an S3 Access Grants resource policy, use the following code example. To use the following example command, replace the *user input placeholders* with your own information.

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
 LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
 }
```

Response:

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

### Step 3: Grant IAM identities in a second account permission to call the S3 Access Grants instance in your account

After the owner of the Amazon S3 data has configured the cross-account policy for the S3 Access Grants instance in account 111122223333, the owner of the second account 444455556666 must create an identity-based policy for its IAM users or roles, and the owner must give them access to the S3 Access Grants instance. In the identity-based policy, include one or more of the following actions, depending on what's granted in the S3 Access Grants instance resource policy and the permissions you want to grant:

- `s3:ListAccessGrants`

- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

Following the AWS cross-account access pattern, the IAM users or roles in the second account `444455556666` must explicitly have one or more of these permissions. For example, grant the `s3:GetDataAccess` permission so that the IAM user or role can call the S3 Access Grants instance in account `111122223333` to request credentials.

To use this example command, replace the *`user input placeholders`* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess",
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }
 ]
}
```

For information on editing IAM identity-based policy, see Editing IAM policies in the *AWS Identity and Access Management guide*.

**Step 4: Create a grant in the S3 Access Grants instance of your account that gives the IAM identity in the second account access to some of your S3 data**

For the final configuration step, you can create a grant in the S3 Access Grants instance in your account 111122223333 that gives access to the IAM identity in the second account 444455556666 to some of the S3 data in your account. You can do this by using the Amazon S3 Console, CLI, API, and SDKs. For more information, see Create grants.

In the grant, specify the AWS ARN of the IAM identity from the second account, and specify which location in your S3 data (a bucket, prefix, or object) that you are granting access to. This location must already be registered with your S3 Access Grants instance. For more information, see Register a location. You can optionally specify a subprefix. For example, if the location you are granting

access to is a bucket, and you want to limit the access further to a specific object in that bucket, then pass the object key name in the `S3SubPrefix` field. Or if you want to limit access to the objects in the bucket with key names that start with a specific prefix, such as `2024-03-research-results/`, then pass `S3SubPrefix=2024-03-research-results/`.

The following is an example CLI command for creating an access grant for an identity in the second account. See Create grants for more information. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

After configuring cross-account access, the user or role in the second account can do the following:

- Calls `ListAccessGrantsInstances` to list the S3 Access Grants instances shared with it through AWS RAM. For more information, see View the details of an S3 Access Grants instance.
- Requests temporary credentials from S3 Access Grants. For more information on how to make these requests, see Request access to Amazon S3 data through S3 Access Grants.

## Using AWS tags with S3 Access Grants

Tags in Amazon S3 Access Grants have similar characteristics to object tags in Amazon S3. Each tag is a key-value pair. The resources in S3 Access Grants that you can tag are S3 Access Grants instances, locations, and grants.

> **Note**
>
> Tagging in S3 Access Grants uses different API operations than object tagging. S3 Access Grants uses the TagResource, UntagResource, and ListTagsForResource API operations, where a resource can be either an S3 Access Grants instance, a registered location, or an access grant.

Similar to object tags, the following limitations apply:

- You can add tags to new S3 Access Grants resources when you create them, or you can add tags to existing resources.
- You can associate up to 10 tags with a resource. If multiple tags are associated with the same resource, they must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Tags are internally represented in UTF-16. In UTF-16, characters consume either 1 or 2 character positions.
- The keys and values are case sensitive.

For more information about tag restrictions, see User-defined tag restrictions in the *AWS Billing User Guide*.

You can tag resources in S3 Access Grants by using the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, or the AWS SDKs.

**Using the AWS CLI**

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can tag an S3 Access Grants resource when you create it or after you have created it. The following examples show how you tag or untag an S3 Access Grants instance. You can perform similar operations for registered locations and access grants.

To use the following example commands, replace the *`user input placeholders`* with your own information.

**Example – Create an S3 Access Grants instance with tags**

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

Response:

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
```

```
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

### Example – Tag an already created S3 Access Grants instance

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

### Example – List tags for the S3 Access Grants instance

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

Response:

```
{
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

### Example – Untag the S3 Access Grants instance

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
```

```
--region us-east-2 \
--tag-keys "tagKey2"
```

#### Using the REST API

You can use the Amazon S3 API to tag, untag, or list tags for an S3 Access Grants instance, registered location, or access grant. For information about the REST API support for managing S3 Access Grants tags, see the following sections in the *Amazon Simple Storage Service API Reference*:

- TagResource
- UntagResource
- ListTagsForResource

## S3 Access Grants limitations

S3 Access Grants has the following limitations:

> **Note**
>
> If your use case exceeds these limitations, contact AWS support to request higher limits.

**S3 Access Grants instance**

You can create **1 S3 Access Grants instance** per AWS Region per account. See Create an S3 Access Grants instance.

**S3 Access Grants location**

You can register **1,000 S3 Access Grants locations** per S3 Access Grants instance. See Register an S3 Access Grants location.

**Grant**

You can create **100,000 grants** per S3 Access Grants instance. See Create a grant.

## S3 Access Grants integrations

S3 Access Grants can be used with the following AWS services and features. This page will be updated as new integrations become available.

**Amazon Athena**

Using IAM Identity Center enabled Athena workgroups

**Amazon EMR**

Launch an Amazon EMR cluster with S3 Access Grants

**Amazon EMR on EKS**

Launch an Amazon EMR on EKS cluster with S3 Access Grants

**Amazon EMR Serverless application**

Launch an Amazon EMR Serverless application with S3 Access Grants

**AWS IAM Identity Center**

Trusted identity propagation across applications

**Amazon SageMaker Studio**

Amazon S3 Access Grants now integrates with Amazon SageMaker Studio

**Open source Python frameworks**

Amazon S3 Access Grants now integrates with open source Python frameworks

## Managing access with ACLs

Access control lists (ACLs) are one of the resource-based options that you can use to manage access to your buckets and objects. You can use ACLs to grant basic read/write permissions to other AWS accounts. There are limits to managing permissions using ACLs.

For example, you can grant permissions only to other AWS accounts; you cannot grant permissions to users in your account. You cannot grant conditional permissions, nor can you explicitly deny permissions. ACLs are suitable for specific scenarios. For example, if a bucket owner allows other AWS accounts to upload objects, permissions to these objects can only be managed using object ACL by the AWS account that owns the object.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

For more information about ACLs, see the following topics.

**Topics**

- Access control list (ACL) overview
- Configuring ACLs
- Policy examples for ACLs

## Access control list (ACL) overview

Amazon S3 access control lists (ACLs) enable you to manage access to buckets and objects. Each bucket and object has an ACL attached to it as a subresource. It defines which AWS accounts or groups are granted access and the type of access. When a request is received against a resource, Amazon S3 checks the corresponding ACL to verify that the requester has the necessary access permissions.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for

each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

When you create a bucket or an object, Amazon S3 creates a default ACL that grants the resource owner full control over the resource. This is shown in the following sample bucket ACL (the default object ACL has the same structure):

**Example**

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The sample ACL includes an `Owner` element that identifies the owner by the AWS account's canonical user ID. For instructions on finding your canonical user ID, see Finding an AWS account canonical user ID. The `Grant` element identifies the grantee (either an AWS account or a

predefined group) and the permission granted. This default ACL has one `Grant` element for the owner. You grant permissions by adding `Grant` elements, with each grant identifying the grantee and the permission.

> **Note**
>
> An ACL can have up to 100 grants.

**Topics**



### Who is a grantee?

A grantee can be an AWS account or one of the predefined Amazon S3 groups. You grant permission to an AWS account using the email address or the canonical user ID. However, if you provide an email address in your grant request, Amazon S3 finds the canonical user ID for that account and adds it to the ACL. The resulting ACLs always contain the canonical user ID for the AWS account, not the email address of the AWS account.

When you grant access rights, you specify each grantee as a *`type`*="*`value`*" pair, where *`type`* is one of the following:

- `id` – If the value specified is the canonical user ID of an AWS account
- `uri` – If you are granting permissions to a predefined group
- `emailAddress` – If the value specified is the email address of an AWS account

> **Important**
>
> Using email addresses to specify a grantee is only supported in the following AWS Regions:
>
> - US East (N. Virginia)

- US West (N. California)
- US West (Oregon)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (Ireland)
- South America (São Paulo)

For a list of all the Amazon S3 supported regions and endpoints, see Regions and Endpoints in the *Amazon Web Services General Reference*.

**Example Example: Email address**

For example, the following `x-amz-grant-read` header grants the AWS accounts identified by email addresses permissions to read object data and its metadata:

```
x-amz-grant-read: emailAddress="xyz@example.com", emailAddress="abc@example.com"
```

**Warning**

When you grant other AWS accounts access to your resources, be aware that the AWS accounts can delegate their permissions to users under their accounts. This is known as *cross-account access*. For information about using cross-account access, see Creating a Role to Delegate Permissions to an IAM User in the *IAM User Guide*.

**Finding an AWS account canonical user ID**

The canonical user ID is associated with your AWS account. This ID is a long string of characters, such as:

`79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be`

For information about how to find the canonical user ID for your account, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.

You can also look up the canonical user ID of an AWS account by reading the ACL of a bucket or an object to which the AWS account has access permissions. When an individual AWS account is granted permissions by a grant request, a grant entry is added to the ACL with the account's canonical user ID.

> **Note**
>
> If you make your bucket public (not recommended), any unauthenticated user can upload objects to the bucket. These anonymous users don't have an AWS account. When an anonymous user uploads an object to your bucket, Amazon S3 adds a special canonical user ID (`65a011a29cdf8ec533ec3d1ccaae921c`) as the object owner in the ACL. For more information, see Amazon S3 bucket and object ownership.

### Amazon S3 predefined groups

Amazon S3 has a set of predefined groups. When granting account access to a group, you specify one of the Amazon S3 URIs instead of a canonical user ID. Amazon S3 provides the following predefined groups:

- **Authenticated Users group** – Represented by `http://acs.amazonaws.com/groups/global/AuthenticatedUsers`.

  This group represents all AWS accounts. **Access permission to this group allows any AWS account to access the resource.** However, all requests must be signed (authenticated).

  > **Warning**
  >
  > When you grant access to the **Authenticated Users group**, any AWS authenticated user in the world can access your resource.

- **All Users group** – Represented by `http://acs.amazonaws.com/groups/global/AllUsers`.

  **Access permission to this group allows anyone in the world access to the resource.** The requests can be signed (authenticated) or unsigned (anonymous). Unsigned requests omit the Authentication header in the request.

> **Warning**
>
> We highly recommend that you never grant the **All Users group** `WRITE`, `WRITE_ACP`, or `FULL_CONTROL` permissions. For example, while `WRITE` permissions do not allow non-owners to overwrite or delete existing objects, `WRITE` permissions still allow anyone to store objects in your bucket, for which you are billed. For more details about these permissions, see the following section What permissions can I grant?.

- **Log Delivery group** – Represented by `http://acs.amazonaws.com/groups/s3/LogDelivery`.

  `WRITE` permission on a bucket enables this group to write server access logs (see Logging requests with server access logging) to the bucket.

> **Note**
>
> When using ACLs, a grantee can be an AWS account or one of the predefined Amazon S3 groups. However, the grantee cannot be an IAM user. For more information about AWS users and permissions within IAM, see Using AWS Identity and Access Management.

### What permissions can I grant?

The following table lists the set of permissions that Amazon S3 supports in an ACL. The set of ACL permissions is the same for an object ACL and a bucket ACL. However, depending on the context (bucket ACL or object ACL), these ACL permissions grant permissions for specific buckets or object operations. The table lists the permissions and describes what they mean in the context of objects and buckets.

For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

### ACL permissions

| Permission | When granted on a bucket | When granted on an object |
|---|---|---|
| `READ` | Allows grantee to list the objects in the bucket | Allows grantee to read the object data and its metadata |

| Permission | When granted on a bucket | When granted on an object |
|---|---|---|
| WRITE | Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects | Not applicable |
| READ_ACP | Allows grantee to read the bucket ACL | Allows grantee to read the object ACL |
| WRITE_ACP | Allows grantee to write the ACL for the applicable bucket | Allows grantee to write the ACL for the applicable object |
| FULL_CONTROL | Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket | Allows grantee the READ, READ_ACP, and WRITE_ACP permissions on the object |

**Warning**

Use caution when granting access permissions to your S3 buckets and objects. For example, granting WRITE access to a bucket allows the grantee to create objects in the bucket. We highly recommend that you read through the entire Access control list (ACL) overview section before granting permissions.

### Mapping of ACL permissions and access policy permissions

As shown in the preceding table, an ACL allows only a finite set of permissions, compared to the number of permissions that you can set in an access policy (see Policy actions for Amazon S3). Each of these permissions allows one or more Amazon S3 operations.

The following table shows how each ACL permission maps to the corresponding access policy permissions. As you can see, access policy allows more permissions than an ACL does. You use ACLs primarily to grant basic read/write permissions, similar to file system permissions. For more information about when to use an ACL, see Identity and Access Management for Amazon S3.

For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

| ACL permission | Corresponding access policy permissions when the ACL permission is granted on a bucket | Corresponding access policy permissions when the ACL permission is granted on an object |
|---|---|---|
| READ | s3:ListBucket , s3:ListBucketVersions , and s3:ListBucketMultipartUploads | s3:GetObject and s3:GetObjectVersion |
| WRITE | s3:PutObject<br><br>Bucket owner can create, overwrite, and delete any object in the bucket, and object owner has FULL_CONTROL over their object.<br><br>In addition, when the grantee is the bucket owner, granting WRITE permission in a bucket ACL allows the s3:DeleteObjectVersion action to be performed on any version in that bucket. | Not applicable |
| READ_ACP | s3:GetBucketAcl | s3:GetObjectAcl and s3:GetObjectVersionAcl |
| WRITE_ACP | s3:PutBucketAcl | s3:PutObjectAcl and s3:PutObjectVersionAcl |
| FULL_CONTROL | Equivalent to granting READ, WRITE, READ_ACP, and WRITE_ACP ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions. | Equivalent to granting READ, READ_ACP, and WRITE_ACP ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions. |

### Condition keys

When you grant access policy permissions, you can use condition keys to constrain the value for the ACL on an object using a bucket policy. The following context keys correspond to ACLs. You can use these context keys to mandate the use of a specific ACL in a request:

- `s3:x-amz-grant-read` - Require read access.
- `s3:x-amz-grant-write` - Require write access.
- `s3:x-amz-grant-read-acp` - Require read access to the bucket ACL.
- `s3:x-amz-grant-write-acp` - Require write access to the bucket ACL.
- `s3:x-amz-grant-full-control` - Require full control.
- `s3:x-amz-acl` - Require a Canned ACL.

For example policies that involve ACL-specific headers, see Granting s3:PutObject permission with a condition requiring the bucket owner to get full control. For a complete list of Amazon S3 specific condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

### `aclRequired` values for common Amazon S3 requests

To identify Amazon S3 requests that required ACLs for authorization, you can use the `aclRequired` value in Amazon S3 server access logs or AWS CloudTrail. The `aclRequired` value that appears in CloudTrail or Amazon S3 server access logs depends on which operations were called and certain information about the requester, object owner, and bucket owner. If no ACLs were required, or if you are setting the `bucket-owner-full-control` canned ACL, or if the requests are allowed by your bucket policy, the `aclRequired` value string is `"-"` in Amazon S3 server access logs and is absent in CloudTrail.

The following tables list the expected `aclRequired` values in CloudTrail or Amazon S3 server access logs for the various Amazon S3 API operations. You can use this information to understand which Amazon S3 operations depend on ACLs for authorization. In the following tables, A, B, and C represent the different accounts associated with the requester, object owner, and bucket owner. Entries with an asterisk (*) indicate any of accounts A, B, or C.

**Note**

`PutObject` operations in the following table, unless specified otherwise, indicate requests that do not set an ACL, unless the ACL is a `bucket-owner-full-control` ACL. A null value for `aclRequired` indicates that `aclRequired` is absent in AWS CloudTrail logs.

### `aclRequired` values for CloudTrail

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | `aclRequired` value | Reason |
|---|---|---|---|---|---|---|
| `GetObject` | A | A | A | Yes or No | null | Same-account access |
| | A | B | A | Yes or No | null | Same-account access with bucket owner enforced |
| | A | A | B | Yes | null | Cross-account access granted by bucket policy |
| | A | A | B | No | Yes | Cross-account access relies on ACL |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | `aclRequired` value | Reason |
|---|---|---|---|---|---|---|
| | A | A | B | Yes | null | Cross-account access granted by bucket policy |
| | A | B | B | No | Yes | Cross-account access relies on ACL |
| | A | B | C | Yes | null | Cross-account access granted by bucket policy |
| | A | B | C | No | Yes | Cross-account access relies on ACL |
| `PutObject` | A | Not applicable | A | Yes or No | null | Same-account access |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | aclRequired value | Reason |
|---|---|---|---|---|---|---|
| | A | Not applicable | B | Yes | null | Cross-account access granted by bucket policy |
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| PutObject with an ACL (except for bucket-owner-full-control ) | * | Not applicable | * | Yes or No | Yes | Request grants ACL |
| ListObjects | A | Not applicable | A | Yes or No | null | Same-account access |
| | A | Not applicable | B | Yes | null | Cross-account access granted by bucket policy |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | aclRequired value | Reason |
|---|---|---|---|---|---|---|
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| DeleteObject | A | Not applicable | A | Yes or No | null | Same-account access |
| | A | Not applicable | B | Yes | null | Cross-account access granted by bucket policy |
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| PutObjectAcl | * | * | * | Yes or No | Yes | Request grants ACL |
| PutBucketAcl | * | Not applicable | * | Yes or No | Yes | Request grants ACL |

**Note**

`REST.PUT.OBJECT` operations in the following table, unless specified otherwise, indicate requests that do not set an ACL, unless the ACL is a `bucket-owner-full-control` ACL. An `aclRequired` value string of "`-`" indicates a null value in Amazon S3 server access logs.

### `aclRequired` values for Amazon S3 server access logs

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | `aclRequired` value | Reason |
|---|---|---|---|---|---|---|
| `REST.GET.OBJECT` | A | A | A | Yes or No | - | Same-account access |
| | A | B | A | Yes or No | - | Same-account access with bucket owner enforced |
| | A | A | B | Yes | - | Cross-account access granted by bucket policy |
| | A | A | B | No | Yes | Cross-account access relies on ACL |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | aclRequired value | Reason |
|---|---|---|---|---|---|---|
| | A | B | B | Yes | - | Cross-account access granted by bucket policy |
| | A | B | B | No | Yes | Cross-account access relies on ACL |
| | A | B | C | Yes | - | Cross-account access granted by bucket policy |
| | A | B | C | No | Yes | Cross-account access relies on ACL |
| REST.PUT.OBJECT | A | Not applicable | A | Yes or No | - | Same-account access |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | aclRequired value | Reason |
|---|---|---|---|---|---|---|
| | A | Not applicable | B | Yes | - | Cross-account access granted by bucket policy |
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| REST.PUT.OBJECT with an ACL (except for bucket-owner-full-control ) | * | Not applicable | * | Yes or No | Yes | Request grants ACL |
| REST.GET.BUCKET | A | Not applicable | A | Yes or No | - | Same-account access |
| | A | Not applicable | B | Yes | - | Cross-account access granted by bucket policy |

| Operation name | Requester | Object owner | Bucket owner | Bucket policy grants access | aclRequired value | Reason |
|---|---|---|---|---|---|---|
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| REST.DELETE.OBJECT | A | Not applicable | A | Yes or No | - | Same-account access |
| | A | Not applicable | B | Yes | - | Cross-account access granted by bucket policy |
| | A | Not applicable | B | No | Yes | Cross-account access relies on ACL |
| REST.PUT.ACL | * | * | * | Yes or No | Yes | Request grants ACL |

### Sample ACL

The following sample ACL on a bucket identifies the resource owner and a set of grants. The format is the XML representation of an ACL in the Amazon S3 REST API. The bucket owner has FULL_CONTROL of the resource. In addition, the ACL shows how permissions are granted on a resource to two AWS accounts, identified by canonical user ID, and two of the predefined Amazon S3 groups discussed in the preceding section.

## Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>Owner-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
```

```
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

  </AccessControlList>
</AccessControlPolicy>
```

### Canned ACL

Amazon S3 supports a set of predefined grants, known as *canned ACLs*. Each canned ACL has a predefined set of grantees and permissions. The following table lists the set of canned ACLs and the associated predefined grants.

| Canned ACL | Applies to | Permissions added to ACL |
| --- | --- | --- |
| `private` | Bucket and object | Owner gets `FULL_CONTROL` . No one else has access rights (default). |
| `public-read` | Bucket and object | Owner gets `FULL_CONTROL` . The `AllUsers` group (see Who is a grantee?) gets `READ` access. |
| `public-read-write` | Bucket and object | Owner gets `FULL_CONTROL` . The `AllUsers` group gets `READ` and `WRITE` access. Granting this on a bucket is generally not recommended. |
| `aws-exec-read` | Bucket and object | Owner gets `FULL_CONTROL` . Amazon EC2 gets `READ` access to `GET` an Amazon Machine Image (AMI) bundle from Amazon S3. |
| `authenticated-read` | Bucket and object | Owner gets `FULL_CONTROL` . The `AuthenticatedUsers` group gets `READ` access. |

| Canned ACL | Applies to | Permissions added to ACL |
| --- | --- | --- |
| `bucket-owner-read` | Object | Object owner gets `FULL_CONTROL` . Bucket owner gets `READ` access. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it. |
| `bucket-owner-full-control` | Object | Both the object owner and the bucket owner get `FULL_CONTROL` over the object. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it. |
| `log-delivery-write` | Bucket | The `LogDelivery` group gets `WRITE` and `READ_ACP` permissions on the bucket. For more information about logs, see (Logging requests with server access logging). |

**Note**

You can specify only one of these canned ACLs in your request.

You specify a canned ACL in your request by using the `x-amz-acl` request header. When Amazon S3 receives a request with a canned ACL in the request, it adds the predefined grants to the ACL of the resource.

## Configuring ACLs

This section explains how to manage access permissions for S3 buckets and objects using access control lists (ACLs). You can add grants to your resource ACL using the AWS Management Console, AWS Command Line Interface (CLI), REST API, or AWS SDKs.

Bucket and object permissions are independent of each other. An object does not inherit the permissions from its bucket. For example, if you create a bucket and grant write access to a user, you can't access that user's objects unless the user explicitly grants you access.

You can grant permissions to other AWS account users or to predefined groups. The user or group that you are granting permissions to is called the *grantee*. By default, the owner, which is the AWS account that created the bucket, has full permissions.

Each permission you grant for a user or group adds an entry in the ACL that is associated with the bucket. The ACL lists grants, which identify the grantee and the permission granted.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**Important**

If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

**Warning**

We highly recommend that you avoid granting write access to the **Everyone (public access)** or **Authenticated Users group (all AWS authenticated users)** groups. For more information about the effects of granting write access to these groups, see Amazon S3 predefined groups.

### Using the S3 console to set ACL permissions for a bucket

The console displays combined access grants for duplicate grantees. To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs.

The following table shows the ACL permissions that you can configure for buckets in the Amazon S3 console.

**Amazon S3 console ACL permissions for buckets**

| Console permission | ACL permission | Access |
|---|---|---|
| Objects - List | `READ` | Allows grantee to list the objects in the bucket. |
| Objects - Write | `WRITE` | Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects. |
| Bucket ACL - Read | `READ_ACP` | Allows grantee to read the bucket ACL. |
| Bucket ACL - Write | `WRITE_ACP` | Allows grantee to write the ACL for the applicable bucket. |
| Everyone (public access): Objects - List | `READ` | Grants public read access for the objects in the bucket. When you grant list access to **Everyone (public access)**, anyone in the world can access the objects in the bucket. |
| Everyone (public access): Bucket ACL - Read | `READ_ACP` | Grants public read access for the bucket ACL. When you grant read access to **Everyone (public access)**, anyone in the world can access the bucket ACL. |

For more information about ACL permissions, see Access control list (ACL) overview.

**Important**

If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

**To set ACL permissions for a bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to set permissions for.
3. Choose **Permissions**.
4. Under **Access control list**, choose **Edit**.

   You can edit the following ACL permissions for the bucket:

   **Objects**

   - **List** – Allows a grantee to list the objects in the bucket.
   - **Write** – Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects.

     In the S3 console, you can only grant write access to the S3 log delivery group and the bucket owner (your AWS account). We highly recommend that you do not grant write access for other grantees. However, if you need to grant write access, you can use the AWS CLI, AWS SDKs, or the REST API.

   **Bucket ACL**

   - **Read** – Allows grantee to read the bucket ACL.
   - **Write** – Allows grantee to write the ACL for the applicable bucket.
5. To change the bucket owner's permissions, beside **Bucket owner (your AWS account)**, clear or select from the following ACL permissions:

- **Objects** – **List** or **Write**
- **Bucket ACL** – **Read** or **Write**

The *owner* refers to the AWS account root user, not an AWS Identity and Access Management IAM user. For more information about the root user, see The AWS account root user in the *IAM User Guide*.

6. To grant or undo permissions for the general public (everyone on the internet), beside **Everyone (public access)**, clear or select from the following ACL permissions:

   - **Objects** – **List**
   - **Bucket ACL** – **Read**

   > **Warning**
   >
   > Use caution when granting the **Everyone** group public access to your S3 bucket. When you grant access to this group, anyone in the world can access your bucket. We highly recommend that you never grant any kind of public write access to your S3 bucket.

7. To grant or undo permissions for anyone with an AWS account, beside **Authenticated Users group (anyone with an AWS account)**, clear or select from the following ACL permissions:

   - **Objects** – **List**
   - **Bucket ACL** – **Read**

8. To grant or undo permissions for Amazon S3 to write server access logs to the bucket, under **S3 log delivery group**, clear or select from the following ACL permissions:

   - **Objects** – **List** or **Write**
   - **Bucket ACL** – **Read** or **Write**

   If a bucket is set up as the target bucket to receive access logs, the bucket permissions must allow the **Log Delivery** group write access to the bucket. When you enable server access logging on a bucket, the Amazon S3 console grants write access to the **Log Delivery** group for the target bucket that you choose to receive the logs. For more information about server access logging, see Enabling Amazon S3 server access logging.

9. To grant access to another AWS account, do the following:

a. Choose **Add grantee**.

b. In the **Grantee** box, enter the canonical ID of the other AWS account.

c. Select from the following ACL permissions:

- **Objects** – **List** or **Write**
- **Bucket ACL** – **Read** or **Write**

> **Warning**
>
> When you grant other AWS accounts access to your resources, be aware that the AWS accounts can delegate their permissions to users under their accounts. This is known as *cross-account access*. For information about using cross-account access, see Creating a Role to Delegate Permissions to an IAM User in the *IAM User Guide*.

10. To remove access to another AWS account, under **Access for other AWS accounts**, choose **Remove**.
11. To save your changes, choose **Save changes**.

### Using the S3 console to set ACL permissions for an object

The console displays combined access grants for duplicate grantees. To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs. The following table shows the ACL permissions that you can configure for objects in the Amazon S3 console.

### Amazon S3 console ACL permissions for objects

| Console permission | ACL permission | Access |
|---|---|---|
| Object - Read | `READ` | Allows grantee to read the object data and its metadata. |
| Object ACL - Read | `READ_ACP` | Allows grantee to read the object ACL. |
| Object ACL - Write | `WRITE_ACP` | Allows grantee to write the ACL for the applicable object |

For more information about ACL permissions, see Access control list (ACL) overview.

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

**To set ACL permissions for an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **objects** list, choose the name of the object for which you want to set permissions.
4. Choose **Permissions**.
5. Under Access control list (ACL), choose **Edit**.

   You can edit the following ACL permissions for the object:

   **Object**

   - **Read** – Allows grantee to read the object data and its metadata.

   **Object ACL**

   - **Read** – Allows grantee to read the object ACL.
   - **Write** – Allows grantee to write the ACL for the applicable object. In the S3 console, you can only grant write access to the bucket owner (your AWS account). We highly recommend that you do not grant write access for other grantees. However, if you need to grant write access, you can use the AWS CLI, AWS SDKs, or the REST API.
6. You can manage object access permissions for the following:

a. **Access for object owner**

The *owner* refers to the AWS account root user, and not an AWS Identity and Access Management IAM user. For more information about the root user, see The AWS account root user in the *IAM User Guide*.

To change the owner's object access permissions, under **Access for object owner**, choose **Your AWS Account (owner)**.

Select the check boxes for the permissions that you want to change, and then choose **Save**.

b. **Access for other AWS accounts**

To grant permissions to an AWS user from a different AWS account, under **Access for other AWS accounts**, choose **Add account**. In the **Enter an ID** field, enter the canonical ID of the AWS user that you want to grant object permissions to. For information about finding a canonical ID, see Your AWS account identifiers in the *Amazon Web Services General Reference*. You can add as many as 99 users.

Select the check boxes for the permissions that you want to grant to the user, and then choose **Save**. To display information about the permissions, choose the Help icons.

c. **Public access**

To grant access to your object to the general public (everyone in the world), under **Public access**, choose **Everyone**. Granting public access permissions means that anyone in the world can access the object.

Select the check boxes for the permissions that you want to grant, and then choose **Save**.

> **Warning**
>
> - Use caution when granting the **Everyone** group anonymous access to your Amazon S3 objects. When you grant access to this group, anyone in the world can access your object. If you need to grant access to everyone, we highly recommend that you only grant permissions to **Read objects**.

- We highly recommend that you *do not* grant the **Everyone** group write object permissions. Doing so allows anyone to overwrite the ACL permissions for the object.

### Using the AWS SDKs

This section provides examples of how to configure access control list (ACL) grants on buckets and objects.

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

Java

This section provides examples of how to configure access control list (ACL) grants on buckets and objects. The first example creates a bucket with a canned ACL (see Canned ACL), creates a list of custom permission grants, and then replaces the canned ACL with an ACL containing the custom grants. The second example shows how to modify an ACL using the `AccessControlList.grantPermission()` method.

**Example Create a bucket and specify a canned ACL that grants permission to the S3 log delivery group**

This example creates a bucket. In the request, the example specifies a canned ACL that grants the Log Delivery group permission to write logs to the bucket.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;
```

```
import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Create a bucket with a canned ACL. This ACL will be replaced by the
            // setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new
 CreateBucketRequest(bucketName, clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
 CanonicalGrantee(s3Client.getS3AccountOwner().getId()),
                    Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
 created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);
```

```
            // Retrieve the bucket's ACL, add another grant, and then save the new
 ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new
 EmailAddressGrantee(userEmailForReadPermission), Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Example Update ACL on an existing object

This example updates the ACL on an object. The example performs the following tasks:

- Retrieves an object's ACL
- Clears the ACL by removing all existing permissions
- Adds two permissions: full access to the owner, and WRITE_ACP (see What permissions can I grant?) to a user identified by an email address
- Saves the ACL to the object

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;
```

```
import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for
 Full
            // Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

**Example Create a bucket and specify a canned ACL that grants permission to the S3 log delivery group**

This C# example creates a bucket. In the request, the code also specifies a canned ACL that grants the Log Delivery group permissions to write the logs to the bucket.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
```

```
                                                    // Add canned ACL.
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
                {
                    BucketName = newBucketName
                });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

**Example Update ACL on an existing object**

This C# example updates the ACL on an existing object. The example performs the following tasks:

- Retrieves an object's ACL.
- Clears the ACL by removing all existing permissions.
- Adds two permissions: full access to the owner, and WRITE_ACP to a user identified by email address.
- Saves the ACL by sending a `PutAcl` request.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after
 we clear the ACL).
                    Owner owner = acl.Owner;

                    // Clear existing grants.
                    acl.Grants.Clear();
```

```
                    // Add a grant to reset the owner's full permission (the
 previous clear statement removed all permissions).
                    S3Grant fullControlGrant = new S3Grant
                    {
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
                        Permission = S3Permission.WRITE_ACP
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
 grantUsingEmail });

                    // Set a new ACL.
                    PutACLResponse response = await client.PutACLAsync(new
 PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

### Using the REST API

Amazon S3 APIs enable you to set an ACL when you create a bucket or an object. Amazon S3 also provides API to set an ACL on an existing bucket or an object. These APIs provide the following methods to set an ACL:

- **Set ACL using request headers—** When you send a request to create a resource (bucket or object), you set an ACL using the request headers. Using these headers, you can either specify a canned ACL or specify grants explicitly (identifying grantee and permissions explicitly).
- **Set ACL using request body—** When you send a request to set an ACL on an existing resource, you can set the ACL either in the request header or in the body.

For information on the REST API support for managing ACLs, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GET Bucket acl
- PUT Bucket acl
- GET Object acl
- PUT Object acl
- PUT Object
- PUT Bucket
- PUT Object - Copy
- Initiate Multipart Upload

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

### Access Control List (ACL)-Specific Request Headers

You can use headers to grant access control list (ACL)-based permissions. By default, all objects are private. Only the owner has full access control. When adding a new object, you can grant permissions to individual AWS accounts or to predefined groups defined by Amazon S3. These permissions are then added to the Access Control List (ACL) on the object. For more information, see Access control list (ACL) overview.

With this operation, you can grant access permissions using one these two methods:

- **Canned ACL (`x-amz-acl`)** — Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, see Canned ACL.
- **Access Permissions** — To explicitly grant access permissions to specific AWS accounts or groups, use the following headers. Each header maps to specific permissions that Amazon S3 supports in an ACL. For more information, see Access control list (ACL) overview. In the header, you specify a list of grantees who get the specific permission.
  - x-amz-grant-read
  - x-amz-grant-write
  - x-amz-grant-read-acp
  - x-amz-grant-write-acp
  - x-amz-grant-full-control

### Using the AWS CLI

For more information about managing ACLs using the AWS CLI, see put-bucket-acl in the *AWS CLI Command Reference*.

> **Important**
>
> If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

## Policy examples for ACLs

You can use condition keys in bucket policies to control access to Amazon S3.

### Topics



### Granting s3:PutObject permission with a condition requiring the bucket owner to get full control

The PUT Object operation allows access control list (ACL)–specific headers that you can use to grant ACL-based permissions. Using these keys, the bucket owner can set a condition to require specific access permissions when the user uploads an object.

Suppose that Account A owns a bucket, and the account administrator wants to grant Dave, a user in Account B, permissions to upload objects. By default, objects that Dave uploads are owned by Account B, and Account A has no permissions on these objects. Because the bucket owner is paying the bills, it wants full permissions on the objects that Dave uploads. The Account A administrator can do this by granting the `s3:PutObject` permission to Dave, with a condition that the request include ACL-specific headers that either grant full permission explicitly or use a canned ACL. For more information, see PUT Object.

#### Require the x-amz-full-control header

You can require the `x-amz-full-control` header in the request with full control permission to the bucket owner. The following bucket policy grants the `s3:PutObject` permission to user Dave with a condition using the `s3:x-amz-grant-full-control` condition key, which requires the request to include the `x-amz-full-control` header.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
```

```
      "Resource": "arn:aws:s3:::awsexamplebucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
      }
    }
  ]
}
```

> **Note**
>
> This example is about cross-account permission. However, if Dave (who is getting the permission) belongs to the AWS account that owns the bucket, this conditional permission is not necessary. This is because the parent account to which Dave belongs owns objects that the user uploads.

### Add explicit deny

The preceding bucket policy grants conditional permission to user Dave in Account B. While this policy is in effect, it is possible for Dave to get the same permission without any condition via some other policy. For example, Dave can belong to a group, and you grant the group `s3:PutObject` permission without any condition. To avoid such permission loopholes, you can write a stricter access policy by adding explicit deny. In this example, you explicitly deny the user Dave upload permission if he does not include the necessary headers in the request granting full permissions to the bucket owner. Explicit deny always supersedes any other permission granted. The following is the revised access policy example with explicit deny added.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
```

```
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

### Test the policy with the AWS CLI

If you have two AWS accounts, you can test the policy using the AWS Command Line Interface (AWS CLI). You attach the policy and use Dave's credentials to test the permission using the following AWS CLI `put-object` command. You provide Dave's credentials by adding the `--profile` parameter. You grant full control permission to the bucket owner by adding the `--grant-full-control` parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

### Require the x-amz-acl header

You can require the `x-amz-acl` header with a canned ACL granting full control permission to the bucket owner. To require the `x-amz-acl` header in the request, you can replace the key-value pair in the `Condition` block and specify the `s3:x-amz-acl` condition key, as shown in the following example.

```
"Condition": {
```

```
    "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
    }
}
```

To test the permission using the AWS CLI, you specify the `--acl` parameter. The AWS CLI then adds the `x-amz-acl` header when it sends the request.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --acl "bucket-owner-full-control" --profile AccountBadmin
```

**Granting s3:PutObject permission with a condition on the x-amz-acl header**

The following bucket policy grants the `s3:PutObject` permission for two AWS accounts if the request includes the `x-amz-acl` header making the object publicly readable. The `Condition` block uses the `StringEquals` condition, and it is provided a key-value pair, `"s3:x-amz-acl": ["public-read"]`, for evaluation. In the key-value pair, the `s3:x-amz-acl` is an Amazon S3–specific key, as indicated by the prefix `s3:`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
         "Sid":"AddCannedAcl",
            "Effect":"Allow",
         "Principal": {
                "AWS": [
                "arn:aws:iam::Account1-ID:root",
    "arn:aws:iam::Account2-ID:root"
    ]
            },
         "Action":"s3:PutObject",
            "Resource": ["arn:aws:s3:::awsexamplebucket1/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

**Important**

Not all conditions make sense for all actions. For example, it makes sense to include an `s3:LocationConstraint` condition on a policy that grants the `s3:CreateBucket` Amazon S3 permission. However, it does not make sense to include this condition on a policy that grants the `s3:GetObject` permission. Amazon S3 can test for semantic errors of this type that involve Amazon S3–specific conditions. However, if you are creating a policy for an IAM user or role and you include a semantically invalid Amazon S3 condition, no error is reported because IAM cannot validate Amazon S3 conditions.

## Blocking public access to your Amazon S3 storage

The Amazon S3 Block Public Access feature provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects don't allow public access. However, users can modify bucket policies, access point policies, or object permissions to allow public access. S3 Block Public Access settings override these policies and permissions so that you can limit public access to these resources.

With S3 Block Public Access, account administrators and bucket owners can easily set up centralized controls to limit public access to their Amazon S3 resources that are enforced regardless of how the resources are created.

For instructions on configuring public block access, see Configuring block public access.

When Amazon S3 receives a request to access a bucket or an object, it determines whether the bucket or the bucket owner's account has a block public access setting applied. If the request was made through an access point, Amazon S3 also checks for block public access settings for the access point. If there is an existing block public access setting that prohibits the requested access, Amazon S3 rejects the request.

Amazon S3 Block Public Access provides four settings. These settings are independent and can be used in any combination. Each setting can be applied to an access point, a bucket, or an entire AWS account. If the block public access settings for the access point, bucket, or account differ, then Amazon S3 applies the most restrictive combination of the access point, bucket, and account settings.

When Amazon S3 evaluates whether an operation is prohibited by a block public access setting, it rejects any request that violates an access point, bucket, or account setting.

**Important**

Public access is granted to buckets and objects through access control lists (ACLs), access point policies, bucket policies, or all. To help ensure that all of your Amazon S3 access points, buckets, and objects have their public access blocked, we recommend that you turn on all four settings for block public access for your account. These settings block public access for all current and future buckets and access points.
Before applying these settings, verify that your applications will work correctly without public access. If you require some level of public access to your buckets or objects, for example to host a static website as described at Hosting a static website using Amazon S3, you can customize the individual settings to suit your storage use cases.
Enabling Block Public Access helps protect your resources by preventing public access from being granted through the resource policies or access control lists (ACLs) that are directly attached to S3 resources. In addition to enabling Block Public Access, carefully inspect the following policies to confirm that they do not grant public access:

- Identity-based policies attached to associated AWS principals (for example, IAM roles)
- Resource-based policies attached to associated AWS resources (for example, AWS Key Management Service (KMS) keys)

**Note**

- You can enable block public access settings only for access points, buckets, and AWS accounts. Amazon S3 doesn't support block public access settings on a per-object basis.
- When you apply block public access settings to an account, the settings apply to all AWS Regions globally. The settings might not take effect in all Regions immediately or simultaneously, but they eventually propagate to all Regions.

**Topics**

- Permissions
- Configuring block public access
- Configuring block public access settings for your account
- Configuring block public access settings for your S3 buckets

## Block public access settings

S3 Block Public Access provides four settings. You can apply these settings in any combination to individual access points, buckets, or entire AWS accounts. If you apply a setting to an account, it applies to all buckets and access points that are owned by that account. Similarly, if you apply a setting to a bucket, it applies to all access points associated with that bucket.

The following table contains the available settings.

| Name | Description |
| --- | --- |
| `BlockPublicAcls` | Setting this option to TRUE causes the following behavior:<br>• PUT Bucket acl and PUT Object acl calls fail if the specified access control list (ACL) is public.<br>• PUT Object calls fail if the request includes a public ACL.<br>• If this setting is applied to an account, then PUT Bucket calls fail if the request includes a public ACL.<br><br>When this setting is set to TRUE, the specified operations fail (whether made through the REST API, AWS CLI, or AWS SDKs). However, existing policies and ACLs for buckets and objects are not modified. This setting enables you to protect against public access while allowing you to audit, refine, or otherwise alter the existing policies and ACLs for your buckets and objects. |

| Name | Description |
| --- | --- |
| | **Note**<br>Access points don't have ACLs associated with them. If you apply this setting to an access point, it acts as a passthrough to the underlying bucket. If an access point has this setting enabled, requests made through the access point behave as though the underlying bucket has this setting enabled, regardless of whether the bucket actually has this setting enabled. |
| `IgnorePublicAcls` | Setting this option to `TRUE` causes Amazon S3 to ignore all public ACLs on a bucket and any objects that it contains. This setting enables you to safely block public access granted by ACLs while still allowing PUT Object calls that include a public ACL (as opposed to `BlockPublicAcls`, which rejects PUT Object calls that include a public ACL). Enabling this setting doesn't affect the persistence of any existing ACLs and doesn't prevent new public ACLs from being set.<br>**Note**<br>Access points don't have ACLs associated with them. If you apply this setting to an access point, it acts as a passthrough to the underlying bucket. If an access point has this setting enabled, requests made through the access point behave as though the underlying bucket has this setting enabled, regardless of whether the bucket actually has this setting enabled. |

| Name | Description |
| --- | --- |
| `BlockPublicPolicy` | Setting this option to TRUE for a bucket causes Amazon S3 to reject calls to PUT Bucket policy if the specified bucket policy allows public access. Setting this option to TRUE for a bucket also causes Amazon S3 to reject calls to PUT access point policy for all of the bucket's same-account access points if the specified policy allows public access.<br><br>Setting this option to TRUE for an access point causes Amazon S3 to reject calls to PUT access point policy and PUT Bucket policy that are made through the access point if the specified policy (for either the access point or the underlying bucket) allows public access.<br><br>You can use this setting to allow users to manage access point and bucket policies without allowing them to publicly share the bucket or the object s it contains. Enabling this setting doesn't affect existing access point or bucket policies.<br><br>**Important**<br><br>To use this setting effectively, we recommend that you apply it at the *account* level. A bucket policy can allow users to alter a bucket's block public access settings. Therefore, users who have permission to change a bucket policy could insert a policy that allows them to disable the block public access settings for the bucket. If this setting is enabled for the entire account, rather than for a specific bucket, Amazon S3 blocks public policies even if a user alters the bucket policy to disable this setting. |

| Name | Description |
| --- | --- |
| `RestrictPublicBuckets` | Setting this option to TRUE restricts access to an access point or bucket with a public policy to only AWS service principals and authorized users within the bucket owner's account and access point owner's account. This setting blocks all cross-account access to the access point or bucket (except by AWS service principals), while still allowing users within the account to manage the access point or bucket.<br><br>Enabling this setting doesn't affect existing access point or bucket policies, except that Amazon S3 blocks public and cross-account access derived from any public access point or bucket policy, including non-public delegation to specific accounts. |

**Important**

- Calls to GET Bucket acl and GET Object acl always return the effective permissions in place for the specified bucket or object. For example, suppose that a bucket has an ACL that grants public access, but the bucket also has the `IgnorePublicAcls` setting enabled. In this case, GET Bucket acl returns an ACL that reflects the access permissions that Amazon S3 is enforcing, rather than the actual ACL that is associated with the bucket.
- Block public access settings don't alter existing policies or ACLs. Therefore, removing a block public access setting causes a bucket or object with a public policy or ACL to again be publicly accessible.

## Performing block public access operations on an access point

To perform block public access operations on an access point, use the AWS CLI service `s3control`.

**Important**

Note that it isn't currently possible to change an access point's block public access settings after creating the access point. Thus, the only way to specify block public access settings for an access point is by including them when creating the access point.

## The meaning of "public"

### ACLs

Amazon S3 considers a bucket or object ACL public if it grants any permissions to members of the predefined `AllUsers` or `AuthenticatedUsers` groups. For more information about predefined groups, see Amazon S3 predefined groups.

### Bucket policies

When evaluating a bucket policy, Amazon S3 begins by assuming that the policy is public. It then evaluates the policy to determine whether it qualifies as non-public. To be considered non-public, a bucket policy must grant access only to fixed values (values that don't contain a wildcard or an AWS Identity and Access Management Policy Variable) for one or more of the following:

- An AWS principal, user, role, or service principal (e.g. `aws:PrincipalOrgID`)
- A set of Classless Inter-Domain Routings (CIDRs), using `aws:SourceIp`. For more information about CIDR, see RFC 4632 on the RFC Editor website.

> **Note**
>
> Bucket policies that grant access conditioned on the `aws:SourceIp` condition key with very broad IP ranges (for example, 0.0.0.0/1) are evaluated as "public." This includes values broader than /8 for IPv4 and /32 for IPv6 (excluding RFC1918 private ranges). Block public access will reject these "public" policies and prevent cross-account access to buckets already using these "public" policies.

- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `s3:x-amz-server-side-encryption-aws-kms-key-id`
- `aws:userid`, outside the pattern "`AROLEID:*`"
- `s3:DataAccessPointArn`

> **Note**
>
> When used in a bucket policy, this value can contain a wildcard for the access point name without rendering the policy public, as long as the account id is fixed. For example, allowing access to `arn:aws:s3:us-west-2:123456789012:accesspoint/*` would permit access to any access point associated with account `123456789012` in Region `us-west-2`, without rendering the bucket policy public. Note that this behavior is different for access point policies. For more information, see Access points.

- `s3:DataAccessPointAccount`

For more information about bucket policies, see Bucket policies for Amazon S3.

**Example : Public bucket policies**

Under these rules, the following example policies are considered public.

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

You can make these policies non-public by including any of the condition keys listed previously, using a fixed value. For example, you can make the last policy preceding non-public by setting `aws:SourceVpc` to a fixed value, like the following.

```
{
  "Principal": "*",
```

```
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

### How Amazon S3 evaluates a bucket policy that contains both public and non-public access grants

This example shows how Amazon S3 evaluates a bucket policy that contains both public and non-public access grants.

Suppose that a bucket has a policy that grants access to a set of fixed principals.
Under the previously described rules, this policy isn't public. Thus, if you enable the `RestrictPublicBuckets` setting, the policy remains in effect as written, because `RestrictPublicBuckets` only applies to buckets that have public policies. However, if you add a public statement to the policy, `RestrictPublicBuckets` takes effect on the bucket. It allows only AWS service principals and authorized users of the bucket owner's account to access the bucket.

As an example, suppose that a bucket owned by "Account-1" has a policy that contains the following:

1. A statement that grants access to AWS CloudTrail (which is an AWS service principal)
2. A statement that grants access to account "Account-2"
3. A statement that grants access to the public, for example by specifying `"Principal": "*"` with no limiting `Condition`

This policy qualifies as public because of the third statement. With this policy in place and `RestrictPublicBuckets` enabled, Amazon S3 allows access only by CloudTrail. Even though statement 2 isn't public, Amazon S3 disables access by "Account-2." This is because statement 3 renders the entire policy public, so `RestrictPublicBuckets` applies. As a result, Amazon S3 disables cross-account access, even though the policy delegates access to a specific account, "Account-2." But if you remove statement 3 from the policy, then the policy doesn't qualify as public, and `RestrictPublicBuckets` no longer applies. Thus, "Account-2" regains access to the bucket, even if you leave `RestrictPublicBuckets` enabled.

### Access points

Amazon S3 evaluates block public access settings slightly differently for access points compared to buckets. The rules that Amazon S3 applies to determine when an access point policy is public are generally the same for access points as for buckets, except in the following situations:

- An access point that has a VPC network origin is always considered non-public, regardless of the contents of its access point policy.
- An access point policy that grants access to a set of access points using `s3:DataAccessPointArn` is considered public. Note that this behavior is different than for bucket policies. For example, a bucket policy that grants access to values of `s3:DataAccessPointArn` that match `arn:aws:s3:us-west-2:123456789012:accesspoint/*` is not considered public. However, the same statement in an access point policy would render the access point public.

## Using IAM Access Analyzer for S3 to review public buckets

You can use IAM Access Analyzer for S3 to review buckets with bucket ACLs, bucket policies, or access point policies that grant public access. IAM Access Analyzer for S3 alerts you to buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings that report the source and level of public or shared access.

In IAM Access Analyzer for S3, you can block all public access to a bucket with a single click. You can also drill down into bucket-level permission settings to configure granular levels of access. For specific and verified use cases that require public or shared access, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket.

In rare events, IAM Access Analyzer for S3 might report no findings for a bucket that an Amazon S3 block public access evaluation reports as public. This happens because Amazon S3 block public access reviews policies for current actions and any potential actions that might be added in the future, leading to a bucket becoming public. On the other hand, IAM Access Analyzer for S3 only analyzes the current actions specified for the Amazon S3 service in the evaluation of access status.

For more information about IAM Access Analyzer for S3, see Reviewing bucket access using IAM Access Analyzer for S3.

## Permissions

To use Amazon S3 Block Public Access features, you must have the following permissions.

| Operation | Required permissions |
| --- | --- |
| GET bucket policy status | `s3:GetBucketPolicyStatus` |
| GET bucket Block Public Access settings | `s3:GetBucketPublicAccessBlock` |
| PUT bucket Block Public Access settings | `s3:PutBucketPublicAccessBlock` |
| DELETE bucket Block Public Access settings | `s3:PutBucketPublicAccessBlock` |
| GET account Block Public Access settings | `s3:GetAccountPublicAccessBlock` |
| PUT account Block Public Access settings | `s3:PutAccountPublicAccessBlock` |
| DELETE account Block Public Access settings | `s3:PutAccountPublicAccessBlock` |
| PUT access point Block Public Access settings | `s3:CreateAccessPoint` |

**Note**

The DELETE operations require the same permissions as the PUT operations. There are no separate permissions for the DELETE operations.

## Configuring block public access

For more information about configuring block public access for your AWS account and your Amazon S3 buckets, see the following topics.

- Configuring block public access settings for your account
- Configuring block public access settings for your S3 buckets

## Configuring block public access settings for your account

Amazon S3 Block Public Access provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects do not allow public access.

For more information, see Blocking public access to your Amazon S3 storage.

> **Note**
>
> Account level settings override settings on individual objects. Configuring your account to block public access will override any public access settings made to individual objects within your account.

You can use the S3 console, AWS CLI, AWS SDKs, and REST API to configure block public access settings for all the buckets in your account. For more information, see the sections below.

To configure block public access settings for your buckets, see Configuring block public access settings for your S3 buckets. For information about access points, see Performing block public access operations on an access point.

### Using the S3 console

Amazon S3 block public access prevents the application of any settings that allow public access to data within S3 buckets. This section describes how to edit block public access settings for all the S3 buckets in your AWS account. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

**To edit block public access settings for all the S3 buckets in an AWS account**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Block Public Access settings for this account**.
3. Choose **Edit** to change the block public access settings for all the buckets in your AWS account.
4. Choose the settings that you want to change, and then choose **Save changes**.
5. When you're asked for confirmation, enter **`confirm`**. Then choose **Confirm** to save your changes.

### Using the AWS CLI

You can use Amazon S3 Block Public Access through the AWS CLI. For more information about setting up and using the AWS CLI, see What is the AWS Command Line Interface?

#### Account

To perform block public access operations on an account, use the AWS CLI service `s3control`. The account-level operations that use this service are as follows:

- PUT PublicAccessBlock (for an account)
- GET PublicAccessBlock (for an account)
- DELETE PublicAccessBlock (for an account)

For additional information and examples, see put-public-access-block in the *AWS CLI Reference*.

### Using the AWS SDKs

Java

The following examples show you how to use Amazon S3 Block Public Access with the AWS SDK for Java to put a public access block configuration on an Amazon S3 account.

```
AWSS3ControlClientBuilder controlClientBuilder =
 AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> This example pertains only to account-level operations, which use the `AWSS3Control` client class. For bucket-level operations, see the preceding example.

Other SDKs

For information about using the other AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

#### Using the REST API

For information about using Amazon S3 Block Public Access through the REST APIs, see the following topics in the *Amazon Simple Storage Service API Reference*.

- Account-level operations
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock

## Configuring block public access settings for your S3 buckets

Amazon S3 Block Public Access provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects do not allow public access.

For more information, see Blocking public access to your Amazon S3 storage.

You can use the S3 console, AWS CLI, AWS SDKs, and REST API to grant public access to one or more buckets. You can also block public access to buckets that are already public. For more information, see the sections below.

To configure block public access settings for every bucket in your account, see Configuring block public access settings for your account. For information about configuring block public access for access points, see Performing block public access operations on an access point.

### Using the S3 console

Amazon S3 Block Public Access prevents the application of any settings that allow public access to data within S3 buckets. This section describes how to edit Block Public Access settings for one or more S3 buckets. For information about blocking public access using the AWS CLI, AWS SDKs, and the Amazon S3 REST APIs, see Blocking public access to your Amazon S3 storage.

You can see if your bucket is publicly accessible in the **Buckets** list. In the **Access** column, Amazon S3 labels the permissions for a bucket as follows:

- **Public** – Everyone has access to one or more of the following: List objects, Write objects, Read and write permissions.
- **Objects can be public** – The bucket is not public, but anyone with the appropriate permissions can grant public access to objects.
- **Buckets and objects not public** – The bucket and objects do not have any public access.
- **Only authorized users of this account** – Access is isolated to IAM users and roles in this account and AWS service principals because there is a policy that grants public access.

You can also filter bucket searches by access type. Choose an access type from the drop-down list that is next to the **Search for buckets** bar.

If you see an `Error` when you list your buckets and their public access settings, you might not have the required permissions. Check to make sure you have the following permissions added to your user or role policy:

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

In some rare cases, requests can also fail because of an AWS Region outage.

#### To edit the Amazon S3 block public access settings for a single S3 bucket

Follow these steps if you need to change the public access settings for a single S3 bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want.
3. Choose **Permissions**.
4. Choose **Edit** to change the public access settings for the bucket. For more information about the four Amazon S3 Block Public Access Settings, see Block public access settings.
5. Choose the setting that you want to change, and then choose **Save**.
6. When you're asked for confirmation, enter **`confirm`**. Then choose **Confirm** to save your changes.

You can change Amazon S3 Block Public Access settings when you create a bucket. For more information, see Creating a bucket.

**Using the AWS CLI**

To block public access on a bucket or to delete the public access block, use the AWS CLI service `s3api`. The bucket-level operations that use this service are as follows:

- PUT PublicAccessBlock (for a bucket)
- GET PublicAccessBlock (for a bucket)
- DELETE PublicAccessBlock (for a bucket)
- GET BucketPolicyStatus

For more information and examples, see put-public-access-block in the *AWS CLI Reference*.

**Using the AWS SDKs**

Java

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
```

```
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> This example pertains only to bucket-level operations, which use the `AmazonS3` client class. For account-level operations, see the following example.

Other SDKs

For information about using the other AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

#### Using the REST API

For information about using Amazon S3 Block Public Access through the REST APIs, see the following topics in the *Amazon Simple Storage Service API Reference*.

- Bucket-level operations
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock
  - GET BucketPolicyStatus

## Reviewing bucket access using IAM Access Analyzer for S3

IAM Access Analyzer for S3 alerts you to S3 buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings into the source and level of public or shared access. For example, IAM Access Analyzer for S3 might show that a bucket has read or write access provided through a bucket access control list (ACL), a bucket policy, a Multi-Region Access Point policy, or an access point policy. With these findings, you can take immediate and precise corrective action to restore your bucket access to what you intended.

When reviewing an at-risk bucket in IAM Access Analyzer for S3, you can block all public access to the bucket with a single click. We recommend that you block all access to your buckets unless you require public access to support a specific use case. Before you block all public access, ensure that your applications will continue to work correctly without public access. For more information, see Blocking public access to your Amazon S3 storage.

You can also drill down into bucket-level permission settings to configure granular levels of access. For specific and verified use cases that require public access, such as static website hosting, public downloads, or cross-account sharing, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket. You can revisit and modify these bucket configurations at any time. You can also download your findings as a CSV report for auditing purposes.

IAM Access Analyzer for S3 is available at no extra cost on the Amazon S3 console. IAM Access Analyzer for S3 is powered by AWS Identity and Access Management (IAM) IAM Access Analyzer. To use IAM Access Analyzer for S3 in the Amazon S3 console, you must visit the IAM console and enable IAM Access Analyzer on a per-Region basis.

For more information about IAM Access Analyzer, see What is IAM Access Analyzer? in the *IAM User Guide*. For more information about IAM Access Analyzer for S3, review the following sections.

**⚠ Important**

- IAM Access Analyzer for S3 requires an account-level analyzer. To use IAM Access Analyzer for S3, you must visit IAM Access Analyzer and create an analyzer that has an account as the zone of trust. For more information, see Enabling IAM Access Analyzer in *IAM User Guide*.
- IAM Access Analyzer for S3 doesn't analyze the access point policy that's attached to cross-account access points. This behavior occurs because the access point and its policy are outside the zone of trust, that is, the account. Buckets that delegate access to a cross-account access point are listed under **Buckets with public access** if you haven't applied the `RestrictPublicBuckets` block public access setting to the bucket or account. When you apply the `RestrictPublicBuckets` block public access setting, the bucket is reported under **Buckets with access from other AWS accounts — including third-party AWS accounts**.
- When a bucket policy or bucket ACL is added or modified, IAM Access Analyzer generates and updates findings based on the change within 30 minutes. Findings related to account

level block public access settings might not be generated or updated for up to 6 hours after you change the settings. Findings related to Multi-Region Access Points might not be generated or updated for up to six hours after the Multi-Region Access Point is created, deleted, or you change its policy.

**Topics**



## What information does IAM Access Analyzer for S3 provide?

IAM Access Analyzer for S3 provides findings for buckets that can be accessed outside your AWS account. Buckets that are listed under **Buckets with public access** can be accessed by anyone on the internet. If IAM Access Analyzer for S3 identifies public buckets, you also see a warning at the top of the page that shows you the number of public buckets in your Region. Buckets listed under **Buckets with access from other AWS accounts — including third-party AWS accounts** are shared conditionally with other AWS accounts, including accounts outside of your organization.

For each bucket, IAM Access Analyzer for S3 provides the following information:

- **Bucket name**
- **Discovered by Access analyzer** - When IAM Access Analyzer for S3 discovered the public or shared bucket access.
- **Shared through** - How the bucket is shared—through a bucket policy, a bucket ACL, a Multi-Region Access Point policy, or an access point policy. Multi-Region Access Points and cross-account access points are reflected under access points. A bucket can be shared through both policies and ACLs. If you want to find and review the source for your bucket access, you can use

the information in this column as a starting point for taking immediate and precise corrective action.

- **Status** - The status of the bucket finding. IAM Access Analyzer for S3 displays findings for all public and shared buckets.
  - **Active** - Finding has not been reviewed.
  - **Archived** - Finding has been reviewed and confirmed as intended.
  - **All** - All findings for buckets that are public or shared with other AWS accounts, including AWS accounts outside of your organization.
- **Access level** - Access permissions granted for the bucket:
  - **List** - List resources.
  - **Read** - Read but not edit resource contents and attributes.
  - **Write** - Create, delete, or modify resources.
  - **Permissions** - Grant or modify resource permissions.
  - **Tagging** - Update tags associated with the resource.

## Enabling IAM Access Analyzer for S3

To use IAM Access Analyzer for S3, you must complete the following prerequisite steps.

1. Grant the required permissions.

   For more information, see Permissions Required to use IAM Access Analyzer in the *IAM User Guide*.
2. Visit IAM to create an account-level analyzer for each Region where you want to use IAM Access Analyzer.

   IAM Access Analyzer for S3 requires an account-level analyzer. To use IAM Access Analyzer for S3, you must create an analyzer that has an account as the zone of trust. For more information, see Enabling IAM Access Analyzer in *IAM User Guide*.

## Blocking all public access

If you want to block all access to a bucket in a single click, you can use the **Block all public access** button in IAM Access Analyzer for S3. When you block all public access to a bucket, no public access is granted. We recommend that you block all public access to your buckets unless you require

public access to support a specific and verified use case. Before you block all public access, ensure that your applications will continue to work correctly without public access.

If you don't want to block all public access to your bucket, you can edit your block public access settings on the Amazon S3 console to configure granular levels of access to your buckets. For more information, see Blocking public access to your Amazon S3 storage.

In rare events, IAM Access Analyzer for S3 might report no findings for a bucket that an Amazon S3 block public access evaluation reports as public. This happens because Amazon S3 block public access reviews policies for current actions and any potential actions that might be added in the future, leading to a bucket becoming public. On the other hand, IAM Access Analyzer for S3 only analyzes the current actions specified for the Amazon S3 service in the evaluation of access status.

**To block all public access to a bucket using IAM Access Analyzer for S3**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane on the left, under **Dashboards**, choose **Access analyzer for S3**.
3. In IAM Access Analyzer for S3, choose a bucket.
4. Choose **Block all public access**.
5. To confirm your intent to block all public access to the bucket, in **Block all public access (bucket settings)**, enter **`confirm`**.

   Amazon S3 blocks all public access to your bucket. The status of the bucket finding updates to **resolved**, and the bucket disappears from the IAM Access Analyzer for S3 listing. If you want to review resolved buckets, open IAM Access Analyzer on the IAM Console.

## Reviewing and changing bucket access

If you did not intend to grant access to the public or other AWS accounts, including accounts outside of your organization, you can modify the bucket ACL, bucket policy, the Multi-Region Access Point policy, or the access point policy to remove the access to the bucket. The **Shared through** column shows all sources of bucket access: bucket policy, bucket ACL, and/or access point policy. Multi-Region Access Points and cross-account access points are reflected under access points.

**To review and change a bucket policy, a bucket ACL, a Multi-Region Access Point, or an access point policy**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. To see whether public access or shared access is granted through a bucket policy, a bucket ACL, a Multi-Region Access Point policy, or an access point policy, look in the **Shared through** column.
4. Under **Buckets**, choose the name of the bucket with the bucket policy, bucket ACL, Multi-Region Access Point policy, or access point policy that you want to change or review.
5. If you want to change or view a bucket ACL:

   a. Choose **Permissions**.

   b. Choose **Access Control List**.

   c. Review your bucket ACL, and make changes as required.

   For more information, see Configuring ACLs.
6. If you want to change or review a bucket policy:

   a. Choose **Permissions**.

   b. Choose **Bucket Policy**.

   c. Review or change your bucket policy as required.

   For more information, see Adding a bucket policy by using the Amazon S3 console.
7. If you want to change or view a Multi-Region Access Point policy:

   a. Choose **Multi-Region Access Point**.

   b. Choose the Multi-Region Access Point name.

   c. Review or change your Multi-Region Access Point policy as required.

   For more information, see Permissions.
8. If you want to review or change an access point policy:

   a. Choose **access points**.

   b. Choose the access point name.

   c. Review or change access as required.

For more information, see Using Amazon S3 access points with the Amazon S3 console.

If you edit or remove a bucket ACL, a bucket policy, or an access point policy to remove public or shared access, the status for the bucket findings updates to resolved. The resolved bucket findings disappear from the IAM Access Analyzer for S3 listing, but you can view them in IAM Access Analyzer.

## Archiving bucket findings

If a bucket grants access to the public or other AWS accounts, including accounts outside of your organization, to support a specific use case (for example, a static website, public downloads, or cross-account sharing), you can archive the finding for the bucket. When you archive bucket findings, you acknowledge and record your intent for the bucket to remain public or shared. Archived bucket findings remain in your IAM Access Analyzer for S3 listing so that you always know which buckets are public or shared.

**To archive bucket findings in IAM Access Analyzer for S3**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. In IAM Access Analyzer for S3, choose an active bucket.
4. To acknowledge your intent for this bucket to be accessed by the public or other AWS accounts, including accounts outside of your organization, choose **Archive**.
5. Enter **`confirm`**, and choose **Archive**.

## Activating an archived bucket finding

After you archive findings, you can always revisit them and change their status back to active, indicating that the bucket requires another review.

**To activate an archived bucket finding in IAM Access Analyzer for S3**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. Choose the archived bucket findings.

4. Choose **Mark as active**.

### Viewing finding details

If you need to see more information about a bucket, you can open the bucket finding details in IAM Access Analyzer on the IAM Console.

**To view finding details in IAM Access Analyzer for S3**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. In IAM Access Analyzer for S3, choose a bucket.
4. Choose **View details**.

   The finding details open in IAM Access Analyzer on the IAM Console.

### Downloading an IAM Access Analyzer for S3 report

You can download your bucket findings as a CSV report that you can use for auditing purposes. The report includes the same information that you see in IAM Access Analyzer for S3 on the Amazon S3 console.

**To download a report**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane on the left, choose **Access analyzer for S3**.
3. In the Region filter, choose the Region.

   IAM Access Analyzer for S3 updates to shows buckets for the chosen Region.
4. Choose **Download report**.

   A CSV report is generated and saved to your computer.

## Verifying bucket ownership with bucket owner condition

Amazon S3 bucket owner condition ensures that the buckets you use in your S3 operations belong to the AWS accounts that you expect.

Most S3 operations read from or write to specific S3 buckets. These operations include uploading, copying, and downloading objects, retrieving or modifying bucket configurations, and retrieving or modifying object configurations. When you perform these operations, you specify the bucket that you want to use by including its name with the request. For example, to retrieve an object from S3, you make a request that specifies the name of a bucket and the object key to retrieve from that bucket.

Because Amazon S3 identifies buckets based on their names, an application that uses an incorrect bucket name in a request could inadvertently perform operations against a different bucket than expected. To help avoid unintentional bucket interactions in situations like this, you can use *bucket owner condition*. Bucket owner condition enables you to verify that the target bucket is owned by the expected AWS account, providing an additional layer of assurance that your S3 operations are having the effects you intend.

**Topics**



## When to use bucket owner condition

We recommend using bucket owner condition whenever you perform a supported S3 operation and know the account ID of the expected bucket owner. Bucket owner condition is available for all S3 object operations and most S3 bucket operations. For a list of S3 operations that don't support bucket owner condition, see Restrictions and limitations.

To see the benefit of using bucket owner condition, consider the following scenario involving AWS customer Bea:

1. Bea develops an application that uses Amazon S3. During development, Bea uses her testing-only AWS account to create a bucket named `bea-data-test`, and configures her application to make requests to `bea-data-test`.
2. Bea deploys her application, but forgets to reconfigure the application to use a bucket in her production AWS account.
3. In production, Bea's application makes requests to `bea-data-test`, which succeed. This results in production data being written to the bucket in Bea's test account.

Bea can help protect against situations like this by using bucket owner condition. With bucket owner condition, Bea can include the AWS account ID of the expected bucket owner in her requests. Amazon S3 then checks the account ID of the bucket owner before processing each request. If the actual bucket owner doesn't match the expected bucket owner, the request fails.

If Bea uses bucket owner condition, the scenario described earlier won't result in Bea's application inadvertently writing to a test bucket. Instead, the requests that her application makes at step 3 will fail with an `Access Denied` error message. By using bucket owner condition, Bea helps eliminate the risk of accidentally interacting with buckets in the wrong AWS account.

## Verifying a bucket owner

To use bucket owner condition, you include a parameter with your request that specifies the expected bucket owner. Most S3 operations involve only a single bucket, and require only this single parameter to use bucket owner condition. For `CopyObject` operations, this first parameter specifies the expected owner of the destination bucket, and you include a second parameter to specify the expected owner of the source bucket.

When you make a request that includes a bucket owner condition parameter, S3 checks the account ID of the bucket owner against the specified parameter before processing the request. If the parameter matches the bucket owner's account ID, S3 processes the request. If the parameter doesn't match the bucket owner's account ID, the request fails with an `Access Denied` error message.

You can use bucket owner condition with the AWS Command Line Interface (AWS CLI), AWS SDKs, and Amazon S3 REST APIs. When using bucket owner condition with the AWS CLI and Amazon S3 REST APIs, use the following parameter names.

| Access method | Parameter for non-copy operations | Copy operation source parameter | Copy operation destination parameter |
|---|---|---|---|
| AWS CLI | `--expected-bucket-owner` | `--expected-source-bucket-owner` | `--expected-bucket-owner` |
| Amazon S3 REST APIs | `x-amz-expected-bucket-owner` header | `x-amz-source-expected-bucket-owner` header | `x-amz-expected-bucket-owner` header |

The parameter names that are required to use bucket owner condition with the AWS SDKs vary depending on the language. To determine the required parameters, see the SDK documentation for your desired language. You can find the SDK documentation at Tools to Build on AWS.

## Examples

The following examples show how you can implement bucket owner condition in Amazon S3 using the AWS CLI or the AWS SDK for Java 2.x.

### Example

#### *Example: Upload an object*

The following example uploads an object to S3 bucket *amzn-s3-demo-bucket1*, using bucket owner condition to ensure that *amzn-s3-demo-bucket1* is owned by AWS account 111122223333.

AWS CLI

```
aws s3api put-object \
                --bucket amzn-s3-demo-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .key("exampleobject")
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

### Example

#### *Example: Copy an object*

The following example copies the object `object1` from S3 bucket *amzn-s3-demo-bucket1* to S3 bucket *amzn-s3-demo-bucket2*. It uses bucket owner condition to ensure that the buckets are owned by the expected accounts according to the following table.

| Bucket | Expected owner |
| --- | --- |
| *amzn-s3-demo-bucket1* | 111122223333 |
| *amzn-s3-demo-bucket2* | 444455556666 |

AWS CLI

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket1/object1 \
                            --bucket amzn-s3-demo-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
                .copySource("amzn-s3-demo-bucket1/object1")
                .destinationBucket("amzn-s3-demo-bucket2")
                .destinationKey("object1copy")
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

**Example**

***Example: Retrieve a bucket policy***

The following example retrieves the access policy for S3 bucket *amzn-s3-demo-bucket1*, using bucket owner condition to ensure that *amzn-s3-demo-bucket1* is owned by AWS account 111122223333.

AWS CLI

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

## Restrictions and limitations

Amazon S3 bucket owner condition has the following restrictions and limitations:

- The value of the bucket owner condition parameter must be an AWS account ID (12-digit numeric value). Service principals aren't supported.
- Bucket owner condition isn't available for CreateBucket, ListBuckets, or any of the operations included in AWS S3 Control. Amazon S3 ignores any bucket owner condition parameters included with requests to these operations.
- Bucket owner condition only verifies that the account specified in the verification parameter owns the bucket. Bucket owner condition doesn't check the configuration of the bucket. It also doesn't guarantee that the bucket's configuration meets any specific conditions or matches any past state.

# Controlling ownership of objects and disabling ACLs for your bucket

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively using access management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs, and we recommend that you keep ACLs disabled except in unusual circumstances where you must control access for each object individually. With ACLs disabled, you can use policies to more easily control access to every object in your bucket, regardless of who uploaded the objects in your bucket.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

**ACLs disabled**

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

For the majority of modern use cases in S3, we recommend that you keep ACLs disabled by applying the Bucket owner enforced setting and using your bucket policy to share data with users outside of your account as needed. This approach simplifies permissions management. You can disable ACLs on both newly created and already existing buckets. For newly created buckets, ACLs are disabled by default. In the case of an existing bucket that already has objects in it, after you disable ACLs, the object and bucket ACLs are no longer part of an access evaluation, and access is granted or denied on the basis of policies. For existing buckets, you can re-enable ACLs at any time after you disable them, and your preexisting bucket and object ACLs are restored.

Before you disable ACLs, we recommend that you review your bucket policy to ensure that it covers all the ways that you intend to grant access to your bucket outside of your account. After you disable ACLs, your bucket accepts only PUT requests that do not specify an ACL or PUT requests with bucket owner full control ACLs, such as the `bucket-owner-full-control` canned ACL or equivalent forms of this ACL expressed in XML. Existing applications that support bucket owner full control ACLs see no impact. PUT requests that contain other ACLs (for example, custom grants to certain AWS accounts) fail and return a `400` error with the error code `AccessControlListNotSupported`.

In contrast, a bucket with the Bucket owner preferred setting continues to accept and honor bucket and object ACLs. With this setting, new objects that are written with the `bucket-owner-full-control` canned ACL are automatically owned by the bucket owner rather than the object writer. All other ACL behaviors remain in place. To require all Amazon S3 PUT operations to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads using this ACL.

To see which Object Ownership settings are applied to your buckets, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to find Object Ownership settings.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

## Object Ownership settings

This table shows the impact that each Object Ownership setting has on ACLs, objects, object ownership, and object uploads.

| Setting | Applies to | Effect on object ownership | Effect on ACLs | Uploads accepted |
|---|---|---|---|---|
| Bucket owner enforced (default) | All new and existing objects | Bucket owner owns every object. | ACLs are disabled and no longer | Uploads with bucket owner full control ACLs |

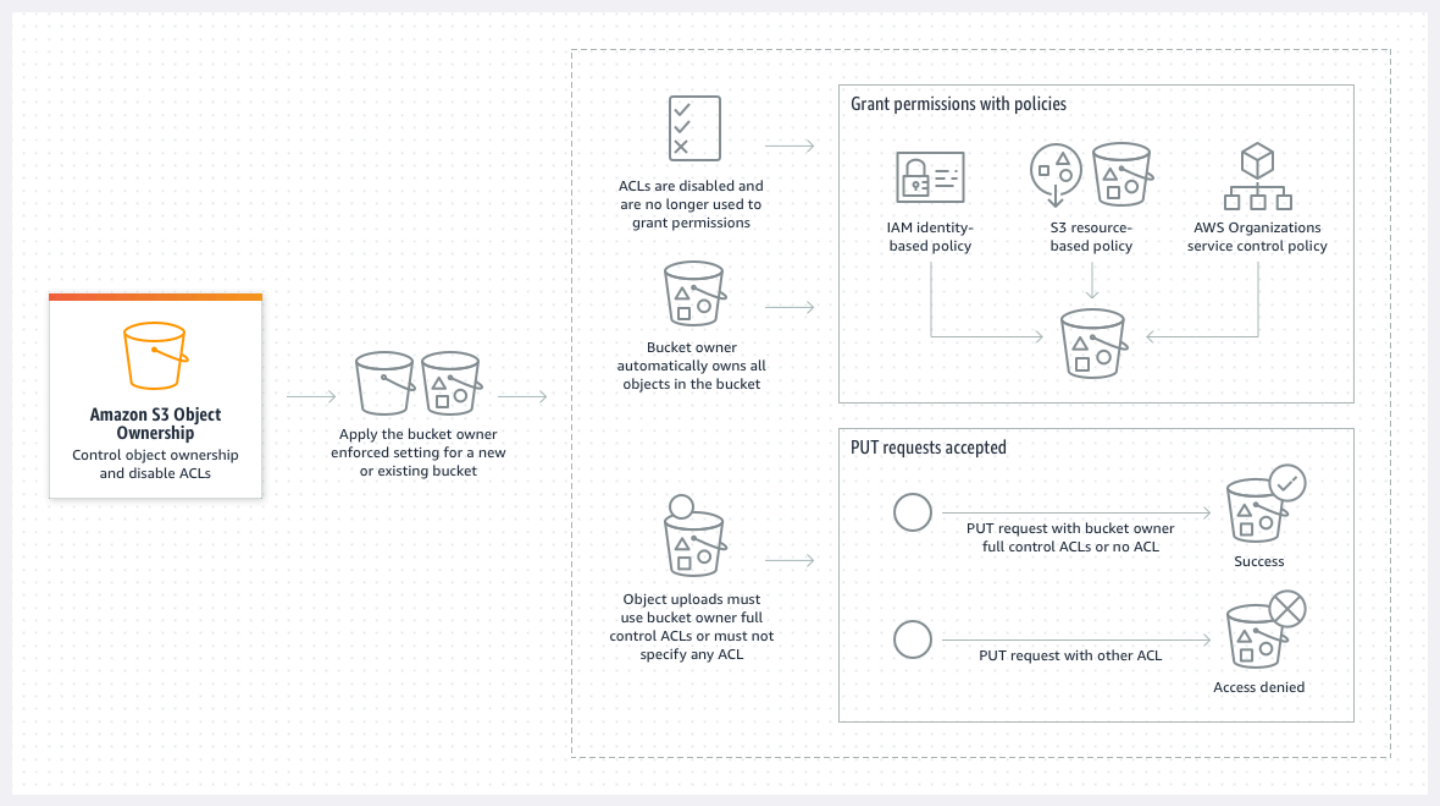
| Setting | Applies to | Effect on object ownership | Effect on ACLs | Uploads accepted |
|---|---|---|---|---|
| | | | affect access permissions to your bucket. Requests to set or update ACLs fail. However, requests to read ACLs are supported.<br><br>Bucket owner has full ownership and control.<br><br>Object writer no longer has full ownership and control. | or uploads that don't specify an ACL |

| Setting | Applies to | Effect on object ownership | Effect on ACLs | Uploads accepted |
|---|---|---|---|---|
| Bucket owner preferred | New objects | If an object upload includes the `bucket-owner-full-control` canned ACL, the bucket owner owns the object.<br><br>Objects uploaded with other ACLs are owned by the writing account. | ACLs can be updated and can grant permissions.<br><br>If an object upload includes the `bucket-owner-full-control` canned ACL, the bucket owner has full control access, and the object writer no longer has full control access. | All uploads |
| Object writer | New objects | Object writer owns the object. | ACLs can be updated and can grant permissions.<br><br>Object writer has full control access. | All uploads |

## Changes introduced by disabling ACLs

When the Bucket owner enforced setting for Object Ownership is applied, ACLs are disabled and you automatically own and take full control over every object in the bucket without taking any additional actions. Bucket owner enforced is the default setting for all newly created buckets. After the Bucket owner enforced setting is applied, you will see three changes:

- All bucket ACLs and object ACLs are disabled, which gives full access to you, as the bucket owner. When you perform a read ACL request on your bucket or object, you will see that full access is given only to the bucket owner.
- You, as the bucket owner, automatically own and have full control over every object in your bucket.
- ACLs no longer affect access permissions to your bucket. As a result, access control for your data is based on policies, such as IAM policies, S3 bucket policies, VPC endpoint policies, and Organizations SCPs.




If you use S3 Versioning, the bucket owner owns and has full control over all object versions in your bucket. Applying the Bucket owner enforced setting does not add a new version of an object.

New objects can be uploaded to your bucket only if they use bucket owner full control ACLs or don't specify an ACL. Object uploads fail if they specify any other ACL. For more information, see Troubleshooting.

Because the following example `PutObject` operation using the AWS Command Line Interface (AWS CLI) includes the `bucket-owner-full-control` canned ACL, the object can be uploaded to a bucket with disabled ACLs.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file --
acl bucket-owner-full-control
```

Because the following `PutObject` operation doesn't specify an ACL, it also succeeds for a bucket with disabled ACLs.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file
```

> **Note**
>
> If other AWS accounts need access to objects after uploading, you must grant additional permissions to those accounts through bucket policies. For more information, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

#### Re-enabling ACLs

You can re-enable ACLs by changing from the Bucket owner enforced setting to another Object Ownership setting at any time. If you used object ACLs for permissions management before you applied the Bucket owner enforced setting and you didn't migrate these object ACL permissions to your bucket policy, after you re-enable ACLs, these permissions are restored. Additionally, objects written to the bucket while the Bucket owner enforced setting was applied are still owned by the bucket owner.

For example, if you change from the Bucket owner enforced setting back to the Object writer setting, you, as the bucket owner, no longer own and have full control over objects that were previously owned by other AWS accounts. Instead, the uploading accounts again own these objects. Objects owned by other accounts use ACLs for permissions, so you can't use policies to grant permissions to these objects. However, you, as the bucket owner, still own any objects that were written to the bucket while the Bucket owner enforced setting was applied. These objects are not owned by the object writer, even if you re-enable ACLs.

For instructions on enabling and managing ACLs using the AWS Management Console, AWS Command Line Interface (CLI), REST API, or AWS SDKs, see Configuring ACLs.

### Prerequisites for disabling ACLs

Before you disable ACLs for an existing bucket, complete the following prerequisites.

**Review bucket and object ACLs and migrate ACL permissions**

When you disable ACLs, permissions granted by bucket and object ACLs no longer affect access. Before you disable ACLs, review your bucket and object ACLs.

If your bucket ACLs grant read or write permissions to others outside of your account, you must migrate these permissions to your bucket policy before you can apply the Bucket owner enforced setting. If you don't migrate bucket ACLs that grant read or write access outside of your account, your request to apply the Bucket owner enforced setting fails and returns the InvalidBucketAclWithObjectOwnership error code.

For example, if you want to disable ACLs for a bucket that receives server access logs, you must migrate the bucket ACL permissions for the S3 log delivery group to the logging service principal in a bucket policy. For more information, see Grant access to the S3 log delivery group for server access logging.

If you want the object writer to maintain full control of the object that they upload, object writer is the best Object Ownership setting for your use case. If you want to control access at the individual object level, bucket owner preferred is the best choice. These use cases are uncommon.

To review ACLs and migrate ACL permissions to bucket policies, see Prerequisites for disabling ACLs.

**Identify requests that required an ACL for authorization**

To identify Amazon S3 requests that required ACLs for authorization, you can use the `aclRequired` value in Amazon S3 server access logs or AWS CloudTrail. If the request required an ACL for authorization or if you have PUT requests that specify an ACL, the string is `Yes`. If no ACLs were required, or if you are setting a `bucket-owner-full-control` canned ACL, or if the requests are allowed by your bucket policy, the `aclRequired` value string is "`-`" in Amazon S3 server access logs and is absent in CloudTrail. For more information about the expected `aclRequired` values, see `aclRequired` values for common Amazon S3 requests.

If you have `PutBucketAcl` or `PutObjectAcl` requests with headers that grant ACL-based permissions, with the exception of the `bucket-owner-full-control` canned ACL, you must remove those headers before you can disable ACLs. Otherwise, your requests will fail.

For all other requests that required an ACL for authorization, migrate those ACL permissions to bucket policies. Then, remove any bucket ACLs before you enable the bucket owner enforced setting.

> **Note**
>
> Do not remove object ACLs. Otherwise, applications that rely on object ACLs for permissions will lose access.

If you see that no requests required an ACL for authorization, you can proceed to disable ACLs. For more information about identifying requests, see Using Amazon S3 server access logs to identify requests and Identifying Amazon S3 requests using CloudTrail.

**Review and update bucket policies that use ACL-related condition keys**

After you apply the Bucket owner enforced setting to disable ACLs, new objects can be uploaded to your bucket only if the request uses bucket owner full control ACLs or doesn't specify an ACL. Before disabling ACLs, review your bucket policy for ACL-related condition keys.

If your bucket policy uses an ACL-related condition key to require the `bucket-owner-full-control` canned ACL (for example, `s3:x-amz-acl`), you don't need to update your bucket policy. The following bucket policy uses the `s3:x-amz-acl` to require the `bucket-owner-full-control` canned ACL for S3 `PutObject` requests. This policy *still* requires the object writer to specify the `bucket-owner-full-control` canned ACL. However, buckets with ACLs disabled still accept this ACL, so requests continue to succeed with no client-side changes required.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with bucket owner full control",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
```

```
                }
            }
        }
    ]
}
```

However, if your bucket policy uses an ACL-related condition key that requires a different ACL, you must remove this condition key. This example bucket policy requires the `public-read` ACL for S3 `PutObject` requests and therefore must be updated before disabling ACLs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "public-read"
                }
            }
        }
    ]
}
```

## Object Ownership permissions

To apply, update, or delete an Object Ownership setting for a bucket, you need the `s3:PutBucketOwnershipControls` permission. To return the Object Ownership setting for a bucket, you need the `s3:GetBucketOwnershipControls` permission. For more information, see Setting Object Ownership when you create a bucket and Viewing the Object Ownership setting for an S3 bucket.

## Disabling ACLs for all new buckets

By default, all new buckets are created with the Bucket owner enforced setting applied and ACLs are disabled. We recommend keeping ACLs disabled. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM policies for access control instead of ACLs. Policies are a simplified and more flexible access control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources.

## Replication and Object Ownership

When you use S3 replication and the source and destination buckets are owned by different AWS accounts, you can disable ACLs (with the Bucket owner enforced setting for Object Ownership) to change replica ownership to the AWS account that owns the destination bucket. This setting mimics the existing owner override behavior without the need of the `s3:ObjectOwnerOverrideToBucketOwner` permission. All objects that are replicated to the destination bucket with the Bucket owner enforced setting are owned by the destination bucket owner. For more information about the owner override option for replication configurations, see Changing the replica owner.

## Setting Object Ownership

You can apply an Object Ownership setting by using the Amazon S3 console, AWS CLI, AWS SDKs, Amazon S3 REST API, or AWS CloudFormation. The following REST API and AWS CLI commands support Object Ownership:

| REST API | AWS CLI | Description |
|---|---|---|
| PutBucketOwnershipControls | put-bucket-ownership-controls | Creates or modifies the Object Ownership setting for an existing S3 bucket. |
| CreateBucket | create-bucket | Creates a bucket using the `x-amz-object-ownership` request header to specify the Object Ownership setting. |

| REST API | AWS CLI | Description |
| --- | --- | --- |
| GetBucketOwnershipControls | get-bucket-ownership-controls | Retrieves the Object Ownership setting for an Amazon S3 bucket. |
| DeleteBucketOwnershipControls | delete-bucket-ownership-controls | Deletes the Object Ownership setting for an Amazon S3 bucket. |

For more information about applying and working with Object Ownership settings, see the following topics.

**Topics**

- Prerequisites for disabling ACLs
- Setting Object Ownership when you create a bucket
- Setting Object Ownership on an existing bucket
- Viewing the Object Ownership setting for an S3 bucket
- Disabling ACLs for all new buckets and enforcing Object Ownership
- Troubleshooting

## Prerequisites for disabling ACLs

If your bucket ACL grants access outside of your AWS account, before you disable ACLs, you must migrate your bucket ACL permissions to your bucket policy and reset your bucket ACL to the default private ACL. If you don't migrate these bucket ACLs, your request to apply the Bucket owner enforced setting to disable ACLs fails and returns the InvalidBucketAclWithObjectOwnership error code. We also recommend that you review your object ACL permissions and migrate them to your bucket policy. For more information about other suggested prerequisites, see Prerequisites for disabling ACLs.

Each of your existing bucket and object ACLs has an equivalent in an IAM policy. The following bucket policy examples show you how `READ` and `WRITE` permissions for bucket and object ACLs map to IAM permissions. For more information about how each ACL translates to IAM permissions, see Mapping of ACL permissions and access policy permissions.

To review and migrate ACL permissions to bucket policies, see the following topics.

**Topics**



### Bucket policies examples

These example bucket policies show you how to migrate `READ` and `WRITE` bucket and object ACL permissions for a third-party AWS account to a bucket policy. `READ_ACP` and `WRITE_ACP` ACLs are less relevant for policies because they grant ACL-related permissions (`s3:GetBucketAcl`, `s3:GetObjectAcl`, `s3:PutBucketAcl`, and `s3:PutObjectAcl`).

#### Example – READ ACL for a bucket

If your bucket had a `READ` ACL that grants AWS account *111122223333* permission to list the contents of your bucket, you can write a bucket policy that grants `s3:ListBucket`, `s3:ListBucketVersions`, `s3:ListBucketMultipartUploads` permissions for your bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to list the objects in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [

                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
```

```
    ]
}
```

### Example – READ ACLs for every object in a bucket

If every object in your bucket has a READ ACL that grants access to AWS account *111122223333*, you can write a bucket policy that grants `s3:GetObject` and `s3:GetObjectVersion` permissions to this account for every object in your bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Read permission for every object in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

This example resource element grants access to a specific object.

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

### Example – WRITE ACL that grants permissions to write objects to a bucket

If your bucket has a WRITE ACL that grants AWS account *111122223333* permission to write objects to your bucket, you can write a bucket policy that grants `s3:PutObject` permission for your bucket.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to write objects to a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

### Using the S3 console to review and migrate ACL permissions

#### To review a bucket's ACL permissions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name.
3. Choose the **Permissions** tab.
4. Under **Access control list (ACL)**, review your bucket ACL permissions.

#### To review an object's ACL permissions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name containing your object.
3. In the **Objects** list, choose your object name.
4. Choose the **Permissions** tab.
5. Under **Access control list (ACL)**, review your object ACL permissions.

**To migrate ACL permissions and update your bucket ACL**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name.
3. On the **Permissions** tab, under **Bucket policy**, choose **Edit**.
4. In the **Policy** box, add or update your bucket policy.

   For example bucket policies, see Bucket policies examples and Example walkthroughs.
5. Choose **Save changes**.
6. Update your bucket ACL to remove ACL grants to other groups or AWS accounts.
7. Apply the **Bucket owner enforced** setting for Object Ownership.

**Using the AWS CLI to review and migrate ACL permissions**

1. To return the bucket ACL for your bucket, use the get-bucket-acl AWS CLI command:

   ```
   aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
   ```

   For example, this bucket ACL grants `WRITE` and `READ` access to a third-party account. In this ACL, the third-party account is identified by the canonical user ID. To apply the Bucket owner enforced setting and disable ACLs, you must migrate these permissions for the third-party account to a bucket policy.

   ```
   {
       "Owner": {
           "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
           "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
       },
       "Grants": [
           {
               "Grantee": {
                   "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                   "ID":
    "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                   "Type": "CanonicalUser"
               },
               "Permission": "FULL_CONTROL"
   ```

```
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        }
    ]
}
```

For other example ACLs, see Example walkthroughs.

2. Migrate your bucket ACL permissions to a bucket policy:

   This example bucket policy grants `s3:PutObject` and `s3:ListBucket` permissions for a third-party account. In the bucket policy, the third-party account is identified by the AWS account ID (*111122223333*).

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PolicyForCrossAccountAllowUpload",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
```

```
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

For more example bucket policies, see Bucket policies examples and Example walkthroughs.

3. To return the ACL for a specific object, use the get-object-acl AWS CLI command.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key EXAMPLE-OBJECT-KEY
```

4. If required, migrate object ACL permissions to your bucket policy.

   This example resource element grants access to a specific object in a bucket policy.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-OBJECT-KEY"
```

5. Reset the ACL for your bucket to the default ACL.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

6. Apply the Bucket owner enforced setting for Object Ownership.

### Example walkthroughs

The following examples show you how to migrate ACL permissions to bucket policies for specific use cases.

### Topics

- Grant access to the S3 log delivery group for server access logging
- Grant public read access to the objects in a bucket
- Grant Amazon ElastiCache (Redis OSS) access to your S3 bucket

### Grant access to the S3 log delivery group for server access logging

If you want to apply the Bucket owner enforced setting to disable ACLs for a server access logging destination bucket (also known as a *target bucket*), you must migrate bucket ACL permissions for the S3 log delivery group to the logging service principal (`logging.s3.amazonaws.com`) in a bucket policy. For more information about log delivery permissions, see Permissions for log delivery.

This bucket ACL grants `WRITE` and `READ_ACP` access to the S3 log delivery group:

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

**To migrate bucket ACL permissions for the S3 log delivery group to the logging service principal in a bucket policy**

1. Add the following bucket policy to your destination bucket, replacing the example values.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. Reset the ACL for your destination bucket to the default ACL.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. Apply the Bucket owner enforced setting for Object Ownership to your destination bucket.

### Grant public read access to the objects in a bucket

If your object ACLs grant public read access to all of the objects in your bucket, you can migrate these ACL permissions to a bucket policy.

This object ACL grants public read access to an object in a bucket:

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

### To migrate public read ACL permissions to a bucket policy

1. To grant public read access to all of the objects in your bucket, add the following bucket policy, replacing the example values.

   ```
   aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
    file://policy.json

   policy.json:
   {
       "Version": "2012-10-17",
   ```

```
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

To grant public access to a specific object in a bucket policy, use the following format for the `Resource` element.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

To grant public access to all of the objects with a specific prefix, use the following format for the `Resource` element.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/PREFIX/*"
```

2. Apply the Bucket owner enforced setting for Object Ownership.

#### Grant Amazon ElastiCache (Redis OSS) access to your S3 bucket

You can export your ElastiCache (Redis OSS) backup to an S3 bucket, which gives you access to the backup from outside ElastiCache. To export your backup to an S3 bucket, you must grant ElastiCache permissions to copy a snapshot to the bucket. If you've granted permissions to ElastiCache in a bucket ACL, you must migrate these permissions to a bucket policy before you apply the Bucket owner enforced setting to disable ACLs. For more information, see Grant ElastiCache access to your Amazon S3 bucket in the *Amazon ElastiCache User Guide*.

The following example shows the bucket ACL permissions that grant permissions to ElastiCache.

```
{
    "Owner": {
```

```
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

**To migrate bucket ACL permissions for ElastiCache (Redis OSS) to a bucket policy**

1. Add the following bucket policy to your bucket, replacing the example values.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
"Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

2. Reset the ACL for your bucket to the default ACL:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. Apply the Bucket owner enforced setting for Object Ownership.

## Setting Object Ownership when you create a bucket

When you create a bucket, you can configure S3 Object Ownership. To set Object Ownership for an existing bucket, see Setting Object Ownership on an existing bucket.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

**ACLs disabled**

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

**Permissions**: To apply the **Bucket owner enforced** setting or the **Bucket owner preferred** setting, you must have the following permissions: `s3:CreateBucket` and `s3:PutBucketOwnershipControls`. No additional permissions are needed when creating a bucket with the **Object writer** setting applied. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

> **Important**
>
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs, and we recommend that you disable ACLs except in unusual circumstances where you need to control access for each object individually. With Object Ownership, you can disable ACLs and rely on policies for access control. When you disable ACLs, you can easily maintain a

bucket with objects uploaded by different AWS accounts. You, as the bucket owner, own all the objects in the bucket and can manage access to them using policies.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   **Note**

   To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.
3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **General purpose**.
7. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, dots (.), and hyphens (-). For best compatibility, we recommend that you avoid using dots (.) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.

After you create the bucket, you cannot change its name. For more information about naming buckets, see Bucket naming rules.

> **Important**
>
> Avoid including sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

8. AWS Management Console allows you to copy an existing bucket's settings to your new bucket. If you do not want to copy the settings of an existing bucket, skip to the next step.

> **Note**
>
> This option:
>
> - Is not available in the AWS CLI and is only available in console
> - Is not available for directory buckets
> - Does not copy the bucket policy from the existing bucket to the new bucket

To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you would like to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window re-opens.

Under **Copy settings from existing bucket**, you will now see the name of the bucket you selected. You will also see a **Restore defaults** option that you can use to remove the copied bucket settings. Review the remaining bucket settings, on the **Create bucket** page. You will see that they now match the settings of the bucket that you selected. You can skip to the final step.

9. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

**ACLs disabled**

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access

permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

  If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

10. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

    By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

> **Note**
>
> To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

11. (Optional) Under **Bucket Versioning**, you can choose if you wish to keep variants of objects in your bucket. For more information about versioning, see Using versioning in S3 buckets.

    To disable or enable versioning on your bucket, choose either **Disable** or **Enable**.

12. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs used to categorize storage.

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

13. Under **Default encryption**, choose **Edit**.

14. To configure default encryption, under **Encryption type**, choose one of the following:

    - **Amazon S3 managed key (SSE-S3)**
    - **AWS Key Management Service key (SSE-KMS)**

    > **Important**
    >
    > If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

    Buckets and new objects are encrypted with server-side encryption with an **Amazon S3 managed key** as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

    For more information about using Amazon S3 server-side encryption to encrypt your data, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

15. If you chose **AWS Key Management Service key (SSE-KMS)**, do the following:

a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

  Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN. For more information on cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information on SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS).
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS).

b. When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

16. (Optional) If you want to enable S3 Object Lock, do the following:

a. Choose **Advanced settings**.

> **Important**
>
> Enabling Object Lock also enables versioning for the bucket. After enabling you must configure the Object Lock default retention and legal hold settings to protect new objects from being deleted or overwritten.

b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

For more information, see Using S3 Object Lock.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning` and `s3:PutBucketObjectLockConfiguration`.

17. Choose **Create bucket**.

### Using the AWS CLI

To set Object Ownership when you create a new bucket, use the `create-bucket` AWS CLI command with the `--object-ownership` parameter.

This example applies the Bucket owner enforced setting for a new bucket using the AWS CLI:

```
aws s3api create-bucket --bucket  amzn-s3-demo-bucket --region us-east-1 --object-
ownership BucketOwnerEnforced
```

> **Important**
>
> If you don't set Object Ownership when you create a bucket by using the AWS CLI, the default setting will be `ObjectWriter` (ACLs enabled).

### Using the AWS SDK for Java

This example sets the Bucket owner enforced setting for a new bucket using the AWS SDK for Java:

```
    // Build the ObjectOwnership for CreateBucket
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

### Using AWS CloudFormation

To use the `AWS::S3::Bucket` AWS CloudFormation resource to set Object Ownership when you create a new bucket, see OwnershipControls within AWS::S3::Bucket in the *AWS CloudFormation User Guide*.

### Using the REST API

To apply the Bucket owner enforced setting for S3 Object Ownership, use the `CreateBucket` API operation with the `x-amz-object-ownership` request header set to `BucketOwnerEnforced`. For information and examples, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

**Next steps**: After you apply the Bucket owner enforced or bucket owner preferred settings for Object Ownership, you can further take the following steps:

- Bucket owner enforced – Require that all new buckets are created with ACLs disabled by using an IAM or Organizations policy.
- Bucket owner preferred – Add an S3 bucket policy to require the `bucket-owner-full-control` canned ACL for all object uploads to your bucket.

## Setting Object Ownership on an existing bucket

You can configure S3 Object Ownership on an existing S3 bucket. To apply Object Ownership when you create a bucket, see Setting Object Ownership when you create a bucket.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

**ACLs disabled**

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

**Prerequisites**: Before you apply the Bucket owner enforced setting to disable ACLs, you must migrate bucket ACL permissions to bucket policies and reset your bucket ACLs to the default private ACL. We also recommend that you migrate object ACL permissions to bucket policies

and edit bucket policies that require ACLs other than bucket owner full control ACLs. For more information, see Prerequisites for disabling ACLs.

**Permissions**: To use this operation, you must have the `s3:PutBucketOwnershipControls` permission. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to apply an S3 Object Ownership setting to.
3. Choose the **Permissions** tab.
4. Under **Object Ownership**, choose **Edit**.
5. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   **ACLs disabled**

   - **Bucket owner enforced** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

     To require that all new buckets are created with ACLs disabled by using IAM or AWS Organizations policies, see Disabling ACLs for all new buckets (bucket owner enforced).

   **ACLs enabled**

   - **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

     If you apply the bucket owner preferred setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that only allows object uploads that use this ACL.
   - **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.
6. Choose **Save**.

### Using the AWS CLI

To apply an Object Ownership setting for an existing bucket, use the `put-bucket-ownership-controls` command with the `--ownership-controls` parameter. Valid values for ownership are `BucketOwnerEnforced`, `BucketOwnerPreferred`, or `ObjectWriter`.

This example applies the Bucket owner enforced setting for an existing bucket by using the AWS CLI:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

For information about `put-bucket-ownership-controls`, see put-bucket-ownership-controls in the *AWS Command Line Interface User Guide*.

### Using the AWS SDK for Java

This example applies the `BucketOwnerEnforced` setting for Object Ownership on an existing bucket by using the AWS SDK for Java:

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
               .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
               .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
                       .build();

         // Send the request to Amazon S3
         s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

### Using AWS CloudFormation

To use AWS CloudFormation to apply an Object Ownership setting for an existing bucket, see AWS::S3::Bucket OwnershipControls in the *AWS CloudFormation User Guide*.

### Using the REST API

To use the REST API to apply an Object Ownership setting to an existing S3 bucket, use `PutBucketOwnershipControls`. For more information, see PutBucketOwnershipControls in the *Amazon Simple Storage Service API Reference*.

**Next steps**: After you apply the Bucket owner enforced or bucket owner preferred settings for Object Ownership, you can further take the following steps:

- Bucket owner enforced – Require that all new buckets are created with ACLs disabled by using an IAM or Organizations policy.
- Bucket owner preferred – Add an S3 bucket policy to require the `bucket-owner-full-control` canned ACL for all object uploads to your bucket.

## Viewing the Object Ownership setting for an S3 bucket

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

### ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

**ACLs enabled**

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

You can view the S3 Object Ownership settings for an Amazon S3 bucket. To set Object Ownership for a new bucket, see Setting Object Ownership when you create a bucket. To set Object Ownership for an existing bucket, see Setting Object Ownership on an existing bucket.

**Permissions:** To use this operation, you must have the `s3:GetBucketOwnershipControls` permission. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to apply an Object Ownership setting to.
3. Choose the **Permissions** tab.
4. Under **Object Ownership**, you can view the Object Ownership settings for your bucket.

**Using the AWS CLI**

To retrieve the S3 Object Ownership setting for an S3 bucket, use the get-bucket-ownership-controls AWS CLI command.

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

**Using the REST API**

To retrieve the Object Ownership setting for an S3 bucket, use the `GetBucketOwnershipControls` API operation. For more information, see GetBucketOwnershipControls.

## Disabling ACLs for all new buckets and enforcing Object Ownership

We recommend that you disable ACLs on your Amazon S3 buckets. You can do this by applying the Bucket owner enforced setting for S3 Object Ownership. When you apply this setting, ACLs are disabled and you automatically own and have full control over all objects in your bucket. To require that all new buckets are created with ACLs disabled, use AWS Identity and Access Management (IAM) policies or AWS Organizations service control policies (SCPs), as described in the next section.

To enforce object ownership for new objects without disabling ACLs, you can apply the Bucket owner preferred setting. When you apply this setting, we strongly recommend that you update your bucket policy to require the `bucket-owner-full-control` canned ACL for all PUT requests to your bucket. Make sure you also update your clients to send the `bucket-owner-full-control` canned ACL to your bucket from other accounts.

### Topics



### Disabling ACLs for all new buckets (bucket owner enforced)

The following example IAM policy denies the `s3:CreateBucket` permission for a specific IAM user or role unless the Bucket owner enforced setting is applied for Object Ownership. The key-value pair in the `Condition` block specifies `s3:x-amz-object-ownership` as its key and the `BucketOwnerEnforced` setting as its value. In other words, the IAM user can create buckets only if they set the Bucket owner enforced setting for Object Ownership and disable ACLs. You can also use this policy as a boundary SCP for your AWS organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
```

```
                }
            }
        }
    ]
}
```

### Requiring the bucket-owner-full-control canned ACL for Amazon S3 PUT operations (bucket owner preferred)

With the Bucket owner preferred setting for Object Ownership, you, as the bucket owner, own and have full control over new objects that other accounts write to your bucket with the `bucket-owner-full-control` canned ACL. However, if other accounts write objects to your bucket without the `bucket-owner-full-control` canned ACL, the object writer maintains full control access. You, as the bucket owner, can implement a bucket policy that allows writes only if they specify the `bucket-owner-full-control` canned ACL.

> **Note**
>
> If you have ACLs disabled with the Bucket owner enforced setting, you, as the bucket owner, automatically own and have full control over all the objects in your bucket. You don't need to use this section to update your bucket policy to enforce object ownership for the bucket owner.

The following bucket policy specifies that account *111122223333* can upload objects to *amzn-s3-demo-bucket* only when the object's ACL is set to `bucket-owner-full-control`. Be sure to replace *111122223333* with your account and *amzn-s3-demo-bucket* with the name of your bucket.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Only allow writes to my bucket with bucket owner full control",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
            ]
         },
```

```
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-acl": "bucket-owner-full-control"
            }
         }
      }
   ]
}
```

The following is an example copy operation that includes the `bucket-owner-full-control` canned ACL by using the AWS Command Line Interface (AWS CLI).

```
aws s3 cp file.txt s3://amzn-s3-demo-bucket --acl bucket-owner-full-control
```

After the bucket policy is put into effect, if the client does not include the `bucket-owner-full-control` canned ACL, the operation fails, and the uploader receives the following error:

An error occurred (AccessDenied) when calling the PutObject operation: Access Denied.

> **Note**
>
> If clients need access to objects after uploading, you must grant additional permissions to the uploading account. For information about granting accounts access to your resources, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

## Troubleshooting

When you apply the Bucket owner enforced setting for S3 Object Ownership, access control lists (ACLs) are disabled and you, as the bucket owner, automatically own all objects in your bucket. ACLs no longer affect permissions for the objects in your bucket. You can use policies to grant permissions. All S3 PUT requests must either specify the `bucket-owner-full-control` canned ACL or not specify an ACL, or these requests will fail. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

If an invalid ACL is specified or bucket ACL permissions grant access outside of your AWS account, you might see the following error responses.

**AccessControlListNotSupported**

After you apply the Bucket owner enforced setting for Object Ownership, ACLs are disabled. Requests to set ACLs or update ACLs fail with a `400` error and return the AccessControlListNotSupported error code. Requests to read ACLs are still supported. Requests to read ACLs always return a response that shows full control for the bucket owner. In your PUT operations, you must either specify bucket owner full control ACLs or not specify an ACL. Otherwise, your PUT operations fail.

The following example `put-object` AWS CLI command includes the `public-read` canned ACL.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object-key-name --body doc-example-body --acl public-read
```

If the bucket uses the Bucket owner enforced setting to disable ACLs, this operation fails, and the uploader receives the following error message:

An error occurred (AccessControlListNotSupported) when calling the PutObject operation: The bucket does not allow ACLs

**InvalidBucketAclWithObjectOwnership**

If you want to apply the Bucket owner enforced setting to disable ACLs, your bucket ACL must give full control only to the bucket owner. Your bucket ACL cannot give access to an external AWS account or any other group. For example, if your `CreateBucket` request sets Bucket owner enforced and specifies a bucket ACL that provides access to an external AWS account, your request fails with a `400` error and returns the InvalidBucketAclWithObjectOwnership error code. Similarly, if your `PutBucketOwnershipControls` request sets Bucket owner enforced on a bucket that has a bucket ACL that grants permissions to others, the request fails.

**Example : Existing bucket ACL grants public read access**

For example, if an existing bucket ACL grants public read access, you cannot apply the Bucket owner enforced setting for Object Ownership until you migrate these ACL permissions to a bucket policy and reset your bucket ACL to the default private ACL. For more information, see Prerequisites for disabling ACLs.

This example bucket ACL grants public read access:

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

The following example `put-bucket-ownership-controls` AWS CLI command applies the Bucket owner enforced setting for Object Ownership:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

Because the bucket ACL grants public read access, the request fails and returns the following error code:

An error occurred (InvalidBucketAclWithObjectOwnership) when calling the PutBucketOwnershipControls operation: Bucket cannot have ACLs set with ObjectOwnership's BucketOwnerEnforced setting

# Logging and monitoring in Amazon S3

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Amazon S3 resources and responding to potential incidents.

For more information, see Monitoring Amazon S3.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Amazon CloudWatch Alarms**

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.

**AWS CloudTrail Logs**

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon S3. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

**Amazon GuardDuty**

Amazon GuardDuty is a threat detection service that continuously monitors your accounts, containers, workloads, and the data within your AWS environment to identify potential threats or security risks to your S3 buckets. GuardDuty also provides rich context about the threats that it detects. GuardDuty monitors AWS CloudTrail management logs for threats and surfaces security relevant information. For example, GuardDuty will include factors of an API request, such as the user that made the request, the location the request was made from, and the

specific API requested, that could be unusual in your environment. GuardDuty S3 Protection monitors the S3 data events collected by CloudTrail and identifies potentially anomalous and malicious behavior in all the S3 buckets in your environment.

**Amazon S3 Access Logs**

Server access logs provide detailed records about requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. For more information, see Logging requests with server access logging.

**AWS Trusted Advisor**

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks.

Trusted Advisor has the following Amazon S3-related checks:

- Logging configuration of Amazon S3 buckets.
- Security checks for Amazon S3 buckets that have open access permissions.
- Fault tolerance checks for Amazon S3 buckets that don't have versioning enabled, or have versioning suspended.

For more information, see AWS Trusted Advisor in the *AWS Support User Guide*.

The following security best practices also address logging and monitoring:

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- Enable AWS Config
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

# Compliance Validation for Amazon S3

The security and compliance of Amazon S3 is assessed by third-party auditors as part of multiple AWS compliance programs, including the following:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS Services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using Amazon S3 is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Amazon S3 is subject to compliance with standards like HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides that discuss architectural considerations and steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance outlines how companies use AWS to help them meet HIPAA requirements.
- AWS Compliance Resources provide several different workbooks and guides that might apply to your industry and location.
- AWS Config can be used to assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub provides you with a comprehensive view of your security state within AWS and helps you check your compliance with security industry standards and best practices.
- Using S3 Object Lock can help you meet technical requirements of financial services regulators (such as the SEC, FINRA, and CFTC) that require write once, read many (WORM) data storage for certain types of books and records information.

- Amazon S3 Inventory can help you audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs.

# Resilience in Amazon S3

The AWS global infrastructure is built around Regions and Availability Zones. AWS Regions provide multiple, physically separated and isolated Availability Zones that are connected with low latency, high throughput, and highly redundant networking. These Availability Zones offer you an effective way to design and operate applications and databases. They are more highly available, fault tolerant, and scalable than traditional single data center infrastructures or multi-data center infrastructures. If you specifically need to replicate your data over greater geographic distances, you can use Replicating objects overview, which enables automatic, asynchronous copying of objects across buckets in different AWS Regions.

Each AWS Region has multiple Availability Zones. You can deploy your applications across multiple Availability Zones in the same Region for fault tolerance and low latency. Availability Zones are connected to each other with fast, private fiber-optic networking, enabling you to easily architect applications that automatically fail over between Availability Zones without interruption.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Amazon S3 offers several features to help support your data resiliency and backup needs.

**Lifecycle configuration**

A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. With lifecycle configuration rules, you can tell Amazon S3 to transition objects to less expensive storage classes, archive them, or delete them. For more information, see Managing your storage lifecycle.

**Versioning**

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. For more information, see Using versioning in S3 buckets.

**S3 Object Lock**

You can use S3 Object Lock to store objects using a *write once, read many* (WORM) model. Using S3 Object Lock, you can prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. S3 Object Lock enables you to meet regulatory requirements

that require WORM storage or simply to add an additional layer of protection against object changes and deletion. For more information, see Using S3 Object Lock.

**Storage classes**

Amazon S3 offers a range of storage classes to choose from depending on the requirements of your workload. The S3 Standard-IA and S3 One Zone-IA storage classes are designed for data you access about once a month and need milliseconds access. The S3 Glacier Instant Retrieval storage class is designed for long-lived archive data accessed with milliseconds access that you access about once a quarter. For archive data that does not require immediate access, such as backups, you can use the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For more information, see Using Amazon S3 storage classes.

The following security best practices also address resilience:

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

## Encryption of Amazon S3 backups

If you are storing backups using Amazon S3, the encryption of your backups depends on the configuration of those buckets. Amazon S3 provides a way to set the default encryption behavior for an S3 bucket. You can set default encryption on a bucket so that all objects are encrypted when they are stored in the bucket. The default encryption supports keys stored in AWS KMS (SSE-KMS). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

For more information about Versioning and Object Lock, see the following topics: Using versioning in S3 buckets Using S3 Object Lock

# Infrastructure security in Amazon S3

As a managed service, Amazon S3 is protected by the AWS global network security procedures that are described in the security pillar of the AWS Well-Architected Framework.

Access to Amazon S3 via the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend also supporting TLS 1.3. (For more information about this recommendation, see Faster AWS cloud connections with TLS 1.3 on the *AWS Security Blog*.) Clients must also support cipher suites with Perfect Forward Secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Additionally, requests must be signed using AWS Signature V4 or AWS Signature V2, requiring valid credentials to be provided.

These APIs are callable from any network location. However, Amazon S3 does support resource-based access policies, which can include restrictions based on the source IP address. You can also use Amazon S3 bucket policies to control access to buckets from specific virtual private cloud (VPC) endpoints, or specific VPCs. Effectively, this isolates network access to a given Amazon S3 bucket from only the specific VPC within the AWS network. For more information, see Controlling access from VPC endpoints with bucket policies.

The following security best practices also address infrastructure security in Amazon S3:

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

# Configuration and vulnerability analysis in Amazon S3

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resources:

- Compliance Validation for Amazon S3
- Shared Responsibility Model
- Amazon Web Services: Overview of Security Processes

The following security best practices also address configuration and vulnerability analysis in Amazon S3:

- Identify and audit all your Amazon S3 buckets
- Enable AWS Config

# Security best practices for Amazon S3

Amazon S3 provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful recommendations rather than prescriptions.

**Topics**



## Amazon S3 security best practices

The following best practices for Amazon S3 can help prevent security incidents.

**Disable access control lists (ACLs)**

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively using access management policies.

A majority of modern use cases in Amazon S3 no longer require the use of access control lists (ACLs). We recommend that you disable ACLs, except in unusual circumstances where you must control access for each object individually. To disable ACLs and take ownership of every object in your bucket, apply the bucket owner enforced setting for S3 Object Ownership. When you disable ACLs, you can easily maintain a bucket with objects uploaded by different AWS accounts.

When ACLs are disabled access control for your data is based on policies, such as the following:

- AWS Identity and Access Management (IAM) user policies
- S3 bucket policies
- Virtual private cloud (VPC) endpoint policies
- AWS Organizations service control policies (SCPs)

Disabling ACLs simplifies permissions management and auditing. ACLs are disabled for new buckets by default. You can also disable ACLs for existing buckets. If you have an existing bucket that already has objects in it, after you disable ACLs, the object and bucket ACLs are no longer part of the access-evaluation process. Instead, access is granted or denied on the basis of policies.

Before you disable ACLs, make sure that you do the following:

- Review your bucket policy to ensure that it covers all the ways that you intend to grant access to your bucket outside of your account.
- Reset your bucket ACL to the default (full control to the bucket owner).

After you disable ACLs, the following behaviors occur:

- Your bucket accepts only PUT requests that do not specify an ACL or PUT requests with bucket owner full control ACLs. These ACLs include the `bucket-owner-full-control` canned ACL or equivalent forms of this ACL that are expressed in XML.
- Existing applications that support bucket owner full control ACLs see no impact.
- PUT requests that contain other ACLs (for example, custom grants to certain AWS accounts) fail and return an HTTP status code `400 (Bad Request)` with the error code `AccessControlListNotSupported`.

For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**Ensure that your Amazon S3 buckets use the correct policies and are not publicly accessible**

Unless you explicitly require anyone on the internet to be able to read or write to your S3 bucket, make sure that your S3 bucket is not public. The following are some of the steps that you can take to block public access:

- Use S3 Block Public Access. With S3 Block Public Access, you can easily set up centralized controls to limit public access to your Amazon S3 resources. These centralized controls are enforced regardless of how the resources are created. For more information, see Blocking public access to your Amazon S3 storage.
- Identify Amazon S3 bucket policies that allow a wildcard identity such as `"Principal": "*"` (which effectively means "anyone"). Also look for policies that allow a wildcard action `"*"` (which effectively allows the user to perform any action in the Amazon S3 bucket).
- Similarly, look for Amazon S3 bucket access control lists (ACLs) that provide read, write, or full-access to "Everyone" or "Any authenticated AWS user."

- Use the `ListBuckets` API operation to scan all of your Amazon S3 buckets. Then use `GetBucketAcl`, `GetBucketWebsite`, and `GetBucketPolicy` to determine whether each bucket has compliant access controls and a compliant configuration.
- Use AWS Trusted Advisor to inspect your Amazon S3 implementation.
- Consider implementing ongoing detective controls by using the s3-bucket-public-read-prohibited and s3-bucket-public-write-prohibited managed AWS Config Rules.

For more information, see Identity and Access Management for Amazon S3.

**Identify potential threats to your Amazon S3 buckets by using Amazon GuardDuty**

Amazon GuardDuty is a threat detection service that identifies potential threats to your accounts, containers, workloads, and the data within your AWS environment. By using machine learning (ML) models, and anomaly and threat detection capabilities, Amazon GuardDuty continuously monitors different data sources to identify and prioritize potential security risks and malicious activities in your environment. When you enable GuardDuty, it offers threat detection for foundational data sources that includes AWS CloudTrail management events, VPC flow logs, and DNS logs. To extend threat detection to data plane events in S3 buckets, you can enable the GuardDuty S3 Protection feature. This feature detects threats such as data exfiltration and suspicious access to S3 buckets via Tor nodes. GuardDuty also establishes a normal baseline pattern in your environment and when it identifies a potentially anomalous behavior, it provides contextual information to help you remediate the potentially compromised S3 bucket or AWS credentials. For more information, see GuardDuty.

**Implement least privilege access**

When granting permissions, you decide who is getting what permissions to which Amazon S3 resources. You enable specific actions that you want to allow on those resources. Therefore, we recommend that you grant only the permissions that are required to perform a task. Implementing least privilege access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

The following tools are available to implement least privilege access:

- Policy actions for Amazon S3 and Permissions Boundaries for IAM Entities
- How Amazon S3 works with IAM
- Access control list (ACL) overview
- Service Control Policies

For guidance on what to consider when choosing one or more of the preceding mechanisms, see Identity and Access Management for Amazon S3.

**Use IAM roles for applications and AWS services that require Amazon S3 access**

In order for applications running on Amazon EC2 or other AWS services to access Amazon S3 resources, they must include valid AWS credentials in their AWS API requests. We recommend not storing AWS credentials directly in the application or Amazon EC2 instance. These are long-term credentials that are not automatically rotated and could have a significant business impact if they are compromised.

Instead, use an IAM role to manage temporary credentials for applications or services that need to access Amazon S3. When you use a role, you don't have to distribute long-term credentials (such as a username and password or access keys) to an Amazon EC2 instance or AWS service, such as AWS Lambda. The role supplies temporary permissions that applications can use when they make calls to other AWS resources.

For more information, see the following topics in the *IAM User Guide*:

- IAM Roles
- Common Scenarios for Roles: Users, Applications, and Services

**Consider encryption of data at rest**

You have the following options for protecting data at rest in Amazon S3:

- **Server-side encryption** – All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

  Amazon S3 also provides these server-side encryption options:

  - Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
  - Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)
  - Server-side encryption with customer-provided keys (SSE-C)

  For more information, see Protecting data with server-side encryption.

- **Client-side encryption** – Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools. As with server-side encryption, client-side encryption can help reduce risk by encrypting the data with a key that is stored in a different mechanism than the mechanism that stores the data itself.

  Amazon S3 provides multiple client-side encryption options. For more information, see Protecting data by using client-side encryption.

**Enforce encryption of data in transit**

You can use HTTPS (TLS) to help prevent potential attackers from eavesdropping on or manipulating network traffic by using person-in-the-middle or similar attacks. We recommend allowing only encrypted connections over HTTPS (TLS) by using the aws:SecureTransport condition in your Amazon S3 bucket policies.

> **Important**
>
> We recommend that your application not pin Amazon S3 TLS certificates as AWS doesn't support pinning of publicly-trusted certificates. S3 automatically renews certificates and renewal can happen any time before certificate expiry. Renewing a certificate generates a new public-private key pair. If you've pinned an S3 certificate which has been recently renewed with a new public key, you won't be able to connect to S3 until your application uses the new certificate.

Also consider implementing ongoing detective controls by using the s3-bucket-ssl-requests-only managed AWS Config rule.

**Consider using S3 Object Lock**

With S3 Object Lock, you can store objects by using a "Write Once Read Many" (WORM) model. S3 Object Lock can help prevent accidental or inappropriate deletion of data. For example, you can use S3 Object Lock to help protect your AWS CloudTrail logs.

For more information, see Using S3 Object Lock.

**Enable S3 Versioning**

S3 Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in

your bucket. With versioning, you can easily recover from both unintended user actions and application failures.

Also consider implementing ongoing detective controls by using the s3-bucket-versioning-enabled managed AWS Config rule.

For more information, see Using versioning in S3 buckets.

**Consider using S3 Cross-Region Replication**

Although Amazon S3 stores your data across multiple geographically diverse Availability Zones by default, compliance requirements might dictate that you store data at even greater distances. With S3 Cross-Region Replication (CRR), you can replicate data between distant AWS Regions to help satisfy these requirements. CRR enables automatic, asynchronous copying of objects across buckets in different AWS Regions. For more information, see Replicating objects overview.

> **Note**
>
> CRR requires both the source and destination S3 buckets to have versioning enabled.

Also consider implementing ongoing detective controls by using the s3-bucket-replication-enabled managed AWS Config rule.

**Consider using VPC endpoints for Amazon S3 access**

A virtual private cloud (VPC) endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. VPC endpoints can help prevent traffic from traversing the open internet.

VPC endpoints for Amazon S3 provide multiple ways to control access to your Amazon S3 data:

- You can control the requests, users, or groups that are allowed through a specific VPC endpoint by using S3 bucket policies.
- You can control which VPCs or VPC endpoints have access to your S3 buckets by using S3 bucket policies.
- You can help prevent data exfiltration by using a VPC that does not have an internet gateway.

For more information, see Controlling access from VPC endpoints with bucket policies.

### Use managed AWS security services to monitor data security

Several managed AWS security services can help you identify, assess, and monitor security and compliance risks for your Amazon S3 data. These services can also help you protect your data from those risks. These services include automated detection, monitoring, and protection capabilities that are designed to scale from Amazon S3 resources for a single AWS account to resources for organizations spanning thousands of accounts.

For more information, see Monitoring data security with managed AWS security services.

## Amazon S3 monitoring and auditing best practices

The following best practices for Amazon S3 can help detect potential security weaknesses and incidents.

### Identify and audit all of your Amazon S3 buckets

Identification of your IT assets is a crucial aspect of governance and security. You need to have visibility of all your Amazon S3 resources to assess their security posture and take action on potential areas of weakness. To audit your resources, we recommend doing the following:

- Use Tag Editor to identify and tag security-sensitive or audit-sensitive resources, then use those tags when you need to search for these resources. For more information, see Searching for Resources to Tag in the *Tagging AWS Resources User Guide*.
- Use S3 Inventory to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. For more information, see Amazon S3 Inventory.
- Create resource groups for your Amazon S3 resources. For more information, see What are resource groups? in the *AWS Resource Groups User Guide*.

### Implement monitoring by using AWS monitoring tools

Monitoring is an important part of maintaining the reliability, security, availability, and performance of Amazon S3 and your AWS solutions. AWS provides several tools and services to help you monitor Amazon S3 and your other AWS services. For example, you can monitor Amazon CloudWatch metrics for Amazon S3, particularly the `PutRequests`, `GetRequests`, `4xxErrors`, and `DeleteRequests` metrics. For more information, see Monitoring metrics with Amazon CloudWatch and Monitoring Amazon S3.

For a second example, see Example: Amazon S3 Bucket Activity. This example describes how to create a CloudWatch alarm that is triggered when an Amazon S3 API call is made to PUT or DELETE a bucket policy, a bucket lifecycle, or a bucket replication configuration, or to PUT a bucket ACL.

**Enable Amazon S3 server access logging**

Server access logging provides detailed records of the requests that are made to a bucket. Server access logs can assist you in security and access audits, help you learn about your customer base, and understand your Amazon S3 bill. For instructions on enabling server access logging, see Logging requests with server access logging.

Also consider implementing ongoing detective controls by using the s3-bucket-logging-enabled AWS Config managed rule.

**Use AWS CloudTrail**

AWS CloudTrail provides a record of actions taken by a user, a role, or an AWS service in Amazon S3. You can use information collected by CloudTrail to determine the following:

- The request that was made to Amazon S3
- The IP address from which the request was made
- Who made the request
- When the request was made
- Additional details about the request

For example, you can identify CloudTrail entries for PUT actions that affect data access, in particular `PutBucketAcl`, `PutObjectAcl`, `PutBucketPolicy`, and `PutBucketWebsite`.

When you set up your AWS account, CloudTrail is enabled by default. You can view recent events in the CloudTrail console. To create an ongoing record of activity and events for your Amazon S3 buckets, you can create a trail in the CloudTrail console. For more information, see Logging data events in the *AWS CloudTrail User Guide*.

When you create a trail, you can configure CloudTrail to log data events. Data events are records of resource operations performed on or within a resource. In Amazon S3, data events record object-level API activity for individual buckets. CloudTrail supports a subset of Amazon S3 object-level API operations, such as `GetObject`, `DeleteObject`, and `PutObject`. For more information about how CloudTrail works with Amazon S3, see Logging Amazon S3 API calls

using AWS CloudTrail. In the Amazon S3 console, you can also configure your S3 buckets to Enabling CloudTrail event logging for S3 buckets and objects.

AWS Config provides a managed rule (`cloudtrail-s3-dataevents-enabled`) that you can use to confirm that at least one CloudTrail trail is logging data events for your S3 buckets. For more information, see cloudtrail-s3-dataevents-enabled in the *AWS Config Developer Guide*.

**Enable AWS Config**

Several of the best practices listed in this topic suggest creating AWS Config rules. AWS Config helps you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config monitors resource configurations so that you can evaluate the recorded configurations against the desired secure configurations. With AWS Config, you can do the following:

- Review changes in configurations and relationships between AWS resources
- Investigate detailed resource-configuration histories
- Determine your overall compliance against the configurations specified in your internal guidelines

Using AWS Config can help you simplify compliance auditing, security analysis, change management, and operational troubleshooting. For more information, see Setting Up AWS Config with the Console in the *AWS Config Developer Guide*. When specifying the resource types to record, ensure that you include Amazon S3 resources.

> **Important**
>
> AWS Config managed rules only supports general purpose buckets when evaluating Amazon S3 resources. AWS Config doesn't record configuration changes for directory buckets. For more information, see AWS Config Managed Rules and List of AWS Config Managed Rules in the *AWS Config Developer Guide*.

For an example of how to use AWS Config, see How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access on the *AWS Security Blog*.

**Discover sensitive data by using Amazon Macie**

Amazon Macie is a security service that discovers sensitive data by using machine learning and pattern matching. Macie provides visibility into data security risks, and enables automated

protection against those risks. With Macie, you can automate the discovery and reporting of sensitive data in your Amazon S3 data estate to gain a better understanding of the data that your organization stores in S3.

To detect sensitive data with Macie, you can use built-in criteria and techniques that are designed to detect a large and growing list of sensitive data types for many countries and regions. These sensitive data types include multiple types of personally identifiable information (PII), financial data, and credentials data. You can also use custom criteria that you define—regular expressions that define text patterns to match and, optionally, character sequences and proximity rules that refine the results.

If Macie detects sensitive data in an S3 object, Macie generates a security finding to notify you. This finding provides information about the affected object, the types and number of occurrences of the sensitive data that Macie found, and additional details to help you investigate the affected S3 bucket and object. For more information, see the Amazon Macie User Guide.

**Use S3 Storage Lens**

S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

With S3 Storage Lens, you can use metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and access-management best practices, and improve the performance of application workloads.

For example, you can identify buckets that don't have S3 Lifecycle rules to abort incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. For more information, see Understanding Amazon S3 Storage Lens.

**Monitor AWS security advisories**

We recommend that you regularly check the security advisories posted in Trusted Advisor for your AWS account. In particular, look for warnings about Amazon S3 buckets with "open access permissions." You can do this programmatically by using describe-trusted-advisor-checks.

Further, actively monitor the primary email address that's registered to each of your AWS accounts. AWS uses this email address to contact you about emerging security issues that might affect you.

AWS operational issues with broad impact are posted on the AWS Health Dashboard - Service health. Operational issues are also posted to individual accounts through the AWS Health Dashboard. For more information, see the AWS Health documentation.

# Monitoring data security with managed AWS security services

Several managed AWS security services can help you identify, assess, and monitor security and compliance risks for your Amazon S3 data. They can also help you protect your data from those risks. These services include automated detection, monitoring, and protection capabilities that are designed to scale from Amazon S3 resources for a single AWS account to resources for organizations spanning thousands of AWS accounts.

AWS detection and response services can help you identify potential security misconfigurations, threats, or unexpected behaviors, so that you can quickly respond to potentially unauthorized or malicious activity in your environment. AWS data protection services can help you monitor and protect your data, accounts, and workloads from unauthorized access. They can also help you discover sensitive data, such as personally identifiable information (PII), in your Amazon S3 data estate.

To help you identify and evaluate data security and compliance risks, managed AWS security services generate findings to notify you of potential security events or issues with your Amazon S3 data. The findings provide relevant details that you can use to investigate, assess, and act upon these risks according to your incident-response workflows and policies. You can access findings data directly by using each service. You can also send the data to other applications, services, and systems, such as your security incident and event management system (SIEM).

To monitor the security of your Amazon S3 data, consider using these managed AWS security services.

**Amazon GuardDuty**

Amazon GuardDuty is a threat-detection service that continuously monitors your AWS accounts and workloads for malicious activity and delivers detailed security findings for visibility and remediation.

With the S3 protection feature in GuardDuty, you can configure GuardDuty to analyze AWS CloudTrail management and data events for your Amazon S3 resources. GuardDuty then monitors those events for malicious and suspicious activity. To inform the analysis and identify potential security risks, GuardDuty uses threat-intelligence feeds and machine learning.

GuardDuty can monitor different kinds of activity for your Amazon S3 resources. For example, CloudTrail management events for Amazon S3 include bucket-level operations, such as `ListBuckets`, `DeleteBucket`, and `PutBucketReplication`. CloudTrail data events

for Amazon S3 include object-level operations, such as `GetObject`, `ListObjects`, and `PutObject`. If GuardDuty detects anomalous or potentially malicious activity, it generates a finding to notify you.

For more information, see Amazon S3 Protection in Amazon GuardDuty in the *Amazon GuardDuty User Guide*.

**Amazon Detective**

Amazon Detective simplifies the investigative process and helps you conduct faster, more effective security investigations. Detective provides prebuilt data aggregations, summaries, and context that can help you analyze and assess the nature and extent of possible security issues.

Detective automatically extracts time-based events, such as API calls from AWS CloudTrail and Amazon VPC Flow Logs for your AWS resources. It also ingests findings generated by Amazon GuardDuty. Detective then uses machine learning, statistical analysis, and graph theory to generate visualizations that help you conduct effective security investigations more quickly.

These visualizations provide a unified, interactive view of resource behaviors and the interactions between them over time. You can explore this behavior graph to examine potentially malicious actions, such as failed login attempts or suspicious API calls. You can also see how these actions affect resources, such as S3 buckets and objects.

For more information, see the Amazon Detective Administration Guide.

**IAM Access Analyzer**

AWS Identity and Access Management Access Analyzer (IAM Access Analyzer) can help you identify resources that are shared with an external entity. You can also use IAM Access Analyzer to validate IAM policies against policy grammar and best practices, and generate IAM policies based on access activity in your AWS CloudTrail logs.

IAM Access Analyzer uses logic-based reasoning to analyze resource policies in your AWS environment, such as bucket policies. With IAM Access Analyzer for S3, you're alerted when an S3 bucket is configured to allow access to anyone on the internet or other AWS accounts, including accounts outside your organization. For example, IAM Access Analyzer for S3 can report that a bucket has read or write access provided through a bucket access control list (ACL), a bucket policy, a Multi-Region Access Point policy, or an access point policy. For each public or shared bucket, you receive findings that indicate the source and level of public or shared access. With these findings, you can take immediate and precise corrective action to restore bucket access to what you intended.

For more information, see Reviewing bucket access using IAM Access Analyzer for S3.

**Amazon Macie**

Amazon Macie is a data security service that discovers sensitive data by using machine learning and pattern matching, provides visibility into data security risks, and enables automated protection against those risks.

With Macie, you can automate discovery and reporting of sensitive data in your S3 buckets to gain a better understanding of the data that your organization stores in Amazon S3. To detect sensitive data, you can use built-in criteria and techniques that Macie provides, custom criteria that you define, or a combination of the two. If Macie detects sensitive data in an S3 object, Macie generates a finding to notify you. This finding provides information about the affected bucket and object, the types and number of occurrences of the sensitive data that Macie found, and additional details to facilitate your investigation.

Macie also provides statistics and other data that offer insight into the security posture of your Amazon S3 data, and it automatically evaluates and monitors your S3 buckets for security and access control. If Macie detects a potential issue with the security or privacy of your data, such as a bucket that becomes publicly accessible, Macie generates a finding for you to review and remediate as necessary.

For more information, see the Amazon Macie User Guide.

**AWS Security Hub**

AWS Security Hub is a security-posture management service that performs security best-practice checks, aggregates alerts and findings from multiple sources into a single format, and enables automated remediation.

Security Hub collects and provides security findings data from integrated AWS Partner Network security solutions and AWS services, including Amazon Detective, Amazon GuardDuty, IAM Access Analyzer, and Amazon Macie. It also generates its own findings by running continuous, automated security checks based on AWS best practices and supported industry standards.

Security Hub then correlates and consolidates findings across providers to help you prioritize and process the most significant findings. It also provides support for custom actions, which you can use to invoke responses or remediation actions for specific classes of findings.

With Security Hub, you can assess the security and compliance status of your Amazon S3 resources, and you can do so as part of a broader analysis of your organization's security

posture in individual AWS Regions and across multiple Regions. This includes analyzing security trends and identifying the highest-priority security issues. You can also aggregate findings from multiple AWS Regions, and monitor and process aggregated findings data from a single Region.

For more information, see Amazon Simple Storage Service controls in the *AWS Security Hub User Guide*.

# Managing your Amazon S3 storage

After you create buckets and upload objects in Amazon S3, you can manage your object storage using features such as versioning, storage classes, object locking, batch operations, replication, tags, and more. The following sections provide detailed information about the storage management capabilities and features that are available in Amazon S3.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**

# Using versioning in S3 buckets

Versioning in Amazon S3 is a means of keeping multiple variants of an object in the same bucket. You can use the S3 Versioning feature to preserve, retrieve, and restore every version of every object stored in your buckets. With versioning you can recover more easily from both unintended user actions and application failures. After versioning is enabled for a bucket, if Amazon S3 receives multiple write requests for the same object simultaneously, it stores all of those objects.

Versioning-enabled buckets can help you recover objects from accidental deletion or overwrite. For example, if you delete an object, Amazon S3 inserts a delete marker instead of removing the object permanently. The delete marker becomes the current object version. If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version. For more information, see Deleting object versions from a versioning-enabled bucket.

By default, S3 Versioning is disabled on buckets, and you must explicitly enable it. For more information, see Enabling versioning on buckets.

**Note**

- The SOAP API does not support S3 Versioning. SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features are not supported for SOAP.
- Normal Amazon S3 rates apply for every version of an object stored and transferred. Each version of an object is the entire object; it is not just a diff from the previous version. Thus, if you have three versions of an object stored, you are charged for three objects.

## Unversioned, versioning-enabled, and versioning-suspended buckets

Buckets can be in one of three states:

- Unversioned (the default)
- Versioning-enabled
- Versioning-suspended

You enable and suspend versioning at the bucket level. After you version-enable a bucket, it can never return to an unversioned state. But you can *suspend* versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket. When you enable versioning in a bucket, all new objects are versioned and given a unique version ID. Objects that already existed in the bucket at the time versioning was enabled will thereafter *always* be versioned and given a unique version ID when they are modified by future requests. Note the following:

- Objects that are stored in your bucket before you set the versioning state have a version ID of `null`. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests. For more information, see Working with objects in a versioning-enabled bucket.
- The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions. When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. For more information, see Working with objects in a versioning-suspended bucket.

## Using S3 Versioning with S3 Lifecycle

To customize your data retention approach and control storage costs, use object versioning with S3 Lifecycle. For more information, see Managing your storage lifecycle. For information about creating S3 Lifecycle configurations using the AWS Management Console, AWS CLI, AWS SDKs, or the REST API, see Setting a lifecycle configuration on a bucket.

> **Important**
>
> If you have an object expiration lifecycle configuration in your unversioned bucket and you want to maintain the same permanent delete behavior when you enable versioning, you must add a noncurrent expiration configuration. The noncurrent expiration lifecycle configuration manages the deletes of the noncurrent object versions in the versioning-enabled bucket. (A versioning-enabled bucket maintains one current, and zero or more noncurrent, object versions.) For more information, see Setting a lifecycle configuration on a bucket.

For information about working with S3 Versioning, see the following topics.

**Topics**

- How S3 Versioning works

- Enabling versioning on buckets
- Configuring MFA delete
- Working with objects in a versioning-enabled bucket
- Working with objects in a versioning-suspended bucket

## How S3 Versioning works

You can use S3 Versioning to keep multiple versions of an object in one bucket so that you can restore objects that are accidentally deleted or overwritten. For example, if you apply S3 Versioning to a bucket, the following changes occur:

- If you delete an object, instead of removing the object permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can then restore the previous version. For more information, see Deleting object versions from a versioning-enabled bucket.
- If you overwrite an object, Amazon S3 adds a new object version in the bucket. The previous version remains in the bucket and becomes a noncurrent version. You can restore the previous version.

> **Note**
>
> Normal Amazon S3 rates apply for every version of an object that is stored and transferred. Each version of an object is the entire object; it is not a diff from the previous version. Thus, if you have three versions of an object stored, you are charged for three objects.

Each S3 bucket that you create has a *versioning* subresource associated with it. (For more information, see Bucket configuration options.) By default, your bucket is *unversioned*, and the versioning subresource stores the empty versioning configuration, as follows.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

To enable versioning, you can send a request to Amazon S3 with a versioning configuration that includes an `Enabled` status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
  <Status>Enabled</Status>
</VersioningConfiguration>
```

To suspend versioning, you set the status value to Suspended.

> **Note**
>
> When you enable versioning on a bucket for the first time, it might take a short amount of time for the change to be fully propagated. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (PUT or DELETE) on objects in the bucket.

The bucket owner and all authorized AWS Identity and Access Management (IAM) users can enable versioning. The bucket owner is the AWS account that created the bucket. For more information about permissions, see Identity and Access Management for Amazon S3.

For more information about enabling and disabling S3 Versioning by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or REST API, see the section called "Enabling versioning on buckets".

**Topics**

- Version IDs
- Versioning workflows

## Version IDs

If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object that is being stored. For example, in one bucket you can have two objects with the same key (object name) but different version IDs, such as `photo.gif` (version 111111) and `photo.gif` (version 121212).
Diagram depicting a versioning-enabled bucket that has two objects with the same key but different version IDs.

Each object has a version ID, whether or not S3 Versioning is enabled. If S3 Versioning is not enabled, Amazon S3 sets the value of the version ID to `null`. If you enable S3 Versioning, Amazon S3 assigns a version ID value for the object. This value distinguishes that object from other versions of the same key.

When you enable S3 Versioning on an existing bucket, objects that are already stored in the bucket are unchanged. Their version IDs (`null`), contents, and permissions remain the same. After you enable S3 Versioning, each object that is added to the bucket gets a version ID, which distinguishes it from other versions of the same key.

Only Amazon S3 generates version IDs, and they cannot be edited. Version IDs are Unicode, UTF-8 encoded, URL-ready, opaque strings that are no more than 1,024 bytes long. The following is an example:

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> For simplicity, the other examples in this topic use much shorter IDs.

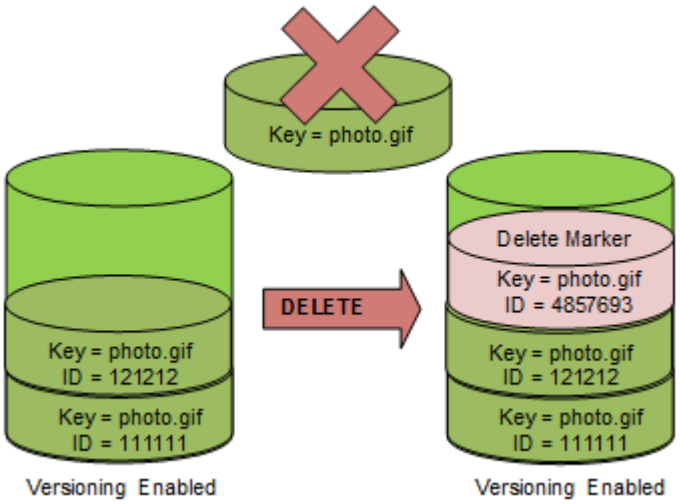
## Versioning workflows

When you PUT an object in a versioning-enabled bucket, the noncurrent version is not overwritten. As shown in the following figure, when a new version of `photo.gif` is PUT into a bucket that already contains an object with the same name, the following behavior occurs:

- The original object (ID = 111111) remains in the bucket.
- Amazon S3 generates a new version ID (121212), and adds this newer version of the object to the bucket.

With this functionality, you can retrieve a previous version of an object if an object has been accidentally overwritten or deleted.

When you DELETE an object, all versions remain in the bucket, and Amazon S3 inserts a delete marker, as shown in the following figure.




The delete marker becomes the current version of the object. By default, GET requests retrieve the most recently stored version. Performing a `GET Object` request when the current version is a delete marker returns a `404 Not Found` error, as shown in the following figure.

However, you can GET a noncurrent version of an object by specifying its version ID. In the following figure, you GET a specific object version, 111111. Amazon S3 returns that object version even though it's not the current version.

For more information, see Retrieving object versions from a versioning-enabled bucket.

You can permanently delete an object by specifying the version that you want to delete. Only the owner of an Amazon S3 bucket or an authorized IAM user can permanently delete a version. If your DELETE operation specifies the `versionId`, that object version is permanently deleted, and Amazon S3 doesn't insert a delete marker.

You can add more security by configuring a bucket to enable multi-factor authentication (MFA) delete. When you enable MFA delete for a bucket, the bucket owner must include two forms of authentication in any request to delete a version or change the versioning state of the bucket. For more information, see Configuring MFA delete.

**When are new versions created for an object?**

New versions of objects are created only when you PUT a new object. Be aware that certain actions, such as `CopyObject`, work by implementing a PUT operation.

Some actions that modify the current object don't create a new version because they don't PUT a new object. This includes actions such as changing the tags on an object.

**Important**

If you notice a significant increase in the number of HTTP 503 (Service Unavailable) responses received for Amazon S3 PUT or DELETE object requests to a bucket that has S3 Versioning enabled, you might have one or more objects in the bucket for which there are millions of versions. For more information, see the S3 Versioning section of Troubleshooting.

## Enabling versioning on buckets

You can use S3 Versioning to keep multiple versions of an object in one bucket. This section provides examples of how to enable versioning on a bucket using the console, REST API, AWS SDKs, and AWS Command Line Interface (AWS CLI).

**Note**

If you enable versioning on a bucket for the first time, it might take up to 15 minutes for the change to be fully propagated. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (PUT or DELETE) on objects in the bucket. Write operations issued before this conversion is complete may apply to unversioned objects.

For more information about S3 Versioning, see Using versioning in S3 buckets. For information about working with objects that are in versioning-enabled buckets, see Working with objects in a versioning-enabled bucket.

To learn more about how to use S3 Versioning to protect data, see Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication.

Each S3 bucket that you create has a *versioning* subresource associated with it. (For more information, see Bucket configuration options.) By default, your bucket is *unversioned*, and the versioning subresource stores the empty versioning configuration, as follows.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

To enable versioning, you can send a request to Amazon S3 with a versioning configuration that includes a status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

To suspend versioning, you set the status value to Suspended.

The bucket owner and all authorized users can enable versioning. The bucket owner is the AWS account that created the bucket (the root account). For more information about permissions, see Identity and Access Management for Amazon S3.

The following sections provide more detail about enabling S3 Versioning using the console, AWS CLI, and the AWS SDKs.

**Using the S3 console**

Follow these steps to use the AWS Management Console to enable versioning on an S3 bucket.

**To enable or disable versioning on an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable versioning for.
3. Choose **Properties**.
4. Under **Bucket Versioning**, choose **Edit**.
5. Choose **Suspend** or **Enable**, and then choose **Save changes**.

> **Note**
>
> You can use AWS multi-factor authentication (MFA) with versioning. When you use MFA with versioning, you must provide your AWS account's access keys and a valid code from the account's MFA device to permanently delete an object version or suspend or reactivate versioning.
>
> To use MFA with versioning, you enable `MFA Delete`. However, you can't enable `MFA Delete` using the AWS Management Console. You must use the AWS Command Line Interface (AWS CLI) or the API. For more information, see Configuring MFA delete.

### Using the AWS CLI

The following example enables versioning on an S3 bucket.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled
```

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

> **Note**
>
> Using MFA delete requires an approved physical or virtual authentication device. For more information about using MFA delete in Amazon S3, see Configuring MFA delete.

For more information about enabling versioning using the AWS CLI, see put-bucket-versioning in the *AWS CLI Command Reference*.

### Using the AWS SDKs

The following examples enable versioning on a bucket and then retrieve versioning status using the AWS SDK for Java and the AWS SDK for .NET. For information about using other AWS SDKs, see the AWS Developer Center.

.NET

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
```

```
namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
 RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
```

```
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response =
 client.PutBucketVersioning(request);
        }


        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
                    BucketName = bucketName
                };

                GetBucketVersioningResponse response =
 client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
}
```

Java

For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
```

```
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest
 =
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);


 s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " +
 conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

Python

The following Python code example creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle that expires noncurrent object versions after 7 days.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
```

```
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
```

```
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

## Configuring MFA delete

When working with S3 Versioning in Amazon S3 buckets, you can optionally add another layer of security by configuring a bucket to enable *MFA (multi-factor authentication) delete*. When you do this, the bucket owner must include two forms of authentication in any request to delete a version or change the versioning state of the bucket.

MFA delete requires additional authentication for either of the following operations:

- Changing the versioning state of your bucket
- Permanently deleting an object version

MFA delete requires two forms of authentication together:

- Your security credentials

- The concatenation of a valid serial number, a space, and the six-digit code displayed on an approved authentication device

MFA delete thus provides added security if, for example, your security credentials are compromised. MFA delete can help prevent accidental bucket deletions by requiring the user who initiates the delete action to prove physical possession of an MFA device with an MFA code and adding an extra layer of friction and security to the delete action.

To identify buckets that have MFA delete enabled, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

The bucket owner, the AWS account that created the bucket (root account), and all authorized users can enable versioning. However, only the bucket owner (root account) can enable MFA delete. For more information, see Securing Access to AWS Using MFA on the AWS Security Blog.

**Note**

To use MFA delete with versioning, you enable `MFA Delete`. However, you cannot enable `MFA Delete` using the AWS Management Console. You must use the AWS Command Line Interface (AWS CLI) or the API.

For examples of using MFA delete with versioning, see the examples section in the topic Enabling versioning on buckets.

You cannot use MFA delete with lifecycle configurations. For more information about lifecycle configurations and how they interact with other configurations, see Lifecycle and other bucket configurations.

To enable or disable MFA delete, you use the same API that you use to configure versioning on a bucket. Amazon S3 stores the MFA delete configuration in the same *versioning* subresource that stores the bucket's versioning status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

To use MFA delete, you can use either a hardware or virtual MFA device to generate an authentication code. The following example shows a generated authentication code displayed on a hardware device.


MFA delete and MFA-protected API access are features intended to provide protection for different scenarios. You configure MFA delete on a bucket to help ensure that the data in your bucket cannot be accidentally deleted. MFA-protected API access is used to enforce another authentication factor (MFA code) when accessing sensitive Amazon S3 resources. You can require any operations against these Amazon S3 resources to be done with temporary credentials created using MFA. For an example, see Requiring MFA.

For more information about how to purchase and activate an authentication device, see Multi-factor authentication.

### To enable S3 Versioning and configure MFA delete

#### Using the AWS CLI

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

#### Using the REST API

For more information about specifying MFA delete using the Amazon S3 REST API, see PutBucketVersioning *Amazon Simple Storage Service API Reference*.

## Working with objects in a versioning-enabled bucket

Objects that are stored in an Amazon S3 bucket before you set the versioning state have a version ID of `null`. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests.

### Transitioning object versions

You can define lifecycle configuration rules for objects that have a well-defined lifecycle to transition object versions to the S3 Glacier Flexible Retrieval storage class at a specific time in the object's lifetime. For more information, see Managing your storage lifecycle.

The topics in this section explain various object operations in a versioning-enabled bucket. For more information about versioning, see Using versioning in S3 buckets.

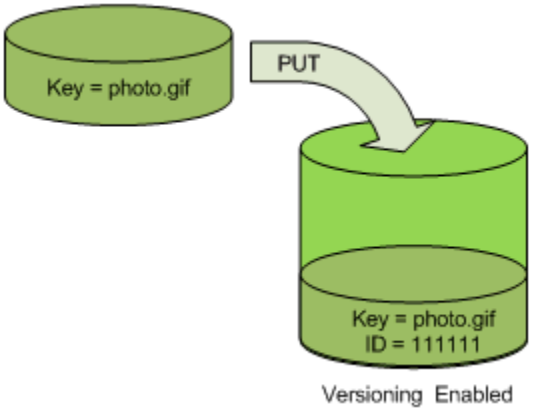
### Topics



## Adding objects to versioning-enabled buckets

After you enable versioning on a bucket, Amazon S3 automatically adds a unique version ID to every object stored (using PUT, POST, or `CopyObject`) in the bucket.

The following figure shows that Amazon S3 adds a unique version ID to an object when it is added to a versioning-enabled bucket.




> **Note**
>
> The version ID values that Amazon S3 assigns are URL safe (can be included as part of a URI).

For more information about versioning, see Using versioning in S3 buckets. You can add object versions to a versioning-enabled bucket using the console, AWS SDKs, and REST API.

**Using the console**

For instructions, see Uploading objects.

**Using the AWS SDKs**

For examples of uploading objects using the AWS SDKs for Java, .NET, and PHP, see Uploading objects. The examples for uploading objects in nonversioned and versioning-enabled buckets are the same, although in the case of versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For information about using other AWS SDKs, see the AWS Developer Center.

**Using the REST API**

**To add objects to versioning-enabled buckets**

1. Enable versioning on a bucket using a `PutBucketVersioning` request.

   For more information, see PutBucketVersioning in the *Amazon Simple Storage Service API Reference*.
2. Send a PUT, POST, or `CopyObject` request to store an object in the bucket.

When you add an object to a versioning-enabled bucket, Amazon S3 returns the version ID of the object in the `x-amz-version-id` response header, as shown in the following example.

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

## Listing objects in a versioning-enabled bucket

This section provides examples of listing object versions from a versioning-enabled bucket. Amazon S3 stores object version information in the *versions* subresource that is associated with the bucket. For more information, see Bucket configuration options. In order to list the objects in a versioning-enabled bucket, you need the `ListBucketVersions` permission.

**Using the S3 console**

Follow these steps to use the Amazon S3 console to see the different versions of an object.

**To see multiple versions of an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. To see a list of the versions of the objects in the bucket, choose the **Show versions** switch.

   For each object version, the console shows a unique version ID, the date and time the object version was created, and other properties. (Objects stored in your bucket before you set the versioning state have a version ID of **null**.)

   To list the objects without the versions, choose the **List versions** switch.

You also can view, download, and delete object versions in the object overview pane on the console. For more information, see Viewing an object overview in the Amazon S3 console.

> **Note**
>
> To access object versions older than 300 versions, you must use the AWS CLI or the object's URL.

> **Important**
>
> You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Using versioning in S3 buckets.

### Using the AWS SDKs

The examples in this section show how to retrieve an object listing from a versioning-enabled bucket. Each request returns up to 1,000 versions, unless you specify a lower number. If the bucket contains more versions than this limit, you send a series of requests to retrieve the list of all versions. This process of returning results in "pages" is called *pagination*.

To show how pagination works, the examples limit each response to two object versions. After retrieving the first page of results, each example checks to determine whether the version list was truncated. If it was, the example continues retrieving pages until all versions have been retrieved.

> **Note**
>
> The following examples also work with a bucket that isn't versioning-enabled, or for objects that don't have individual versions. In those cases, Amazon S3 returns the object listing with a version ID of `null`.

For information about using other AWS SDKs, see the AWS Developer Center.

Java

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
```

```
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
 versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing =
 s3Client.listNextBatchOfVersions(versionListing);
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " +
 numPages + " pages");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.

                    // For this example we limit response to return list of 2
 versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
 s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
```

```
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

### Using the REST API

### Example — Listing all object versions in a bucket

To list all the versions of all the objects in a bucket, you use the `versions` subresource in a `GET Bucket` request. Amazon S3 can retrieve a maximum of 1,000 objects, and each object version counts fully as an object. Therefore, if a bucket contains two keys (for example, `photo.gif` and `picture.jpg`), and the first key has 990 versions and the second key has 400 versions, a single request would retrieve all 990 versions of `photo.gif` and only the most recent 10 versions of `picture.jpg`.

Amazon S3 returns object versions in the order in which they were stored, with the most recently stored returned first.

In a `GET Bucket` request, include the `versions` subresource.

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

**Example — Retrieving all versions of a key**

To retrieve a subset of object versions, you use the request parameters for `GET Bucket`. For more information, see GET Bucket.

1. Set the `prefix` parameter to the key of the object that you want to retrieve.
2. Send a `GET Bucket` request using the `versions` subresource and `prefix`.

   ```
   GET /?versions&prefix=objectName HTTP/1.1
   ```

**Example — Retrieving objects using a prefix**

The following example retrieves objects whose key is or begins with `myObject`.

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

You can use the other request parameters to retrieve a subset of all versions of the object. For more information, see GET Bucket in the *Amazon Simple Storage Service API Reference*.

**Example — Retrieving a listing of additional objects if the response is truncated**

If the number of objects that could be returned in a `GET` request exceeds the value of `max-keys`, the response contains `<isTruncated>true</isTruncated>`, and includes the first key (in `NextKeyMarker`) and the first version ID (in `NextVersionIdMarker`) that satisfy the request, but were not returned. You use those returned values as the starting position in a subsequent request to retrieve the additional objects that satisfy the `GET` request.

Use the following process to retrieve additional objects that satisfy the original `GET Bucket versions` request from a bucket. For more information about `key-marker`, `version-id-marker`, `NextKeyMarker`, and `NextVersionIdMarker`, see GET Bucket in the *Amazon Simple Storage Service API Reference*.

The following are additional responses that satisfy the original `GET` request:

- Set the value of `key-marker` to the key returned in `NextKeyMarker` in the previous response.
- Set the value of `version-id-marker` to the version ID returned in `NextVersionIdMarker` in the previous response.
- Send a `GET Bucket versions` request using `key-marker` and `version-id-marker`.

**Example — Retrieving objects starting with a specified key and version ID**

```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

**Using the AWS CLI**

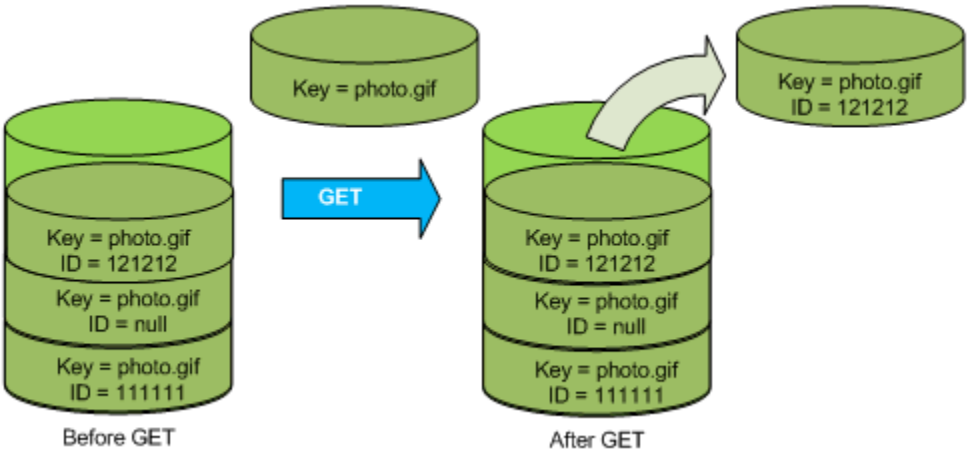
The following command returns metadata about all versions of the objects in a bucket.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket1
```

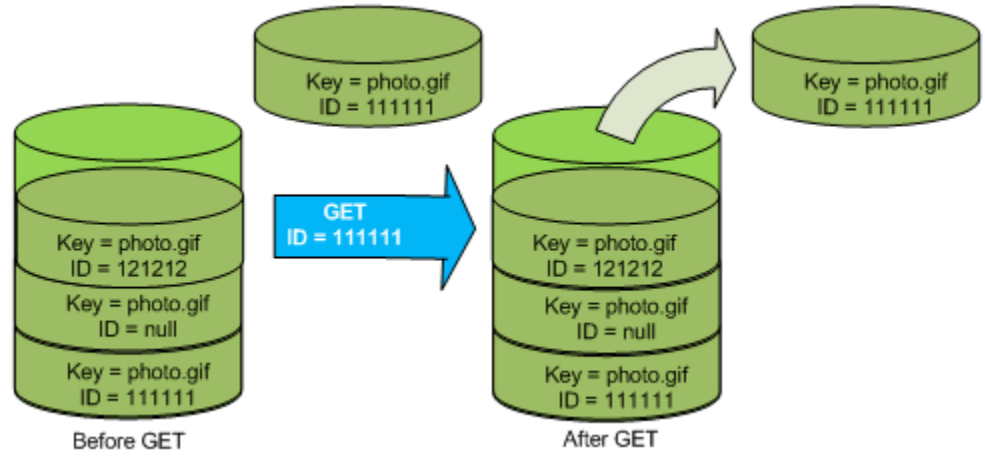
For more information about `list-object-versions` see list-object-versions in the *AWS CLI Command Reference*.

## Retrieving object versions from a versioning-enabled bucket

Versioning in Amazon S3 is a way of keeping multiple variants of an object in the same bucket. A simple `GET` request retrieves the current version of an object. The following figure shows how `GET` returns the current version of the object, `photo.gif`.




To retrieve a specific version, you have to specify its version ID. The following figure shows that a `GET versionId` request retrieves the specified version of the object (not necessarily the current one).




You can retrieve object versions in Amazon S3 using the console, AWS SDKs, or REST API.

> **Note**
>
> To access object versions older than 300 versions, you must use the AWS CLI or the object's URL.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to retrieve.
6. Choose **Actions**, choose **Download**, and save the object.

You also can view, download, and delete object versions in the object overview panel. For more information, see Viewing an object overview in the Amazon S3 console.

> **Important**
>
> You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Using versioning in S3 buckets.

#### Using the AWS SDKs

The examples for uploading objects in nonversioned and versioning-enabled buckets are the same. However, for versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For examples of downloading objects using AWS SDKs for Java, .NET, and PHP, see Downloading objects.

For examples of listing the version of objects using AWS SDKs for .NET and Rust, see List the version of objects in an Amazon S3 bucket.

#### Using the REST API

##### To retrieve a specific object version

1. Set `versionId` to the ID of the version of the object that you want to retrieve.
2. Send a `GET Object versionId` request.

**Example — Retrieving a versioned object**

The following request retrieves version `L4kqtJlcpXroDTDmpUMLUo` of `my-image.jpg`.

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

You can retrieve just the metadata of an object (not the content). For information, see the section called “Retrieving version metadata”.

For information about restoring a previous object version, see the section called “Restoring previous versions”.

**Retrieving the metadata of an object version**

If you only want to retrieve the metadata of an object (and not its content), you use the `HEAD` operation. By default, you get the metadata of the most recent version. To retrieve the metadata of a specific object version, you specify its version ID.

**To retrieve the metadata of an object version**

1. Set `versionId` to the ID of the version of the object whose metadata you want to retrieve.
2. Send a `HEAD Object versionId` request.

**Example — Retrieving the metadata of a versioned object**

The following request retrieves the metadata of version `3HL4kqCxf3vjVBH40Nrjfkd` of `my-image.jpg`.

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

The following shows a sample response.

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

### Restoring previous versions

You can use versioning to retrieve previous versions of an object. There are two approaches to doing so:

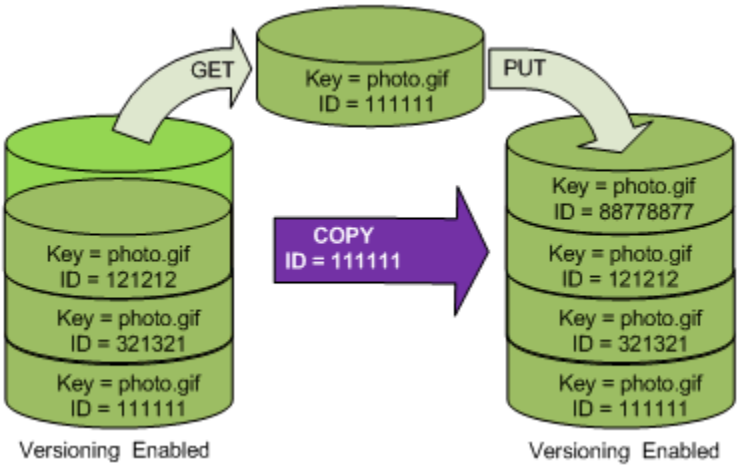
- Copy a previous version of the object into the same bucket.

  The copied object becomes the current version of that object and all object versions are preserved.

- Permanently delete the current version of the object.

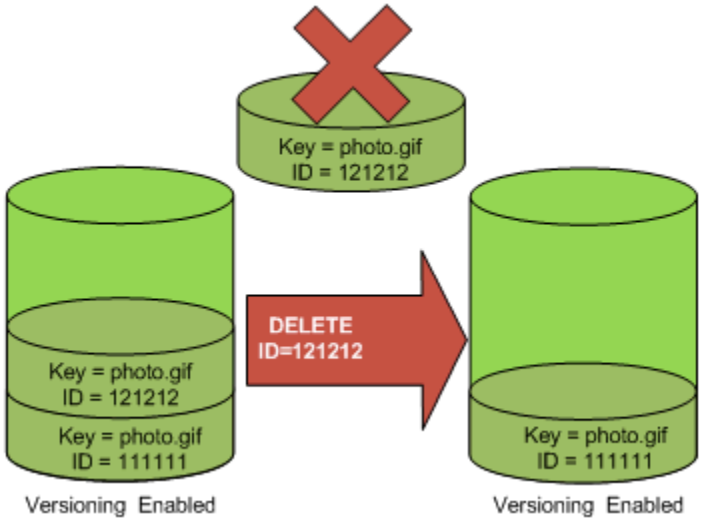
  When you delete the current object version, you, in effect, turn the previous version into the current version of that object.

Because all object versions are preserved, you can make any earlier version the current version by copying a specific version of the object into the same bucket. In the following figure, the source object (ID = 111111) is copied into the same bucket. Amazon S3 supplies a new ID (88778877) and it becomes the current version of the object. So, the bucket has both the original object version (111111) and its copy (88778877). For more information about getting a previous version and then uploading it to make it the current version, see Retrieving object versions from a versioning-enabled bucket and Uploading objects.




A subsequent GET retrieves version 88778877.

The following figure shows how deleting the current version (121212) of an object leaves the previous version (111111) as the current object. For more information about deleting an object, see Deleting a single object.




A subsequent GET retrieves version 111111.

**Note**

To restore object versions in batches, you can use the `CopyObject` operation. The `CopyObject` operation copies each object that is specified in the manifest. However, be aware that objects aren't necessarily copied in the same order as they appear in the manifest. For versioned buckets, if preserving current/non-current version order

is important, you should copy all non-current versions first. Then, after the first job is complete, copy the current versions in a subsequent job.

### To restore previous object versions

#### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to retrieve.
6. Choose **Actions**, choose **Download**, and save the object.

You also can view, download, and delete object versions in the object overview panel. For more information, see Viewing an object overview in the Amazon S3 console.

**Important**

You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Using versioning in S3 buckets.

#### Using the AWS SDKs

For information about using other AWS SDKs, see the AWS Developer Center.

Python

The following Python code example restores a versioned object's previous version by deleting all versions that occurred after the specified rollback version.

```
def rollback_object(bucket, object_key, version_id):
    """
```

```
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

## Deleting object versions from a versioning-enabled bucket

You can delete object versions from Amazon S3 buckets whenever you want. You can also define lifecycle configuration rules for objects that have a well-defined lifecycle to request Amazon S3 to expire current object versions or permanently remove noncurrent object versions. When your bucket has versioning enabled or the versioning is suspended, the lifecycle configuration actions work as follows:

- The `Expiration` action applies to the current object version. Instead of deleting the current object version, Amazon S3 retains the current version as a noncurrent version by adding a *delete marker*, which then becomes the current version.
- The `NoncurrentVersionExpiration` action applies to noncurrent object versions, and Amazon S3 permanently removes these object versions. You cannot recover permanently removed objects.

For more information about S3 Lifecycle, see Managing your storage lifecycle and Examples of S3 Lifecycle configuration.

To see how many current and noncurrent object versions that your buckets have, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to optimize your storage costs. For a complete list of metrics, see S3 Storage Lens metrics glossary.

> **Note**
>
> Normal Amazon S3 rates apply for every version of an object that is stored and transferred, including noncurrent object versions. For more information, see Amazon S3 pricing.

### Delete request use cases

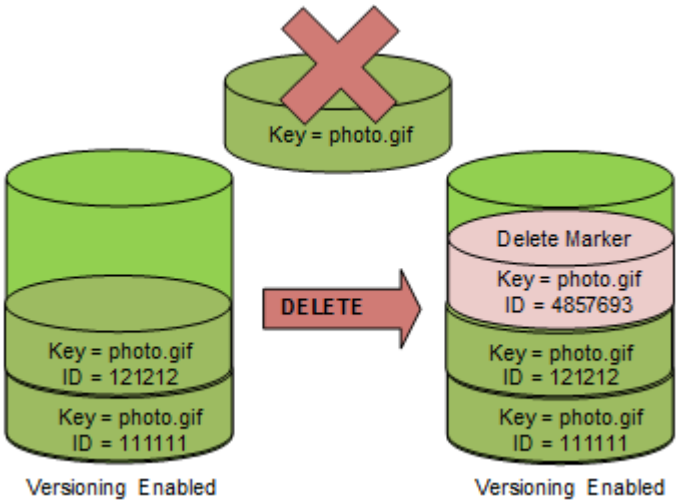
A `DELETE` request has the following use cases:

- When versioning is enabled, a simple `DELETE` cannot permanently delete an object. (A simple `DELETE` request is a request that doesn't specify a version ID.) Instead, Amazon S3 inserts a

delete marker in the bucket, and that marker becomes the current version of the object with a new ID.

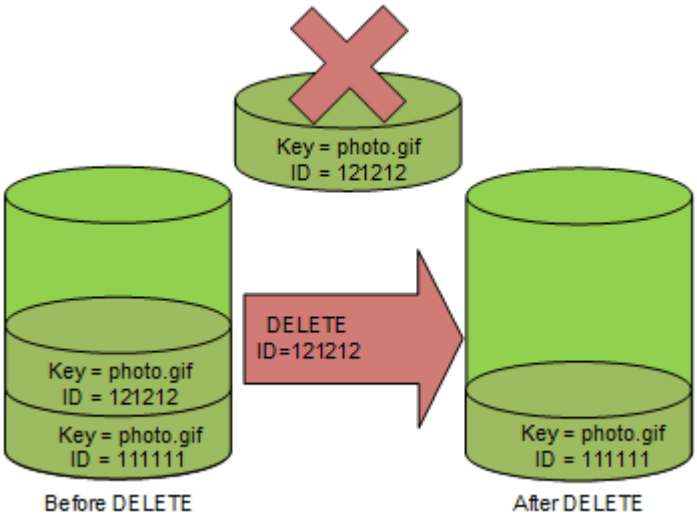
When you try to GET an object whose current version is a delete marker, Amazon S3 behaves as though the object has been deleted (even though it has not been erased) and returns a 404 error. For more information, see Working with delete markers.

The following figure shows that a simple DELETE does not actually remove the specified object. Instead, Amazon S3 inserts a delete marker.




- To delete versioned objects permanently, you must use `DELETE Object versionId`.

  The following figure shows that deleting a specified object version permanently removes that object.



### To delete object versions

You can delete object versions in Amazon S3 using the console, AWS SDKs, the REST API, or the AWS Command Line Interface.

#### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to permanently delete.
6. Choose **Delete**.
7. In **Permanently delete objects?**, enter **`permanently delete`**.

   > **Warning**
   >
   > When you permanently delete an object version, the action cannot be undone.

8. Choose **Delete objects**.

   Amazon S3 deletes the object version.

#### Using the AWS SDKs

For examples of deleting objects using the AWS SDKs for Java, .NET, and PHP, see Deleting Amazon S3 objects. The examples for deleting objects in nonversioned and versioning-enabled buckets are the same. However, for versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For information about using other AWS SDKs, see the AWS Developer Center.

Python

The following Python code example permanently deletes a versioned object by deleting all of its versions.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

### Using the REST API

### To delete a specific version of an object

- In a DELETE, specify a version ID.

### Example — Deleting a specific version

The following example deletes version `UIORUnfnd89493jJFJ` of `photo.gif`.

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

### Using the AWS CLI

The following command deletes an object named test.txt from a bucket named *amzn-s3-demo-bucket1*. To remove a specific version of an object, you must be the bucket owner and you must use the version Id subresource.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket1 --key test.txt --version-
id versionID
```

For more information about `delete-object` see delete-object in the *AWS CLI Command Reference*.

For more information about deleting object versions, see the following topics:

- Working with delete markers
- Removing delete markers to make an older version current
- Deleting an object from an MFA delete-enabled bucket

### Working with delete markers

A *delete marker* in Amazon S3 is a placeholder (or marker) for a versioned object that was specified in a simple `DELETE` request. A simple `DELETE` request is a request that doesn't specify a version ID. Because the object is in a versioning-enabled bucket, the object is not deleted. But the delete marker makes Amazon S3 behave as if the object is deleted. You can use an Amazon S3 API `DELETE` call on a delete marker. To do this, you must make the `DELETE` request by using an AWS Identity and Access Management (IAM) user or role with the appropriate permissions.

A delete marker has a *key name* (or *key*) and version ID like any other object. However, a delete marker differs from other objects in the following ways:

- A delete marker doesn't have data associated with it.
- A delete marker isn't associated with an access control list (ACL) value.
- If you issue a `GET` request for a delete marker, the `GET` request doesn't retrieve anything because a delete marker has no data. Specifically, when your `GET` request doesn't specify a `versionId`, you get a 404 (Not Found) error.

Delete markers accrue a minimal charge for storage in Amazon S3. The storage size of a delete marker is equal to the size of the key name of the delete marker. A key name is a sequence of

Unicode characters. The UTF-8 encoding for the key name adds 1-4 bytes of storage to your bucket for each character in the name. Delete markers are stored in the S3 Standard storage class.

If you want to find out how many delete markers you have and what storage class they're stored in, you can use Amazon S3 Storage Lens. For more information, see Assessing your storage activity and usage with Amazon S3 Storage Lens and Amazon S3 Storage Lens metrics glossary.

For more information about key names, see Creating object key names. For information about deleting a delete marker, see Managing delete markers.

Only Amazon S3 can create a delete marker, and it does so whenever you send a `DeleteObject` request on an object in a versioning-enabled or suspended bucket. The object specified in the `DELETE` request is not actually deleted. Instead, the delete marker becomes the current version of the object. The object's key name (or key) becomes the key of the delete marker.

When you get an object without specifying a `versionId` in your request, if its current version is a delete marker, Amazon S3 responds with the following:

- A 404 (Not Found) error
- A response header, `x-amz-delete-marker: true`

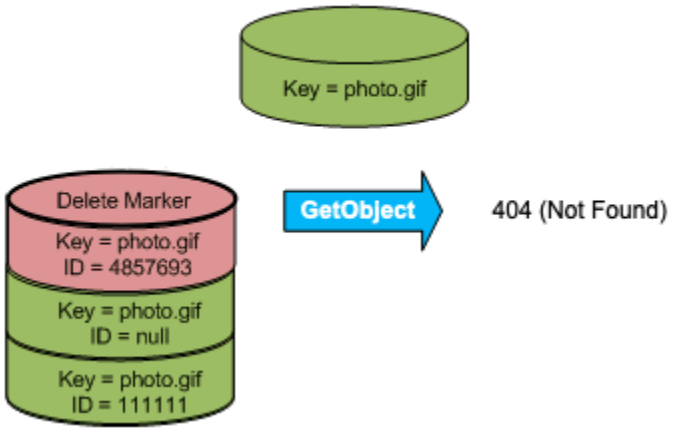
When you get an object by specifying a `versionId` in your request, if the specified version is a delete marker, Amazon S3 responds with the following:

- A 405 (Method Not Allowed) error
- A response header, `x-amz-delete-marker: true`
- A response header, `Last-Modified: timestamp` (only when using the HeadObject or GetObject API operations)

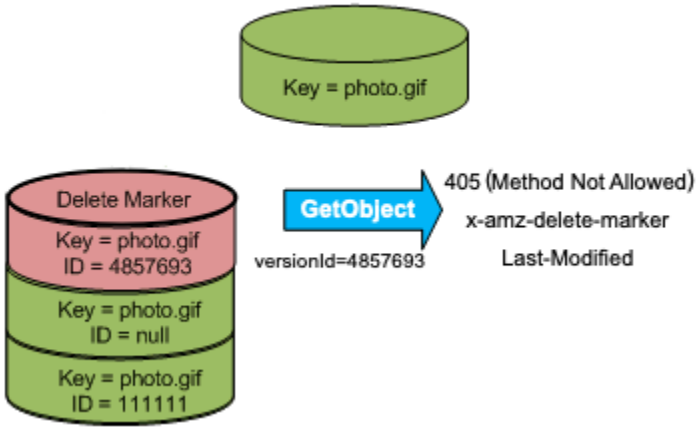
The `x-amz-delete-marker: true` response header tells you that the object accessed was a delete marker. This response header never returns `false`, because when the value is `false`, the current or specified version of the object is not a delete marker.

The `Last-Modified` response header provides the creation time of the delete markers.

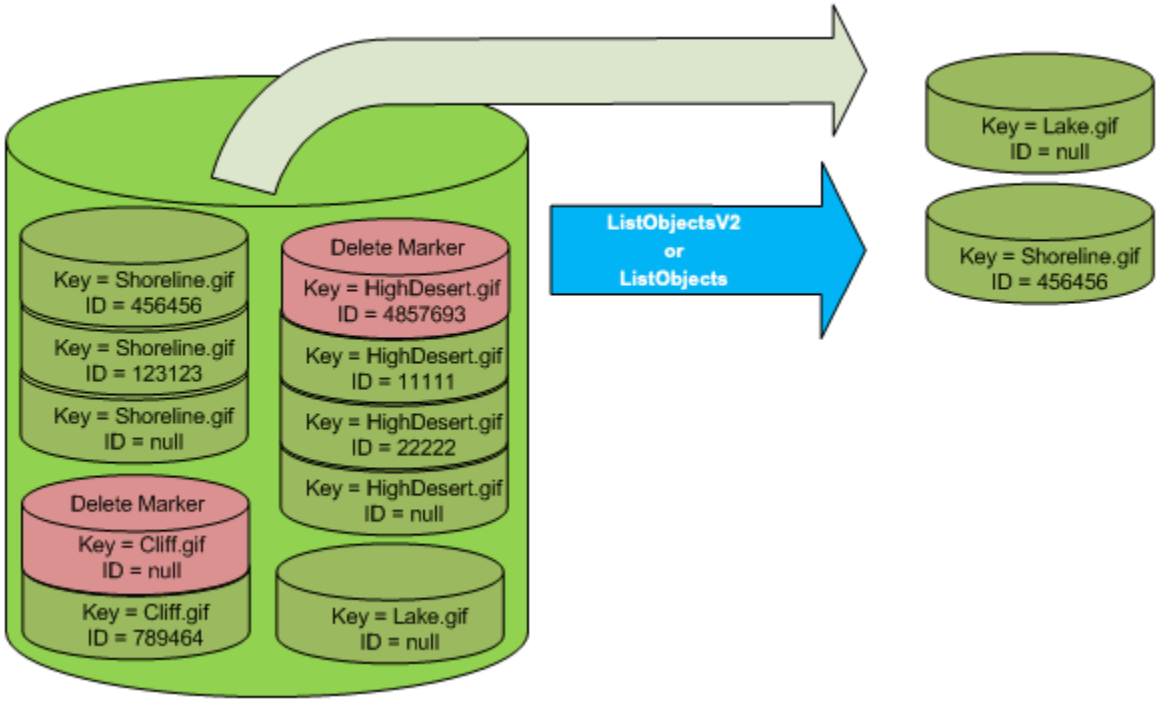
The following figure shows how a `GetObject` API call on an object whose current version is a delete marker responds with a 404 (Not Found) error and the response header includes `x-amz-delete-marker: true`.




If you make a `GetObject` call on an object by specifying a `versionId` in your request, and if the specified version is a delete marker, Amazon S3 responds with a 405 (Method Not Allowed) error and the response headers include `x-amz-delete-marker: true` and `Last-Modified: timestamp`.




The only way to list delete markers (and other versions of an object) is by using the `versions` subresource in a ListObjectVersions request. The following figure shows that a ListObjectsV2 or ListObjects request doesn't return objects whose current version is a delete marker.




### Managing delete markers

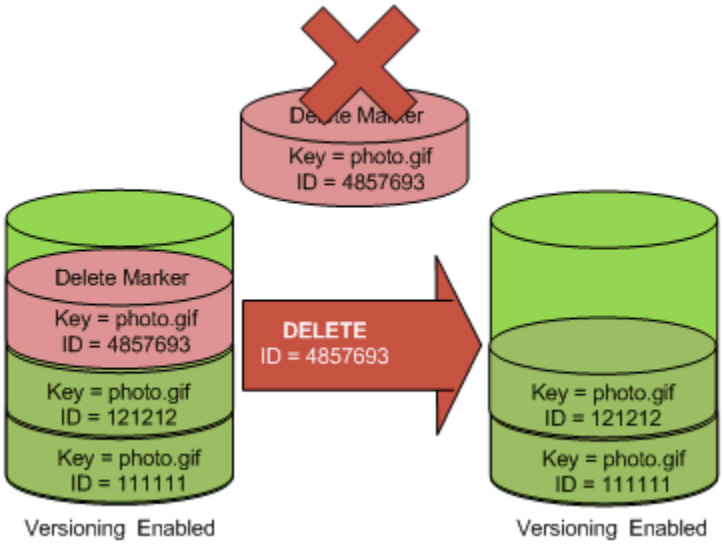
#### Configuring lifecycle to clean up expired delete markers automatically

An expired object delete marker is one where all object versions are deleted and only a single delete marker remains. If the lifecycle configuration is set to delete current versions, or the `ExpiredObjectDeleteMarker` action is explicitly set, Amazon S3 removes the expired object's delete marker. For an example, see Example 7: Removing expired object delete markers.

#### Removing delete markers to make an older version current

When you delete an object in a versioning-enabled bucket, all versions remain in the bucket, and Amazon S3 creates a delete marker for the object. To undelete the object, you must delete this delete marker. For more information about versioning and delete markers, see Using versioning in S3 buckets.

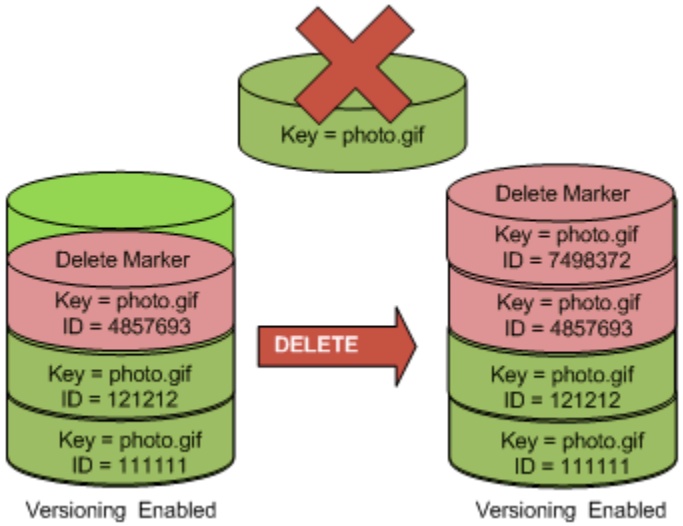
To delete a delete marker permanently, you must include its version ID in a `DeleteObject versionId` request. The following figure shows how a `DeleteObject versionId` request permanently removes a delete marker.




The effect of removing the delete marker is that a simple `GET` request will now retrieve the current version ID (121212) of the object.

> **Note**
>
> If you use a `DeleteObject` request where the current version is a delete marker (without specifying the version ID of the delete marker), Amazon S3 does not delete the delete marker, but instead `PUT`s another delete marker.

To delete a delete marker with a `NULL` version ID, you must pass the `NULL` as the version ID in the `DeleteObject` request. The following figure shows how a simple `DeleteObject` request made without a version ID where the current version is a delete marker, removes nothing, but instead adds an additional delete marker with a unique version ID (7498372).




### Using the S3 console

Use the following steps to recover deleted objects that are not folders from your S3 bucket, including objects that are within those folders.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want.
3. To see a list of the **versions** of the objects in the bucket, choose the **List versions** switch. You'll be able to see the delete markers for deleted objects.
4. To undelete an object, you must delete the delete marker. Select the check box next to the **delete marker** of the object to recover, and then choose **Delete**.
5. Confirm the deletion on the **Delete objects** page.
   a. For **Permanently delete objects?** enter **`permanently delete`**.
   b. Choose **Delete objects**.

> **Note**
>
> You can't use the Amazon S3 console to undelete folders. You must use the AWS CLI or SDK. For examples, see How can I retrieve an Amazon S3 object that was deleted in a versioning-enabled bucket? in the AWS Knowledge Center.

### Using the REST API

#### To permanently remove a delete marker

1. Set `versionId` to the ID of the version to the delete marker you want to remove.
2. Send a `DELETE Object versionId` request.

#### Example — Removing a delete marker

The following example removes the delete marker for `photo.gif` version 4857693.

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

When you delete a delete marker, Amazon S3 includes the following in the response.

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

### Using the AWS SDKs

For information about using other AWS SDKs, see the AWSDeveloper Center.

Python

The following Python code example demonstrates how to remove a delete marker from an object and thus makes the most recent non-current version, the current version of the object.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.
```

```
    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

### Deleting an object from an MFA delete-enabled bucket

If a bucket's versioning configuration is MFA delete enabled, the bucket owner must include the `x-amz-mfa` request header in requests to permanently delete an object version or change the versioning state of the bucket. Requests that include `x-amz-mfa` must use HTTPS.

The header's value is the concatenation of your authentication device's serial number, a space, and the authentication code displayed on it. If you don't include this request header, the request fails.

For more information about authentication devices, see Multi-factor Authentication.

**Example — Deleting an object from an MFA delete-enabled bucket**

The following example deletes `my-image.jpg` (with the specified version), which is in a bucket configured with MFA delete enabled.

Note the space between *[SerialNumber]* and *[AuthenticationCode]*. For more information, see DeleteObject in the *Amazon Simple Storage Service API Reference*.

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

For more information about enabling MFA delete, see Configuring MFA delete.

## Configuring versioned object permissions

Permissions for objects in Amazon S3 are set at the version level. Each version has its own object owner. The AWS account that creates the object version is the owner. So, you can set different permissions for different versions of the same object. To do so, you must specify the version ID of the object whose permissions you want to set in a `PUT Object versionId acl` request. For a detailed description and instructions on using ACLs, see Identity and Access Management for Amazon S3.

**Example — Setting permissions for an object version**

The following request sets the permission of the grantee, `BucketOwner@amazon.com`, to `FULL_CONTROL` on the key, `my-image.jpg`, version ID, `3HL4kqtJvjVBH40Nrjfkd`.

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
```

```
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>BucketOwner@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
```

Likewise, to get the permissions of a specific object version, you must specify its version ID in a `GET Object versionId acl` request. You need to include the version ID because, by default, `GET Object acl` returns the permissions of the current version of the object.

**Example — Retrieving the permissions for a specified object version**

In the following example, Amazon S3 returns the permissions for the key, `my-image.jpg`, version ID, `DVBH40Nr8X8gUMLUo`.

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

For more information, see GetObjectAcl in the *Amazon Simple Storage Service API Reference*.

## Working with objects in a versioning-suspended bucket

In Amazon S3, you can suspend versioning to stop accruing new versions of the same object in a bucket. You might do this because you only want a single version of an object in a bucket. Or, you might not want to accrue charges for multiple versions.

When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. The topics in this section explain various object operations in a versioning-suspended bucket, including adding, retrieving, and deleting objects.

For more information about S3 Versioning, see Using versioning in S3 buckets. For more information about retrieving object versions, see Retrieving object versions from a versioning-enabled bucket.

**Topics**

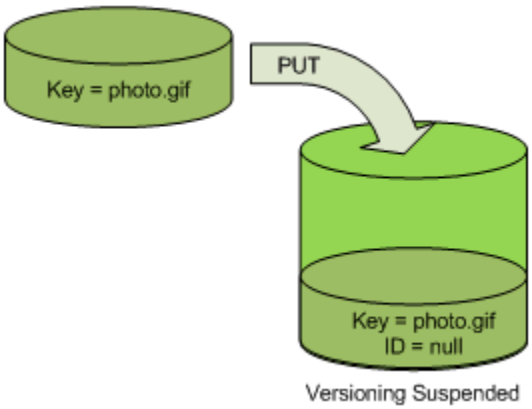
- Adding objects to versioning-suspended buckets
- Retrieving objects from versioning-suspended buckets
- Deleting objects from versioning-suspended buckets

## Adding objects to versioning-suspended buckets

You can add objects to versioning-suspended buckets in Amazon S3 to create the object with a null version ID or overwrite any object version with a matching version ID.

After you suspend versioning on a bucket, Amazon S3 automatically adds a `null` version ID to every subsequent object stored thereafter (using `PUT`, `POST`, or `CopyObject`) in that bucket.

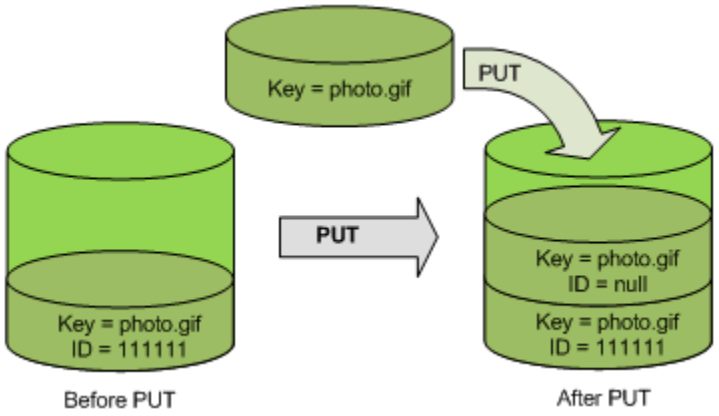
The following figure shows how Amazon S3 adds the version ID of `null` to an object when it is added to a version-suspended bucket.



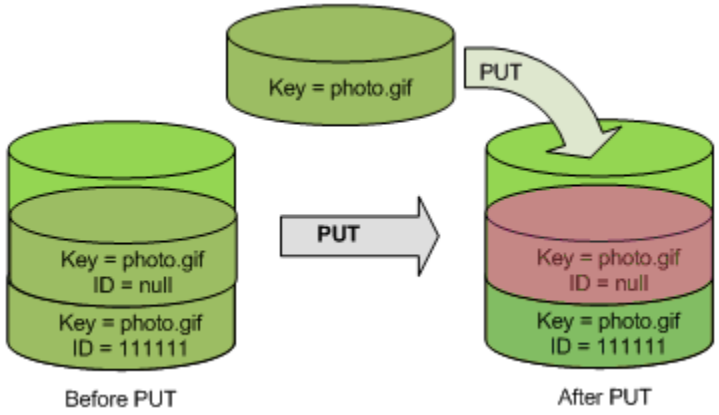

If a null version is already in the bucket and you add another object with the same key, the added object overwrites the original null version.

If there are versioned objects in the bucket, the version you PUT becomes the current version of the object. The following figure shows how adding an object to a bucket that contains versioned objects does not overwrite the object already in the bucket.

In this case, version 111111 was already in the bucket. Amazon S3 attaches a version ID of null to the object being added and stores it in the bucket. Version 111111 is not overwritten.



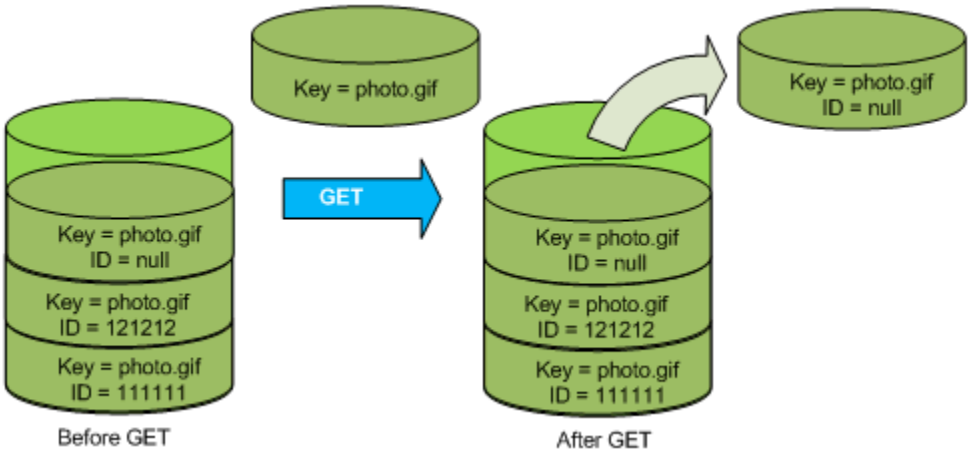

If a null version already exists in a bucket, the null version is overwritten, as shown in the following figure.




Although the key and version ID (`null`) of the null version are the same before and after the PUT, the contents of the null version originally stored in the bucket are replaced by the contents of the object PUT into the bucket.

## Retrieving objects from versioning-suspended buckets

A `GET Object` request returns the current version of an object whether you've enabled versioning on a bucket or not. The following figure shows how a simple GET returns the current version of an object.



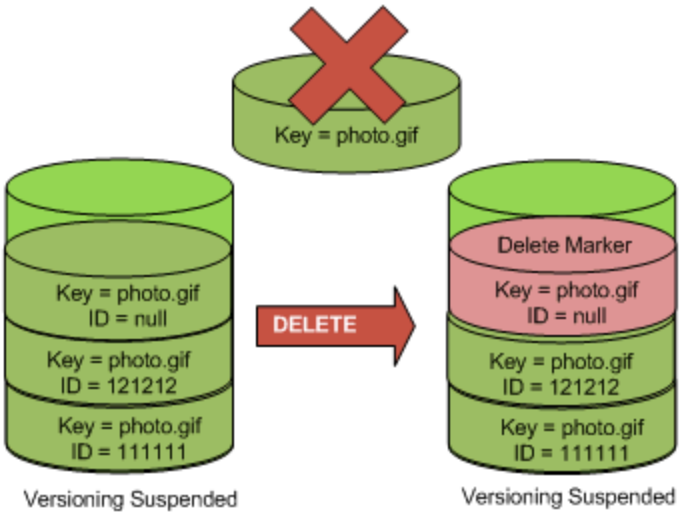

## Deleting objects from versioning-suspended buckets

You can delete objects from versioning-suspended buckets to remove an object with a null version ID.

If versioning is suspended for a bucket, a `DELETE` request:

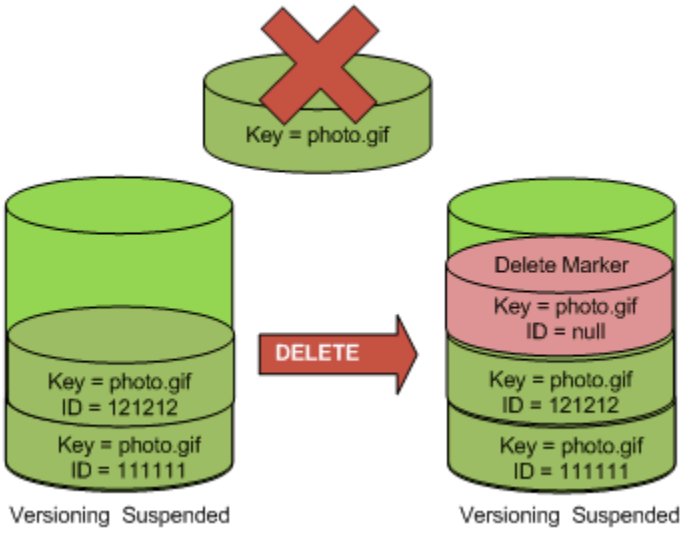
- Can only remove an object whose version ID is `null`.
- Doesn't remove anything if there isn't a null version of the object in the bucket.
- Inserts a delete marker into the bucket.

The following figure shows how a simple `DELETE` removes a null version. (A simple `DELETE` request is a request that doesn't specify a version ID.) Amazon S3 inserts a delete marker in its place with a version ID of `null`.

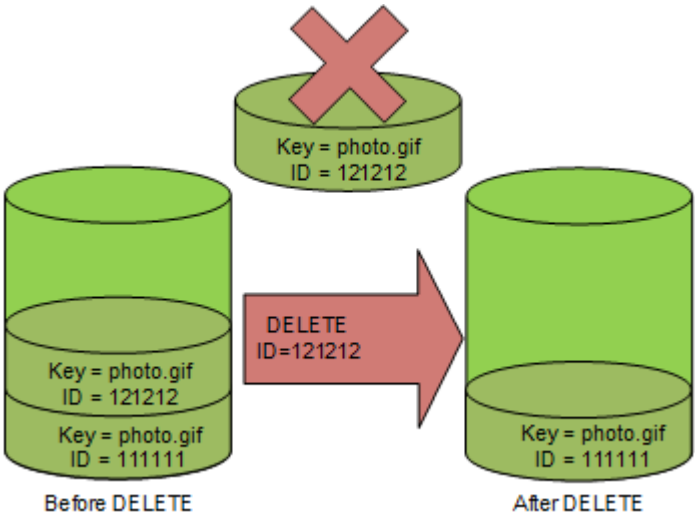


Remember that a delete marker doesn't have content, so you lose the content of the null version when a delete marker replaces it.

The following figure shows a bucket that doesn't have a null version. In this case, the DELETE removes nothing; Amazon S3 just inserts a delete marker.




Even in a versioning-suspended bucket, the bucket owner can permanently delete a specified version by including the version ID in the DELETE request. The following figure shows that deleting a specified object version permanently removes that version of the object. Only the bucket owner can delete a specified object version.



# Using AWS Backup for Amazon S3

Amazon S3 is natively integrated with AWS Backup, a fully managed, policy-based service that you can use to centrally define backup policies to protect your data in Amazon S3. After you define your backup policies and assign Amazon S3 resources to the policies, AWS Backup automates the creation of Amazon S3 backups and securely stores the backups in an encrypted backup vault that you designate in your backup plan.

When using AWS Backup for Amazon S3, you can perform the following actions:

- Create continuous backups and periodic backups. Continuous backups are useful for point-in-time restore, and periodic backups are useful to meet your long-term data-retention needs.
- Automate backup scheduling and retention by centrally configuring backup policies.
- Restore backups of Amazon S3 data to a point in time that you specify.

Along with AWS Backup, you can use S3 Versioning and S3 Replication to help recover from accidental deletions and perform your own self-recovery operations.

**Prerequisites**

You must activate S3 Versioning on your bucket before AWS Backup can back it up.

> **Note**
>
> We recommend that you set a lifecycle expiration rule for versioning-enabled buckets that are being backed up. If you do not set a lifecycle expiration period, your Amazon S3 storage costs might increase because AWS Backup retains all versions of your Amazon S3 data.

**Getting started**

To get started with AWS Backup for Amazon S3, see Creating Amazon S3 backups in the *AWS Backup Developer Guide*.

**Restrictions and limitations**

To learn about the limitations, see Creating Amazon S3 backups in the *AWS Backup Developer Guide*.

# Working with archived objects

To reduce your storage costs for infrequently accessed objects, you can *archive* those objects. When you archive an object, it is moved into low-cost storage, which means that you can't access it in real time.

Although archived objects are not accessible in real time, you can restore them in minutes or hours, depending on the storage class. You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the REST API, the AWS SDKs, and the AWS Command Line Interface (AWS CLI). For instructions, see Restoring an archived object.

Amazon S3 objects in the following storage classes or tiers are archived and are not accessible in real time:

- The S3 Glacier Flexible Retrieval storage class
- The S3 Glacier Deep Archive storage class
- The S3 Intelligent-Tiering Archive Access tier
- The S3 Intelligent-Tiering Deep Archive Access tier

To restore archived objects, you must do the following:

- For objects in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, you must initiate the restore request and wait until a temporary copy of the object is available. When a temporary copy of the restored object is created, the object's storage class remains the same. (A HeadObject or GetObject API operation request returns S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive as the storage class.)
- For objects in the S3 Intelligent-Tiering Archive Access and S3 Intelligent-Tiering Deep Archive Access tiers, you must initiate the restore request and wait until the object is moved into the Frequent Access tier.

For more information about how all Amazon S3 storage classes compare, see Using Amazon S3 storage classes. For more information about S3 Intelligent-Tiering, see the section called “How S3 Intelligent-Tiering works”.

## Restoring objects from S3 Glacier

When you use S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 restores a temporary copy of the object only for the specified duration. After that, it deletes the restored object copy. You can modify the expiration period of a restored copy by reissuing a restore request. In this case, Amazon S3 updates the expiration period relative to the current time.

> **Note**
>
> When you restore an archived object from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, you pay for both the archived object and the copy that you restored temporarily. For information about pricing, see Amazon S3 pricing.

## Restoring objects from S3 Intelligent-Tiering

When you restore an object from the S3 Intelligent-Tiering Archive Access tier or S3 Intelligent-Tiering Deep Archive Access tier, the object moves back into the S3 Intelligent-Tiering Frequent Access tier. If the object is not accessed after 30 consecutive days, it automatically moves into the Infrequent Access tier. After a minimum of 90 consecutive days of no access, the object moves into the S3 Intelligent-Tiering Archive Access tier. If the object is not accessed after a minimum of 180 consecutive days, the object moves into the Deep Archive Access tier.

> **Note**
>
> Unlike in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, restore requests for S3 Intelligent-Tiering objects don't accept the `Days` value.

## Using S3 Batch Operations with restore requests

To restore more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

## Restore time

Amazon S3 calculates the expiration time of the restored object copy by adding the number of days specified in the restoration request to the time when the requested restoration is completed. Amazon S3 then rounds the resulting time to the next day at midnight Universal Coordinated Time (UTC). For example, suppose that a restored object copy was created on October 15, 2012, at 10:30 AM UTC, and the restoration period was specified as 3 days. In this case, the restored copy expires on October 19, 2012, at 00:00 UTC, at which time Amazon S3 deletes the object copy.

The time it takes a restore job to finish depends on which archive storage class or storage tier you use and which retrieval option you specify: Expedited (only available for S3 Glacier Flexible Retrieval and S3 Intelligent-Tiering Archive Access), Standard, or Bulk. For more information, see Archive retrieval options.

You can be notified when your restore is complete by using Amazon S3 Event Notifications. For more information, see Amazon S3 Event Notifications.

**Topics**

- Archive retrieval options
- Restoring an archived object

## Archive retrieval options

The following are the available retrieval options when restoring an archived object in Amazon S3:

- **Expedited** – Quickly access your data that is stored in the S3 Glacier Flexible Retrieval storage class or S3 Intelligent-Tiering Archive Access tier. You can use this option when occasional urgent requests for a subset of archives are required. For all but the largest archived objects (250 MB+), data that is accessed by using expedited retrievals is typically made available within 1–5 minutes.

> **Note**
>
> Expedited retrievals are a premium feature and are charged at the Expedited request and retrieval rate.
> For information about Amazon S3 pricing, see Amazon S3 Pricing.

Provisioned capacity helps ensure that retrieval capacity for expedited retrievals from S3 Glacier Flexible Retrieval is available when you need it. For more information, see Provisioned capacity.

- **Standard** – Access any of your archived objects within several hours. Standard is the default option for retrieval requests that do not specify the retrieval option. Standard retrievals typically finish within 3–5 hours for objects stored in the S3 Glacier Flexible Retrieval storage class or S3 Intelligent-Tiering Archive Access tier. These retrievals typically finish within 12 hours for objects stored in the S3 Glacier Deep Archive storage class or S3 Intelligent-Tiering Deep Archive Access tier. Standard retrievals are free for objects that are stored in S3 Intelligent-Tiering.

  > **Note**
  >
  > - For objects stored in the S3 Glacier Flexible Retrieval storage class or the S3 Intelligent-Tiering Archive Access tier, Standard retrievals initiated by using the S3 Batch Operations restore operation typically start within minutes and finish within 3-5 hours.
  > - For objects in the S3 Glacier Deep Archive storage class or the S3 Intelligent-Tiering Deep Archive Access tier, Standard retrievals initiated by using the Batch Operations restore operation typically start within 9 hours and finish within 12 hours.

- **Bulk** – Access your data by using the lowest-cost retrieval option in Amazon S3 Glacier. With Bulk retrievals, you can retrieve large amounts, even petabytes, of data inexpensively.

  For objects that are stored in the S3 Glacier Flexible Retrieval storage class or the S3 Intelligent-Tiering Archive Access tier, Bulk retrievals typically finish within 5–12 hours. For objects stored in the S3 Glacier Deep Archive storage class or the S3 Intelligent-Tiering Deep Archive Access tier, these retrievals typically finish within 48 hours.

  Bulk retrievals are free for objects that are stored in the S3 Glacier Flexible Retrieval or S3 Intelligent-Tiering storage classes.

The following table summarizes the archive retrieval options. For information about pricing, see Amazon S3 pricing.

To make an `Expedited`, `Standard`, or `Bulk` retrieval, set the `Tier` request element in the RestoreObject REST API operation request to the option that you want, or the equivalent in the

AWS Command Line Interface (AWS CLI) or AWS SDKs. If you purchased provisioned capacity, all Expedited retrievals are automatically served through your provisioned capacity.

### Provisioned capacity

Provisioned capacity helps ensure that your retrieval capacity for Expedited retrievals from S3 Glacier Flexible Retrieval is available when you need it. Each unit of capacity provides that at least three Expedited retrievals can be performed every 5 minutes, and it provides up to 150 megabytes per second (MBps) of retrieval throughput.

If your workload requires highly reliable and predictable access to a subset of your data in minutes, consider purchasing provisioned retrieval capacity. Without provisioned capacity, Expedited retrievals might not be accepted during periods of high demand. If you require access to Expedited retrievals under all circumstances, we recommend that you purchase provisioned retrieval capacity.

Provisioned capacity units are allocated to an AWS account. Thus, the requester of the Expedited data retrieval should purchase the provisioned capacity unit, not the bucket owner.

You can purchase provisioned capacity by using the Amazon S3 console, the Amazon S3 Glacier console, the Purchase Provisioned Capacity REST API operation, the AWS SDKs, or the AWS CLI. For provisioned capacity pricing information, see Amazon S3 pricing.

### S3 Glacier restore initiation request rates

When you initiate restore requests for objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class, a retrieval-requests quota is applied for your AWS account. S3 Glacier supports restore requests at a rate of 1,000 transactions per second. If this rate is exceeded otherwise valid requests are throttled or rejected and Amazon S3 returns a `ThrottlingException` error.

Optionally, you can also use S3 Batch Operations to retrieve a large number of objects stored in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive with a single request. For more information, see Performing large-scale batch operations on Amazon S3 objects.

## Restoring an archived object

Amazon S3 objects in the following storage classes or tiers are archived and are not accessible in real time:

- The S3 Glacier Flexible Retrieval storage class

- The S3 Glacier Deep Archive storage class
- The S3 Intelligent-Tiering Archive Access tier
- The S3 Intelligent-Tiering Deep Archive Access tier

Amazon S3 objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes are not immediately accessible. To access an object in these storage classes, you must restore a temporary copy of the object to its S3 bucket for a specified duration (number of days). If you want a permanent copy of the object, restore the object, and then create a copy of it in your Amazon S3 bucket. Copying restored objects isn't supported in the Amazon S3 console. For this type of copy operation, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the REST API. Unless you make a copy and change its storage class, the object will still be stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For information about using these storage classes, see Storage classes for rarely accessed objects.

To access objects in the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers, you must initiate a restore request and wait until the object is moved into the Frequent Access tier. When you restore an object from the Archive Access tier or Deep Archive Access tier, the object moves back into the Frequent Access tier. For information about using these storage classes, see Storage class for automatically optimizing data with changing or unknown access patterns.

For general information about archived objects, see Working with archived objects.

**Note**

- When you restore an archived object from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, you pay for both the archived object and the copy that you restored temporarily.
- When you restore an object from S3 Intelligent-Tiering, there are no retrieval charges for Standard or Bulk retrievals.
- Subsequent restore requests called on archived objects that have already been restored are billed as GET requests. For information about pricing, see Amazon S3 pricing.

## Restoring an archived object

You can restore an archived object by using the Amazon S3 console, the Amazon S3 REST API, the AWS SDKs, the AWS Command Line Interface (AWS CLI), or S3 Batch Operations.

### Using the S3 console

#### Restore objects using the Amazon S3 console

Use the following procedure to Restore an object that has been archived to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, or the S3 Intelligent-Tiering Archive Access or Deep Archive Access storage tiers.

**To restore an archived object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to restore.
4. In the **Objects** list, select the object or objects that you want to restore, choose **Actions**, and then choose **Initiate restore**.
5. If you're restoring from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, enter the number of days that you want your archived data to be accessible in the **Number of days that the restored copy is available** box.
6. For **Retrieval tier**, do one of the following:

   - Choose **Bulk retrieval** or **Standard retrieval**, and then choose **Initiate restore**.
   - Choose **Expedited retrieval** (available only for S3 Glacier Flexible Retrieval or S3 Intelligent-Tiering Archive Access). If you're restoring an object in S3 Glacier Flexible Retrieval, you can choose whether to buy provisioned capacity for your Expedited retrieval. If you want to purchase provisioned capacity, proceed to the next step. If you don't, choose **Initiate restore**.

   > **Note**
   >
   > Objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers are automatically restored to the Frequent Access tier.

7. (Optional) If you're restoring an object in S3 Glacier Flexible Retrieval and you chose **Expedited retrieval**, you can choose whether to buy provisioned capacity. Provisioned capacity is available only for objects in S3 Glacier Flexible Retrieval. If you already have provisioned capacity, choose **Initiate restore** to start a provisioned retrieval.

If you have provisioned capacity, all of your Expedited retrievals are served by your provisioned capacity. For more information, see Provisioned capacity.

- If you don't have provisioned capacity and you don't want to buy it, choose **Initiate restore**.
- If you don't have provisioned capacity, but you want to buy provisioned capacity units (PCUs), choose **Purchase PCUs**. In the **Purchase PCUs** dialog box, choose how many PCUs you want to buy, confirm your purchase, and then choose **Purchase PCUs**. When you get the **Purchase succeeded** message, choose **Initiate restore** to start provisioned retrieval.

### Using the AWS CLI

#### Restore objects from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive

The following example uses the `restore-object` command to restore the object *dir1/example.obj* in the bucket *amzn-s3-demo-bucket* for 25 days.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

If the JSON syntax used in the example results in an error on a Windows client, replace the restore request with the following syntax:

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

#### Restore objects from S3 Intelligent-Tiering Archive Access and Deep Archive Access

The following example uses the `restore-object` command to restore the object *dir1/example.obj* in the bucket *amzn-s3-demo-bucket* to the Frequent Access tier.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> Unlike in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, restore requests for S3 Intelligent-Tiering objects don't accept the `Days` value.

### Monitor restore status

To monitor the status of your `restore-object` request, use the following `head-object` command:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

For more information, see restore-object in the *AWS CLI Command Reference*.

### Using the REST API

Amazon S3 provides an API operation for you to initiate the restoration of an archived object. For more information, see RestoreObject in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

For examples of how to restore archived objects in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive with the AWS SDKs, see Use `RestoreObject` with an AWS SDK or CLI.

### Using S3 Batch Operations

To restore more than one archived object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

To create a Batch Operations job, you must have a manifest that contains only the objects that you want to restore. You can create a manifest by using S3 Inventory, or you can supply a CSV file with the necessary information. For more information, see the section called “Specifying a manifest”.

Before creating and running S3 Batch Operations jobs, you must grant permissions to Amazon S3 to perform S3 Batch Operations on your behalf. For the required permissions, see the section called “Granting permissions”.

> **Note**
>
> Batch Operations jobs can operate either on S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage class objects *or* on S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tier objects. Batch Operations can't operate on both types of archived objects in the same job. To restore objects of both types, you *must* create separate Batch Operations jobs.

For more information about using Batch Operations to restore archive objects, see the section called "Restore objects".

**To create an S3 Initiate Restore Object Batch Operations job**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose **Create job**.
4. For **AWS Region**, choose the Region where you want to create your job.
5. Under **Manifest format**, choose the type of manifest to use.
   - If you choose **S3 inventory report**, enter the path to the `manifest.json` object that Amazon S3 generated as part of the CSV-formatted inventory report. If you want to use a manifest version other than the most recent, enter the version ID for the `manifest.json` object.
   - If you choose **CSV**, enter the path to a CSV-formatted manifest object. The manifest object must follow the format described in the console. If you want to use a version other than the most recent, you can optionally include the version ID for the manifest object.
6. Choose **Next**.
7. In the **Operation** section, choose **Restore**.
8. In the **Restore** section, for **Restore source**, choose either **Glacier Flexible Retrieval or Glacier Deep Archive** or **Intelligent-Tiering Archive Access tier or Deep Archive Access tier**.

   If you chose **Glacier Flexible Retrieval or Glacier Deep Archive**, enter a number for **Number of days that the restored copy is available**.

   For **Retrieval tier**, choose the tier that you want to use.
9. Choose **Next**.
10. On the **Configure additional options** page, fill out the following sections:
    - In the **Additional options** section, provide a description for the job and specify a priority number for the job. Higher numbers indicate a higher priority. For more information, see the section called "Assigning job priority".

- In the **Completion report** section, select whether Batch Operations should create a completion report. For more information about completion reports, see the section called "Completion reports".
- In the **Permissions** section, you must grant permissions to Amazon S3 to perform Batch Operations on your behalf. For the required permissions, see the section called "Granting permissions".
- (Optional) In the **Job tags** section, add tags in key-value pairs. For more information, see the section called "Using tags".

When you're finished, choose **Next**.

11. On the **Review** page, verify the settings. If you need to make changes, choose **Previous**. Otherwise, choose **Create job**.

For more information about Batch Operations, see Restore objects with Batch Operations and Creating an S3 Batch Operations job.

## Checking the restore status and expiration date

You can check the status of a restore request or the expiration date by using the Amazon S3 console, Amazon S3 Event Notifications, the AWS CLI, or the Amazon S3 REST API.

> **Note**
>
> Objects restored from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes are stored only for the number of days that you specify. The following procedures return the expiration date for these copies.
> Objects restored from the S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tiers don't have expiration dates and instead are moved back to the Frequent Access tier.

### Using the S3 console

#### To check an object's restore status and expiration date in the Amazon S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the name of the bucket that contains the object that you are restoring.
4. In the **Objects** list, select the object that you are restoring. The object's details page appears.

   - If the restoration isn't finished, at the top of the page, you see a section that says **Restoration in progress**.
   - If the restoration is finished, at the top of the page, you see a section that says **Restoration complete**. If you're restoring from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, this section also displays the **Restoration expiry date**. Amazon S3 will remove the restored copy of your archived object on this date.

#### Using Amazon S3 Event Notifications

You can be notified of object restoration completion by using the `s3:ObjectRestore:Completed` action with the Amazon S3 Event Notifications feature. For more information about enabling event notifications, see Enabling notifications by using Amazon SQS, Amazon SNS, and AWS Lambda. For more information about the various `ObjectRestore` event types, see the section called "Supported event types for SQS, SNS, and Lambda".

#### Using the AWS CLI

#### Check an object's restore status and expiration date with the AWS CLI

The following example uses the `head-object` command to view metadata for the object *`dir1/example.obj`* in the bucket *`amzn-s3-demo-bucket`*. When you run this command on an object being restored Amazon S3 returns if the restore is ongoing and (if applicable) the expiration date.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

Expected output (restore ongoing):

```
{
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
```

```
    "StorageClass": "GLACIER"
}
```

Expected output (restore finished):

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

For more information about `head-object`, see head-object in the *AWS CLI Command Reference*.

#### Using the REST API

Amazon S3 provides an API operation for you to retrieve object metadata. To check the restoration status and expiration date of an archived object using the REST API, see HeadObject in the *Amazon Simple Storage Service API Reference*.

## Upgrading the speed of an in-progress restore

You can upgrade the speed of your restoration while it is in progress.

#### To upgrade an in-progress restore to a faster tier

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to restore.
4. In the **Objects** list, select the object that you are restoring. The object's details page appears. On the object's details page, choose **Upgrade retrieval tier**. For information about checking the restoration status of an object, see Checking the restore status and expiration date.
5. Choose the tier that you want to upgrade to, and then choose **Initiate restore**.

# Using S3 Object Lock

S3 Object Lock can help prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock uses a *write-once-read-many* (WORM) model to store objects. You can use Object Lock to help meet regulatory requirements that require WORM storage, or to add another layer of protection against object changes or deletion.

> **Note**
>
> S3 Object Lock has been assessed by Cohasset Associates for use in environments that are subject to SEC 17a-4, CFTC, and FINRA regulations. For more information about how Object Lock relates to these regulations, see the Cohasset Associates Compliance Assessment.

Object Lock provides two ways to manage object retention: *retention periods* and *legal holds*. An object version can have a retention period, a legal hold, or both.

- **Retention period** – A retention period specifies a fixed period of time during which an object remains locked. You can set a unique retention period for individual objects. Additionally, you can set a default retention period on an S3 bucket. You may also restrict the minimum and maximum allowable retention periods with the `s3:object-lock-remaining-retention-days` condition key in the bucket policy. This helps you establish a range of retention periods and by restricting retention periods that may be shorter or longer than this range.
- **Legal hold** – A legal hold provides the same protection as a retention period, but it has no expiration date. Instead, a legal hold remains in place until you explicitly remove it. Legal holds are independent from retention periods and are placed on individual object versions.

Object Lock works only in buckets that have S3 Versioning enabled. When you lock an object version, Amazon S3 stores the lock information in the metadata for that object version. Placing a retention period or a legal hold on an object protects only the version that's specified in the request. Retention periods and legal holds don't prevent new versions of the object from being created, or delete markers to be added on top of the object. For information about S3 Versioning, see Using versioning in S3 buckets.

If you put an object into a bucket that already contains an existing protected object with the same object key name, Amazon S3 creates a new version of that object. The existing protected version of the object remains locked according to its retention configuration.

# How S3 Object Lock works

**Topics**



## Retention periods

A *retention period* protects an object version for a fixed amount of time. When you place a retention period on an object version, Amazon S3 stores a timestamp in the object version's metadata to indicate when the retention period expires. After the retention period expires, the object version can be overwritten or deleted.

You can place a retention period explicitly on an individual object version or on a bucket's properties so that it applies to all objects in the bucket automatically. When you apply a retention period to an object version explicitly, you specify a *Retain Until Date* for the object version. Amazon S3 stores this date in the object version's metadata.

You can also set a retention period in a bucket's properties. When you set a retention period on a bucket, you specify a duration, in either days or years, for how long to protect every object version placed in the bucket. When you place an object in the bucket, Amazon S3 calculates a *Retain Until Date* for the object version by adding the specified duration to the object version's creation timestamp. The object version is then protected exactly as though you explicitly placed an individual lock with that retention period on the object version.

> **Note**
>
> When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.

Like all other Object Lock settings, retention periods apply to individual object versions. Different versions of a single object can have different retention modes and periods.

For example, suppose that you have an object that is 15 days into a 30-day retention period, and you PUT an object into Amazon S3 with the same name and a 60-day retention period. In this case, your PUT request succeeds, and Amazon S3 creates a new version of the object with a 60-day retention period. The older version maintains its original retention period and becomes deletable in 15 days.

After you've applied a retention setting to an object version, you can extend the retention period. To do this, submit a new Object Lock request for the object version with a *Retain Until Date* that is later than the one currently configured for the object version. Amazon S3 replaces the existing retention period with the new, longer period. Any user with permissions to place an object retention period can extend a retention period for an object version. To set a retention period, you must have the `s3:PutObjectRetention` permission.

When you set a retention period on an object or S3 bucket, you must select one of two retention modes: *compliance* or *governance*.

## Retention modes

S3 Object Lock provides two retention modes that apply different levels of protection to your objects:

- Compliance mode
- Governance mode

In *compliance* mode, a protected object version can't be overwritten or deleted by any user, including the root user in your AWS account. When an object is locked in compliance mode, its retention mode can't be changed, and its retention period can't be shortened. Compliance mode helps ensure that an object version can't be overwritten or deleted for the duration of the retention period.

> **Note**
>
> The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

In *governance* mode, users can't overwrite or delete an object version or alter its lock settings unless they have special permissions. With governance mode, you protect objects against being deleted by most users, but you can still grant some users permission to alter the retention settings or delete the objects if necessary. You can also use governance mode to test retention-period settings before creating a compliance-mode retention period.

To override or remove governance-mode retention settings, you must have the `s3:BypassGovernanceRetention` permission and must explicitly include `x-amz-bypass-governance-retention:true` as a request header with any request that requires overriding governance mode.

> **Note**
>
> By default, the Amazon S3 console includes the `x-amz-bypass-governance-retention:true` header. If you try to delete objects protected by *governance* mode and have the `s3:BypassGovernanceRetention` permission, the operation will succeed.

## Legal holds

With Object Lock, you can also place a *legal hold* on an object version. Like a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated fixed amount of time and remains in effect until removed. Legal holds can be freely placed and removed by any user who has the `s3:PutObjectLegalHold` permission.

Legal holds are independent from retention periods. Placing a legal hold on an object version doesn't affect the retention mode or retention period for that object version.

For example, suppose that you place a legal hold on an object version and that object version is also protected by a retention period. If the retention period expires, the object doesn't lose its WORM protection. Rather, the legal hold continues to protect the object until an authorized user explicitly removes the legal hold. Similarly, if you remove a legal hold while an object version has a retention period in effect, the object version remains protected until the retention period expires.

## Best practices for using S3 Object Lock

Consider using *Governance mode* if you want to protect objects from being deleted by most users during a pre-defined retention period, but at the same time want some users with special permissions to have the flexibility to alter the retention settings or delete the objects.

Consider using *Compliance mode* if you never want any user, including the root user in your AWS account, to be able to delete the objects during a pre-defined retention period. You can use this mode in case you have a requirement to store compliant data.

You can use *Legal Hold* when you are not sure for how long you want your objects to stay immutable. This could be because you have an upcoming external audit of your data and want to keep objects immutable till the audit is complete. Alternately, you may have an ongoing project utilizing a dataset that you want to keep immutable until the project is complete.

### Required permissions

Object Lock operations require specific permissions. Depending on the exact operation that you're attempting, you might need any of the following permissions:

- `s3:BypassGovernanceRetention`
- `s3:GetBucketObjectLockConfiguration`
- `s3:GetObjectLegalHold`
- `s3:GetObjectRetention`
- `s3:PutBucketObjectLockConfiguration`
- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

For a complete list of Amazon S3 permissions with descriptions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For information about using conditions with permissions, see Bucket policy examples using condition keys.

## Object Lock considerations

Amazon S3 Object Lock can help prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely.

You can use the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API to view or set Object Lock information. For general information about S3 Object Lock capabilities, see Using S3 Object Lock.

**Important**

- After you enable Object Lock on a bucket, you can't disable Object Lock or suspend versioning for that bucket.
- S3 buckets with Object Lock can't be used as destination buckets for server access logs. For more information, see the section called "Logging server access".

**Topics**



## Permissions for viewing lock information

You can programmatically view the Object Lock status of an Amazon S3 object version by using the HeadObject or GetObject operations. Both operations return the retention mode, retain until date, and legal hold status for the specified object version. Additionally, you can view the Object Lock status for multiple objects in your S3 bucket using S3 Inventory.

To view an object version's retention mode and retention period, you must have the `s3:GetObjectRetention` permission. To view an object version's legal hold status, you must have the `s3:GetObjectLegalHold` permission. To view a bucket's default retention configuration, you must have the `s3:GetBucketObjectLockConfiguration` permission. If you make a request for an Object Lock configuration on a bucket that doesn't have S3 Object Lock enabled, Amazon S3 returns an error.

## Bypassing governance mode

If you have the `s3:BypassGovernanceRetention` permission, you can perform operations on object versions that are locked in governance mode as if they were unprotected. These operations include deleting an object version, shortening the retention period, or removing the Object Lock retention period by placing a new `PutObjectRetention` request with empty parameters.

To bypass governance mode, you must explicitly indicate in your request that you want to bypass this mode. To do this, include the `x-amz-bypass-governance-retention:true` header with your `PutObjectRetention` API operation request, or use the equivalent parameter with requests made through the AWS CLI or AWS SDKs. The S3 console automatically applies this header for requests made through the S3 console if you have the `s3:BypassGovernanceRetention` permission.

> **Note**
>
> Bypassing governance mode doesn't affect an object version's legal hold status. If an object version has a legal hold enabled, the legal hold remains and prevents requests to overwrite or delete the object version.

## Using Object Lock with S3 Replication

You can use Object Lock with S3 Replication to enable automatic, asynchronous copying of locked objects and their retention metadata, across S3 buckets. This means that for replicated objects, Amazon S3 takes the object lock configuration of the source bucket. In other words, if the source bucket has Object Lock enabled, the destination buckets must also have Object Lock enabled. If an object is directly uploaded to the destination bucket (outside of S3 Replication), it takes the Object Lock set on the destination bucket. When you use replication, objects in a *source bucket* are replicated to one or more *destination buckets*.

To set up replication on a bucket with Object Lock enabled, you can use the S3 console, AWS CLI, Amazon S3 REST API, or AWS SDKs.

> **Note**
>
> To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use

to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see Setting up permissions for live replication.
For general information about S3 Replication, see Replicating objects overview.
For examples of setting up S3 Replication, see Examples for configuring live replication.

## Using Object Lock with Amazon S3 Inventory

You can configure Amazon S3 Inventory to create lists of the objects in an S3 bucket on a defined schedule. You can configure Amazon S3 Inventory to include the following Object Lock metadata for your objects:

- The retain until date
- The retention mode
- The legal hold status

For more information, see Amazon S3 Inventory.

## Managing S3 Lifecycle policies with Object Lock

Object lifecycle management configurations continue to function normally on protected objects, including placing delete markers. However, a locked version of an object cannot be deleted by a S3 Lifecycle expiration policy. Object Lock is maintained regardless of which storage class the object resides in and throughout S3 Lifecycle transitions between storage classes.

For more information about managing object lifecycles, see Managing your storage lifecycle.

## Managing delete markers with Object Lock

Although you can't delete a protected object version, you can still create a delete marker for that object. Placing a delete marker on an object doesn't delete the object or its object versions. However, it makes Amazon S3 behave in most ways as though the object has been deleted. For more information, see Working with delete markers.

> **Note**
>
> Delete markers are not WORM-protected, regardless of any retention period or legal hold in place on the underlying object.

## Using S3 Storage Lens with Object Lock

To see metrics for Object Lock-enabled storage bytes and object count, you can use Amazon S3 Storage Lens. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity.

For more information, see Using S3 Storage Lens to protect your data.

For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

## Uploading objects to an Object Lock enabled bucket

The `Content-MD5` header is required for any request to upload an object with a retention period configured using Object Lock. The MD5 digest is a way to verify the integrity of your object after uploading it to a bucket. After uploading the object, Amazon S3 calculates the MD5 digest of the object and compares it to the value that you provided. The request succeeds only if the two digests match. The S3 console automatically adds this header, however you must specify this header when using the PutObject API.

For more information, see Using Content-MD5 when uploading objects.

## Configuring events and notifications

You can use Amazon S3 Event Notifications to track access and changes to your Object Lock configurations and data by using AWS CloudTrail. For information about CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

You can also use Amazon CloudWatch to generate alerts based on this data. For information about CloudWatch, see the What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*.

## Setting limits on retention periods with a bucket policy

You can set minimum and maximum allowable retention periods for a bucket by using a bucket policy. The maximum retention period is 100 years.

The following example shows a bucket policy that uses the `s3:object-lock-remaining-retention-days` condition key to set a maximum retention period of 10 days.

```
{
    "Version": "2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
        }
    ]
}
```

**Note**

If your bucket is the destination bucket for a replication configuration, you can set up minimum and maximum allowable retention periods for object replicas that are created by using replication. To do so, you must allow the `s3:ReplicateObject` action in your bucket policy. For more information about replication permissions, see the section called "Setting up permissions".

For more information about bucket policies, see the following topics:

- Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*
- Object operations
- Bucket policy examples using condition keys

# Configuring S3 Object Lock

With Amazon S3 Object Lock, you can store objects in Amazon S3 by using a *write-once-read-many* (WORM) model. You can use S3 Object Lock to prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. For general information about Object Lock capabilities, see Using S3 Object Lock.

Before you lock any objects, you must enable S3 Versioning and Object Lock on a bucket. Afterward, you can set a retention period, a legal hold, or both.

To work with Object Lock, you must have certain permissions. For a list of the permissions related to various Object Lock operations, see the section called "Required permissions".

**Important**

- After you enable Object Lock on a bucket, you can't disable Object Lock or suspend versioning for that bucket.
- S3 buckets with Object Lock can't be used as destination buckets for server access logs. For more information, see the section called "Logging server access".

**Topics**



## Enable Object Lock when creating a new S3 bucket

You can enable Object Lock when creating a new S3 bucket by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name for your bucket.

   > **Note**
   >
   > After you create a bucket, you can't change its name. For more information about naming buckets, see Bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.
6. Under **Object Ownership**, choose to disable or enable access control lists (ACLs) and control ownership of objects uploaded in your bucket.
7. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.
8. Under **Bucket Versioning**, choose **Enabled**.

   Object Lock works only with versioned buckets.
9. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs that are used to categorize storage and allocate costs.
10. Under **Advanced settings**, find **Object Lock** and choose **Enable**.

    You must acknowledge that enabling Object Lock will permanently allow objects in this bucket to be locked.
11. Choose **Create bucket**.

#### Using the AWS CLI

The following `create-bucket` example creates a new S3 bucket named *amzn-s3-demo-bucket1* with Object Lock enabled:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket1 --object-lock-enabled-for-bucket
```

For more information and examples, see create-bucket in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

#### Using the REST API

You can use the REST API to create a new S3 bucket with Object Lock enabled. For more information, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

#### Using the AWS SDKs

For examples of how to enable Object Lock when creating a new S3 bucket with the AWS SDKs, see Use `CreateBucket` with an AWS SDK or CLI.

For examples of how to get the current Object Lock configuration with the AWS SDKs, see Use `GetObjectLockConfiguration` with an AWS SDK or CLI.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Work with Amazon S3 object lock features using an AWS SDK.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

### Enable Object Lock on an existing S3 bucket

You can enable Object Lock for an existing S3 bucket by using the Amazon S3 console, the AWS CLI, AWS SDKs, or Amazon S3 REST API.

#### Using the S3 console

> **Note**
>
> Object Lock works only with versioned buckets.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to enable Object Lock on.
4. Choose the **Properties** tab.
5. Under **Properties**, scroll down to the **Object Lock** section, and choose **Edit**.
6. Under **Object Lock**, choose **Enable**.

   You must acknowledge that enabling Object Lock will permanently allow objects in this bucket to be locked.
7. Choose **Save changes**.

### Using the AWS CLI

The following `put-object-lock-configuration` example command sets a 50-day Object Lock retention period on a bucket named *amzn-s3-demo-bucket1*:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

For more information and examples, see put-object-lock-configuration in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

### Using the REST API

You can use the Amazon S3 REST API to enable Object Lock on an existing S3 bucket. For more information, see PutObjectLockConfiguration in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

For examples of how to enable Object Lock for an existing S3 bucket with the AWS SDKs, see Use `PutObjectLockConfiguration` with an AWS SDK or CLI.

For examples of how to get the current Object Lock configuration with the AWS SDKs, see Use `GetObjectLockConfiguration` with an AWS SDK or CLI.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Work with Amazon S3 object lock features using an AWS SDK.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

## Set or modify a legal hold on an S3 object

You can set or remove a legal hold on an S3 object by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

> **Important**
>
> - If you want to set a legal hold on an object, the object's bucket must already have Object Lock enabled.
> - When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.

For more information, see the section called “Legal holds”.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want to set or modify a legal hold on.
4. In the **Objects** list, select the object that you want to set or modify a legal hold on.

5. On the **Object properties** page, find the **Object Lock legal hold** section, and choose **Edit**.
6. Choose **Enable** to set a legal hold or **Disable** to remove a legal hold.
7. Choose **Save changes**.

### Using the AWS CLI

The following `put-object-legal-hold` example sets a legal hold on the object *my-image.fs* in the bucket named *amzn-s3-demo-bucket1*:

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=ON"
```

The following `put-object-legal-hold` example removes a legal hold on the object *my-image.fs* in the bucket named *amzn-s3-demo-bucket1*:

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=OFF"
```

For more information and examples, see put-object-legal-hold in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

### Using the REST API

You can use the REST API to set or modify a legal hold on an object. For more information, see PutObjectLegalHold in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

For examples of how to set a legal hold on an object with the AWS SDKs, see Use `PutObjectLegalHold` with an AWS SDK or CLI.

For examples of how to get the current legal hold status with the AWS SDKs, see Get the legal hold configuration of an Amazon S3 object using an AWS SDK.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Work with Amazon S3 object lock features using an AWS SDK.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

## Set or modify a retention period on an S3 object

You can set or modify a retention period on an S3 object by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

> **⚠ Important**
>
> - If you want to set a retention period on an object, the object's bucket must already have Object Lock enabled.
> - When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.
> - The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

For more information, see Retention periods.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want to set or modify a retention period on.
4. In the **Objects** list, select the object that you want to set or modify a retention period on.
5. On the **Object properties** page, find the **Object Lock retention** section, and choose **Edit**.
6. Under **Retention**, choose **Enable** to set a retention period or **Disable** to remove a retention period.

7. If you chose **Enable**, under **Retention mode**, choose either **Governance mode** or **Compliance mode**. For more information, see Retention modes.
8. Under **Retain until date**, choose the date that you want to have the retention period end on. During this period, your object is WORM-protected and can't be overwritten or deleted. For more information, see Retention periods.
9. Choose **Save changes**.

### Using the AWS CLI

The following `put-object-retention` example sets a retention period on the object *my-image.fs* in the bucket named *amzn-s3-demo-bucket1* until January 1, 2025:

```
aws s3api put-object-retention --bucket amzn-s3-demo-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

For more information and examples, see put-object-retention in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

### Using the REST API

You can use the REST API to set a retention period on an object. For more information, see PutObjectRetention in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

For examples of how to set a retention period on an object with the AWS SDKs, see Use `PutObjectRetention` with an AWS SDK or CLI.

For examples of how to get the retention period on an object with the AWS SDKs, see Use `GetObjectRetention` with an AWS SDK or CLI.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Work with Amazon S3 object lock features using an AWS SDK.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

## Set or modify a default retention period on an S3 bucket

You can set or modify a default retention period on an S3 bucket by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API. You specify a duration, in either days or years, for how long to protect every object version placed in the bucket.

> **Important**
>
> - If you want to set a default retention period on a bucket, the bucket must already have Object Lock enabled.
> - When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.
> - The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

For more information, see Retention periods.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to set or modify a default retention period on.
4. Choose the **Properties** tab.
5. Under **Properties**, scroll down to the **Object Lock** section, and choose **Edit**.
6. Under **Default retention**, choose **Enable** to set a default retention or **Disable** to remove a default retention.

7. If you chose **Enable**, under **Retention mode**, choose either **Governance mode** or **Compliance mode**. For more information, see Retention modes.
8. Under **Default retention period**, choose the number of days or years that you want the retention period to last for. Objects placed in this bucket will be locked for this number of days or years. For more information, see Retention periods.
9. Choose **Save changes**.

### Using the AWS CLI

The following `put-object-lock-configuration` example command sets a 50-day Object Lock retention period on the bucket named *amzn-s3-demo-bucket1* by using compliance mode:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

The following `put-object-lock-configuration` example removes the default retention configuration on a bucket:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

For more information and examples, see put-object-lock-configuration in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

### Using the REST API

You can use the REST API to set a default retention period on an existing S3 bucket. For more information, see PutObjectLockConfiguration in the *Amazon Simple Storage Service API Reference*.

### Using the AWS SDKs

For examples of how to set a default retention period on an existing S3 bucket with the AWS SDKs, see Use `PutObjectLockConfiguration` with an AWS SDK or CLI.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Work with Amazon S3 object lock features using an AWS SDK.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs.

## Using Amazon S3 storage classes

Each object in Amazon S3 has a storage class associated with it. For example, if you list the objects in an S3 bucket, the console shows the storage class for all the objects in the list. Amazon S3 offers a range of storage classes for the objects that you store. You choose a class depending on your use case scenario and performance access requirements. All of these storage classes offer high durability.

The following sections provide details of the various storage classes and how to set the storage class for your objects.

### Topics



## Storage classes for frequently accessed objects

For performance-sensitive use cases (those that require millisecond access time) and frequently accessed data, Amazon S3 provides the following storage classes:

- **S3 Standard** – The default storage class. If you don't specify the storage class when you upload an object, Amazon S3 assigns the S3 Standard storage class.

- **S3 Express One Zone** – Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud object storage class available today, with data access speed up to 10x faster and with request costs 50 percent lower than S3 Standard. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. For more information, see What is S3 Express One Zone?.
- **Reduced Redundancy** – The Reduced Redundancy Storage (RRS) storage class is designed for noncritical, reproducible data that can be stored with less redundancy than the S3 Standard storage class.

  > **Important**
  >
  > We recommend not using this storage class. The S3 Standard storage class is more cost-effective.

  For durability, RRS objects have an average annual expected loss of 0.01 percent of objects. If an RRS object is lost, when requests are made to that object, Amazon S3 returns a 405 error.

## Storage class for automatically optimizing data with changing or unknown access patterns

**S3 Intelligent-Tiering** is an Amazon S3 storage class that's designed to optimize storage costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead. S3 Intelligent-Tiering is the only cloud storage class that delivers automatic cost savings by moving data on a granular object level between access tiers when access patterns change. S3 Intelligent-Tiering is the ideal storage class when you want to optimize storage costs for data that has unknown or changing access patterns. There are no retrieval fees for S3 Intelligent-Tiering.

For a small monthly object monitoring and automation fee, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers. S3 Intelligent-Tiering delivers automatic storage cost savings in three low-latency and high-throughput access tiers. For data that can be accessed asynchronously, you can choose to activate automatic archiving capabilities within the S3 Intelligent-Tiering storage class. S3 Intelligent-Tiering is designed for 99.9% availability and 99.999999999% durability.

S3 Intelligent-Tiering automatically stores objects in three access tiers:

- **Frequent Access** – Objects that are uploaded or transitioned to S3 Intelligent-Tiering are automatically stored in the Frequent Access tier.
- **Infrequent Access** – S3 Intelligent-Tiering moves objects that have not been accessed in 30 consecutive days to the Infrequent Access tier.
- **Archive Instant Access** – With S3 Intelligent-Tiering, any existing objects that have not been accessed for 90 consecutive days are automatically moved to the Archive Instant Access tier.

In addition to these three tiers, S3 Intelligent-Tiering offers two optional archive access tiers:

- **Archive Access** – S3 Intelligent-Tiering provides you with the option to activate the Archive Access tier for data that can be accessed asynchronously. After activation, the Archive Access tier automatically archives objects that have not been accessed for a minimum of 90 consecutive days.
- **Deep Archive Access** – S3 Intelligent-Tiering provides you with the option to activate the Deep Archive Access tier for data that can be accessed asynchronously. After activation, the Deep Archive Access tier automatically archives objects that have not been accessed for a minimum of 180 consecutive days.

> **Note**
>
> - Only activate the Archive Access tier for 90 days if you want to bypass the Archive Instant Access tier. The Archive Access tier delivers slightly lower-cost storage with minute-to-hour retrieval times. The Archive Instant Access tier delivers millisecond access and high-throughput performance.
> - Activate the Archive Access and Deep Archive Access tiers only if your objects can be accessed asynchronously by your application. If the object that you are retrieving is stored in the Archive Access or Deep Archive Access tiers, first restore the object by using `RestoreObject`.

You can move newly created data to S3 Intelligent-Tiering, setting it as your default storage class. You can also choose to activate one or both of the archive access tiers by using the `PutBucketIntelligentTieringConfiguration` API operation, the AWS CLI, or the Amazon

S3 console. For more information about using S3 Intelligent-Tiering and activating the archive access tiers, see Using S3 Intelligent-Tiering.

To access objects in the Archive Access or Deep Archive Access tiers, you first need to restore them. For more information, see Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers.

> **Note**
>
> If the size of an object is less than 128 KB, it is not monitored and not eligible for auto-tiering. Smaller objects are always stored in the Frequent Access tier. For more information about S3 Intelligent-Tiering, see S3 Intelligent-Tiering access tiers.

## Storage classes for infrequently accessed objects

The **S3 Standard-IA** and **S3 One Zone-IA** storage classes are designed for long-lived and infrequently accessed data. (IA stands for *infrequent access*.) S3 Standard-IA and S3 One Zone-IA objects are available for millisecond access (similar to the S3 Standard storage class). Amazon S3 charges a retrieval fee for these objects, so they are most suitable for infrequently accessed data. For pricing information, see Amazon S3 pricing.

For example, you might choose the S3 Standard-IA and S3 One Zone-IA storage classes to do the following:

- For storing backups.
- For older data that is accessed infrequently, but that still requires millisecond access. For example, when you upload data, you might choose the S3 Standard storage class, and use lifecycle configuration to tell Amazon S3 to transition the objects to the S3 Standard-IA or S3 One Zone-IA class.

  For more information about lifecycle management, see Managing your storage lifecycle.

> **Note**
>
> The S3 Standard-IA and S3 One Zone-IA storage classes are suitable for objects larger than 128 KB that you plan to store for at least 30 days. If an object is less than 128 KB, Amazon S3 charges you for 128 KB. If you delete an object before the end of the 30-day minimum

storage duration period, you are charged for 30 days. Objects that are deleted, overwritten, or transitioned to a different storage class before 30 days will incur the normal storage usage charge plus a pro-rated charge for the remainder of the 30-day minimum. For pricing information, see Amazon S3 pricing.

These storage classes differ as follows:

- **S3 Standard-IA** – Amazon S3 stores the object data redundantly across multiple geographically separated Availability Zones (similar to the S3 Standard storage class). S3 Standard-IA objects are resilient to the loss of an Availability Zone. This storage class offers greater availability and resiliency than the S3 One Zone-IA class.
- **S3 One Zone-IA** – Amazon S3 stores the object data in only one Availability Zone, which makes it less expensive than S3 Standard-IA. However, the data is not resilient to the physical loss of the Availability Zone resulting from disasters, such as earthquakes and floods. The S3 One Zone-IA storage class is as durable as S3 Standard-IA, but it is less available and less resilient. For a comparison of storage class durability and availability, see Comparing the Amazon S3 storage classes at the end of this section. For pricing information, see Amazon S3 pricing.

We recommend the following:

- **S3 Standard-IA** – Use for your primary or only copy of data that can't be re-created.
- **S3 One Zone-IA** – Use if you can re-create the data if the Availability Zone fails, and for object replicas when configuring S3 Cross-Region Replication (CRR).

## Storage classes for rarely accessed objects

The **S3 Glacier Instant Retrieval**, **S3 Glacier Flexible Retrieval**, and **S3 Glacier Deep Archive** storage classes are designed for low-cost, long-term data storage and data archiving. These storage classes offer the same durability and resiliency as the S3 Standard and S3 Standard-IA storage classes. For more information about S3 Glacier storage classes, see Long-term data storage using S3 Glacier storage classes.

Amazon S3 provides the following S3 Glacier storage classes:

- **S3 Glacier Instant Retrieval** – Use for long-term data that's rarely accessed and requires milliseconds retrieval. Data in this storage class is available for real-time access.

- **S3 Glacier Flexible Retrieval** – Use for archives where portions of the data might need to be retrieved in minutes. Data in this storage class is archived, and not available for real-time access.
- **S3 Glacier Deep Archive** – Use for archiving data that rarely needs to be accessed. Data in this storage class is archived, and not available for real-time access.

### Retrieving archived objects

You can set the storage class of an object to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive in the same ways that you do for the other storage classes as described in the section Setting the storage class of an object. However, S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive objects are archived, and not available for real-time access. For more information, see Archival storage.

> **Note**
>
> When you use S3 Glacier storage classes, your objects remain in Amazon S3. You can't access them directly through the separate Amazon S3 Glacier service. For information about the Amazon S3 Glacier service, see the Amazon S3 Glacier Developer Guide.

## Storage class for Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use the same API operations and features on AWS Outposts as you do on Amazon S3, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS CLI, AWS SDKs, or REST API.

S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS). The S3 Outposts storage class is available only for objects stored in buckets on Outposts. If you try to use this storage class with an S3 bucket in an AWS Region, an `InvalidStorageClass` error occurs. In addition, if you try to use other S3 storage classes with objects stored in S3 on Outposts buckets, the same error occurs.

Objects stored in the S3 Outposts (OUTPOSTS) storage class are always encrypted by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

You can also explicitly choose to encrypt objects stored in the S3 Outposts storage class by using server-side encryption with customer-provided encryption keys (SSE-C). For more information, see Using server-side encryption with customer-provided keys (SSE-C).

> **Note**
>
> S3 on Outposts doesn't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

For more information about S3 on Outposts, see What is Amazon S3 on Outposts?

## Comparing the Amazon S3 storage classes

The following table compares the storage classes, including their availability, durability, minimum storage duration, and other considerations.

| Storage Class | Designed for | Durability (designed for) | Availability (designed for) | Availability Zones | Min storage duration | Min billable object size | Other Considerations |
|---|---|---|---|---|---|---|---|
| STANDARD | Frequently accessed data | 99.999999999% | 99.99% | >= 3 | None | None | None |
| STANDARD_IA | Long-lived, infrequently accessed data | 99.999999999% | 99.9% | >= 3 | 30 days | 128 KB | Per GB retrieval fees apply. |
| INTELLIGENT_TIERING | Long-lived data with changing or unknown access patterns | 99.999999999% | 99.9% | >= 3 | 30 days | None | Monitoring and automation fees per object apply. No retrieval fees. |
| ONEZONE_IA | Long-lived, infrequently accessed, non-critical data | 99.999999999% | 99.5% | 1 | 30 days | 128 KB | Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone. |
| GLACIER | Long-term data archiving with retrieval times ranging from minutes to hours | 99.999999999% | 99.99% (after you restore objects) | >= 3 | 90 days | None | Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects. |
| DEEP_ARCHIVE | Archiving rarely accessed data with a default retrieval time of 12 hours | 99.999999999% | 99.99% (after you restore objects) | >= 3 | 180 days | None | Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects. |
| RRS (Not recommended) | Frequently accessed, non-critical data | 99.99% | 99.99% | >= 3 | None | None | None |

* S3 Glacier Flexible Retrieval requires 40 KB of additional metadata for each archived object. This includes 32 KB of metadata charged at the S3 Glacier Flexible Retrieval rate (required to identify and retrieve your data), and an additional 8 KB data charged at the S3 Standard rate. The S3 Standard rate is required to maintain the user-defined name and metadata for objects archived to S3 Glacier Flexible Retrieval. For more information about storage classes, see Amazon S3 storage classes.

** S3 Glacier Deep Archive requires 40 KB of additional metadata for each archived object. This includes 32 KB of metadata charged at the S3 Glacier Deep Archive rate (required to identify and retrieve your data), and an additional 8 KB data charged at the S3 Standard rate. The S3 Standard rate is required to maintain the user-defined name and metadata for objects archived to Amazon S3 Glacier Deep Archive. For more information about storage classes, see Amazon S3 storage classes.

Be aware that all of the storage classes except for S3 One Zone-IA and S3 Express One Zone are designed to be resilient to the physical loss of an Availability Zone resulting from disasters. Also, consider costs, in addition to the performance requirements of your application scenario. For storage class pricing, see Amazon S3 pricing.

## Setting the storage class of an object

To set and update object storage classes, you can use the Amazon S3 console, AWS SDKs, or the AWS Command Line Interface (AWS CLI). All of these approaches use Amazon S3 API operations to send requests to Amazon S3.

Amazon S3 API operations support setting (or updating) the storage class of objects as follows:

- When creating a new object, you can specify its storage class. For example, when creating objects by using the PUT Object, POST Object, and Initiate Multipart Upload API operations, you add the `x-amz-storage-class` request header to specify a storage class. If you don't add this header, Amazon S3 uses S3 Standard, the default storage class.
- You can also change the storage class of an object that is already stored in Amazon S3 to any other storage class by making a copy of the object by using the PUT Object - Copy API operation. However, you can't use PUT Object - Copy to copy objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. You also can't transition from S3 One Zone-IA to S3 Glacier Instant Retrieval.

  You copy the object in the same bucket by using the same key name and specifying the request headers as follows:

  - Set the `x-amz-metadata-directive` header to `COPY`.
  - Set the `x-amz-storage-class` header to the storage class that you want to use.

  In a versioning-enabled bucket, you can't change the storage class of a specific version of an object. When you copy the object, Amazon S3 gives it a new version ID.

- You can change an object's storage class using the Amazon S3 console if the object size is less than 160 GB. If larger, we recommend adding an S3 Lifecycle configuration to change the object's storage class.
- If you use the Amazon S3 console to change the storage class for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing the storage class for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission. If the destination bucket policy denies the `s3:GetObjectTagging` action, the storage class for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.
- You can direct Amazon S3 to change the storage class of objects by adding an S3 Lifecycle configuration to a bucket. For more information, see Managing your storage lifecycle.
- When setting up a replication configuration, you can set the storage class for replicated objects to any other storage class. However, you can't replicate objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For more information, see Replication configuration.

### Restricting access policy permissions to a specific storage class

When you grant access policy permissions for Amazon S3 operations, you can use the `s3:x-amz-storage-class` condition key to restrict which storage class to use when storing uploaded objects. For example, when you grant the `s3:PutObject` permission, you can restrict object uploads to a specific storage class. For an example policy, see Example: Restricting object uploads to objects with a specific storage class.

For more information about using conditions in policies and a complete list of Amazon S3 condition keys, see the following topics:

- Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*
- Bucket policy examples using condition keys

## Long-term data storage using S3 Glacier storage classes

Amazon S3 offers multiple S3 Glacier storage classes that are designed to provide cost-effective solutions to storing long-term data that isn't accessed often. The S3 Glacier storage classes are:

- S3 Glacier Instant Retrieval

- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

You choose one of these storage classes based on how often you access your data and how fast you need to retrieve it. Each of these storage classes offer the same durability and resiliency as the S3 Standard storage class, but at lower storage costs. For more information about the S3 Glacier storage classes, see https://aws.amazon.com/s3/storage-classes/glacier/.

**Topics**



## Comparing the S3 Glacier storage classes

Each S3 Glacier storage class has a minimum storage duration for all objects. If you delete, overwrite, or transition the object to a different storage class before the minimum, you are charged for the full minimum storage duration.

Some S3 Glacier storage classes are archival, which means the objects stored in those classes are archived and not available for real-time access. For more information, see Archival storage.

Storage classes designed for less frequent access patterns with longer retrieval times offer lower storage costs. For pricing information, see https://aws.amazon.com/s3/pricing/.

The following table summarizes the key points to consider when choosing a S3 Glacier storage class:

## S3 Glacier Instant Retrieval

We recommend using S3 Glacier Instant Retrieval for long-term data that's accessed once per quarter and requires millisecond retrieval times. This storage class is ideal for performance-

sensitive use cases such as image hosting, file-sharing applications, and storing medical records for access during appointments.

S3 Glacier Instant Retrieval storage class offers real-time access to your objects with the same latency and throughput performance as the S3 Standard-IA storage class. When compared to S3 Standard-IA, S3 Glacier Instant Retrieval has lower storage costs but higher data access costs.

There is a minimum object size of 128 KB for data stored in the S3 Glacier Instant Retrieval storage class. This storage class also has a minimum storage duration period of 90 days.

## S3 Glacier Flexible Retrieval

We recommend using S3 Glacier Flexible Retrieval for archive data that's accessed one to two times a year and doesn't require immediate access. S3 Glacier Flexible Retrieval offers flexible retrieval times to help you balance costs, with access times ranging from a few minutes to hours, and free bulk retrievals. This storage class is ideal for backup and disaster recovery.

Objects stored in S3 Glacier Flexible Retrieval are archived and not available for real-time access. For more information, see Archival storage. To access these objects, you first initiate a restore request which creates a temporary copy of the object that you can access when the request completes. For information, see Working with archived objects. When you restore an object, you can choose a retrieval tier to meet your use case, with lower costs for longer restore times.

The following retrieval tiers are available for S3 Glacier Flexible Retrieval:

- **Expedited retrieval** – Typically restores the object in 1–5 minutes. Expedited retrievals are subject to demand, so to make sure you have reliable and predictable restore times, we recommend that you purchase provisioned retrieval capacity. For more information, see Provisioned capacity.
- **Standard retrieval** – Typically restores the object in 3–5 hours, or within 1 minute to 5 hours when you use S3 Batch Operations. For more information, see Restore objects with Batch Operations.
- **Bulk retrieval** – Typically restores the object within 5–12 hours. Bulk retrievals are free.

The minimum storage duration for objects in S3 Glacier Flexible Retrieval storage class is 90 days.

S3 Glacier Flexible Retrieval requires 40 KB of additional metadata for each object. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for

S3 Glacier Flexible Retrieval. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

## S3 Glacier Deep Archive

We recommend using S3 Glacier Deep Archive for archive data that's accessed less than once a year. This storage class is designed for retaining data sets for multiple years to meet compliance requirements and can also be used for backup or disaster recovery or any infrequently accessed data that you can wait up to 72 hours to retrieve. S3 Glacier Deep Archive is the lowest-cost storage option in AWS.

Objects stored in S3 Glacier Deep Archive are archived and not available for real-time access. For more information, see Archival storage. To access these objects, you first initiate a restore request which creates a temporary copy of the object that you can access when the request completes. For information, see Working with archived objects. When you restore an object, you can choose a retrieval tier to meet your use case, with lower costs for longer restore times.

The following retrieval tiers are available for S3 Glacier Deep Archive:

- **Standard retrieval** – Typically restores the object within 12 hours, or within 9–12 hours when you use S3 Batch Operations. For more information, see Restore objects with Batch Operations.
- **Bulk retrieval** – Typically restores the object within 48 hours at a fraction of the cost of the Standard retrieval tier.

The minimum storage duration for objects in S3 Glacier Deep Archive storage class is 180 days.

S3 Glacier Deep Archive requires 40 KB of additional metadata for each object. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for S3 Glacier Deep Archive. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

## Archival storage

S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are archival storage classes. This means that when you store an object in these storage classes that object is archived, and cannot be accessed directly. To access an archived object, you submit a restore request for it, and then wait for the service to restore the object. The restore request restores a temporary copy of the object, and that copy is deleted when the duration you specified in the request expires. For more information see Working with archived objects.

These storage classes require 40 KB of additional metadata for each archived object. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for that storage class. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

Objects in these storage classes are billed at S3 Standard storage class rates when you upload them using multipart uploads. For more information, see Multipart upload and pricing.

You can restore archived objects in these storage classes with up to 1,000 transactions per second (TPS) of object restore requests per account per AWS Region.

### How these storage classes differ from the S3 Glacier service

The S3 Glacier storage classes are part of the Amazon S3 service and store data as objects in S3 buckets. You can manage objects in these storage classes using the S3 console or programmatically using the S3 APIs or SDKs. When you store objects in S3 Glacier storage classes, you can use S3 features such as advanced encryption, object tagging, and S3 Lifecycle configurations to help manage data accessibility and cost.

> **Important**
>
> We recommend using the S3 Glacier storage classes within the Amazon S3 service for all of your long-term data.

The Amazon S3 Glacier (S3 Glacier) service is a separate service that stores data as archives within vaults. This service doesn't support Amazon S3 features and doesn't provide console support for data upload and download operations. We don't recommend using the S3 Glacier service for your long-term data. Data stored in this service isn't accessible from the Amazon S3 service. If you are looking for information on the S3 Glacier service, see the Amazon S3 Glacier Developer Guide. To transfer data from the Amazon S3 Glacier service to a storage class in Amazon S3 see Data Transfer from Amazon S3 Glacier Vaults to Amazon S3 in the AWS solutions library.

## Amazon S3 Intelligent-Tiering

The S3 Intelligent-Tiering storage class is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change, without operational overhead or impact on performance. For a small monthly object monitoring and

automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers.

S3 Intelligent-Tiering delivers automatic storage cost savings in three low latency and high throughput access tiers. For data that can be accessed asynchronously, you can choose to activate automatic archiving capabilities within the S3 Intelligent-Tiering storage class. There are no retrieval charges in S3 Intelligent-Tiering. If an object in the Infrequent Access tier or Archive Instant Access tier is accessed later, it is automatically moved back to the Frequent Access tier. No additional tiering charges apply when objects are moved between access tiers within the S3 Intelligent-Tiering storage class.

S3 Intelligent-Tiering is the recommended storage class for data with unknown, changing, or unpredictable access patterns, independent of object size or retention period, such as data lakes, data analytics, and new applications.

For information about using S3 Intelligent-Tiering, see the following sections:

**Topics**



## How S3 Intelligent-Tiering works

The Amazon S3 Intelligent-Tiering storage class automatically stores objects in three access tiers. One tier is optimized for frequent access, one lower-cost tier is optimized for infrequent access, and another very low-cost tier is optimized for rarely accessed data. For a low monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects to the Infrequent Access tier when they haven't been accessed for 30 consecutive days. After 90 days of no access, the objects are moved to the Archive Instant Access tier without performance impact or operational overhead.

To get the lowest storage cost for data that can be accessed in minutes to hours, activate archiving capabilities to add two additional access tiers. You can tier down objects to the Archive Access tier, the Deep Archive Access tier, or both. With Archive Access, S3 Intelligent-Tiering moves objects that have not been accessed for a minimum of 90 consecutive days to the Archive Access tier. With Deep Archive Access, S3 Intelligent-Tiering moves objects to the Deep Archive Access tier after a

minimum of 180 consecutive days of no access. For both tiers, you can configure the number of days of no access based on your needs.

The following actions constitute access that prevents tiering your objects down to the Archive Access tier or the Deep Archive Access tier:

- Downloading or copying an object through the Amazon S3 console.
- Invoking CopyObject, UploadPartCopy, or replicating objects with S3 Batch Replication. In these cases, the source objects of the copy or replication operations are tiered up.
- Invoking GetObject, PutObject, RestoreObject, CompleteMultipartUpload, ListParts, or SelectObjectContent.

For example, if your objects are accessed through `SelectObjectContent` before your specified number of days of no access (for example, 180 days), that action resets the timer. Your objects won't move to the Archive Access tier or the Deep Archive Access tier until the time after the last `SelectObjectContent` request reaches your specified number of days.

If an object in the Infrequent Access tier or Archive Instant Access tier is accessed later, it is automatically moved back to the Frequent Access tier.

The following actions constitute access that automatically moves objects from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier:

- Downloading or copying an object through the Amazon S3 console.
- Invoking CopyObject, UploadPartCopy, or replicating objects with Batch Replication. In these cases, the source objects of the copy or replication operations are tiered up.
- Invoking GetObject, PutObject, RestoreObject, CompleteMultipartUpload, or ListParts.

Other actions **don't** constitute access that automatically moves objects from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier. The following is a sample, not a definitive list, of such actions:

- Invoking HeadObject, GetObjectTagging, PutObjectTagging, ListObjects, ListObjectsV2, or ListObjectVersions.
- Invoking SelectObjectContent doesn't constitute access that tiers objects up to a Frequent Access tier. In addition, it doesn't prevent tiering objects down from the Frequent Access tier to the Infrequent Access tier, and then to the Archive Instant Access tier.

You can configure S3 Intelligent-Tiering as your default storage class for newly created data by specifying `INTELLIGENT-TIERING` in your PutBucketIntelligentTieringConfiguration request header. S3 Intelligent-Tiering is designed for 99.9% availability and 99.999999999% durability.

> **Note**
>
> If the size of an object is less than 128 KB, it is not monitored and is not eligible for automatic tiering. Smaller objects are always stored in the Frequent Access tier.

## S3 Intelligent-Tiering access tiers

The following section explains the different automatic and optional access tiers. When objects move between access tiers, the storage class remains the same (S3 Intelligent-Tiering).

Frequent Access tier (automatic)

This is the default access tier that any object created or transitioned to S3 Intelligent-Tiering begins its lifecycle in. An object remains in this tier as long as it is being accessed. The Frequent Access tier provides low latency and high-throughput performance.

Infrequent Access tier (automatic)

If an object is not accessed for 30 consecutive days, the object moves to the Infrequent Access tier. The Infrequent Access tier provides low latency and high-throughput performance.

Archive Instant Access tier (automatic)

If an object is not accessed for 90 consecutive days, the object moves to the Archive Instant Access tier. The Archive Instant Access tier provides low latency and high-throughput performance.

Archive Access tier (optional)

S3 Intelligent-Tiering provides you with the option to activate the Archive Access tier for data that can be accessed asynchronously. After activation, the Archive Access tier automatically archives objects that have not been accessed for a minimum of 90 consecutive days. You can extend the last access time for archiving to a maximum of 730 days. The Archive Access tier has the same performance as the S3 Glacier Flexible Retrieval storage class.

Standard retrieval times for this access tier can range from 3–5 hours. If you initiate your restore request by using S3 Batch Operations, your restore starts within minutes. For more

information about retrieval options and times, see the section called "Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers".

> **Note**
>
> Only activate the Archive Access tier for 90 days if you want to bypass the Archive Instant Access tier. The Archive Access tier delivers slightly lower storage costs, with minute-to-hour retrieval times. The Archive Instant Access tier delivers millisecond access and high-throughput performance.

Deep Archive Access tier (optional)

S3 Intelligent-Tiering provides you with the option to activate the Deep Archive Access tier for data that can be accessed asynchronously. After activation, the Deep Archive Access tier automatically archives objects that have not been accessed for a minimum of 180 consecutive days. You can extend the last access time for archiving to a maximum of 730 days. The Deep Archive Access tier has the same performance as the S3 Glacier Deep Archive storage class.

Standard retrieval of objects in this access tier occurs within 12 hours. If you initiate your restore request by using S3 Batch Operations, your restore starts within 9 hours. For more information about retrieval options and times, see the section called "Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers".

> **Note**
>
> Activate the Archive Access and Deep Archive Access tiers only if your objects can be accessed asynchronously by your application. If the object that you are retrieving is stored in the Archive Access or Deep Archive Access tiers, you must first restore the object by using the `RestoreObject` operation.

## Using S3 Intelligent-Tiering

You can use the S3 Intelligent-Tiering storage class to automatically optimize storage costs. S3 Intelligent-Tiering delivers automatic cost savings by moving data on a granular object level between access tiers when access patterns change. For data that can be accessed asynchronously,

you can choose to enable automatic archiving within the S3 Intelligent-Tiering storage class using the AWS Management Console, AWS CLI, or Amazon S3 API.

## Moving data to S3 Intelligent-Tiering

There are two ways to move data into S3 Intelligent-Tiering. You can directly PUT data into S3 Intelligent-Tiering by specifying `INTELLIGENT_TIERING` in the `x-amz-storage-class` header or configure S3 Lifecycle configurations to transition objects from S3 Standard or S3 Standard-Infrequent Access to S3 Intelligent-Tiering.

### Uploading data to S3 Intelligent-Tiering using Direct PUT

When you upload an object to the S3 Intelligent-Tiering storage class using the PUT API operation, you specify S3 Intelligent-Tiering in the `x-amz-storage-class` request header.

The following request stores the image, `my-image.jpg`, in the `myBucket` bucket. The request uses the `x-amz-storage-class` header to request that the object is stored using the S3 Intelligent-Tiering storage class.

**Example**

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

### Transitioning data to S3 Intelligent-Tiering from S3 Standard or S3 Standard-Infrequent Access using S3 Lifecycle

You can add rules to an S3 Lifecycle configuration to tell Amazon S3 to transition objects from one storage class to another. For information on supported transitions and related constraints, see Transitioning objects using S3 Lifecycle.

You can specify S3 Lifecycle configurations at the bucket or prefix level. In this S3 Lifecycle configuration rule, the filter specifies a key prefix (`documents/`). Therefore, the rule applies to objects with key name prefix `documents/`, such as `documents/doc1.txt` and `documents/doc2.txt`. The rule specifies a `Transition` action directing Amazon S3 to transition objects to

the S3 Intelligent-Tiering storage class 0 days after creation. In this case, objects are eligible for transition to S3 Intelligent-Tiering at midnight UTC following creation.

**Example**

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
 </Rule>
</LifecycleConfiguration>
```

## Enabling S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers

To get the lowest storage cost on data that can be accessed in minutes to hours, you can activate one or both of the archive access tiers by creating a bucket, prefix, or object tag level configuration using the AWS Management Console, AWS CLI, or Amazon S3 API.

### Using the S3 console

### To enable S3 Intelligent-Tiering automatic archiving

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want.
3. Choose **Properties**.
4. Navigate to the **S3 Intelligent-Tiering Archive configurations** section and choose **Create configuration**.
5. In the **Archive configuration settings** section, specify a descriptive configuration name for your S3 Intelligent-Tiering Archive configuration.
6. Under **Choose a configuration scope**, choose a configuration scope to use. Optionally, you can limit the configuration scope to specified objects within a bucket using a shared prefix, object tag, or combination of the two.

a. To limit the scope of the configuration, select **Limit the scope of this configuration using one or more filters**.

b. To limit the scope of the configuration using a single prefix, enter the prefix under **Prefix**.

c. To limit the scope of the configuration using object tags, select **Add tag** and enter a value for Key.

7. Under **Status**, select **Enable**.
8. In the **Archive settings** section, select one or both of the Archive Access tiers to enable.
9. Choose **Create**.

### Using the AWS CLI

You can use the following AWS CLI commands to manage S3 Intelligent-Tiering configurations:

- `delete-bucket-intelligent-tiering-configuration`
- `get-bucket-intelligent-tiering-configuration`
- `list-bucket-intelligent-tiering-configurations`
- `put-bucket-intelligent-tiering-configuration`

For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

When using the AWS CLI, you cannot specify the configuration as an XML file. You must specify the JSON instead. The following is an example XML S3 Intelligent-Tiering configuration and equivalent JSON that you can specify in an AWS CLI command.

The following example puts an S3 Intelligent-Tiering configuration to the specified bucket.

**Example `put-bucket-intelligent-tiering-configuration`**

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
```

```
    },
    "And": {
      "Prefix": "string",
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
```

```
   </Filter>
   <Status>string</Status>
   <Tiering>
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

#### Using the PUT API operation

You can use the `PutBucketIntelligentTieringConfiguration` operation for a specified bucket and up to 1,000 S3 Intelligent-Tiering configurations per bucket. You can define which objects within a bucket are eligible for the archive access tiers using a shared prefix or object tag. Using a shared prefix or object tag allows you to align to specific business applications, workflows, or internal organizations. You also have the flexibility to activate the Archive Access tier, the Deep Archive Access tier, or both.

### Getting started with S3 Intelligent-Tiering

To learn more about how to use S3 Intelligent-Tiering, see Tutorial: Getting started using S3 Intelligent-Tiering.

## Managing S3 Intelligent-Tiering

The S3 Intelligent-Tiering storage class delivers automatic storage cost savings in three low-latency and high-throughput access tiers. It also offers optional archive capabilities to help you get the lowest storage costs in the cloud for data that can be accessed in minutes to hours. The S3 Intelligent-Tiering storage class supports all Amazon S3 features, including the following:

- S3 Inventory, for verifying the access tier of objects
- S3 Replication, for replicating data to any AWS Region
- S3 Storage Lens, for viewing storage usage and activity metrics
- Server-side encryption, for protecting object data
- S3 Object Lock, for preventing accidental deletion of data
- AWS PrivateLink, for accessing Amazon S3 through a private endpoint in a virtual private cloud (VPC)

## Identifying which S3 Intelligent-Tiering access tier objects are stored in

To get a list of your objects and their corresponding metadata, including their S3 Intelligent-Tiering access tier, you can use the section called "Managing inventory". S3 Inventory provides CSV, ORC, or Parquet output files that list your objects and their corresponding metadata. You can receive these inventory reports on either a daily or weekly basis for an Amazon S3 bucket or a shared prefix. (*Shared prefix* refers to objects that have names that begin with a common string.)

## Viewing the archive status of an object within S3 Intelligent-Tiering

To receive notice when an object within the S3 Intelligent-Tiering storage class has moved to either the Archive Access tier or the Deep Archive Access tier, you can set up S3 Event Notifications. For more information, see Enabling event notifications.

Amazon S3 can publish event notifications to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

The following is an example of a message that Amazon S3 sends to publish an `s3:IntelligentTiering` event. For more information, see the section called "Event message structure".

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
```

```
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
   ]
}
```

You can also use a HEAD object request to view an object's archive status. If an object is stored in the S3 Intelligent-Tiering storage class and is in one of the archive tiers, the HEAD object response shows the current archive tier. To show the archive tier, the request uses the x-amz-archive-status header.

The following HEAD object request returns the metadata of an object (in this case, *my-image.jpg*).

**Example**

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

You can also use HEAD object requests to monitor the status of a `restore-object` request. If the archive restoration is in progress, the HEAD object response includes the x-amz-restore header.

The following sample HEAD object response shows an object archived by using S3 Intelligent-Tiering with a restore request in progress.

**Example**

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

## Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers

To access objects in the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers, you must initiate a restore request, and then wait until the object is moved into the Frequent Access tier. For more information about archived objects, see the section called “Working with archived objects”.

When you restore an object from the Archive Access tier or Deep Archive Access tier, the object moves back into the Frequent Access tier. Afterwards, if the object isn't accessed for 30 consecutive days, it automatically moves into the Infrequent Access tier. Then, after a minimum of 90 consecutive days of no access, the object moves into the Archive Access tier. After a minimum of 180 consecutive days of no access, the object moves into the Deep Archive Access tier. For more information, see the section called “How S3 Intelligent-Tiering works”.

You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the Amazon S3 REST API, the AWS SDKs, or the AWS Command Line Interface (AWS CLI). For more information, see the section called “Working with archived objects”.

# Managing your storage lifecycle

To manage your objects so that they're stored cost effectively throughout their lifecycle, create an *Amazon S3 Lifecycle configuration*. An Amazon S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:

- **Transition actions** – These actions define when objects transition to another storage class. For example, you might choose to transition objects to the S3 Standard-IA storage class 30 days after creating them, or archive objects to the S3 Glacier Flexible Retrieval storage class one year after creating them. For more information, see Using Amazon S3 storage classes.

  There are costs associated with lifecycle transition requests. For pricing information, see Amazon S3 pricing.
- **Expiration actions** – These actions define when objects expire. Amazon S3 deletes expired objects on your behalf.

  Lifecycle expiration costs depend on when you choose to expire objects. For more information, see Expiring objects.

> **Important**
>
> You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

### Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old.

### Changes in billing

If there is any delay between when an object becomes eligible for a lifecycle action and when Amazon S3 transfers or expires your object, billing changes are applied as soon as the object becomes eligible for the lifecycle action. For example, if an object is scheduled to expire and

Amazon S3 doesn't immediately expire the object, you won't be charged for storage after the expiration time.

The one exception to this behavior is if you have a lifecycle rule to transition to the S3 Intelligent-Tiering storage class. In that case, billing changes don't occur until the object has transitioned to S3 Intelligent-Tiering.

For more information about S3 Lifecycle rules, see Lifecycle configuration elements.

**Monitoring the effect of lifecycle rules**

To monitor the effect of updates made by active lifecycle rules, see the section called "How do I monitor the actions taken by my lifecycle rules?".

## Managing object lifecycle

Define S3 Lifecycle configuration rules for objects that have a well-defined lifecycle. For example:

- If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you might want to delete them.
- Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them, but your organization or regulations might require you to archive them for a specific period. After that, you can delete them.
- You might upload some types of data to Amazon S3 primarily for archival purposes. For example, you might archive digital media, financial and healthcare records, raw genomics sequence data, long-term database backups, and data that must be retained for regulatory compliance.

With S3 Lifecycle configuration rules, you can tell Amazon S3 to transition objects to less-expensive storage classes, or archive or delete them.

## Creating a lifecycle configuration

An S3 Lifecycle configuration is an XML file that consists of a set of rules with predefined actions that you want Amazon S3 to perform on objects during their lifetime.

You can create a lifecycle configuration by using the Amazon S3 console, REST API, AWS SDKs, and the AWS Command Line Interface (AWS CLI). For more information, see Setting a lifecycle configuration on a bucket.

Amazon S3 provides a set of REST API operations for managing lifecycle configuration on a bucket. Amazon S3 stores the configuration as a *lifecycle subresource* that's attached to your bucket. For details, see the following:

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

For more information about creating a lifecycle configuration, see the following topics:

**Topics**



## Transitioning objects using Amazon S3 Lifecycle

You can add rules in an S3 Lifecycle configuration to tell Amazon S3 to transition objects to another Amazon S3 storage class. For more information about storage classes, see Using Amazon S3 storage classes. Some examples of when you might use S3 Lifecycle configurations in this way include the following:

- When you know that objects are infrequently accessed, you might transition them to the S3 Standard-IA storage class.
- You might want to archive objects that you don't need to access in real time to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.

### Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today

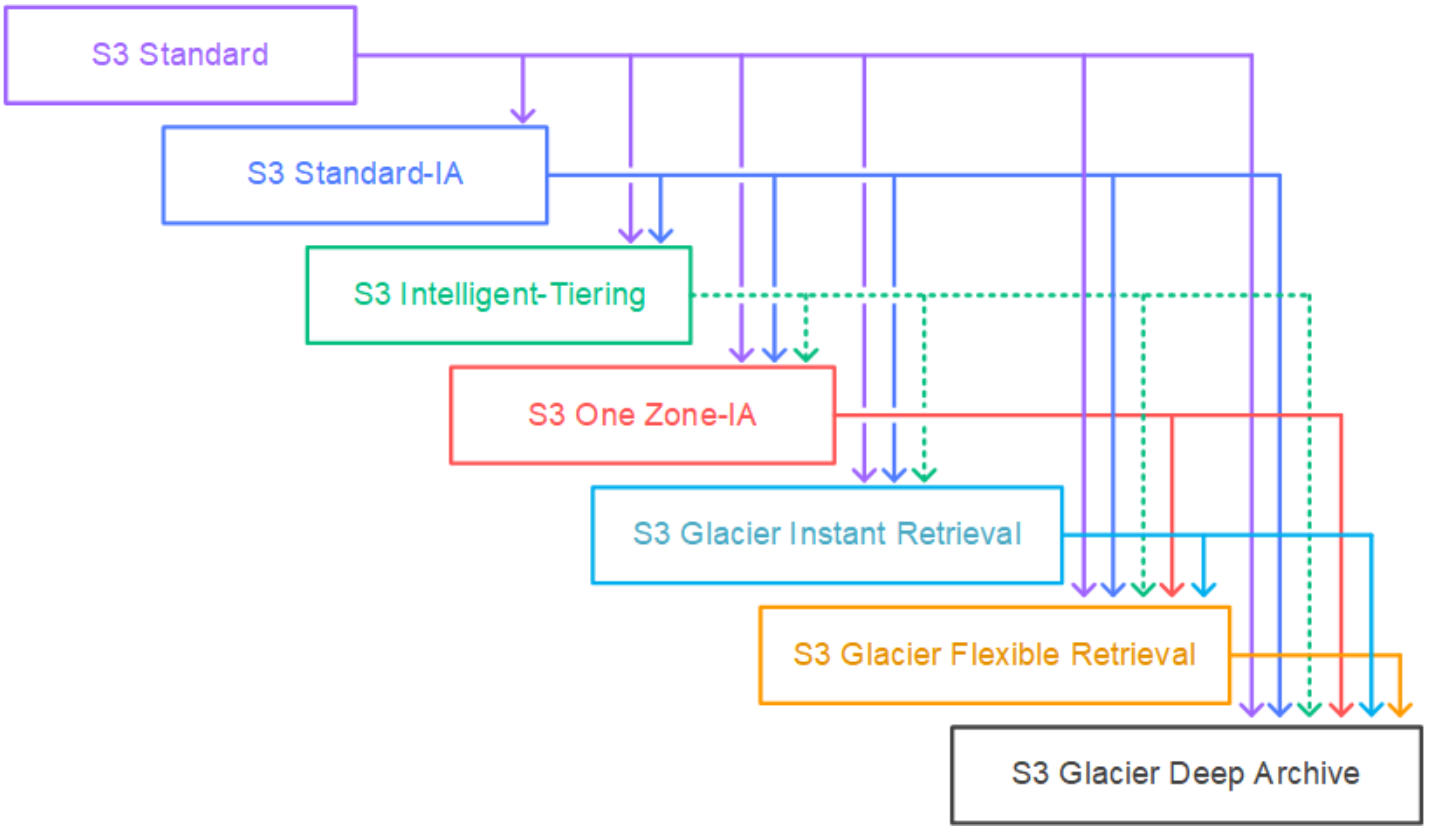
with a transition action that causes objects with a specific prefix to transition to a different storage class 30 days after creation, Amazon S3 will queue for transition any existing objects that are more than 30 days old and that have the specified prefix.

> **Important**
>
> You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

## Supported transitions and related constraints

In an S3 Lifecycle configuration, you can define rules to transition objects from one storage class to another to save on storage costs. When you don't know the access patterns of your objects, or if your access patterns are changing over time, you can transition the objects to the S3 Intelligent-Tiering storage class for automatic cost savings. For information about storage classes, see Using Amazon S3 storage classes.

Amazon S3 supports a waterfall model for transitioning between storage classes, as shown in the following diagram.




**Supported lifecycle transitions**

Amazon S3 supports the following lifecycle transitions between storage classes using an S3 Lifecycle configuration.

You *can transition* from the following:

- The S3 Standard storage class to any other storage class.
- The S3 Standard-IA storage class to the S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
- The S3 Intelligent-Tiering storage class to the S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.

> **Note**
>
> There are some exceptions for transitioning objects from the S3 Intelligent-Tiering storage class to S3 One Zone-IA and some S3 Glacier storage classes. For more information, see the section called “Unsupported lifecycle transitions”.

- The S3 One Zone-IA storage class to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.
- The S3 Glacier Instant Retrieval storage class to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.
- The S3 Glacier Flexible Retrieval storage class to the S3 Glacier Deep Archive storage class.
- Any storage class to the S3 Glacier Deep Archive storage class.

> **Note**
>
> There are no data retrieval charges for lifecycle transitions. However, there are per-request ingestion charges when using PUT, COPY, or lifecycle rules to move data into any S3 storage class. Consider the ingestion or transition cost before moving objects into any storage class. For more information about cost considerations, see Amazon S3 pricing.

### Unsupported lifecycle transitions

Amazon S3 does not support any of the following lifecycle transitions.

You *can't transition* from the following:

- For versioning enabled or versioning suspended buckets, any objects with a `Pending` replication status.
- Any storage class to the S3 Standard storage class.
- Any storage class to the Reduced Redundancy Storage (RRS) class.
- The S3 One Zone-IA storage class to the S3 Intelligent-Tiering, S3 Standard-IA, or S3 Glacier Instant Retrieval storage classes.
- The S3 Intelligent-Tiering storage class (all tiers) to the S3 Standard-IA storage class.
- The S3 Intelligent-Tiering storage class Archive Instant Access tier to S3 One Zone-IA.
- The S3 Intelligent-Tiering storage class Archive Access tier to S3 One Zone-IA or S3 Glacier Instant Retrieval.
- The S3 Intelligent-Tiering storage class Deep Archive Access tier to S3 One Zone-IA, S3 Glacier Instant Retrieval, or S3 Glacier Flexible Retrieval.

### Constraints

Lifecycle storage class transitions have the following constraints:

#### Object Size and Transitions from S3 Standard or S3 Standard-IA to S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA

When you transition objects from the S3 Standard or S3 Standard-IA storage classes to S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA, the following object size constraints apply:

- **Larger objects** – For the following transitions, there is a cost benefit to transitioning larger objects:
  - From the S3 Standard or S3 Standard-IA storage classes to S3 Intelligent-Tiering.
  - From the S3 Standard storage class to S3 Standard-IA or S3 One Zone-IA.
- **Objects smaller than 128 KiB** – For the following transitions, Amazon S3 does not transition objects that are smaller than 128 KiB:
  - From the S3 Standard or S3 Standard-IA storage classes to S3 Intelligent-Tiering or S3 Glacier Instant Retrieval.
  - From the S3 Standard storage class to S3 Standard-IA or S3 One Zone-IA.

> **Note**
>
> You can filter lifecycle rules based on object size.

> **Important**
>
> When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:
>
> - Permanent deletion takes precedence over transition.
> - Transition takes precedence over creation of delete markers.
> - When an object is eligible for both a S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

**Minimum Days for Transition to S3 Standard-IA or S3 One Zone-IA**

Before you transition objects to S3 Standard-IA or S3 One Zone-IA, you must store them for at least 30 days in Amazon S3. For example, you cannot create a Lifecycle rule to transition objects to the S3 Standard-IA storage class one day after you create them. Amazon S3 doesn't support this transition within the first 30 days because newer objects are often accessed more frequently or deleted sooner than is suitable for S3 Standard-IA or S3 One Zone-IA storage.

Similarly, if you are transitioning noncurrent objects (in versioned buckets), you can transition only objects that are at least 30 days noncurrent to S3 Standard-IA or S3 One Zone-IA storage. For a list of minimum storage duration for all storage class, see Comparing the Amazon S3 storage classes.

**Minimum 30-Day Storage Charge for S3 Standard-IA and S3 One Zone-IA**

The S3 Standard-IA and S3 One Zone-IA storage classes have a minimum 30-day storage charge. Therefore, you can't specify a single Lifecycle rule for both an S3 Standard-IA or S3 One Zone-IA transition and an S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive transition when the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive transition occurs less than 30 days after the S3 Standard-IA or S3 One Zone-IA transition.

The same 30-day minimum applies when you specify a transition from S3 Standard-IA storage to S3 One Zone-IA. You can specify two rules to accomplish this, but you pay minimum storage charges. For more information about cost considerations, see Amazon S3 pricing.

**Manage an object's complete lifecycle**

You can combine these S3 Lifecycle actions to manage an object's complete lifecycle. For example, suppose that the objects you create have a well-defined lifecycle. Initially, the objects are frequently accessed for a period of 30 days. Then, objects are infrequently accessed for up to 90 days. After that, the objects are no longer needed, so you might choose to archive or delete them.

In this scenario, you can create an S3 Lifecycle rule in which you specify the initial transition action to S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage, another transition action to S3 Glacier Flexible Retrieval storage for archiving, and an expiration action. As you move the

objects from one storage class to another, you save on storage costs. For more information about cost considerations, see Amazon S3 pricing.

## Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival)

By using an S3 Lifecycle configuration, you can transition objects to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes for archiving. When you choose the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class, your objects remain in Amazon S3. You cannot access them directly through the separate Amazon S3 Glacier service. For more general information about S3 Glacier see, What is Amazon S3 Glacier in the *Amazon S3 Glacier Developer Guide*.

Before you archive objects, review the following sections for relevant considerations.

### General considerations

The following are the general considerations for you to consider before you archive objects:

- Encrypted objects remain encrypted throughout the storage class transition process.
- Objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes are not available in real time.

  Archived objects are Amazon S3 objects, but before you can access an archived object, you must first restore a temporary copy of it. The restored object copy is available only for the duration that you specify in the restore request. After that, Amazon S3 deletes the temporary copy, and the object remains archived in S3 Glacier Flexible Retrieval.

  You can restore an object by using the Amazon S3 console or programmatically by using the AWS SDK wrapper libraries or the Amazon S3 REST API in your code. For more information, see Restoring an archived object.
- Objects that are stored in the S3 Glacier Flexible Retrieval storage class can only be transitioned to the S3 Glacier Deep Archive storage class.

  You can use an S3 Lifecycle configuration rule to convert the storage class of an object from S3 Glacier Flexible Retrieval to the S3 Glacier Deep Archive storage class only. If you want to change the storage class of an object that is stored in S3 Glacier Flexible Retrieval to a storage class other than S3 Glacier Deep Archive, you must use the restore operation to make a temporary copy of the object first. Then use the copy operation to overwrite the object specifying S3

Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, or Reduced Redundancy as the storage class.

- The transition of objects to the S3 Glacier Deep Archive storage class can go only one way.

  You cannot use an S3 Lifecycle configuration rule to convert the storage class of an object from S3 Glacier Deep Archive to any other storage class. If you want to change the storage class of an archived object to another storage class, you must use the restore operation to make a temporary copy of the object first. Then use the copy operation to overwrite the object specifying S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or Reduced Redundancy Storage as the storage class.

  > **Note**
  >
  > The Copy operation for restored objects isn't supported in the Amazon S3 console for objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For this type of Copy operation, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the REST API.

  The objects that are stored in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes are visible and available only through Amazon S3. They are not available through the separate Amazon S3 Glacier service.

  These are Amazon S3 objects, and you can access them only by using the Amazon S3 console or the Amazon S3 API. You cannot access the archived objects through the separate Amazon S3 Glacier console or the Amazon S3 Glacier API.

### Cost considerations

If you are planning to archive infrequently accessed data for a period of months or years, the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes can reduce your storage costs. However, to ensure that the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class is appropriate for you, consider the following:

- **Storage overhead charges** – When you transition objects to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class, a fixed amount of storage is added to each object to accommodate metadata for managing the object.

- For each object archived to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. Amazon S3 stores this metadata so that you can get a real-time list of your archived objects by using the Amazon S3 API. For more information, see Get Bucket (List Objects). You are charged S3 Standard rates for this additional storage.
- For each object that is archived to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive rates for this additional storage.

If you are archiving small objects, consider these storage charges. Also consider aggregating many small objects into a smaller number of large objects to reduce overhead costs.

- **Number of days you plan to keep objects archived** – S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are long-term archival solutions. The minimal storage duration period is 90 days for the S3 Glacier Flexible Retrieval storage class and 180 days for S3 Glacier Deep Archive. Deleting data that is archived to Amazon S3 Glacier doesn't incur charges if the objects you delete are archived for more than the minimal storage duration period. If you delete or overwrite an archived object within the minimal duration period, Amazon S3 charges a prorated early deletion fee. For information about the early deletion fee, see the "How am I charged for deleting objects from Amazon S3 Glacier that are less than 90 days old?" question on the Amazon S3 FAQ.
- **S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive transition request charges** – Each object that you transition to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class constitutes one transition request. There is a cost for each such request. If you plan to transition a large number of objects, consider the request costs. If you are archiving a mix of objects that includes small objects, especially those under 128KB, we recommend using the lifecycle object size filter to filter out small objects from your transition to reduce request costs. S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive do not automatically block transition of objects under 128KB.
- **S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive data restore charges** – S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are designed for long-term archival of data that you access infrequently. For information about data restoration charges, see the "How much does it cost to retrieve data from Amazon S3 Glacier?" question on the Amazon S3 FAQ. For information about how to restore data from Amazon S3 Glacier, see Restoring an archived object.

When you archive objects to Amazon S3 Glacier by using S3 Lifecycle management, Amazon S3 transitions these objects asynchronously. There might be a delay between the transition date in the S3 Lifecycle configuration rule and the date of the physical transition. You are charged Amazon S3 Glacier prices based on the transition date specified in the rule. For more information, see the Amazon S3 Glacier section of the Amazon S3 FAQ.

The Amazon S3 product detail page provides pricing information and example calculations for archiving Amazon S3 objects. For more information, see the following topics:

- "How is my storage charge calculated for Amazon S3 objects archived to Amazon S3 Glacier?" on the Amazon S3 FAQ.
- "How am I charged for deleting objects from Amazon S3 Glacier that are less than 90 days old?" on the Amazon S3 FAQ.
- "How much does it cost to retrieve data from Amazon S3 Glacier?" on the Amazon S3 FAQ.
- Amazon S3 pricing for storage costs for the different storage classes.

### Restoring archived objects

Archived objects aren't accessible in real time. You must first initiate a restore request and then wait until a temporary copy of the object is available for the duration that you specify in the request. After you receive a temporary copy of the restored object, the object's storage class remains S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. (A HeadObject or GetObject API operation request will return S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive as the storage class.)

> **Note**
>
> When you restore an archive, you are paying for both the archive (S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive rate) and a copy that you restored temporarily (S3 Standard storage rate). For information about pricing, see Amazon S3 pricing.

You can restore an object copy programmatically or by using the Amazon S3 console. Amazon S3 processes only one restore request at a time per object. For more information, see Restoring an archived object.

# Expiring objects

When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 takes an action based on which S3 Versioning state the bucket is in.

- **Nonversioned bucket** – Amazon S3 queues the object for removal and removes it asynchronously, permanently removing the object.
- **Versioning-enabled bucket** – If the current object version is not a delete marker, Amazon S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and the delete marker the current version.
- **Versioning-suspended bucket** – Amazon S3 creates a delete marker with null as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.

For a versioned bucket (that is, versioning-enabled or versioning-suspended), there are several considerations that guide how Amazon S3 handles the Expiration action. For versioning-enabled or versioning-suspended buckets, the following applies:

- Object expiration applies only to an object's current version (it has no impact on noncurrent object versions).
- Amazon S3 doesn't take any action if there are one or more object versions and the delete marker is the current version.
- If the current object version is the only object version and it is also a delete marker (also referred as an *expired object delete marker*, where all object versions are deleted and you only have a delete marker remaining), Amazon S3 removes the expired object delete marker. You can also use the expiration action to direct Amazon S3 to remove any expired object delete markers. For example, see Example 7: Removing expired object delete markers.
- You can use the `NoncurrentVersionExpiration` action element to direct Amazon S3 to permanently delete noncurrent versions of objects. These deleted objects can't be recovered. You can base this expiration on a certain number of days since the objects became noncurrent. In addition to the number of days, you can also provide a maximum number of noncurrent versions to retain (between 1 and 100). This value specifies how many newer noncurrent versions must exist before Amazon S3 can perform the associated action on a given version. To specify the maximum number of noncurrent versions, you must also provide a `Filter` element. If you don't specify a `Filter` element, Amazon S3 generates an `InvalidRequest` error when you provide a maximum number of noncurrent versions. For more information about using the

`NoncurrentVersionExpiration` action element, see the section called "Elements to describe lifecycle actions".

- Amazon S3 doesn't take any action on noncurrent versions of objects that have the S3 Object Lock configuration applied.
- For objects with a `Pending` replication status, Amazon S3 doesn't take any action on current or non-current versions of objects.

For more information, see Using versioning in S3 buckets.

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and an S3 Standard-IA (or an S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

### Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects with a specific prefix to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old and that have the specified prefix.

**Important**

You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

## How to find when objects will expire

To find when an object is scheduled to expire, use the HeadObject or GetObject API operation. These API operations return response headers that provide the date and time at which the object is no longer cacheable.

**Note**

- There may be a delay between the expiration date and the date at which Amazon S3 removes an object. You are not charged for expiration or the storage time associated with an object that has expired.
- Before updating, disabling, or deleting Lifecycle rules, use the `LIST` API operations (such as ListObjectsV2, ListObjectVersions, and ListMultipartUploads) or Amazon S3 Inventory to verify that Amazon S3 has transitioned and expired eligible objects based on your use cases.

## Minimum storage duration charge

If you create an S3 Lifecycle expiration rule that causes objects that have been in S3 Standard-IA or S3 One Zone-IA storage for less than 30 days to expire, you are charged for 30 days. If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Flexible Retrieval storage for less than 90 days to expire, you are charged for 90 days. If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Deep Archive storage for less than 180 days to expire, you are charged for 180 days.

For more information, see Amazon S3 pricing.

# Setting a lifecycle configuration on a bucket

This section explains how you can set an Amazon S3 Lifecycle configuration on a bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API. For information about S3 Lifecycle configuration, see Managing your storage lifecycle.

You can use lifecycle rules to define actions that you want Amazon S3 to take during an object's lifetime (for example, transition objects to another storage class, archive them, or delete them after a specified period of time).

Before you set a lifecycle configuration, note the following:

**Lifecycle configuration propagation delay**

When you add an S3 Lifecycle configuration to a bucket, there is usually some lag before a new or updated Lifecycle configuration is fully propagated to all the Amazon S3 systems. Expect a delay of a few minutes before the configuration fully takes effect. This delay can also occur when you delete an S3 Lifecycle configuration.

**Transition or expiration delay**

There's a delay between when a lifecycle rule is satisfied and when the action for the rule is completed. For example, suppose that a set of objects is expired by a lifecycle rule on January 1. Even though the expiration rule has been satisfied on January 1, Amazon S3 might not actually delete these objects until days or even weeks later. This delay occurs because S3 Lifecycle queues objects for transitions or expirations asynchronously. However, changes in billing are usually applied when the lifecycle rule is satisfied, even if the action isn't complete. For more information, see Changes in billing. To monitor the effect of updates made by active lifecycle rules, see the section called “How do I monitor the actions taken by my lifecycle rules?”

**Disabling or deleting lifecycle rules**

When you disable or delete lifecycle rules, Amazon S3 stops scheduling new objects for deletion or transition after a small delay. Any objects that were already scheduled are unscheduled and are not deleted or transitioned.

> **Note**
>
> Before updating, disabling, or deleting lifecycle rules, use the `LIST` API operations (such as ListObjectsV2, ListObjectVersions, and ListMultipartUploads) or Amazon S3 Inventory

to verify that Amazon S3 has transitioned and expired eligible objects based on your use cases. If you're experiencing any issues with updating, disabling, or deleting lifecycle rules, see Troubleshoot Amazon S3 Lifecycle issues.

### Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects with a specific prefix to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old and that have the specified prefix.

### Monitoring the effect of lifecycle rules

To monitor the effect of updates made by active lifecycle rules, see the section called “How do I monitor the actions taken by my lifecycle rules?”

### Changes in billing

There might be a lag between when the Lifecycle configuration rules are satisfied and when the action triggered by satisfying the rule is taken. However, changes in billing happen as soon as the Lifecycle configuration rule is satisfied, even if the action isn't yet taken.

For example, after the object expiration time, you aren't charged for storage, even if the object isn't deleted immediately. Likewise, as soon as the object transition time elapses, you're charged S3 Glacier Flexible Retrieval storage rates, even if the object isn't immediately transitioned to the S3 Glacier Flexible Retrieval storage class.

However, lifecycle transitions to the S3 Intelligent-Tiering storage class are the exception. Changes in billing don't happen until after the object has transitioned into the S3 Intelligent-Tiering storage class.

### Multiple or conflicting rules

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.

- When an object is eligible for both an S3 Glacier Flexible Retrieval and an S3 Standard-IA (or an S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

**Using the S3 console**

You can define lifecycle rules for all objects or a subset of objects in a bucket by using a shared prefix (objects names that begin with a common string) or a tag. In your lifecycle rule, you can define actions specific to current and noncurrent object versions. For more information, see the following:

- Managing your storage lifecycle
- Using versioning in S3 buckets

**To create a lifecycle rule**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to create a lifecycle rule for.
3. Choose the **Management** tab, and choose **Create lifecycle rule**.
4. In **Lifecycle rule name**, enter a name for your rule.

   The name must be unique within the bucket.
5. Choose the scope of the lifecycle rule:

   - To apply this lifecycle rule to *all objects with a specific prefix or tag*, choose **Limit the scope to specific prefixes or tags**.
     - To limit the scope by prefix, in **Prefix**, enter the prefix.
     - To limit the scope by tag, choose **Add tag**, and enter the tag key and value.

     For more information about object name prefixes, see Creating object key names. For more information about object tags, see Categorizing your storage using tags.
   - To apply this lifecycle rule to *all objects in the bucket*, choose **This rule applies to *all* objects in the bucket**, and then choose **I acknowledge that this rule applies to all objects in the bucket**.

6. To filter a rule by object size, you can select **Specify minimum object size**, **Specify maximum object size**, or both options.

   - When you're specifying a value for **Minimum object size** or **Maximum object size**, the value must be larger than 0 bytes and up to 5 TB. You can specify this value in bytes, KB, MB, or GB.
   - When you're specifying both values, the maximum object size must be larger than the minimum object size.

   > **Note**
   >
   > The **Minimum object size** and **Maximum object size** filters exclude the specified values. For example, if you set a filter to expire objects that have a **Minimum object size** of 128 KB, objects that are exactly 128 KB don't expire. Instead, the rule applies only to objects that are greater than 128 KB in size.

7. Under **Lifecycle rule actions**, choose the actions that you want your lifecycle rule to perform:

   - Transition *current* versions of objects between storage classes
   - Transition *previous* versions of objects between storage classes
   - Expire *current* versions of objects

   > **Note**
   >
   > For buckets that don't have S3 Versioning enabled, expiring current versions causes Amazon S3 to permanently delete the objects. For more information, see the section called “Lifecycle actions and bucket versioning state”.

   - Permanently delete *previous* versions of objects
   - Delete expired delete markers or incomplete multipart uploads

   Depending on the actions that you choose, different options appear.

8. To transition *current* versions of objects between storage classes, under **Transition current versions of objects between storage classes**:

a. In **Storage class transitions**, choose the storage class to transition to. For a list of possible transitions, see the section called "Supported lifecycle transitions". You can choose from the following storage classes:

- S3 Standard-IA
- S3 Intelligent-Tiering
- S3 One Zone-IA
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

b. In **Days after object creation**, enter the number of days after creation to transition the object.

For more information about storage classes, see Using Amazon S3 storage classes. You can define transitions for current or previous object versions or for both current and previous versions. Versioning enables you to keep multiple versions of an object in one bucket. For more information about versioning, see Using the S3 console.

> **Important**
>
> When you choose the S3 Glacier Flexible Retrieval or Glacier Deep Archive storage class, your objects remain in Amazon S3. You cannot access them directly through the separate Amazon S3 Glacier service. For more information, see Transitioning objects using Amazon S3 Lifecycle.

9. To transition *noncurrent* versions of objects between storage classes, under **Transition noncurrent versions of objects between storage classes**:

a. In **Storage class transitions**, choose the storage class to transition to. For a list of possible transitions, see the section called "Supported lifecycle transitions". You can choose from the following storage classes:

- S3 Standard-IA
- S3 Intelligent-Tiering
- S3 One Zone-IA
- S3 Glacier Flexible Retrieval

- S3 Glacier Deep Archive

b. In **Days after object becomes noncurrent**, enter the number of days after creation to transition the object.

10. To expire *current* versions of objects, under **Expire current versions of objects**, in **Number of days after object creation**, enter the number of days.

> **Important**
>
> In a nonversioned bucket the expiration action results in Amazon S3 permanently removing the object. For more information about lifecycle actions, see Elements to describe lifecycle actions.

11. To permanently delete previous versions of objects, under **Permanently delete noncurrent versions of objects**, in **Days after objects become noncurrent**, enter the number of days. You can optionally specify the number of newer versions to retain by entering a value under **Number of newer versions to retain**.
12. Under **Delete expired delete markers or incomplete multipart uploads**, choose **Delete expired object delete markers** and **Delete incomplete multipart uploads**. Then, enter the number of days after the multipart upload initiation that you want to end and clean up incomplete multipart uploads.

    For more information about multipart uploads, see Uploading and copying objects using multipart upload.
13. Choose **Create rule**.

    If the rule does not contain any errors, Amazon S3 enables it, and you can see it on the **Management** tab under **Lifecycle rules**.

For information about AWS CloudFormation templates and examples, see Working with AWS CloudFormation templates and AWS::S3::Bucket in the *AWS CloudFormation User Guide*.

**Using the AWS CLI**

You can use the following AWS CLI commands to manage S3 Lifecycle configurations:

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`

- `delete-bucket-lifecycle`

For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

The Amazon S3 Lifecycle configuration is an XML file. But when you're using the AWS CLI, you cannot specify the XML format. You must specify the JSON format instead. The following are example XML lifecycle configurations and the equivalent JSON configurations that you can specify in an AWS CLIcommand.

Consider the following example S3 Lifecycle configuration.

**Example Example 1**

**Example**

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
```

```
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
        }
```

### Example Example 2

### Example

XML

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
```

```
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

You can test the `put-bucket-lifecycle-configuration` as follows.

**To test the configuration**

1. Save the JSON Lifecycle configuration in a file (for example, *`lifecycle.json`*).
2. Run the following AWS CLI command to set the Lifecycle configuration on your bucket. Replace the *`user input placeholders`* with your own information.

```
$ aws s3api put-bucket-lifecycle-configuration  \
```

```
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

3. To verify, retrieve the S3 Lifecycle configuration by using the `get-bucket-lifecycle-configuration` AWS CLI command as follows:

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket
```

4. To delete the S3 Lifecycle configuration, use the `delete-bucket-lifecycle` AWS CLI command as follows:

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

### Using the AWS SDKs

Java

You can use the AWS SDK for Java to manage the S3 Lifecycle configuration of a bucket. For more information about managing S3 Lifecycle configuration, see Managing your storage lifecycle.

> **Note**
>
> When you add S3 Lifecycle configuration to a bucket, Amazon S3 replaces the bucket's current Lifecycle configuration, if there is one. To update a configuration, you retrieve it, make the desired changes, and then add the revised configuration to the bucket.

The following example shows how to use the AWS SDK for Java to add, update, and delete the Lifecycle configuration of a bucket. The example does the following:

- Adds a Lifecycle configuration to a bucket.
- Retrieves the Lifecycle configuration and updates it by adding another rule.
- Adds the modified Lifecycle configuration to the bucket. Amazon S3 replaces the existing configuration.

- Retrieves the configuration again and verifies that it has the right number of rules by printing the number of rules.
- Deletes the Lifecycle configuration and verifies that it has been deleted by attempting to retrieve it again.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

import java.io.IOException;
import java.util.Arrays;

public class LifecycleConfiguration {

        public static void main(String[] args) throws IOException {
                Regions clientRegion = Regions.DEFAULT_REGION;
                String bucketName = "*** Bucket name ***";

                // Create a rule to archive objects with the "glacierobjects/"
 prefix to Glacier
                // immediately.
                BucketLifecycleConfiguration.Rule rule1 = new
 BucketLifecycleConfiguration.Rule()
                                .withId("Archive immediately rule")
                                .withFilter(new LifecycleFilter(new
 LifecyclePrefixPredicate("glacierobjects/")))
```

```
                                .addTransition(new
Transition().withDays(0).withStorageClass(StorageClass.Glacier))
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Create a rule to transition objects to the Standard-Infrequent
Access storage
                // class
                // after 30 days, then to Glacier after 365 days. Amazon S3 will
delete the
                // objects after 3650 days.
                // The rule applies to all objects with the tag "archive" set to
"true".
                BucketLifecycleConfiguration.Rule rule2 = new
BucketLifecycleConfiguration.Rule()
                                .withId("Archive and then delete rule")
                                .withFilter(new LifecycleFilter(new
LifecycleTagPredicate(new Tag("archive", "true"))))
                                .addTransition(new Transition().withDays(30)

.withStorageClass(StorageClass.StandardInfrequentAccess))
                                .addTransition(new
Transition().withDays(365).withStorageClass(StorageClass.Glacier))
                                .withExpirationInDays(3650)
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Add the rules to a new BucketLifecycleConfiguration.
                BucketLifecycleConfiguration configuration = new
BucketLifecycleConfiguration()
                                .withRules(Arrays.asList(rule1, rule2));

                try {
                        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                                        .withCredentials(new
ProfileCredentialsProvider())
                                        .withRegion(clientRegion)
                                        .build();

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);
```

```
                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        configuration.getRules().add(new
BucketLifecycleConfiguration.Rule().withId("NewRule")
                                        .withFilter(new LifecycleFilter(new
LifecycleAndOperator(
                                                        Arrays.asList(new
LifecyclePrefixPredicate("YearlyDocuments/"),
                                                                        new
LifecycleTagPredicate(new Tag(

   "expire_after",

   "ten_years"))))))
                                        .withExpirationInDays(3650)

.withStatus(BucketLifecycleConfiguration.ENABLED));

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration now has three rules.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);
                        System.out.println("Expected # of rules = 3; found: " +
configuration.getRules().size());

                        // Delete the configuration.
                        s3Client.deleteBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration has been deleted by
attempting to retrieve it.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);
                        String s = (configuration == null) ? "No configuration
found." : "Configuration found.";
                        System.out.println(s);
                } catch (AmazonServiceException e) {
```

```
                        // The call was transmitted successfully, but Amazon S3
 couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
 client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }
}
```

.NET

You can use the AWS SDK for .NET to manage the S3 Lifecycle configuration on a bucket. For more information about managing Lifecycle configuration, see Managing your storage lifecycle.

> **Note**
>
> When you add a Lifecycle configuration, Amazon S3 replaces the existing configuration on the specified bucket. To update a configuration, you must first retrieve the Lifecycle configuration, make the changes, and then add the revised Lifecycle configuration to the bucket.

The following example shows how to use the AWS SDK for .NET to add, update, and delete a bucket's Lifecycle configuration. The code example does the following:

- Adds a Lifecycle configuration to a bucket.
- Retrieves the Lifecycle configuration and updates it by adding another rule.
- Adds the modified Lifecycle configuration to the bucket. Amazon S3 replaces the existing Lifecycle configuration.
- Retrieves the configuration again and verifies it by printing the number of rules in the configuration.
- Deletes the Lifecycle configuration and verifies the deletion.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
                                 Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
 LifecyclePrefixPredicate()
                                     {
                                         Prefix = "glacierobjects/"
                                     }
```

```
                                    },
                                    Status = LifecycleRuleStatus.Enabled,
                                    Transitions = new List<LifecycleTransition>
                                    {
                                         new LifecycleTransition
                                         {
                                              Days = 0,
                                              StorageClass = S3StorageClass.Glacier
                                         }
                                     },
                               },
                               new LifecycleRule
                               {
                                    Id = "Archive and then delete rule",
                                     Filter = new LifecycleFilter()
                                    {
                                        LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                        {
                                            Prefix = "projectdocs/"
                                        }
                                    },
                                    Status = LifecycleRuleStatus.Enabled,
                                    Transitions = new List<LifecycleTransition>
                                    {
                                         new LifecycleTransition
                                         {
                                              Days = 30,
                                              StorageClass =
S3StorageClass.StandardInfrequentAccess
                                         },
                                         new LifecycleTransition
                                         {
                                           Days = 365,
                                           StorageClass = S3StorageClass.Glacier
                                         }
                                    },
                                    Expiration = new LifecycleRuleExpiration()
                                    {
                                          Days = 3650
                                    }
                               }
                          }
                };
```

```
                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
```

```
        }
    }
}
```

Ruby

You can use the AWS SDK for Ruby to manage S3 Lifecycle configuration on a bucket by using the class AWS::S3::BucketLifecycleConfiguration. For more information about managing lifecycle configuration, see Managing your storage lifecycle.

#### Using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API related to the S3 Lifecycle configuration.

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

### Troubleshooting S3 Lifecycle

For common issues that might occur when working with S3 Lifecycle, see the section called "Troubleshoot lifecycle issues".

## Lifecycle and other bucket configurations

In addition to S3 Lifecycle configurations, you can associate other configurations with your bucket. This section explains how S3 Lifecycle configuration relates to other bucket configurations.

### Lifecycle and versioning

You can add S3 Lifecycle configurations to unversioned buckets and versioning-enabled buckets. For more information, see Using versioning in S3 buckets.

A versioning-enabled bucket maintains one current object version, and zero or more noncurrent object versions. You can define separate Lifecycle rules for current and noncurrent object versions.

For more information, see Lifecycle configuration elements.

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both a S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

## Lifecycle configuration on MFA-enabled buckets

Lifecycle configuration on multi-factor authentication (MFA)-enabled buckets is not supported.

## Lifecycle and logging

Amazon S3 Lifecycle actions are not captured by AWS CloudTrail object level logging. CloudTrail captures API requests made to external Amazon S3 endpoints, whereas S3 Lifecycle actions are performed using internal Amazon S3 endpoints. Amazon S3 server access logs can be enabled in an S3 bucket to capture S3 Lifecycle-related actions such as object transition to another storage class and object expiration resulting in permanent deletion or logical deletion. For more information, see the section called “Logging server access”.

If you have logging enabled on your bucket, Amazon S3 server access logs report the results of the following operations.

| Operation log | Description |
| --- | --- |
| `S3.EXPIRE.OBJECT` | Amazon S3 permanently deletes the object because of the Lifecycle expiration action. |

| Operation log | Description |
| --- | --- |
| S3.CREATE.DELETEMARKER | Amazon S3 logically deletes the current version and adds a delete marker in a versioning-enabled bucket. |
| S3.TRANSITION_SIA.OBJECT | Amazon S3 transitions the object to the S3 Standard-IA storage class. |
| S3.TRANSITION_ZIA.OBJECT | Amazon S3 transitions the object to the S3 One Zone-IA storage class. |
| S3.TRANSITION_INT.OBJECT | Amazon S3 transitions the object to the S3 Intelligent-Tiering storage class. |
| S3.TRANSITION_GIR.OBJECT | Amazon S3 initiates the transition of the object to the S3 Glacier Instant Retrieval storage class. |
| S3.TRANSITION.OBJECT | Amazon S3 initiates the transition of the object to the S3 Glacier Flexible Retrieval storage class. |
| S3.TRANSITION_GDA.OBJECT | Amazon S3 initiates the transition of the object to the S3 Glacier Deep Archive storage class. |
| S3.DELETE.UPLOAD | Amazon S3 aborts an incomplete multipart upload. |

**Note**

Amazon S3 server access log records are generally delivered on a best-effort basis and cannot be used for complete accounting of all Amazon S3 requests.

### Troubleshooting S3 Lifecycle

For more information about troubleshooting common issues with S3 Lifecycle, see Troubleshoot Amazon S3 Lifecycle issues.

### More info

- Lifecycle configuration elements
- Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival)
- Setting a lifecycle configuration on a bucket

## Configuring Lifecycle event notifications

You can set up an Amazon S3 event notification to receive notice when Amazon S3 deletes an object or transitions it to another Amazon S3 storage class following an S3 Lifecycle rule.

By using the `LifecycleExpiration` event types, you can receive notifications whenever Amazon S3 deletes an object based on your S3 Lifecycle configuration. The `s3:LifecycleExpiration:Delete` event type notifies you when an object in an unversioned bucket is deleted. It also notifies you when an object version is permanently deleted by an S3 Lifecycle configuration. The `s3:LifecycleExpiration:DeleteMarkerCreated` event type notifies you when S3 Lifecycle creates a delete marker when a current version of an object in a versioned bucket is deleted. For more information, see Delete object version.

By using the `s3:LifecycleTransition` event type, you can receive notification when an object is transitioned from one Amazon S3 storage class to another by an S3 Lifecycle configuration.

Amazon S3 can publish event notifications to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

For instructions on how to configure Amazon S3 Event Notifications, see Enabling event notifications.

The following is an example of a message that Amazon S3 sends to publish an `s3:LifecycleExpiration:Delete` event. For more information, see Event message structure.

```
{
```

```
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"expiration/delete",
               "sequencer":"0055AED6DCD90281E5",
            }
         }
      }
   ]
}
```

Messages that Amazon S3 sends to publish an `s3:LifecycleTransition` event also include the following information.

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
```

```
    }
}
```

## Lifecycle configuration elements

### Topics



You specify an Amazon S3 Lifecycle configuration as XML, consisting of one or more Lifecycle rules.

```
<LifecycleConfiguration>
    <Rule>
         ...
    </Rule>
    <Rule>
         ...
    </Rule>
</LifecycleConfiguration>
```

Each rule consists of the following:

- Rule metadata that includes a rule ID, and a status that indicates whether the rule is enabled or disabled. If a rule is disabled, Amazon S3 doesn't perform any actions specified in the rule.
- A filter that identifies the objects to which the rule applies. You can specify a filter by using the object size, the object key prefix, one or more object tags, or a combination of filters.
- One or more transition or expiration actions with a date or a time period in the object's lifetime when you want Amazon S3 to perform the specified action.

The following sections describe the XML elements in an S3 Lifecycle configuration. For example configurations, see Examples of S3 Lifecycle configuration.

### ID element

An S3 Lifecycle configuration can have up to 1,000 rules. This limit is not adjustable. The `<ID>` element uniquely identifies a rule. ID length is limited to 255 characters.

### Status element

The `<Status>` element value can be either `Enabled` or `Disabled`. If a rule is disabled, Amazon S3 doesn't perform any of the actions defined in the rule.

### Filter element

A Lifecycle rule can apply to all or a subset of objects in a bucket based on the `<Filter>` element that you specify in the Lifecycle rule.

You can filter objects by key prefix, object tags, or a combination of both (in which case Amazon S3 uses a logical `AND` to combine the filters). Consider the following examples:

- **Specifying a filter by using key prefixes** – This example shows an S3 Lifecycle rule that applies to a subset of objects based on the key name prefix (`logs/`). For example, the Lifecycle rule applies to the objects `logs/mylog.txt`, `logs/temp1.txt`, and `logs/test.txt`. The rule does not apply to the object `example.jpg`.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

  If you want to apply a lifecycle action to a subset of objects based on different key name prefixes, specify separate rules. In each rule, specify a prefix-based filter. For example, to describe a lifecycle action for objects with the key prefixes `projectA/` and `projectB/`, you specify two rules as follows:

```
<LifecycleConfiguration>
```

```
    <Rule>
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

For more information about object keys, see Creating object key names.

- **Specifying a filter based on object tags** – In the following example, the Lifecycle rule specifies a filter based on a tag (*key*) and value (*value*). The rule then applies only to a subset of objects with the specific tag.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions
        ...
    </Rule>
</LifecycleConfiguration>
```

You can specify a filter based on multiple tags. You must wrap the tags in the <And> element, as shown in the following example. The rule directs Amazon S3 to perform lifecycle actions on objects with two tags (with the specific tag key and value).

```
<LifecycleConfiguration>
    <Rule>
```

```
      <Filter>
         <And>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
         </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

The Lifecycle rule applies to objects that have both of the tags specified. Amazon S3 performs a logical AND. Note the following:

- Each tag must match *both* the key and value exactly. If you specify only a <Key> element and no <Value> element, the rule will apply only to objects that match the tag key and that do *not* have a value specified.
- The rule applies to a subset of objects that has all the tags specified in the rule. If an object has additional tags specified, the rule will still apply.

> **Note**
>
> When you specify multiple tags in a filter, each tag key must be unique.

- **Specifying a filter based on both the prefix and one or more tags** – In a Lifecycle rule, you can specify a filter based on both the key prefix and one or more tags. Again, you must wrap all of these filter elements in the <And> element, as follows:

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
```

```
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
       </Filter>
       <Status>Enabled</Status>
       transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

Amazon S3 combines these filters by using a logical AND. That is, the rule applies to the subset of objects with the specified key prefix and the specified tags. A filter can have only one prefix, and zero or more tags.

- You can specify an **empty filter**, in which case the rule applies to all objects in the bucket.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- To filter a rule by **object size**, you can specify a minimum size (ObjectSizeGreaterThan) or a maximum size (ObjectSizeLessThan), or you can specify a range of object sizes.

  Object size values are in bytes. Maximum filter size is 5 TB. Some storage classes have minimum object size limitations. For more information, see Comparing the Amazon S3 storage classes.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
```

```
</LifecycleConfiguration>
```

> **Note**
>
> The `ObjectSizeGreaterThan` and `ObjectSizeLessThan` filters exclude the specified values. For example, if you set objects sized 128 KB to 1024 KB to move from the S3 Standard storage class to the S3 Standard-IA storage class, objects that are exactly 1024 KB and 128 KB won't transition to S3 Standard-IA. Instead, the rule will apply only to objects that are greater than 128 KB and less than 1024 KB in size.

If you're specifying an object size range, the `ObjectSizeGreaterThan` integer must be less than the `ObjectSizeLessThan` value. When using more than one filter, you must wrap the filters in an <And> element. The following example shows how to specify objects in a range between 500 bytes and 64,000 bytes.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

## Elements to describe lifecycle actions

You can direct Amazon S3 to perform specific actions in an object's lifetime by specifying one or more of the following predefined actions in an S3 Lifecycle rule. The effect of these actions depends on the versioning state of your bucket.

- **`Transition` action element** – You specify the `Transition` action to transition objects from one storage class to another. For more information about transitioning objects, see Supported

transitions and related constraints. When a specified date or time period in the object's lifetime is reached, Amazon S3 performs the transition.

For a versioned bucket (versioning-enabled or versioning-suspended bucket), the `Transition` action applies to the current object version. To manage noncurrent versions, Amazon S3 defines the `NoncurrentVersionTransition` action (described later in this topic).

- **`Expiration` action element** – The `Expiration` action expires objects identified in the rule and applies to eligible objects in any of the Amazon S3 storage classes. For more information about storage classes, see Using Amazon S3 storage classes. Amazon S3 makes all expired objects unavailable. Whether the objects are permanently removed depends on the versioning state of the bucket.
  - **Nonversioned bucket** – The `Expiration` action results in Amazon S3 permanently removing the object.
  - **Versioned bucket** – For a versioned bucket (that is, versioning-enabled or versioning-suspended), there are several considerations that guide how Amazon S3 handles the `Expiration` action. For versioning-enabled or versioning-suspended buckets, the following applies:
    - The `Expiration` action applies only to the current version (it has no impact on noncurrent object versions).
    - Amazon S3 doesn't take any action if there are one or more object versions and the delete marker is the current version.
    - If the current object version is the only object version and it is also a delete marker (also referred as an *expired object delete marker*, where all object versions are deleted and you only have a delete marker remaining), Amazon S3 removes the expired object delete marker. You can also use the expiration action to direct Amazon S3 to remove any expired object delete markers. For an example, see Example 7: Removing expired object delete markers.

    For more information, see Using versioning in S3 buckets.

    Also consider the following when setting up Amazon S3 to manage expiration:

    - **Versioning-enabled bucket**

      If the current object version is not a delete marker, Amazon S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and the delete marker the current version.

- **Versioning-suspended bucket**

  In a versioning-suspended bucket, the expiration action causes Amazon S3 to create a delete marker with `null` as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.

In addition, Amazon S3 provides the following actions that you can use to manage noncurrent object versions in a versioned bucket (that is, versioning-enabled and versioning-suspended buckets).

- **`NoncurrentVersionTransition` action element** – Use this action to specify when Amazon S3 transitions objects to the specified storage class. You can base this expiration on a certain number of days since the objects became noncurrent. In addition to the number of days, you can also provide a maximum number of noncurrent versions to retain (between 1 and 100). This value determines how many newer noncurrent versions must exist before Amazon S3 can perform the associated action on a given version. Amazon S3 will transition any additional noncurrent versions beyond the specified number to retain.

  To specify the maximum number of noncurrent versions, you must also provide a `Filter` element. If you don't specify a `Filter` element, Amazon S3 generates an `InvalidRequest` error when you provide a maximum number of noncurrent versions.

  For more information about transitioning objects, see Supported transitions and related constraints. For details about how Amazon S3 calculates the date when you specify the number of days in the `NoncurrentVersionTransition` action, see Lifecycle rules: Based on an object's age.
- **`NoncurrentVersionExpiration` action element** – Use this action to direct Amazon S3 to permanently delete noncurrent versions of objects. These deleted objects can't be recovered. You can base this expiration on a certain number of days since the objects became noncurrent. In addition to the number of days, you can also provide a maximum number of noncurrent versions to retain (between 1 and 100). This value specifies how many newer noncurrent versions must exist before Amazon S3 can perform the associated action on a given version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain.

  To specify the maximum number of noncurrent versions, you must also provide a `Filter` element. If you don't specify a `Filter` element, Amazon S3 generates an `InvalidRequest` error when you provide a maximum number of noncurrent versions.

Delayed removal of noncurrent objects can be helpful when you need to correct any accidental deletes or overwrites. For example, you can configure an expiration rule to delete noncurrent versions five days after they become noncurrent. For example, suppose that on 1/1/2014 at 10:30 AM UTC, you create an object called `photo.gif` (version ID 111111). On 1/2/2014 at 11:30 AM UTC, you accidentally delete `photo.gif` (version ID 111111), which creates a delete marker with a new version ID (such as version ID 4857693). You now have five days to recover the original version of `photo.gif` (version ID 111111) before the deletion is permanent. On 1/8/2014 at 00:00 UTC, the Lifecycle rule for expiration runs and permanently deletes `photo.gif` (version ID 111111), five days after it became a noncurrent version.

For details about how Amazon S3 calculates the date when you specify the number of days in an `NoncurrentVersionExpiration` action, see Lifecycle rules: Based on an object's age.

> **Note**
>
> Object expiration lifecycle configurations don't remove incomplete multipart uploads. To remove incomplete multipart uploads, you must use the `AbortIncompleteMultipartUpload` Lifecycle configuration action that's described later in this section.

In addition to the transition and expiration actions, you can use the following Lifecycle configuration actions to direct Amazon S3 to stop incomplete multipart uploads or to remove expired object delete markers:

- **`AbortIncompleteMultipartUpload` action element** – Use this element to set a maximum time (in days) that you want to allow multipart uploads to remain in progress. If the applicable multipart uploads (determined by the key name `prefix` specified in the Lifecycle rule) aren't successfully completed within the predefined time period, Amazon S3 stops the incomplete multipart uploads. For more information, see Aborting a multipart upload.

  > **Note**
  >
  > You can't specify this lifecycle action in a rule that has a filter that uses object tags.

- **`ExpiredObjectDeleteMarker` action element** – In a versioning-enabled bucket, a delete marker with zero noncurrent versions is referred to as an *expired object delete marker*. You can

use this lifecycle action to direct Amazon S3 to remove expired object delete markers. For an example, see Example 7: Removing expired object delete markers.

> **Note**
>
> You can't specify this lifecycle action in a rule that has a filter that uses object tags.

### How Amazon S3 calculates how long an object has been noncurrent

In a versioning-enabled bucket, you can have multiple versions of an object. There is always one current version, and zero or more noncurrent versions. Each time you upload an object, the current version is retained as the noncurrent version and the newly added version, the successor, becomes the current version. To determine the number of days an object is noncurrent, Amazon S3 looks at when its successor was created. Amazon S3 uses the number of days since its successor was created as the number of days an object is noncurrent.

> **Restoring previous versions of an object when using S3 Lifecycle configurations**
>
> As explained in Restoring previous versions, you can use either of the following two methods to retrieve previous versions of an object:
>
> - **Method 1 – Copy a noncurrent version of the object into the same bucket.** The copied object becomes the current version of that object, and all object versions are preserved.
> - **Method 2 – Permanently delete the current version of the object.** When you delete the current object version, you, in effect, turn the noncurrent version into the current version of that object.
>
> When you're using S3 Lifecycle configuration rules with versioning-enabled buckets, we recommend as a best practice that you use Method 1.
> S3 Lifecycle operates under an eventually consistent model. A current version that you permanently deleted might not disappear until the changes propagate to all of the Amazon S3 systems. (Therefore, Amazon S3 might be temporarily unaware of this deletion.) In the meantime, the lifecycle rule that you configured to expire noncurrent objects might permanently remove noncurrent objects, including the one that you want to restore. So, copying the old version, as recommended in Method 1, is the safer alternative.

### Lifecycle actions and bucket versioning state

#### Lifecycle rules: Based on an object's age

You can specify a time period, in the number of days from the creation (or modification) of the object, when Amazon S3 can take the specified action.

When you specify the number of days in the `Transition` and `Expiration` actions in an S3 Lifecycle configuration, note the following:

- The value that you specify is the number of days since object creation when the action will occur.
- Amazon S3 calculates the time by adding the number of days specified in the rule to the object creation time and rounding the resulting time to the next day at midnight UTC. For example, if an object was created on 1/15/2014 at 10:30 AM UTC and you specify 3 days in a transition rule, then the transition date of the object would be calculated as 1/19/2014 00:00 UTC.

> **Note**
>
> Amazon S3 maintains only the last modified date for each object. For example, the Amazon S3 console shows the **Last modified** date in the object's **Properties** pane. When you initially create a new object, this date reflects the date that the object is created. If you replace the object, the date changes accordingly. Therefore, the creation date is synonymous with the **Last modified** date.

When specifying the number of days in the `NoncurrentVersionTransition` and `NoncurrentVersionExpiration` actions in a Lifecycle configuration, note the following:

- The value that you specify is the number of days from when the version of the object becomes noncurrent (that is, when the object is overwritten or deleted) that Amazon S3 will perform the action on the specified object or objects.
- Amazon S3 calculates the time by adding the number of days specified in the rule to the time when the new successor version of the object is created and rounding the resulting time to the next day at midnight UTC. For example, in your bucket, suppose that you have a current version of an object that was created on 1/1/2014 at 10:30 AM UTC. If the new version of the object that replaces the current version is created on 1/15/2014 at 10:30 AM UTC, and you specify 3 days in a transition rule, the transition date of the object is calculated as 1/19/2014 00:00 UTC.

### Lifecycle rules: Based on a specific date

When specifying an action in an S3 Lifecycle rule, you can specify a date when you want Amazon S3 to take the action. When the specific date arrives, Amazon S3 applies the action to all qualified objects (based on the filter criteria).

If you specify an S3 Lifecycle action with a date that is in the past, all qualified objects become immediately eligible for that lifecycle action.

**Important**

The date-based action is not a one-time action. Amazon S3 continues to apply the date-based action even after the date has passed, as long as the rule status is `Enabled`. For example, suppose that you specify a date-based `Expiration` action to delete all objects (assume that no filter is specified in the rule). On the specified date, Amazon S3 expires all the objects in the bucket. Amazon S3 also continues to expire any new objects that you create in the bucket. To stop the lifecycle action, you must either remove the action from the lifecycle rule, disable the rule, or delete the rule from the lifecycle configuration.

The date value must conform to the ISO 8601 format. The time is always midnight UTC.

**Note**

You can't create date-based Lifecycle rules by using the Amazon S3 console, but you can view, disable, or delete such rules.

## Examples of S3 Lifecycle configuration

This section provides examples of S3 Lifecycle configuration. Each example shows how you can specify the XML in each of the example scenarios.

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

**Topics**



## Example 1: Specifying a filter

Each S3 Lifecycle rule includes a filter that you can use to identify a subset of objects in your bucket to which the S3 Lifecycle rule applies. The following S3 Lifecycle configurations show examples of how you can specify a filter.

- In this S3 Lifecycle configuration rule, the filter specifies a key name prefix (`tax/`). Therefore, the rule applies to objects with the prefix `tax/`, such as `tax/doc1.txt` and `tax/doc2.txt`.

  The rule specifies two actions that direct Amazon S3 to do the following:

  - Transition objects to the S3 Glacier Flexible Retrieval storage class 365 days (one year) after creation.

- Delete objects (the `Expiration` action) 3,650 days (10 years) after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

  Instead of specifying the object age in terms of days after creation, you can specify a date for each action. However, you can't use both `Date` and `Days` in the same rule.

- If you want the S3 Lifecycle rule to apply to all objects in the bucket, specify an empty prefix. In the following configuration, the rule specifies a `Transition` action that directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class 0 days after creation. This rule means that the objects are eligible for archival to S3 Glacier Flexible Retrieval at midnight UTC following creation. For more information about lifecycle constraints, see Constraints.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- You can specify zero or one prefix and zero or more object tags in a filter. The following example code applies the S3 Lifecycle rule to a subset of objects with the `tax/` prefix and to objects that have two tags with a specific key and value. When you specify more than one filter, you must include the <And> element as shown (Amazon S3 applies a logical `AND` to combine the specified filter conditions).

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

> **Note**
>
> If you have one or more prefixes that start with the same characters, you can include all of those prefixes in your rule by specifying a partial prefix with no trailing slash (/) in the filter. For example, suppose that you have these prefixes:
>
> ```
> sales1999/
> sales2000/
> sales2001/
> ```
>
> To include all three prefixes in your rule, specify `<Prefix>sales</Prefix>` in your lifecycle rule.

- Instead of using a prefix, you can filter objects based only on tags. For example, the following S3 Lifecycle rule applies to objects that have the two specified tags (it does not specify any prefix).

```
...
<Filter>
```

```
    <And>
       <Tag>
          <Key>key1</Key>
          <Value>value1</Value>
       </Tag>
       <Tag>
          <Key>key2</Key>
          <Value>value2</Value>
       </Tag>
     </And>
</Filter>
...
```

If you want to *exclude* a particular prefix from your lifecycle rule, use tags to tag all of the objects in the prefixes that you want to *include* in the rule.

## Example 2: Disabling a Lifecycle rule

You can temporarily disable an S3 Lifecycle rule. The following S3 Lifecycle configuration specifies two rules:

- Rule 1 directs Amazon S3 to transition objects with the `logs/` prefix to the S3 Glacier Flexible Retrieval storage class soon after creation.
- Rule 2 directs Amazon S3 to transition objects with the `documents/` prefix to the S3 Glacier Flexible Retrieval storage class soon after creation.

In the configuration, Rule 1 is enabled and Rule 2 is disabled. Amazon S3 ignores disabled rules.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
```

```
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

### Example 3: Tiering down the storage class over an object's lifetime

In this example, you use S3 Lifecycle configuration to tier down the storage class of objects over their lifetime. Tiering down can help reduce storage costs. For more information about pricing, see Amazon S3 pricing.

The following S3 Lifecycle configuration specifies a rule that applies to objects with the key name prefix `logs/`. The rule specifies the following actions:

- Two transition actions:
  - Transition objects to the S3 Standard-IA storage class 30 days after creation.
  - Transition objects to the S3 Glacier Flexible Retrieval storage class 90 days after creation.
- One expiration action that directs Amazon S3 to delete objects a year after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
```

```
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

**Note**

You can use one rule to describe all S3 Lifecycle actions if all actions apply to the same set of objects (identified by the filter). Otherwise, you can add multiple rules with each specifying a different filter.

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

## Example 4: Specifying multiple rules

You can specify multiple rules if you want different S3 Lifecycle actions of different objects. The following S3 Lifecycle configuration has two rules:

- Rule 1 applies to objects with the key name prefix `classA/`. It directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class one year after creation and expire these objects 10 years after creation.
- Rule 2 applies to objects with key name prefix `classB/`. It directs Amazon S3 to transition objects to the S3 Standard-IA storage class 90 days after creation and delete them one year after creation.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

## Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets

You might specify an S3 Lifecycle configuration in which you specify overlapping prefixes, or actions.

Generally, S3 Lifecycle optimizes for cost. For example, if two expiration policies overlap, the shorter expiration policy is honored so that data is not stored for longer than expected. Likewise, if two transition policies overlap, S3 Lifecycle transitions your objects to the lower-cost storage class.

In both cases, S3 Lifecycle tries to choose the path that is least expensive for you. An exception to this general rule is with the S3 Intelligent-Tiering storage class. S3 Intelligent-Tiering is favored by S3 Lifecycle over any storage class, aside from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.

The following examples show how Amazon S3 resolves potential conflicts.

### Example 1: Overlapping prefixes (no conflict)

The following example configuration has two rules that specify overlapping prefixes as follows:

- The first rule specifies an empty filter, indicating all objects in the bucket.

- The second rule specifies a key name prefix (`logs/`), indicating only a subset of objects.

Rule 1 requests Amazon S3 to delete all objects one year after creation. Rule 2 requests Amazon S3 to transition a subset of objects to the S3 Standard-IA storage class 30 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

Since there is no conflict in this case, Amazon S3 will transition the objects with the `logs/` prefix to the S3 Standard-IA storage class 30 days after creation. When any object reaches one year after creation, it will be deleted.

**Example 2: Conflicting lifecycle actions**

In this example configuration, there are two rules that direct Amazon S3 to perform two different actions on the same set of objects at the same time in the objects' lifetime:

- Both rules specify the same key name prefix, so both rules apply to the same set of objects.
- Both rules specify the same 365 days after object creation when the rules apply.
- One rule directs Amazon S3 to transition objects to the S3 Standard-IA storage class and another rule wants Amazon S3 to expire the objects at the same time.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

In this case, because you want objects to expire (to be removed), there is no point in changing the storage class, so Amazon S3 chooses the expiration action on these objects.

**Example 3: Overlapping prefixes resulting in conflicting lifecycle actions**

In this example, the configuration has two rules, which specify overlapping prefixes as follows:

- Rule 1 specifies an empty prefix (indicating all objects).
- Rule 2 specifies a key name prefix (`logs/`) that identifies a subset of all objects.

For the subset of objects with the `logs/` key name prefix, S3 Lifecycle actions in both rules apply. One rule directs Amazon S3 to transition objects 10 days after creation, and another rule directs Amazon S3 to transition objects 365 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
```

```
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

In this case, Amazon S3 chooses to transition them 10 days after creation.

**Example 4: Tag-based filtering and resulting conflicting lifecycle actions**

Suppose that you have the following S3 Lifecycle configuration that has two rules, each specifying a tag filter:

- Rule 1 specifies a tag-based filter (`tag1/value1`). This rule directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class 365 days after creation.
- Rule 2 specifies a tag-based filter (`tag2/value2`). This rule directs Amazon S3 to expire objects 14 days after creation.

The S3 Lifecycle configuration is shown in following example.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
```

```
          <Key>tag1</Key>
          <Value>value1</Value>
       </Tag>
     </Filter>
     <Status>Enabled</Status>
     <Transition>
       <StorageClass>GLACIER<StorageClass>
       <Days>365</Days>
     </Transition>
   </Rule>
   <Rule>
     <ID>Rule 2</ID>
     <Filter>
       <Tag>
          <Key>tag2</Key>
          <Value>value2</Value>
       </Tag>
     </Filter>
     <Status>Enabled</Status>
     <Expiration>
       <Days>14</Days>
     </Expiration>
    </Rule>
</LifecycleConfiguration>
```

If an object has both tags, then Amazon S3 has to decide which rule to follow. In this case, Amazon S3 expires the object 14 days after creation. The object is removed, and therefore the transition action does not apply.

**Important**

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Example 5: Overlapping filters, conflicting lifecycle actions, and what Amazon S3 does with nonversioned buckets.

### Example 6: Specifying a lifecycle rule for a versioning-enabled bucket

Suppose that you have a versioning-enabled bucket, which means that for each object, you have a current version and zero or more noncurrent versions. (For more information about S3 Versioning, see Using versioning in S3 buckets.) In this example, you want to maintain one year's worth of history, and delete the noncurrent versions. S3 Lifecycle configurations support keeping 1 to 100 versions of any object.

To save storage costs, you want to move noncurrent versions to S3 Glacier Flexible Retrieval 30 days after they become noncurrent (assuming that these noncurrent objects are cold data for which you don't need real-time access). In addition, you expect the frequency of access of the current versions to diminish 90 days after creation, so you might choose to move these objects to the S3 Standard-IA storage class.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

## Example 7: Removing expired object delete markers

A versioning-enabled bucket has one current version and zero or more noncurrent versions for each object. When you delete an object, note the following:

- If you don't specify a version ID in your delete request, Amazon S3 adds a delete marker instead of deleting the object. The current object version becomes noncurrent, and the delete marker becomes the current version.
- If you specify a version ID in your delete request, Amazon S3 deletes the object version permanently (a delete marker is not created).
- A delete marker with zero noncurrent versions is referred to as an *expired object delete marker*.

This example shows a scenario that can create expired object delete markers in your bucket, and how you can use S3 Lifecycle configuration to direct Amazon S3 to remove the expired object delete markers.

Suppose that you write an S3 Lifecycle configuration that uses the `NoncurrentVersionExpiration` action to remove the noncurrent versions 30 days after they become noncurrent and retains at most 10 noncurrent versions, as shown in the following example.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

The `NoncurrentVersionExpiration` action does not apply to the current object versions. It removes only the noncurrent versions.

For current object versions, you have the following options to manage their lifetime, depending on whether the current object versions follow a well-defined lifecycle:

- **The current object versions follow a well-defined lifecycle.**

In this case, you can use an S3 Lifecycle configuration with the `Expiration` action to direct Amazon S3 to remove the current versions, as shown in the following example.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <Expiration>
           <Days>60</Days>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

In this example, Amazon S3 removes current versions 60 days after they are created by adding a delete marker for each of the current object versions. This process makes the current version noncurrent, and the delete marker becomes the current version. For more information, see Using versioning in S3 buckets.

> **Note**
>
> You cannot specify both a `Days` and an `ExpiredObjectDeleteMarker` tag on the same rule. When you specify the `Days` tag, Amazon S3 automatically performs `ExpiredObjectDeleteMarker` cleanup when the delete markers are old enough to satisfy the age criteria. To clean up delete markers as soon as they become the only version, create a separate rule with only the `ExpiredObjectDeleteMarker` tag.

The `NoncurrentVersionExpiration` action in the same S3 Lifecycle configuration removes noncurrent objects 30 days after they become noncurrent. Thus, in this example, all object versions are permanently removed 90 days after object creation. Although expired object delete markers are created during this process, Amazon S3 detects and removes the expired object delete markers for you.

- **The current object versions don't have a well-defined lifecycle.**

In this case, you might remove the objects manually when you don't need them, creating a delete marker with one or more noncurrent versions. If your S3 Lifecycle configuration with the `NoncurrentVersionExpiration` action removes all the noncurrent versions, you now have expired object delete markers.

Specifically for this scenario, S3 Lifecycle configuration provides an `Expiration` action that you can use to remove the expired object delete markers.

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

By setting the `ExpiredObjectDeleteMarker` element to `true` in the `Expiration` action, you direct Amazon S3 to remove the expired object delete markers.

> **Note**
>
> When you use the `ExpiredObjectDeleteMarker` S3 Lifecycle action, the rule cannot specify a tag-based filter.

## Example 8: Lifecycle configuration to abort multipart uploads

You can use the Amazon S3 multipart upload REST API operations to upload large objects in parts. For more information about multipart uploads, see Uploading and copying objects using multipart upload.

By using an S3 Lifecycle configuration, you can direct Amazon S3 to stop incomplete multipart uploads (identified by the key name prefix specified in the rule) if they aren't completed within a specified number of days after initiation. When Amazon S3 aborts a multipart upload, it deletes all the parts associated with the multipart upload. This process helps control your storage costs by ensuring that you don't have incomplete multipart uploads with parts that are stored in Amazon S3.

> **Note**
>
> When you use the `AbortIncompleteMultipartUpload` S3 Lifecycle action, the rule cannot specify a tag-based filter.

The following is an example S3 Lifecycle configuration that specifies a rule with the `AbortIncompleteMultipartUpload` action. This action directs Amazon S3 to stop incomplete multipart uploads seven days after initiation.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

## Example 9: Lifecycle configuration using size-based rules

You can create rules that transition objects based only on their size. You can specify a minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`), or you can specify

a range of object sizes in bytes. When using more than one filter, such as a prefix and size rule, you must wrap the filters in an <And> element.

> **Note**
>
> The ObjectSizeGreaterThan and ObjectSizeLessThan filters exclude the specified values. For more information, see the section called "Filter element".

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

If you're specifying a range by using both the ObjectSizeGreaterThan and ObjectSizeLessThan elements, the maximum object size must be larger than the minimum object size. When using more than one filter, you must wrap the filters in an <And> element. The following example shows how to specify objects in a range between 500 bytes and 64,000 bytes.

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

You can also create rules to specifically expire noncurrent objects that have no data, including noncurrent delete marker objects created in a versioning-enabled bucket. The following example uses the `NoncurrentVersionExpiration` action to remove noncurrent versions 30 days after they become noncurrent and retain at most 10 noncurrent versions of objects. It also uses the `ObjectSizeLessThan` element to filter only objects with no data.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

# Amazon S3 Inventory

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

You can use Amazon S3 Inventory to help manage your storage. For example, you can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. You can also simplify and speed up business workflows and big data jobs by using Amazon S3 Inventory, which provides a scheduled alternative to the Amazon S3 synchronous

`List` API operations. Amazon S3 Inventory does not use the `List` API operations to audit your objects and does not affect the request rate of your bucket.

Amazon S3 Inventory provides comma-separated values (CSV), Apache optimized row columnar (ORC) or Apache Parquet output files that list your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or objects with a shared prefix (that is, objects that have names that begin with a common string). If you set up a weekly inventory, a report is generated every Sunday (UTC time zone) after the initial report. For information about Amazon S3 Inventory pricing, see Amazon S3 pricing.

You can configure multiple inventory lists for a bucket. When you're configuring an inventory list, you can specify the following:

- What object metadata to include in the inventory
- Whether to list all object versions or only current versions
- Where to store the inventory list file output
- Whether to generate the inventory on a daily or weekly basis
- Whether to encrypt the inventory list file

You can query Amazon S3 Inventory with standard SQL queries by using Amazon Athena, Amazon Redshift Spectrum, and other tools, such as Presto, Apache Hive, and Apache Spark. For more information about using Athena to query your inventory files, see the section called “Querying inventory with Athena”.

## Source and destination buckets

The bucket that the inventory lists objects for is called the *source bucket*. The bucket where the inventory list file is stored is called the *destination bucket*.

**Source bucket**

The inventory lists the objects that are stored in the source bucket. You can get an inventory list for an entire bucket, or you can filter the list by object key name prefix.

The source bucket:

- Contains the objects that are listed in the inventory
- Contains the configuration for the inventory

**Destination bucket**

Amazon S3 Inventory list files are written to the destination bucket. To group all the inventory list files in a common location in the destination bucket, you can specify a destination prefix in the inventory configuration.

The destination bucket:

- Contains the inventory file lists.
- Contains the manifest files that list all the inventory list files that are stored in the destination bucket. For more information, see Inventory manifest.
- Must have a bucket policy to give Amazon S3 permission to verify ownership of the bucket and permission to write files to the bucket.
- Must be in the same AWS Region as the source bucket.
- Can be the same as the source bucket.
- Can be owned by a different AWS account than the account that owns the source bucket.

## Amazon S3 Inventory list

An inventory list file contains a list of the objects in the source bucket and metadata for each object. An inventory list file is stored in the destination bucket with one of the following formats:

- As a CSV file compressed with GZIP
- As an Apache optimized row columnar (ORC) file compressed with ZLIB
- As an Apache Parquet file compressed with Snappy

> **Note**
>
> Objects in Amazon S3 Inventory reports aren't guaranteed to be sorted in any order.

An inventory list file contains a list of the objects in the source bucket and metadata for each listed object:

- **Bucket name** – The name of the bucket that the inventory is for.

- **Key name** – The object key name (or key) that uniquely identifies the object in the bucket. When you're using the CSV file format, the key name is URL-encoded and must be decoded before you can use it.
- **Version ID** – The object version ID. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects that are added to the bucket. For more information, see Using versioning in S3 buckets. (This field is not included if the list is configured only for the current version of the objects.)
- **IsLatest** – Set to `True` if the object is the current version of the object. (This field is not included if the list is configured only for the current version of the objects.)
- **Delete marker** – Set to `True` if the object is a delete marker. For more information, see Using versioning in S3 buckets. (This field is automatically added to your report if you've configured the report to include all versions of your objects).
- **Size** – The object size in bytes, not including the size of incomplete multipart uploads, object metadata, and delete markers.
- **Last modified date** – The object creation date or the last modified date, whichever is the latest.
- **ETag** – The entity tag (ETag) is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag can be an MD5 digest of the object data. Whether it is depends on how the object was created and how it is encrypted.
- **Storage class** – The storage class that's used for storing the object. Set to `STANDARD`, `REDUCED_REDUNDANCY`, `STANDARD_IA`, `ONEZONE_IA`, `INTELLIGENT_TIERING`, `GLACIER`, `DEEP_ARCHIVE`, `OUTPOSTS`, `GLACIER_IR`, or `SNOW`. For more information, see Using Amazon S3 storage classes.
- **Multipart upload flag** – Set to `True` if the object was uploaded as a multipart upload. For more information, see Uploading and copying objects using multipart upload.
- **Replication status** – Set to `PENDING`, `COMPLETED`, `FAILED`, or `REPLICA`. For more information, see Getting replication status information.
- **Encryption status** – The server-side encryption status, depending on what kind of encryption key is used— server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). Set to `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C`, or `NOT-SSE`. A status of `NOT-SSE` means that the object is not encrypted with server-side encryption. For more information, see Protecting data with encryption.

- **S3 Object Lock retain until date** – The date until which the locked object cannot be deleted. For more information, see Using S3 Object Lock.
- **S3 Object Lock retention mode** – Set to `Governance` or `Compliance` for objects that are locked. For more information, see Using S3 Object Lock.
- **S3 Object Lock legal hold status** – Set to `On` if a legal hold has been applied to an object. Otherwise, it is set to `Off`. For more information, see Using S3 Object Lock.
- **S3 Intelligent-Tiering access tier** – Access tier (frequent or infrequent) of the object if it is stored in the S3 Intelligent-Tiering storage class. Set to `FREQUENT`, `INFREQUENT`, `ARCHIVE_INSTANT_ACCESS`, `ARCHIVE`, or `DEEP_ARCHIVE`. For more information, see Storage class for automatically optimizing data with changing or unknown access patterns.
- **S3 Bucket Key status** – Set to `ENABLED` or `DISABLED`. Indicates whether the object uses an S3 Bucket Key for SSE-KMS. For more information, see Using Amazon S3 Bucket Keys.
- **Checksum algorithm** – Indicates the algorithm that's used to create the checksum for the object.
- **Object access control list** – An access control list (ACL) for each object that defines which AWS accounts or groups are granted access to this object and the type of access that is granted. The Object ACL field is defined in JSON format. An S3 Inventory report includes ACLs that are associated with objects in your source bucket, even when ACLs are disabled for the bucket. For more information, see Working with the Object ACL field and Access control list (ACL) overview.

**Note**

The Object ACL field is defined in JSON format. An inventory report displays the value for the Object ACL field as a base64-encoded string.
For example, suppose that you have the following Object ACL field in JSON format:

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
}
```

The Object ACL field is encoded and shown as the following base64-encoded string:

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
```

To get the decoded value in JSON for the Object ACL field, you can query this field in Amazon Athena. For query examples, see Querying Amazon S3 Inventory with Amazon Athena.

- **Object owner** – The owner of the object.

> **Note**
>
> When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 queues the object for removal and removes it asynchronously. Therefore, there might be a delay between the expiration date and the date when Amazon S3 removes an object. The inventory report includes the objects that have expired but haven't been removed yet. For more information about expiration actions in S3 Lifecycle, see Expiring objects.

We recommend that you create a lifecycle policy that deletes old inventory lists. For more information, see Managing your storage lifecycle.

The `s3:PutInventoryConfiguration` permission allows a user to both select all the metadata fields that are listed earlier for each object when configuring an inventory list and to specify the destination bucket to store the inventory. A user with read access to objects in the destination bucket can access all object metadata fields that are available in the inventory list. To restrict access to an inventory report, see Grant permissions for S3 Inventory and S3 analytics.

## Inventory consistency

All of your objects might not appear in each inventory list. The inventory list provides eventual consistency for PUT requests (of both new objects and overwrites) and for `DELETE` requests. Each inventory list for a bucket is a snapshot of bucket items. These lists are eventually consistent (that is, a list might not include recently added or deleted objects).

To validate the state of an object before you take action on the object, we recommend that you perform a `HeadObject` REST API request to retrieve metadata for the object, or to check the object's properties in the Amazon S3 console. You can also check object metadata with the AWS

CLI or the AWS SDKS. For more information, see `HeadObject` in the *Amazon Simple Storage Service API Reference*.

For more information about working with Amazon S3 Inventory, see the following topics.

**Topics**

- Configuring Amazon S3 Inventory
- Setting up Amazon S3 Event Notifications for inventory completion
- Locating your inventory list
- Querying Amazon S3 Inventory with Amazon Athena
- Converting empty version ID strings in Amazon S3 Inventory reports to null strings
- Working with the Object ACL field

## Configuring Amazon S3 Inventory

Amazon S3 Inventory provides a flat file list of your objects and metadata, on a schedule that you define. You can use S3 Inventory as a scheduled alternative to the Amazon S3 synchronous `List` API operation. S3 Inventory provides comma-separated values (CSV), Apache optimized row columnar (ORC), or Apache Parquet (Parquet) output files that list your objects and their corresponding metadata.

You can configure S3 Inventory to create inventory lists on a daily or weekly basis for an S3 bucket or for objects that share a prefix (objects that have names that begin with the same string). For more information, see Amazon S3 Inventory.

This section describes how to configure an inventory, including details about the inventory source and destination buckets.

**Topics**

- Overview
- Creating a destination bucket policy
- Granting Amazon S3 permission to use your customer managed key for encryption
- Configuring inventory by using the S3 console
- Using the REST API to work with S3 Inventory

## Overview

Amazon S3 Inventory helps you manage your storage by creating lists of the objects in an S3 bucket on a defined schedule. You can configure multiple inventory lists for a bucket. The inventory lists are published to CSV, ORC, or Parquet files in a destination bucket.

The easiest way to set up an inventory is by using the Amazon S3 console, but you can also use the Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs. The console performs the first step of the following procedure for you: adding a bucket policy to the destination bucket.

**To set up Amazon S3 Inventory for an S3 bucket**

1. **Add a bucket policy for the destination bucket.**

   You must create a bucket policy on the destination bucket that grants permissions to Amazon S3 to write objects to the bucket in the defined location. For an example policy, see Grant permissions for S3 Inventory and S3 analytics.

2. **Configure an inventory to list the objects in a source bucket and publish the list to a destination bucket.**

   When you configure an inventory list for a source bucket, you specify the destination bucket where you want the list to be stored, and whether you want to generate the list daily or weekly. You can also configure whether to list all object versions or only current versions and what object metadata to include.

   Some object metadata fields in S3 Inventory report configurations are optional, meaning that they're available by default but they can be restricted when you grant a user the `s3:PutInventoryConfiguration` permission. You can control whether users can include these optional metadata fields in their reports by using the `s3:InventoryAccessibleOptionalFields` condition key.

   For more information about the optional metadata fields available in S3 Inventory, see OptionalFields in the *Amazon Simple Storage Service API Reference*. For more information about restricting access to certain optional metadata fields in an inventory configuration, see Control S3 Inventory report configuration creation.

   You can specify that the inventory list file be encrypted by using server-side encryption with an Amazon S3 managed key (SSE-S3) or an AWS Key Management Service (AWS KMS) customer managed key (SSE-KMS).

> **Note**
>
> The AWS managed key (`aws/s3`) is not supported for SSE-KMS encryption with S3 Inventory.

For more information about SSE-S3 and SSE-KMS, see Protecting data with server-side encryption. If you plan to use SSE-KMS encryption, see Step 3.

- For information about how to use the console to configure an inventory list, see Configuring inventory by using the S3 console.
- To use the Amazon S3 API to configure an inventory list, use the PutBucketInventoryConfiguration REST API operation or the equivalent from the AWS CLI or AWS SDKs.

3. **To encrypt the inventory list file with SSE-KMS, grant Amazon S3 permission to use the AWS KMS key.**

   You can configure encryption for the inventory list file by using the Amazon S3 console, Amazon S3 REST API, AWS CLI, or AWS SDKs. Whichever way you choose, you must grant Amazon S3 permission to use the customer managed key to encrypt the inventory file. You grant Amazon S3 permission by modifying the key policy for the customer managed key that you want to use to encrypt the inventory file. For more information, see Granting Amazon S3 permission to use your customer managed key for encryption.

   The destination bucket that stores the inventory list file can be owned by a different AWS account than the account that owns the source bucket. If you use SSE-KMS encryption for the cross-account operations of Amazon S3 Inventory, we recommend that you use a fully qualified KMS key ARN when you configure S3 inventory. For more information, see Using SSE-KMS encryption for cross-account operations and ServerSideEncryptionByDefault in the *Amazon Simple Storage Service API Reference*.

## Creating a destination bucket policy

If you create your inventory configuration through the Amazon S3 console, Amazon S3 automatically creates a bucket policy on the destination bucket that grants Amazon S3 write permission to the bucket. However, if you create your inventory configuration through the AWS CLI, AWS SDKs, or the Amazon S3 REST API, you must manually add a bucket policy on the destination

bucket. For more information, see Grant permissions for S3 Inventory and S3 analytics. The S3 Inventory destination bucket policy allows Amazon S3 to write data for the inventory reports to the bucket.

If an error occurs when you try to create the bucket policy, you are given instructions on how to fix it. For example, if you choose a destination bucket in another AWS account and don't have permissions to read and write to the bucket policy, you see an error message.

In this case, the destination bucket owner must add the bucket policy to the destination bucket. If the policy is not added to the destination bucket, you won't get an inventory report because Amazon S3 doesn't have permission to write to the destination bucket. If the source bucket is owned by a different account than that of the current user, the correct account ID of the source bucket owner must be substituted in the policy.

## Granting Amazon S3 permission to use your customer managed key for encryption

To grant Amazon S3 permission to use your AWS Key Management Service (AWS KMS) customer managed key for server-side encryption, you must use a key policy. To update your key policy so that you can use your customer managed key, use the following procedure.

**To grant Amazon S3 permissions to encrypt by using your customer managed key**

1. Using the AWS account that owns the customer managed key, sign into the AWS Management Console.
2. Open the AWS KMS console at https://console.aws.amazon.com/kms.
3. To change the AWS Region, use the Region selector in the upper-right corner of the page.
4. In the left navigation pane, choose **Customer managed keys**.
5. Under **Customer managed keys**, choose the customer managed key that you want to use to encrypt your inventory files.
6. In the **Key policy** section, choose **Switch to policy view**.
7. To update the key policy, choose **Edit**.
8. On the **Edit key policy** page, add the following lines to the existing key policy. For *source-account-id* and *amzn-s3-demo-source-bucket*, supply the appropriate values for your use case.

```
{
```

```
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::amzn-s3-demo-source-bucket"
     }
   }
}
```

9. Choose **Save changes**.

For more information about creating customer managed keys and using key policies, see the following links in the *AWS Key Management Service Developer Guide*:

- Managing keys
- Key policies in AWS KMS

## Configuring inventory by using the S3 console

Use these instructions to configure inventory by using the S3 console.

> **Note**
>
> It might take up to 48 hours for Amazon S3 to deliver the first inventory report.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**. In the **Buckets** list, choose the name of the bucket that you want to configure Amazon S3 Inventory for.

3. Choose the **Management** tab.
4. Under **Inventory configurations**, choose **Create inventory configuration**.
5. For **Inventory configuration name**, enter a name.
6. For **Inventory scope**, do the following:

   - Enter an optional prefix.
   - Choose which object versions to include, either **Current versions only** or **Include all versions**.
7. Under **Report details**, choose the location of the AWS account that you want to save the reports to: **This account** or **A different account**.
8. Under **Destination**, choose the destination bucket where you want the inventory reports to be saved.

   The destination bucket must be in the same AWS Region as the bucket for which you are setting up the inventory. The destination bucket can be in a different AWS account. When specifying the destination bucket, you can also include an optional prefix to group your inventory reports together.

   Under the **Destination** bucket field, you see the **Destination bucket permission** statement that is added to the destination bucket policy to allow Amazon S3 to place data in that bucket. For more information, see Creating a destination bucket policy.
9. Under **Frequency**, choose how often the report will be generated, **Daily** or **Weekly**.
10. For **Output format**, choose one of the following formats for the report:

    - **CSV** – If you plan to use this inventory report with S3 Batch Operations or if you want to analyze this report in another tool, such as Microsoft Excel, choose **CSV**.
    - **Apache ORC**
    - **Apache Parquet**
11. Under **Status**, choose **Enable** or **Disable**.
12. To configure server-side encryption, under **Inventory report encryption**, follow these steps:

    a. Under **Server-side encryption**, choose either **Do not specify an encryption key** or **Specify an encryption key** to encrypt data.

       - To keep the bucket settings for default server-side encryption of objects when storing them in Amazon S3, choose **Do not specify an encryption key**. As long as the bucket

destination has S3 Bucket Keys enabled, the copy operation applies an S3 Bucket Key at the destination bucket.

> **Note**
>
> If the bucket policy for the specified destination requires objects to be encrypted before storing them in Amazon S3, you must choose **Specify an encryption key**. Otherwise, copying objects to the destination will fail.

- To encrypt objects before storing them in Amazon S3, choose **Specify an encryption key**.

b. If you chose **Specify an encryption key**, under **Encryption type**, you must choose either **Amazon S3 managed key (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**.

SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt each object. SSE-KMS provides you with more control over your key. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3). For more information about SSE-KMS, see Using server-side encryption with AWS KMS keys (SSE-KMS).

> **Note**
>
> To encrypt the inventory list file with SSE-KMS, you must grant Amazon S3 permission to use the customer managed key. For instructions, see Grant Amazon S3 Permission to Encrypt Using Your KMS Keys.

c. If you chose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify your AWS KMS key through one of the following options.

> **Note**
>
> If the destination bucket that stores the inventory list file is owned by a different AWS account, make sure that you use a fully qualified KMS key ARN to specify your KMS key.

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose a symmetric encryption KMS key from the list of available keys. Make sure the KMS key is in the same Region as your bucket.

  > **Note**
  >
  > Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list. However, the AWS managed key (`aws/s3`) is not supported for SSE-KMS encryption with S3 Inventory.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

13. For **Additional metadata fields**, select one or more of the following to add to the inventory report:

    - **Size** – The object size in bytes, not including the size of incomplete multipart uploads, object metadata, and delete markers.
    - **Last modified date** – The object creation date or the last modified date, whichever is the latest.
    - **Multipart upload** – Specifies that the object was uploaded as a multipart upload. For more information, see Uploading and copying objects using multipart upload.
    - **Replication status** – The replication status of the object. For more information, see Getting replication status information.
    - **Encryption status** – The server-side encryption type that's used to encrypt the object. For more information, see Protecting data with server-side encryption.
    - **Bucket Key status** – Indicates whether a bucket-level key generated by AWS KMS applies to the object. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.
    - **Object access control list** – An access control list (ACL) for each object that defines which AWS accounts or groups are granted access to this object and the type of access that is granted. For more information about this field, see Working with the Object ACL field. For more information about ACLs, see Access control list (ACL) overview.

- **Object owner** – The owner of the object.
- **Storage class** – The storage class that's used for storing the object.
- **Intelligent-Tiering: Access tier** – Indicates the access tier (frequent or infrequent) of the object if it was stored in the S3 Intelligent-Tiering storage class. For more information, see Storage class for automatically optimizing data with changing or unknown access patterns.
- **ETag** – The entity tag (ETag) is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag might or might not be an MD5 digest of the object data. Whether it is depends on how the object was created and how it is encrypted. For more information, see Object in the *Amazon Simple Storage Service API Reference*.
- **Checksum algorithm** – Indicates the algorithm that is used to create the checksum for the object.
- **All Object Lock configurations** – The Object Lock status of the object, including the following settings:
  - **Object Lock: Retention mode** – The level of protection applied to the object, either *Governance* or *Compliance*.
  - **Object Lock: Retain until date** – The date until which the locked object cannot be deleted.
  - **Object Lock: Legal hold status** – The legal hold status of the locked object.

  For information about S3 Object Lock, see How S3 Object Lock works.

For more information about the contents of an inventory report, see Amazon S3 Inventory list.

For more information about restricting access to certain optional metadata fields in an inventory configuration, see Control S3 Inventory report configuration creation.

14. Choose **Create**.

## Using the REST API to work with S3 Inventory

The following are the REST operations that you can use to work with Amazon S3 Inventory.

- DeleteBucketInventoryConfiguration
- GetBucketInventoryConfiguration
- ListBucketInventoryConfigurations

- PutBucketInventoryConfiguration

## Setting up Amazon S3 Event Notifications for inventory completion

You can set up an Amazon S3 event notification to receive notice when the manifest checksum file is created, which indicates that an inventory list has been added to the destination bucket. The manifest is an up-to-date list of all the inventory lists at the destination location.

Amazon S3 can publish events to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

The following notification configuration defines that all `manifest.checksum` files newly added to the destination bucket are processed by the AWS Lambda `cloud-function-list-write`.

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>checksum</Value>
              </FilterRule>
          </S3Key>
     </Filter>
     <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
Cloudcode>
     <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
  </NotificationConfiguration>
```

For more information, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

## Locating your inventory list

When an inventory list is published, the manifest files are published to the following location in the destination bucket.

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
 destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- *destination-prefix* is the object key name prefix that is optionally specified in the inventory configuration. You can use this prefix to group all the inventory list files in a common location within the destination bucket.
- *source-bucket* is the source bucket that the inventory list is for. The source bucket name is added to prevent collisions when multiple inventory reports from different source buckets are sent to the same destination bucket.
- *config-ID* is added to prevent collisions with multiple inventory reports from the same source bucket that are sent to the same destination bucket. The *config-ID* comes from the inventory report configuration, and is the name for the report that is defined during setup.
- *YYYY-MM-DDTHH-MMZ* is the timestamp that consists of the start time and the date when the inventory report generation process begins scanning the bucket; for example, `2016-11-06T21-32Z`.
- `manifest.json` is the manifest file.
- `manifest.checksum` is the MD5 hash of the content of the `manifest.json` file.
- `symlink.txt` is the Apache Hive-compatible manifest file.

The inventory lists are published daily or weekly to the following location in the destination bucket.

```
destination-prefix/source-bucket/config-ID/data/example-file-name.csv.gz
...
destination-prefix/source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix* is the object key name prefix that is optionally specified in the inventory configuration. You can use this prefix to group all the inventory list files in a common location in the destination bucket.

- *source-bucket* is the source bucket that the inventory list is for. The source bucket name is added to prevent collisions when multiple inventory reports from different source buckets are sent to the same destination bucket.
- *example-file-name*.csv.gz is one of the CSV inventory files. ORC inventory names end with the file name extension .orc, and Parquet inventory names end with the file name extension .parquet.

## Inventory manifest

The manifest files manifest.json and symlink.txt describe where the inventory files are located. Whenever a new inventory list is delivered, it is accompanied by a new set of manifest files. These files might overwrite each other. In versioning-enabled buckets, Amazon S3 creates new versions of the manifest files.

Each manifest contained in the manifest.json file provides metadata and other basic information about an inventory. This information includes the following:

- The source bucket name
- The destination bucket name
- The version of the inventory
- The creation timestamp in the epoch date format that consists of the start time and the date when the inventory report generation process begins scanning the bucket
- The format and schema of the inventory files
- A list of the inventory files that are in the destination bucket

Whenever a manifest.json file is written, it is accompanied by a manifest.checksum file that is the MD5 hash of the content of the manifest.json file.

### Example Inventory manifest in a manifest.json file

The following examples show an inventory manifest in a manifest.json file for CSV, ORC, and Parquet-formatted inventories.

CSV

The following is an example of a manifest in a manifest.json file for a CSV-formatted inventory.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 Size, LastModifiedDate, ETag, StorageClass, IsMultipartUploaded,
 ReplicationStatus, EncryptionStatus, ObjectLockRetainUntilDate, ObjectLockMode,
 ObjectLockLegalHoldStatus, IntelligentTieringAccessTier, BucketKeyStatus,
 ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner",
    "files": [
        {
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
}
```

ORC

The following is an example of a manifest in a `manifest.json` file for an ORC-formatted inventory.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boole
    "files": [
        {
            "key": "inventory/example-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
```

```
}
```

Parquet

The following is an example of a manifest in a `manifest.json` file for a Parquet-formatted inventory.

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8);
 required binary key (UTF8); optional binary version_id (UTF8); optional boolean
 is_latest; optional boolean is_delete_marker; optional int64 size; optional
 int64 last_modified_date (TIMESTAMP_MILLIS); optional binary e_tag (UTF8);
 optional binary storage_class (UTF8); optional boolean is_multipart_uploaded;
 optional binary replication_status (UTF8); optional binary encryption_status
 (UTF8); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);}",
    "files": [
        {
           "key": "inventory/example-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
}
```

The `symlink.txt` file is an Apache Hive-compatible manifest file that allows Hive to automatically discover inventory files and their associated data files. The Hive-compatible manifest works with the Hive-compatible services Athena and Amazon Redshift Spectrum. It also works with Hive-compatible applications, including Presto, Apache Hive, Apache Spark, and many others.

**Important**

The `symlink.txt` Apache Hive-compatible manifest file does not currently work with AWS Glue.
Reading the `symlink.txt` file with Apache Hive and Apache Spark is not supported for ORC and Parquet-formatted inventory files.

## Querying Amazon S3 Inventory with Amazon Athena

You can query Amazon S3 Inventory files with standard SQL queries by using Amazon Athena in all Regions where Athena is available. To check for AWS Region availability, see the AWS Region Table.

Athena can query Amazon S3 Inventory files in Apache optimized row columnar (ORC), Apache Parquet, or comma-separated values (CSV) format. When you use Athena to query inventory files, we recommend that you use ORC-formatted or Parquet-formatted inventory files. The ORC and Parquet formats provide faster query performance and lower query costs. ORC and Parquet are self-describing, type-aware columnar file formats designed for Apache Hadoop. The columnar format lets the reader read, decompress, and process only the columns that are required for the current query. The ORC and Parquet formats for Amazon S3 Inventory are available in all AWS Regions.

**To use Athena to query Amazon S3 Inventory files**

1. Create an Athena table. For information about creating a table, see Creating Tables in Amazon Athena in the *Amazon Athena User Guide*.
2. Create your query by using one of the following sample query templates, depending on whether you're querying an ORC-formatted, a Parquet-formatted, or a CSV-formatted inventory report.

   - When you're using Athena to query an ORC-formatted inventory report, use the following sample query as a template.

     The following sample query includes all the optional fields in an ORC-formatted inventory report.

     To use this sample query, do the following:

     - Replace *`your_table_name`* with the name of the Athena table that you created.

- Remove any optional fields that you did not choose for your inventory so that the query corresponds to the fields chosen for your inventory.
- Replace the following bucket name and inventory location (the configuration ID) as appropriate for your configuration.

  `s3://amzn-s3-demo-bucket/config-ID/hive/`
- Replace the *2022-01-01-00-00* date under `projection.dt.range` with the first day of the time range within which you partition the data in Athena. For more information, see Partitioning data in Athena.

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size bigint,
         last_modified_date timestamp,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date bigint,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
```

```
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- When you're using Athena to query a Parquet-formatted inventory report, use the sample query for an ORC-formatted report. However, use the following Parquet SerDe in place of the ORC SerDe in the `ROW FORMAT SERDE` statement.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

- When you're using Athena to query a CSV-formatted inventory report, use the following sample query as a template.

  The following sample query includes all the optional fields in an CSV-formatted inventory report.

  To use this sample query, do the following:

  - Replace *`your_table_name`* with the name of the Athena table that you created.
  - Remove any optional fields that you did not choose for your inventory so that the query corresponds to the fields chosen for your inventory.
  - Replace the following bucket name and inventory location (the configuration ID) as appropriate for your configuration.

    `s3://`*`amzn-s3-demo-bucket`*`/`*`config-ID`*`/hive/`
  - Replace the *`2022-01-01-00-00`* date under `projection.dt.range` with the first day of the time range within which you partition the data in Athena. For more information, see Partitioning data in Athena.

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
```

```
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. You can now run various queries on your inventory, as shown in the following examples. Replace each *user input placeholder* with your own information.

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the encryption status for a provided report date.
SELECT encryption_status, count(*) FROM your_table_name WHERE dt = 'YYYY-MM-DD-HH-
MM' GROUP BY encryption_status;

# Get the encryption status for inventory report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your_table_name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

When you configure S3 Inventory to add the Object Access Control List (Object ACL) field to an inventory report, the report displays the value for the Object ACL field as a base64-encoded string. To get the decoded value in JSON for the Object ACL field, you can query this field by using Athena. See the following query examples. For more information about the Object ACL field, see Working with the Object ACL field.

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
```

```
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
FROM your_table_name;
```

For more information about using Athena, see the Amazon Athena User Guide.

# Converting empty version ID strings in Amazon S3 Inventory reports to null strings

> **Note**
>
> **The following procedure applies only to Amazon S3 Inventory reports that include all versions, and only if the "all versions" reports are used as manifests for S3 Batch Operations on buckets that have S3 Versioning enabled.** You are not required to convert strings for S3 Inventory reports that specify the current version only.

You can use S3 Inventory reports as manifests for S3 Batch Operations. However, when S3 Versioning is enabled on a bucket, S3 Inventory reports that include all versions mark any null-versioned objects with empty strings in the version ID field. When an Inventory Report includes all object version IDs, Batch Operations recognizes `null` strings as version IDs, but not empty strings.

When an S3 Batch Operations job uses an "all versions" S3 Inventory report as a manifest, it fails all tasks on objects that have an empty string in the version ID field. To convert empty strings in the version ID field of the S3 Inventory report to `null` strings for Batch Operations, use the following procedure.

**Update an Amazon S3 Inventory report for use with Batch Operations**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Navigate to your S3 Inventory report. The inventory report is located in the destination bucket that you specified while configuring your inventory report. For more information about locating inventory reports, see Locating your inventory list.
    a. Choose the destination bucket.
    b. Choose the folder. The folder is named after the original source bucket.
    c. Choose the folder named after the inventory configuration.
    d. Select the check box next to the folder named **hive**. At the top of the page, choose **Copy S3 URI** to copy the S3 URI for the folder.
3. Open the Amazon Athena console at https://console.aws.amazon.com/athena/.
4. In the query editor, choose **Settings**, then choose **Manage**. On the **Manage settings** page, for **Location of query result**, choose an S3 bucket to store your query results in.

5. In the query editor, create an Athena table to hold the data in the inventory report using the following command. Replace *table_name* with a name of your choosing, and in the LOCATION clause, insert the S3 URI that you copied earlier. Then choose **Run** to run the query.

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. To clear the query editor, choose **Clear**. Then load the inventory report into the table using the following command. Replace *table_name* with the one that you chose in the prior step. Then choose **Run** to run the query.

```
MSCK REPAIR TABLE table_name;
```

7. To clear the query editor, choose **Clear**. Run the following SELECT query to retrieve all entries in the original inventory report and replace any empty version IDs with null strings. Replace *table_name* with the one that you chose earlier, and replace *YYYY-MM-DD-HH-MM* in the WHERE clause with the date of the inventory report that you want this tool to run on. Then choose **Run** to run the query.

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. Return to the Amazon S3 console (https://console.aws.amazon.com/s3/), and navigate to the S3 bucket that you chose for **Location of query result** earlier. Inside, there should be a series of folders ending with the date.

   For example, you should see something like **s3://*DOC-EXAMPLE-BUCKET*/*query-result-location*/Unsaved/2021/10/07/**. You should see .csv files containing the results of the SELECT query that you ran.

   Choose the CSV file with the latest modified date. Download this file to your local machine for the next step.

9. The generated CSV file contains a header row. To use this CSV file as input for an S3 Batch Operations job, you must remove the header row, because Batch Operations doesn't support header rows on CSV manifests.

To remove the header row, you can run one of the following commands on the file. Replace *file.csv* with the name of your CSV file.

**For macOS and Linux machines**, run the `tail` command in a Terminal window.

```
tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
```

**For Windows machines**, run the following script in a Windows PowerShell window. Replace *File-location* with the path to your file, and *file.csv* with the file name.

```
$ins = New-Object System.IO.StreamReader File-location\file.csv
$outs = New-Object System.IO.StreamWriter File-location\temp.csv
try {
    $skip = 0
    while ( !$ins.EndOfStream ) {
        $line = $ins.ReadLine();
        if ( $skip -ne 0 ) {
            $outs.WriteLine($line);
        } else {
            $skip = 1
        }
    }
} finally {
    $outs.Close();
    $ins.Close();
}
Move-Item File-location\temp.csv File-location\file.csv -Force
```

10. After removing the header row from the CSV file, you are ready to use it as a manifest in an S3 Batch Operations job. Upload the CSV file to an S3 bucket or location of your choosing, and then create a Batch Operations job using the CSV file as the manifest.

    For more information about creating a Batch Operations job, see Creating an S3 Batch Operations job.

## Working with the Object ACL field

An Amazon S3 Inventory report contains a list of the objects in the S3 source bucket and metadata for each object. The Object access control list (ACL) field is a metadata field that is available in

Amazon S3 Inventory. Specifically, the Object ACL field contains the access control list (ACL) for each object. The ACL for an object defines which AWS accounts or groups are granted access to this object and the type of access that is granted. For more information, see Access control list (ACL) overview and Amazon S3 Inventory list.

The Object ACL field in Amazon S3 Inventory reports is defined in JSON format. The JSON data includes the following fields:

- `version` – The version of the Object ACL field format in the inventory reports. It's in date format `yyyy-mm-dd`.
- `status` – Possible values are `AVAILABLE` or `UNAVAILABLE` to indicate whether an Object ACL is available for an object. When the status for the Object ACL is `UNAVAILABLE`, the value of the Object Owner field in the inventory report is also `UNAVAILABLE`.
- `grants` – Grantee-permission pairs that list the permission status of each grantee that is granted by the Object ACL. The available values for a grantee are `CanonicalUser` and `Group`. For more information about grantees, see Grantees in access control lists.

  For a grantee with the `Group` type, a grantee-permission pair includes the following attributes:

  - `uri` – A predefined Amazon S3 group.
  - `permission` – The ACL permissions that are granted on the object. For more information, see ACL permissions on an object.
  - `type` – The type `Group`, which denotes that the grantee is group.

  For a grantee with the `CanonicalUser` type, a grantee-permission pair includes the following attributes:

  - `canonicalId` – An obfuscated form of the AWS account ID. The canonical user ID for an AWS account is specific to that account. You can retrieve the canonical user ID. For more information see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.

    **Note**

    If a grantee in an ACL is the email address of an AWS account, S3 Inventory uses the `canonicalId` of that AWS account and the `CanonicalUser` type to specify this grantee. For more information, see Grantees in access control lists.

- `permission` – The ACL permissions that are granted on the object. For more information, see ACL permissions on an object.
- `type` – The type `CanonicalUser`, which denotes that the grantee is an AWS account.

The following example shows possible values for the Object ACL field in JSON format:

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
        "type": "CanonicalUser"
    }]
}
```

**Note**

The Object ACL field is defined in JSON format. An inventory report displays the value for the Object ACL field as a base64-encoded string.
For example, suppose that you have the following Object ACL field in JSON format:

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
}
```

The Object ACL field is encoded and shown as the following base64-encoded string:

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
```

To get the decoded value in JSON for the Object ACL field, you can query this field in Amazon Athena. For query examples, see Querying Amazon S3 Inventory with Amazon Athena.

# Replicating objects overview

You can use replication to enable automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. You can replicate objects to a single destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Regions or within the same Region as the source bucket.

There are two types of replication: *live replication* and *on-demand replication*.

- **Live replication** – **To automatically replicate new and updated objects** as they are written to the source bucket, use live replication. Live replication doesn't replicate any objects that existed in the bucket before you set up replication. To replicate objects that existed before you set up replication, use on-demand replication.
- **On-demand replication** – **To replicate existing objects** from the source bucket to one or more destination buckets on demand, use S3 Batch Replication. For more information about replicating existing objects, see When to use S3 Batch Replication.

There are two forms of live replication: *Cross-Region Replication (CRR)* and *Single-Region Replication (SRR)*.

- **Cross-Region Replication (CRR)** – You can use CRR to replicate objects across Amazon S3 buckets in different AWS Regions. For more information about CRR, see the section called “When to use Cross-Region Replication”.
- **Single-Region Replication (SRR)** – You can use SRR to copy objects across Amazon S3 buckets in the same AWS Region. For more information about SRR, see the section called “When to use Same-Region Replication”.

### Topics

- When to use Same-Region Replication
- When to use two-way replication (bi-directional replication)
- When to use S3 Batch Replication
- Workload requirements and live replication
- What does Amazon S3 replicate?
- Requirements and considerations for replication
- Setting up live replication
- Managing or pausing live replication
- Monitoring progress with replication metrics and S3 Event Notifications
- Replicating existing objects with S3 Batch Replication

## Why use replication?

Replication can help you do the following:

- **Replicate objects while retaining metadata** – You can use replication to make copies of your objects that retain all metadata, such as the original object creation times and version IDs. This capability is important if you must ensure that your replica is identical to the source object.
- **Replicate objects into different storage classes** – You can use replication to directly put objects into S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or another storage class in the destination buckets. You can also replicate your data to the same storage class and use lifecycle configurations on the destination buckets to move your objects to a colder storage class as they age.
- **Maintain object copies under different ownership** – Regardless of who owns the source object, you can tell Amazon S3 to change replica ownership to the AWS account that owns the destination bucket. This is referred to as the *owner override* option. You can use this option to restrict access to object replicas.
- **Keep objects stored over multiple AWS Regions** – To ensure geographic differences in where your data is kept, you can set multiple destination buckets across different AWS Regions. This feature might help you meet certain compliance requirements.
- **Replicate objects within 15 minutes** – To replicate your data in the same AWS Region or across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3 RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes

(backed by a service-level agreement). For more information, see the section called "Using S3 Replication Time Control".

> **Note**
>
> S3 RTC does not apply to Batch Replication. Batch Replication is an on-demand replication job, and can be tracked with S3 Batch Operations. For more information, see Tracking job status and completion reports.

- **Sync buckets, replicate existing objects, and replicate previously failed or replicated objects** – To sync buckets and replicate existing objects, use Batch Replication as an on-demand replication action. For more information about when to use Batch Replication, see When to use S3 Batch Replication.
- **Replicate objects and fail over to a bucket in another AWS Region** – To keep all metadata and objects in sync across buckets during data replication, use two-way replication (also known as bi-directional replication) rules before configuring Amazon S3 Multi-Region Access Point failover controls. Two-way replication rules help ensure that when data is written to the S3 bucket that traffic fails over to, that data is then replicated back to the source bucket.

## When to use Cross-Region Replication

S3 Cross-Region Replication (CRR) is used to copy objects across Amazon S3 buckets in different AWS Regions. CRR can help you do the following:

- **Meet compliance requirements** – Although Amazon S3 stores your data across multiple geographically distant Availability Zones by default, compliance requirements might dictate that you store data at even greater distances. To satisfy these requirements, use Cross-Region Replication to replicate data between distant AWS Regions.
- **Minimize latency** – If your customers are in two geographic locations, you can minimize latency in accessing objects by maintaining object copies in AWS Regions that are geographically closer to your users.
- **Increase operational efficiency** – If you have compute clusters in two different AWS Regions that analyze the same set of objects, you might choose to maintain object copies in those Regions.

## When to use Same-Region Replication

Same-Region Replication (SRR) is used to copy objects across Amazon S3 buckets in the same AWS Region. SRR can help you do the following:

- **Aggregate logs into a single bucket** – If you store logs in multiple buckets or across multiple accounts, you can easily replicate logs into a single, in-Region bucket. Doing so allows for simpler processing of logs in a single location.
- **Configure live replication between production and test accounts** – If you or your customers have production and test accounts that use the same data, you can replicate objects between those multiple accounts, while maintaining object metadata.
- **Abide by data sovereignty laws** – You might be required to store multiple copies of your data in separate AWS accounts within a certain Region. Same-Region Replication can help you automatically replicate critical data when compliance regulations don't allow the data to leave your country.

## When to use two-way replication (bi-directional replication)

- **Build shared datasets across multiple AWS Regions** – With replica modification sync, you can easily replicate metadata changes, such as object access control lists (ACLs), object tags, or object locks, on replication objects. This two-way replication is important if you want to keep all objects and object metadata changes in sync. You can enable replica modification sync on a new or existing replication rule when performing two-way replication between two or more buckets in the same or different AWS Regions.
- **Keep data synchronized across Regions during failover** – You can synchronize data in buckets between AWS Regions by configuring two-way replication rules with S3 Cross-Region Replication (CRR) directly from a Multi-Region Access Point. To make an informed decision on when to initiate failover, you can also enable S3 replication metrics so that you can monitor the replication in Amazon CloudWatch, in S3 Replication Time Control (S3 RTC), or from the Multi-Region Access Point.
- **Make your application highly available** – Even in the event of a Regional traffic disruption, you can use two-way replication rules to keep all metadata and objects in sync across buckets during data replication.

## When to use S3 Batch Replication

Batch Replication replicates existing objects to different buckets as an on-demand option. Unlike live replication, these jobs can be run as needed. Batch Replication can help you do the following:

- **Replicate existing objects** – You can use Batch Replication to replicate objects that were added to the bucket before Same-Region Replication or Cross-Region Replication were configured.
- **Replicate objects that previously failed to replicate** – You can filter a Batch Replication job to attempt to replicate objects with a replication status of **FAILED**.
- **Replicate objects that were already replicated** – You might be required to store multiple copies of your data in separate AWS accounts or AWS Regions. Batch Replication can replicate existing objects to newly added destinations.
- **Replicate replicas of objects that were created from a replication rule** – Replication configurations create replicas of objects in destination buckets. Replicas of objects can be replicated only with Batch Replication.

## Workload requirements and live replication

Depending on your workload requirements, some types of replication will be better suited to your use case than others. Use the following table to determine which type of replication to use for your situation, and whether to use S3 Replication Time Control (S3 RTC) for your workload. S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement, or SLA). For more information, see the section called “Using S3 Replication Time Control”.

**Workload requirements for replication comparison**

| Workload requirement | S3 RTC (15-minute SLA) | Cross-Region Replication (CRR) | Single-Region Replication (SRR) |
|---|---|---|---|
| Replicate objects between different AWS accounts | Yes | Yes | Yes |
| Replicate objects within the same AWS Region within | No | No | Yes |

| Workload requirement | S3 RTC (15-minute SLA) | Cross-Region Replication (CRR) | Single-Region Replication (SRR) |
|---|---|---|---|
| 24-48 hours (not SLA backed) | | | |
| Replicate objects between different AWS Regions within 24-48 hours (not SLA backed) | No | Yes | No |
| Predictable replication time: Backed by SLA to replicate 99.9 percent of objects within 15 minutes | Yes | No | No |

## What does Amazon S3 replicate?

Amazon S3 replicates only specific items in buckets that are configured for replication.

**Topics**

- What is replicated with replication configurations?
- What isn't replicated with replication configurations?
- How default bucket encryption affects replication

### What is replicated with replication configurations?

By default, Amazon S3 replicates the following:

- Objects created after you add a replication configuration.
- Unencrypted objects.
- Objects encrypted using customer provided keys (SSE-C), objects encrypted at rest under an Amazon S3 managed key (SSE-S3) or a KMS key stored in AWS Key Management Service (SSE-KMS). For more information, see the section called “Replicating encrypted objects”.

- Object metadata from the source objects to the replicas. For information about replicating metadata from the replicas to the source objects, see Replicating metadata changes with Amazon S3 replica modification sync.
- Only objects in the source bucket for which the bucket owner has permissions to read objects and access control lists (ACLs).

  For more information about resource ownership, see Amazon S3 bucket and object ownership.
- Object ACL updates, unless you direct Amazon S3 to change the replica ownership when source and destination buckets aren't owned by the same accounts.

  For more information, see Changing the replica owner.

  It can take a while until Amazon S3 can bring the two ACLs in sync. This change in ownership applies only to objects created after you add a replication configuration to the bucket.
- Object tags, if there are any.
- S3 Object Lock retention information, if there is any.

  When Amazon S3 replicates objects that have retention information applied, it applies those same retention controls to your replicas, overriding the default retention period configured on your destination buckets. If you don't have retention controls applied to the objects in your source bucket, and you replicate into destination buckets that have a default retention period set, the destination bucket's default retention period is applied to your object replicas. For more information, see Using S3 Object Lock.

**How delete operations affect replication**

If you delete an object from the source bucket, the following actions occur by default:

- If you make a DELETE request without specifying an object version ID, Amazon S3 adds a delete marker. Amazon S3 deals with the delete marker as follows:
  - If you are using the latest version of the replication configuration (that is, you specify the `Filter` element in a replication configuration rule), Amazon S3 does not replicate the delete marker by default. However, you can add *delete marker replication* to non-tag-based rules. For more information, see Replicating delete markers between buckets.
  - If you don't specify the `Filter` element, Amazon S3 assumes that the replication configuration is version V1, and it replicates delete markers that resulted from user actions.

However, if Amazon S3 deletes an object due to a lifecycle action, the delete marker is not replicated to the destination buckets.

- If you specify an object version ID to delete in a DELETE request, Amazon S3 deletes that object version in the source bucket. But it doesn't replicate the deletion in the destination buckets. In other words, it doesn't delete the same object version from the destination buckets. This protects data from malicious deletions.

## What isn't replicated with replication configurations?

By default, Amazon S3 doesn't replicate the following:

- Objects in the source bucket that are replicas that were created by another replication rule. For example, suppose you configure replication where bucket A is the source and bucket B is the destination. Now suppose that you add another replication configuration where bucket B is the source and bucket C is the destination. In this case, objects in bucket B that are replicas of objects in bucket A are not replicated to bucket C.

  To replicate objects that are replicas, use Batch Replication. Learn more about configuring Batch Replication at Replicating existing objects.

- Objects in the source bucket that have already been replicated to a different destination. For example, if you change the destination bucket in an existing replication configuration, Amazon S3 won't replicate the objects again.

  To replicate previously replicated objects, use Batch Replication. Learn more about configuring Batch Replication at Replicating existing objects.

- Batch Replication does not support re-replicating objects that were deleted with the version ID of the object from the destination bucket. To re-replicate these objects, you can copy the source objects in place with a Batch Copy job. Copying those objects in place creates new versions of the objects in the source bucket and initiates replication automatically to the destination. For more information about how to use Batch Copy, see, Examples that use Batch Operations to copy objects.

- By default, when replicating from a different AWS account, delete markers added to the source bucket are not replicated.

  For information about how to replicate delete markers, see Replicating delete markers between buckets.

- Objects that are stored in the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access, or S3 Intelligent-Tiering Deep Archive Access storage classes or tiers. You cannot replicate these objects until you restore them and copy them to a different storage class.

  To learn more about S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive, see Storage classes for rarely accessed objects.

  To learn more about the S3 Intelligent-Tiering, see Amazon S3 Intelligent-Tiering.

- Objects in the source bucket that the bucket owner doesn't have sufficient permissions to replicate.

  For information about how an object owner can grant permissions to a bucket owner, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- Updates to bucket-level subresources.

  For example, if you change the lifecycle configuration or add a notification configuration to your source bucket, these changes are not applied to the destination bucket. This feature makes it possible to have different configurations on source and destination buckets.

- Actions performed by lifecycle configuration.

  For example, if lifecycle configuration is enabled only on your source bucket, Amazon S3 creates delete markers for expired objects but doesn't replicate those markers. If you want the same lifecycle configuration applied to both the source and destination buckets, enable the same lifecycle configuration on both. For more information about lifecycle configuration, see Managing your storage lifecycle.

- When you're using tag-based replication rules with live replication, new objects must be tagged with the matching replication rule tag in the `PutObject` operation. Otherwise, the objects won't be replicated. If objects are tagged after the `PutObject` operation, those objects also won't be replicated.

  To replicate objects that have been tagged after the `PutObject` operation, you must use S3 Batch Replication. For more information about Batch Replication, see Replicating existing objects.

### How default bucket encryption affects replication

When you enable default encryption for a replication destination bucket, the following encryption behavior applies:

- If objects in the source bucket are not encrypted, the replica objects in the destination bucket are encrypted by using the default encryption settings of the destination bucket. As a result, the entity tags (ETags) of the source objects differ from the ETags of the replica objects. If you have applications that use ETags, you must update those applications to account for this difference.
- If objects in the source bucket are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), the replica objects in the destination bucket use the same type of encryption as the source objects. The default encryption settings of the destination bucket are not used.

## Requirements and considerations for replication

Amazon S3 replication requires the following:

- The source bucket owner must have the source and destination AWS Regions enabled for their account. The destination bucket owner must have the destination Region enabled for their account.

  For more information about enabling or disabling an AWS Region, see Managing AWS Regions in the *AWS General Reference*.
- Both source and destination buckets must have versioning enabled. For more information about versioning, see Using versioning in S3 buckets.
- Amazon S3 must have permissions to replicate objects from the source bucket to the destination bucket or buckets on your behalf. For more information about these permissions, see Setting up permissions for live replication.
- If the owner of the source bucket doesn't own the object in the bucket, the object owner must grant the bucket owner `READ` and `READ_ACP` permissions with the object access control list (ACL). For more information, see Access control list (ACL) overview.
- If the source bucket has S3 Object Lock enabled, the destination buckets must also have S3 Object Lock enabled.

To enable replication on a bucket that has Object Lock enabled, you must use the AWS Command Line Interface, REST API, or AWS SDKs. For more general information, see Using S3 Object Lock.

> **Note**
>
> You must grant two new permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two new permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission, it satisfies the requirement. For more information, see Setting up permissions for live replication.

For more information, see Setting up live replication.

If you are setting the replication configuration in a *cross-account scenario*, where the source and destination buckets are owned by different AWS accounts, the following additional requirement applies:

- The owner of the destination buckets must grant the owner of the source bucket permissions to replicate objects with a bucket policy. For more information, see Granting permissions when the source and destination buckets are owned by different AWS accounts.
- The destination buckets cannot be configured as Requester Pays buckets. For more information, see Using Requester Pays buckets for storage transfers and usage.

## Considerations for replication

Before you create a replication configuration, be aware of the following considerations.

**Topics**

- Lifecycle configuration and object replicas
- Versioning configuration and replication configuration
- Using S3 Replication with S3 Intelligent-Tiering
- Logging configuration and replication configuration
- CRR and the destination Region
- S3 Batch Replication

- S3 Replication Time Control

### Lifecycle configuration and object replicas

The time it takes for Amazon S3 to replicate an object depends on the size of the object. For large objects, it can take several hours. Although it might take a while before a replica is available in the destination, it takes the same amount of time to create the replica as it took to create the corresponding object in the source bucket. If a lifecycle configuration is enabled on a destination bucket, the lifecycle rules honor the original creation time of the object, not when the replica became available in the destination bucket.

Replication configuration requires the bucket to be versioning-enabled. When you enable versioning on a bucket, keep the following in mind:

- If you have an object Expiration lifecycle configuration, after you enable versioning, add a `NonCurrentVersionExpiration` policy to maintain the same permanent delete behavior as before you enabled versioning.
- If you have a Transition lifecycle configuration, after you enable versioning, consider adding a `NonCurrentVersionTransition` policy.

### Versioning configuration and replication configuration

Both the source and destination buckets must be versioning-enabled when you configure replication on a bucket. After you enable versioning on both the source and destination buckets and configure replication on the source bucket, you will encounter the following issues:

- If you attempt to disable versioning on the source bucket, Amazon S3 returns an error. You must remove the replication configuration before you can disable versioning on the source bucket.
- If you disable versioning on the destination bucket, replication fails. The source object has the replication status `FAILED`.

### Using S3 Replication with S3 Intelligent-Tiering

S3 Intelligent-Tiering is a storage class that is designed to optimize storage costs by automatically moving data to the most cost-effective access tier. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers.

Replicating objects stored in S3 Intelligent-Tiering with S3 Batch Replication or invoking CopyObject or UploadPartCopy constitutes access. In these cases, the source objects of the copy or replication operations are tiered up.

For more information about S3 Intelligent-Tiering see, Amazon S3 Intelligent-Tiering.

**Logging configuration and replication configuration**

If Amazon S3 delivers logs to a bucket that has replication enabled, it replicates the log objects.

If server access logs (Logging requests with server access logging) or AWS CloudTrail logs (Logging Amazon S3 API calls using AWS CloudTrail) are enabled on your source or destination bucket, Amazon S3 includes replication-related requests in the logs. For example, Amazon S3 logs each object that it replicates.

**CRR and the destination Region**

Amazon S3 Cross-Region Replication (CRR) is used to copy objects across S3 buckets in different AWS Regions. You might choose the Region for your destination bucket based on either your business needs or cost considerations. For example, inter-Region data transfer charges vary depending on the Regions that you choose.

Suppose that you chose US East (N. Virginia) (us-east-1) as the Region for your source bucket. If you choose US West (Oregon) (us-west-2) as the Region for your destination buckets, you pay more than if you choose the US East (Ohio) (us-east-2) Region. For pricing information, see "Data Transfer Pricing" in Amazon S3 pricing.

There are no data transfer charges associated with Same-Region Replication (SRR).

**S3 Batch Replication**

For information about considerations for Batch Replication, see S3 Batch Replication considerations.

**S3 Replication Time Control**

For information about best practices and considerations for S3 Replication Time Control (S3 RTC), see Best practices and guidelines for S3 RTC.

# Setting up live replication

> **Note**
>
> Objects that existed before you set up replication aren't replicated automatically. In other words, Amazon S3 doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration, use S3 Batch Replication. Learn more about configuring Batch Replication at Replicating existing objects.

To enable live replication—Same-Region Replication (SRR) or Cross-Region Replication (CRR)—add a replication configuration to your source bucket. This configuration tells Amazon S3 to replicate objects as specified. In the replication configuration, you must provide the following:

- **The destination buckets** – The bucket or buckets where you want Amazon S3 to replicate the objects.
- **The objects that you want to replicate** – You can replicate all of the objects in the source bucket or a subset. You identify a subset by providing a key name prefix, one or more object tags, or both in the configuration.

  For example, if you configure a replication rule to replicate only objects with the key name prefix `Tax/`, Amazon S3 replicates objects with keys such as `Tax/doc1` or `Tax/doc2`. But it doesn't replicate objects with the key `Legal/doc3`. If you specify both a prefix and one or more tags, Amazon S3 replicates only objects that have the specific key prefix and tags.
- **An AWS Identity and Access Management (IAM) role** – Amazon S3 assumes this IAM role to replicate objects on your behalf.

In addition to these minimum requirements, you can choose the following options:

- **Replica storage class** – By default, Amazon S3 stores object replicas using the same storage class as the source object. You can specify a different storage class for the replicas.
- **Replica ownership** – Amazon S3 assumes that an object replica continues to be owned by the owner of the source object. So when it replicates objects, it also replicates the corresponding object access control list (ACL) or S3 Object Ownership setting. If the source and destination buckets are owned by different AWS accounts, you can configure replication to change the owner of a replica to the AWS account that owns the destination bucket.

You can configure replication by using the REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or the Amazon S3 console.

Amazon S3 also provides API operations to support setting up replication rules. For more information, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

**Topics**



## Replication configuration

Amazon S3 stores a replication configuration as XML. In the replication configuration XML file, you specify an AWS Identity and Access Management (IAM) role and one or more rules.

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3 can't replicate objects without your permission. You grant permissions with the IAM role that you specify in the replication configuration. Amazon S3 assumes the IAM role to replicate objects on your behalf. You must grant the required permissions to the IAM role first. For more information about managing permissions, see Setting up permissions for live replication.

You add one rule in a replication configuration in the following scenarios:

- You want to replicate all objects.
- You want to replicate one subset of objects. You identify the object subset by adding a filter in the rule. In the filter, you specify an object key prefix, tags, or a combination of both, to identify the subset of objects that the rule applies to. The filters target objects that match the exact values that you specify.

You add multiple rules in a replication configuration if you want to replicate different subsets of objects. In each rule, you specify a filter that selects a different subset of objects. For example, you might choose to replicate objects that have either `tax/` or `document/` key prefixes. To do this, you add two rules, one that specifies the `tax/` key prefix filter and another that specifies the `document/` key prefix. For more information about object key prefix, see Organizing objects using prefixes.

The following sections provide additional information.

**Topics**



### Basic rule configuration

Each rule must include the rule's status and priority. The rule must also indicate whether to replicate delete markers.

- `Status` indicates whether the rule is enabled or disabled by using the values `Enabled` or `Disabled`. If a rule is disabled, Amazon S3 doesn't perform the actions specified in the rule.
- `Priority` indicates which rule has precedence whenever two or more replication rules conflict. Amazon S3 attempts to replicate objects according to all replication rules. However, if there are two or more rules with the same destination bucket, then objects are replicated according to the rule with the highest priority. The higher the number, the higher the priority.
- `DeleteMarkerReplication` indicates whether to replicate delete markers by using the values `Enabled` or `Disabled`.

In the destination configuration, you must provide the name of the bucket or buckets where you want Amazon S3 to replicate objects.

The following example shows the minimum requirements for a V2 rule. For backward compatibility, Amazon S3 continues to support the XML V1 format. For more information, see Backward compatibility.

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

You can also specify other configuration options. For example, you might choose to use a storage class for object replicas that differs from the class for the source object.

**Optional: Specifying a filter**

To choose a subset of objects that the rule applies to, add an optional filter. You can filter by object key prefix, object tags, or a combination of both. If you filter on both a key prefix and object tags, Amazon S3 combines the filters by using a logical AND operator. In other words, the rule applies to a subset of objects with a specific key prefix and specific tags.

**Filter based on object key prefix**

To specify a rule with a filter based on an object key prefix, use the following code. You can specify only one prefix.

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

### Filter based on object tags

To specify a rule with a filter based on object tags, use the following code. You can specify one or more object tags.

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
</Rule>
...
```

### Filter with a key prefix and object tags

To specify a rule filter with a combination of a key prefix and object tags, use the following code. You wrap these filters in an <And> parent element. Amazon S3 performs a logical AND operation to combine these filters. In other words, the rule applies to a subset of objects with both a specific key prefix and specific tags.

```
<Rule>
    ...
```

```
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
    ...
</Rule>
...
```

**Note**

- If you specify a rule with an empty `<Filter>` element, your rule applies to all objects in your bucket.
- When you're using tag-based replication rules with live replication, new objects must be tagged with the matching replication rule tag in the `PutObject` operation. Otherwise, the objects won't be replicated. If objects are tagged after the `PutObject` operation, those objects also won't be replicated.

  To replicate objects that have been tagged after the `PutObject` operation, you must use S3 Batch Replication. For more information about Batch Replication, see Replicating existing objects.

### Additional destination configurations

In the destination configuration, you specify the bucket or buckets where you want Amazon S3 to replicate objects. You can set configurations to replicate objects from one source bucket to one or more destination buckets.

```
...
<Destination>
    <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
```

```
</Destination>
...
```

You can add the following options in the <Destination> element.

### Topics

- Specify storage class
- Add multiple destination buckets
- Specify different parameters for each replication rule with multiple destination buckets
- Change replica ownership
- Enable S3 Replication Time Control
- Replicate objects created with server-side encryption by using AWS KMS

### Specify storage class

You can specify the storage class for the object replicas. By default, Amazon S3 uses the storage class of the source object to create object replicas, as in the following example.

```
...
<Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
</Destination>
...
```

### Add multiple destination buckets

You can add multiple destination buckets in a single replication configuration, as follows.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
```

```
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

### Specify different parameters for each replication rule with multiple destination buckets

When adding multiple destination buckets in a single replication configuration, you can specify different parameters for each replication rule, as follows.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
```

```
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

### Change replica ownership

When the source and destination buckets aren't owned by the same accounts, you can change the ownership of the replica to the AWS account that owns the destination bucket. To do so, add the `AccessControlTranslation` element. This element takes the value `Destination`.

```
...
<Destination>
   <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

If you don't add the `AccessControlTranslation` element to the replication configuration, the replicas are owned by the same AWS account that owns the source object. For more information, see Changing the replica owner.

### Enable S3 Replication Time Control

You can enable S3 Replication Time Control (S3 RTC) in your replication configuration. S3 RTC replicates most objects in seconds and 99.99 percent of objects within 15 minutes (backed by a service-level agreement).

> **Note**
>
> Only a value of `<Minutes>15</Minutes>` is accepted for `EventThreshold` and `Time`.

```
...
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
  <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
</Destination>
...
```

For more information, see Meeting compliance requirements using S3 Replication Time Control (S3 RTC). For API examples, see PutBucketReplication in the *Amazon Simple Storage Service API Reference*.

### Replicate objects created with server-side encryption by using AWS KMS

Your source bucket might contain objects that were created with server-side encryption by using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, Amazon S3 doesn't replicate these objects. You can optionally direct Amazon S3 to replicate these objects. To do so, first explicitly opt into this feature by adding the `SourceSelectionCriteria` element. Then provide the AWS KMS key (for the AWS Region of the destination bucket) to use for encrypting object replicas. The following example shows how to specify these elements.

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

For more information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

### Example replication configurations

To get started, you can add the following example replication configurations to your bucket, as appropriate.

> **Important**
>
> To add a replication configuration to a bucket, you must have the `iam:PassRole` permission. This permission allows you to pass the IAM role that grants Amazon S3 replication permissions. You specify the IAM role by providing the Amazon Resource Name (ARN) that is used in the `Role` element in the replication configuration XML. For more information, see Granting a User Permissions to Pass a Role to an AWS service in the *IAM User Guide*.

### Example 1: Replication configuration with one rule

The following basic replication configuration specifies one rule. The rule specifies an IAM role that Amazon S3 can assume and a single destination bucket for object replicas. The `Status` value of `Enabled` indicates that the rule is in effect.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
```

```
  <Rule>
    <Status>Enabled</Status>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

To choose a subset of objects to replicate, you can add a filter. In the following configuration, the filter specifies an object key prefix. This rule applies to objects that have the prefix *Tax/* in their key names.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

If you specify the `Filter` element, you must also include the `Priority` and `DeleteMarkerReplication` elements. In this example, `Priority` is irrelevant because there is only one rule.

In the following configuration, the filter specifies one prefix and two tags. The rule applies to the subset of objects that have the specified key prefix and tags. Specifically, it applies to object that have the *Tax/* prefix in their key names and the two specified object tags. `Priority` doesn't apply because there is only one rule.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>

    </Filter>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

You can specify a storage class for the object replicas as follows.

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
```

```
        <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

You can specify any storage class that Amazon S3 supports.

**Example 2: Replication configuration with two rules**

**Example**

In the following replication configuration:

- Each rule filters on a different key prefix so that each rule applies to a distinct subset of objects. In this example, Amazon S3 replicates objects with the key names *Tax/doc1.pdf* and *Project/project1.txt*, but it doesn't replicate objects with the key name *PersonalDoc/documentA*.
- Rule priority is irrelevant because the rules apply to two distinct sets of objects. The next example shows what happens when rule priority is applied.
- The second rule specifies the S3 Standard-IA storage class for object replicas. Amazon S3 uses the specified storage class for those object replicas.

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
     ...
```

```
  </Rule>
 <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

### Example 3: Replication configuration with two rules with overlapping prefixes

In this configuration, the two rules specify filters with overlapping key prefixes, *star/* and *starship/*. Both rules apply to objects with the key name *starship-x*. In this case, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority.

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
```

```
    </Destination>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

**Example 4: Example walkthroughs**

For example walkthroughs, see Examples for configuring live replication.

For more information about the XML structure of replication configuration, see PutBucketReplication in the *Amazon Simple Storage Service API Reference*.

**Backward compatibility**

The latest version of the replication configuration XML is V2. XML V2 replication configurations are those that contain the `Filter` element for rules, and rules that specify S3 Replication Time Control (S3 RTC).

To see your replication configuration version, you can use the `GetBucketReplication` API operation. For more information, see GetBucketReplication in the *Amazon Simple Storage Service API Reference*.

For backward compatibility, Amazon S3 continues to support the XML V1 replication configuration. If you've used XML V1 replication configuration, consider the following issues that affect backward compatibility:

- Replication configuration XML V2 includes the `Filter` element for rules. With the `Filter` element, you can specify object filters based on the object key prefix, tags, or both to scope the objects that the rule applies to. Replication configuration XML V1 supports filtering based

only on the key prefix. In that case, you add the `Prefix` directly as a child element of the `Rule` element, as in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix>key-prefix</Prefix>
    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

For backward compatibility, Amazon S3 continues to support the V1 configuration.

- When you delete an object from your source bucket without specifying an object version ID, Amazon S3 adds a delete marker. If you use V1 of the replication configuration XML, Amazon S3 replicates delete markers that result from user actions. In other words, Amazon S3 replicates the delete marker only if a user deletes an object. If an expired object is removed by Amazon S3 (as part of a lifecycle action), Amazon S3 does not replicate the delete marker.

  In V2 replication configurations, you can enable delete marker replication for non-tag-based rules. For more information, see Replicating delete markers between buckets.

## Setting up permissions for live replication

When setting up live replication, you must acquire the necessary permissions as follows:

- Amazon S3 needs permissions to replicate objects on your behalf. You grant these permissions by creating an IAM role and then specifying that role in your replication configuration.
- When the source and destination buckets aren't owned by the same accounts, the owner of the destination bucket must grant the source bucket owner permissions to store the replicas.

### Topics

- Creating an IAM role
- Granting permissions when the source and destination buckets are owned by different AWS accounts

- Granting permissions for S3 Batch Operations
- Changing replica ownership
- Enable receiving replicated objects from a source bucket

### Creating an IAM role

By default, all Amazon S3 resources—buckets, objects, and related subresources—are private, and only the resource owner can access the resource. Amazon S3 needs permissions to read and replicate objects from the source bucket. You grant these permissions by creating an IAM role and specifying the role in your replication configuration.

This section explains the trust policy and minimum required permissions policy. The example walkthroughs provide step-by-step instructions to create an IAM role. For more information, see Examples for configuring live replication.

- The following example shows a *trust policy*, where you identify Amazon S3 as the service principal who can assume the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- The following example shows a *trust policy*, where you identify Amazon S3 and S3 Batch Operations as service principals. This is useful if you are creating a Batch Replication job. For more information, see Create a Batch Replication job for a first replication rule or new destination.

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Principal":{
            "Service": [
              "s3.amazonaws.com",
              "batchoperations.s3.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

For more information about IAM roles, see IAM Roles in the *IAM User Guide*.

- The following example shows an *access policy*, where you grant the role permissions to perform replication tasks on your behalf. When Amazon S3 assumes the role, it has the permissions that you specify in this policy. In this policy, *amzn-s3-demo-bucket1* is the source bucket, and *amzn-s3-demo-bucket2* is the destination bucket.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-bucket1"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
         ]
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      }
   ]
}
```

The access policy grants permissions for the following actions:

- `s3:GetReplicationConfiguration` and `s3:ListBucket` – Permissions for these actions on the *amzn-s3-demo-bucket1* bucket (the source bucket) allow Amazon S3 to retrieve the replication configuration and list the bucket content. (The current permissions model requires the `s3:ListBucket` permission for accessing delete markers.)
- `s3:GetObjectVersionForReplication` and `s3:GetObjectVersionAcl` – Permissions for these actions are granted on all objects to allow Amazon S3 to get a specific object version and access control list (ACL) associated with the objects.
- `s3:ReplicateObject` and `s3:ReplicateDelete` – Permissions for these actions on all objects in the *amzn-s3-demo-bucket2* bucket (the destination bucket) allow Amazon S3 to replicate objects or delete markers to the destination bucket. For information about delete markers, see How delete operations affect replication.

  > **Note**
  >
  > Permissions for the `s3:ReplicateObject` action on the *amzn-s3-demo-bucket2* bucket (the destination bucket) also allow replication of metadata such as object tags and ACLS. Therefore you do not need to explicitly grant permission for the `s3:ReplicateTags` action.

- `s3:GetObjectVersionTagging` – Permissions for this action on objects in the *amzn-s3-demo-bucket1* bucket (the source bucket) allow Amazon S3 to read object tags for replication. For more information, see Categorizing your storage using tags. If Amazon S3 doesn't have these permissions, it replicates the objects, but not the object tags.

For a list of Amazon S3 actions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

> **Important**
>
> The AWS account that owns the IAM role must have permissions for the actions that it grants to the IAM role.
>
> For example, suppose that the source bucket contains objects owned by another AWS account. The owner of the objects must explicitly grant the AWS account that owns the IAM role the required permissions through the object ACL. Otherwise, Amazon S3 can't access the objects, and replication of the objects fails. For information about ACL permissions, see Access control list (ACL) overview.
>
> The permissions described here are related to the minimum replication configuration. If you choose to add optional replication configurations, you must grant additional permissions to Amazon S3.

### Granting permissions when the source and destination buckets are owned by different AWS accounts

When the source and destination buckets aren't owned by the same accounts, the owner of the destination bucket must also add a bucket policy to grant the owner of the source bucket permissions to perform replication actions, as follows. In this policy, *amzn-s3-demo-bucket2* is the destination bucket.

> **Note**
>
> The ARN format of the role might appear different. If the role was created by using the console, the ARN format is `arn:aws:iam::`*`account-ID`*`:role/service-role/`*`role-name`*. If the role was created by using the AWS CLI, the ARN format is `arn:aws:iam::`*`account-ID`*`:role/`*`role-name`*. For more information, see IAM roles in the *IAM User Guide*.

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
```

```
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      },
      {
         "Sid":"Permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action": [
            "s3:List*",
            "s3:GetBucketVersioning",
            "s3:PutBucketVersioning"
         ],
         "Resource":"arn:aws:s3::amzn-s3-demo-bucket2"
      }
   ]
}
```

For an example, see Configuring replication when source and destination buckets are owned by different accounts.

If objects in the source bucket are tagged, note the following:

- If the source bucket owner grants Amazon S3 permission for the `s3:GetObjectVersionTagging` and `s3:ReplicateTags` actions to replicate object tags (through the IAM role), Amazon S3 replicates the tags along with the objects. For information about the IAM role, see Creating an IAM role.

- If the owner of the destination bucket doesn't want to replicate the tags, they can add the following statement to the destination bucket policy to explicitly deny permission for the `s3:ReplicateTags` action. In this policy, *amzn-s3-demo-bucket2* is the destination bucket.

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-id:role/service-role/source-account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      }
   ]
...
```

### Granting permissions for S3 Batch Operations

S3 Batch Replication provides you a way to replicate objects that existed before a replication configuration was in place, objects that have previously been replicated, and objects that have failed replication. You may create a one-time Batch Replication job when creating the first rule in a new replication configuration or when adding a new destination to an existing configuration through the AWS Management Console. You may also initiate Batch Replication for an existing replication configuration by creating a Batch Operations job.

For Batch Replication IAM role and policy examples see, Configuring IAM policies for Batch Replication.

### Changing replica ownership

When different AWS accounts own the source and destination buckets, you can tell Amazon S3 to change the ownership of the replica to the AWS account that owns the destination bucket. For more information about owner override, see Changing the replica owner.

### Enable receiving replicated objects from a source bucket

You can quickly generate the policies needed to enable receiving replicated objects from a source bucket through the AWS Management Console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the bucket that you want to use as a destination bucket.
4. Choose the **Management** tab, and scroll down to **Replication rules**.
5. For **Actions**, choose **Receive replicated objects**.

   Follow the prompts and enter the AWS account ID of the source bucket account and choose **Generate policies**. This will generate an Amazon S3 bucket policy and a KMS key policy.
6. To add this policy to your existing bucket policy, either choose **Apply settings** or choose **Copy** to manually copy the changes.
7. (Optional) Copy the AWS KMS policy to your desired KMS key policy on the AWS Key Management Service console.

## Examples for configuring live replication

The following examples show how to configure live replication for common use cases.

> **Note**
>
> Live replication refers to Same-Region Replication (SRR) and Cross-Region Replication (CRR). Live replication doesn't replicate any objects that existed in the bucket before you set up replication. To replicate objects that existed before you set up replication, use on-demand replication. To sync buckets and replicate existing objects on demand, see Replicating existing objects.

These examples demonstrate how to create a replication configuration by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDKs (AWS SDK for Java and AWS SDK for .NET examples are shown).

For information about installing and configuring the AWS CLI, see the following topics in the *AWS Command Line Interface User Guide*.

- Installing the AWS Command Line Interface
- Configuring the AWS CLI – You must set up at least one profile. If you are exploring cross-account scenarios, set up two profiles.

For information about the AWS SDKs, see AWS SDK for Java and AWS SDK for .NET.

> **Tip**
>
> For a step-by-step tutorial that demonstrates how to use live replication to replicate data, see Tutorial: Replicating data within and between AWS Regions using S3 Replication.

**Topics**



### Configuring replication for source and destination buckets owned by the same account

Replication is the automatic, asynchronous copying of objects across buckets in the same or different AWS Regions. Replication copies newly created objects and object updates from a source bucket to a destination bucket or buckets. For more information, see Replicating objects overview.

When you configure replication, you add replication rules to the source bucket. Replication rules define which source bucket objects to replicate and the destination bucket or buckets where the replicated objects are stored. You can create a rule to replicate all the objects in a bucket or a subset of objects with a specific key name prefix, one or more object tags, or both. A destination bucket can be in the same AWS account as the source bucket, or it can be in a different account.

If you specify an object version ID to delete, Amazon S3 deletes that object version in the source bucket. But it doesn't replicate the deletion in the destination bucket. In other words, it doesn't delete the same object version from the destination bucket. This protects data from malicious deletions.

When you add a replication rule to a bucket, the rule is enabled by default, so it starts working as soon as you save it.

In this example, you set up replication for source and destination buckets that are owned by the same AWS account. Examples are provided for using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), and the AWS SDK for Java and AWS SDK for .NET.

**Using the S3 console**

To configure a replication rule when the destination bucket is in the same AWS account as the source bucket, follow these steps.

If the destination bucket is in a different account from the source bucket, you must add a bucket policy to the destination bucket to grant the owner of the source bucket account permission to replicate objects in the destination bucket. For more information, see Granting permissions when the source and destination buckets are owned by different AWS accounts.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want.
4. Choose the **Management** tab, scroll down to **Replication rules**, and then choose **Create replication rule**.
5. In the **Replication rule configuration** section, under **Replication rule name**, enter a name for your rule to help identify the rule later. The name is required and must be unique within the bucket.
6. Under **Status**, **Enabled** is selected by default. An enabled rule starts to work as soon as you save it. If you want to enable the rule later, choose **Disabled**.
7. If the bucket has existing replication rules, you are instructed to set a priority for the rule. You must set a priority for the rule to avoid conflicts caused by objects that are included in the scope of more than one rule. In the case of overlapping rules, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority. For more information about rule priority, see Replication configuration.
8. Under **Source bucket**, you have the following options for setting the replication source:
   - To replicate the whole bucket, choose **Apply to all objects in the bucket**.
   - To replicate all objects that have the same prefix, choose **Limit the scope of this rule using one or more filters**. This limits replication to all objects that have names that begin with the prefix that you specify (for example `pictures`). Enter a prefix in the **Prefix** box.

> **Note**
>
> If you enter a prefix that is the name of a folder, you must use **/** (forward slash) as the last character (for example, `pictures/`).

- To replicate all objects with one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. Repeat the procedure to add another tag. You can combine a prefix and tags. For more information about object tags, see Categorizing your storage using tags.

The new replication configuration XML schema supports prefix and tag filtering and the prioritization of rules. For more information about the new schema, see Backward compatibility. For more information about the XML used with the Amazon S3 API that works behind the user interface, see Replication configuration. The new schema is described as *replication configuration XML V2*.

9. Under **Destination**, choose the bucket where you want Amazon S3 to replicate objects.

> **Note**
>
> The number of destination buckets is limited to the number of AWS Regions in a given partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions). To request an increase in your destination bucket quota, you can use service quotas.

- To replicate to a bucket or buckets in your account, choose **Choose a bucket in this account**, and enter or browse for the destination bucket names.
- To replicate to a bucket or buckets in a different AWS account, choose **Specify a bucket in another account**, and enter the destination bucket account ID and bucket name.

  If the destination is in a different account from the source bucket, you must add a bucket policy to the destination buckets to grant the owner of the source bucket account permission to replicate objects. For more information, see Granting permissions when the source and destination buckets are owned by different AWS accounts.

Optionally, if you want to help standardize ownership of new objects in the destination bucket, choose **Change object ownership to the destination bucket owner**. For more information about this option, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Note**
>
> If versioning is not enabled on the destination bucket, you get a warning that contains an **Enable versioning** button. Choose this button to enable versioning on the bucket.

10. Set up an AWS Identity and Access Management (IAM) role that Amazon S3 can assume to replicate objects on your behalf.

    To set up an IAM role, in the **IAM role** section, select one of the following from the **IAM role** dropdown list:

    - We highly recommend that you choose **Create new role** to have Amazon S3 create a new IAM role for you. When you save the rule, a new policy is generated for the IAM role that matches the source and destination buckets that you choose.
    - You can choose to use an existing IAM role. If you do, you must choose a role that grants Amazon S3 the necessary permissions for replication. Replication fails if this role does not grant Amazon S3 sufficient permissions to follow your replication rule.

    > **Important**
    >
    > When you add a replication rule to a bucket, you must have the `iam:PassRole` permission to be able to pass the IAM role that grants Amazon S3 replication permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

11. To replicate objects in the source bucket that are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), under **Encryption**, select **Replicate objects encrypted with AWS KMS**. Under **AWS KMS keys for encrypting destination objects** are the source keys that you allow replication to use. All source KMS keys are included by default. To narrow the KMS key selection, you can choose an alias or key ID.

Objects encrypted by AWS KMS keys that you do not select are not replicated. A KMS key or a group of KMS keys is chosen for you, but you can choose the KMS keys if you want. For information about using AWS KMS with replication, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

> **Important**
>
> When you replicate objects that are encrypted with AWS KMS, the AWS KMS request rate doubles in the source Region and increases in the destination Region by the same amount. These increased call rates to AWS KMS are due to the way that data is re-encrypted by using the KMS key that you define for the replication destination Region. AWS KMS has a request rate quota that is per calling account per Region. For information about the quota defaults, see AWS KMS Quotas - Requests per Second: Varies in the *AWS Key Management Service Developer Guide*.
>
> If your current Amazon S3 PUT object request rate during replication is more than half the default AWS KMS rate limit for your account, we recommend that you request an increase to your AWS KMS request rate quota. To request an increase, create a case in the AWS Support Center at Contact Us. For example, suppose that your current PUT object request rate is 1,000 requests per second and you use AWS KMS to encrypt your objects. In this case, we recommend that you ask AWS Support to increase your AWS KMS rate limit to 2,500 requests per second, in both your source and destination Regions (if different), to ensure that there is no throttling by AWS KMS.
>
> To see your PUT object request rate in the source bucket, view `PutRequests` in the Amazon CloudWatch request metrics for Amazon S3. For information about viewing CloudWatch metrics, see Using the S3 console.

If you chose to replicate objects encrypted with AWS KMS, do the following:

- Under **AWS KMS key for encrypting destination objects** , specify your KMS key in one of the following ways:
  - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

- To enter the KMS key Amazon Resource Name (ARN), choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears. This encrypts the replicas in the destination bucket. You can find the ARN for your KMS key in the IAM Console, under **Encryption keys**.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can only use KMS keys that are enabled in the same AWS Region as the bucket. When you choose **Choose from your KMS keys**, the S3 console lists only 100 KMS keys per Region. If you have more than 100 KMS keys in the same Region, you can see only the first 100 KMS keys in the S3 console. To use a KMS key that is not listed in the console, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN.
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS).

12. Under **Destination storage class**, if you want to replicate your data into a specific storage class in the destination, choose **Change the storage class for the replicated objects**. Then choose the storage class that you want to use for the replicated objects in the destination. If you don't

choose this option, the storage class for replicated objects is the same class as the original objects.

13. You have the following additional options while setting the **Additional replication options**:

    - If you want to enable S3 Replication Time Control (S3 RTC) in your replication configuration, select **Replication Time Control (RTC)**. For more information about this option, see Meeting compliance requirements using S3 Replication Time Control (S3 RTC).
    - If you want to enable S3 Replication metrics in your replication configuration, select **Replication metrics and events**. For more information see, Monitoring progress with replication metrics and S3 Event Notifications.
    - If you want to enable delete marker replication in your replication configuration, select **Delete marker replication**. For more information see, Replicating delete markers between buckets.
    - If you want to enable Amazon S3 replica modification sync in your replication configuration, select **Replica modification sync**. For more information see, Replicating metadata changes with Amazon S3 replica modification sync.

    > **Note**
    >
    > When you use S3 RTC or S3 Replication metrics, additional fees apply.

14. To finish, choose **Save**.
15. After you save your rule, you can edit, enable, disable, or delete your rule by selecting your rule and choosing **Edit rule**.

### Using the AWS CLI

To use the AWS CLI to set up replication when the source and destination buckets are owned by the same AWS account, you do the following:

- Create source and destination buckets
- Enable versioning on the buckets
- Create an IAM role that gives Amazon S3 permission to replicate objects
- Add the replication configuration to the source bucket

To verify your setup, you test it.

**To set up replication when source and destination buckets are owned by the same AWS account**

1. Set a credentials profile for the AWS CLI. In this example, we use the profile name `acctA`. For information about setting credential profiles, see Named Profiles in the *AWS Command Line Interface User Guide*.

   > **Important**
   >
   > The profile you use for this exercise must have the necessary permissions. For example, in the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if the profile you use has the `iam:PassRole` permission. For more information, see Granting a User Permissions to Pass a Role to an AWS Service in the *IAM User Guide*. If you use administrator credentials to create a named profile, you can perform all the tasks.

2. Create a *`source`* bucket and enable versioning on it. The following code creates a *`source`* bucket in the US East (N. Virginia) (us-east-1) Region.

   ```
   aws s3api create-bucket \
   --bucket source \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket source \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. Create a *`destination`* bucket and enable versioning on it. The following code creates a *`destination`* bucket in the US West (Oregon) (us-west-2) Region.

   > **Note**
   >
   > To set up replication configuration when both source and destination buckets are in the same AWS account, you use the same profile. This example uses `acctA`. To test replication configuration when the buckets are owned by different AWS accounts,

you specify different profiles for each. This example uses the `acctB` profile for the destination bucket.

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. Create an IAM role. You specify this role in the replication configuration that you add to the *source* bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create a role.
   - Attach a permissions policy to the role.

   a. Create the IAM role.

      i. Copy the following trust policy and save it to a file named `s3-role-trust-policy.json` in the current directory on your local computer. This policy grants Amazon S3 service principal permissions to assume the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
```

```
}
```

ii. Run the following command to create a role.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. Attach a permissions policy to the role.

i. Copy the following permissions policy and save it to a file named `s3-role-permissions-policy.json` in the current directory on your local computer. This policy grants permissions for various Amazon S3 bucket and object actions.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
```

```
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::destination-bucket/*"
      }
   ]
}
```

ii. Run the following command to create a policy and attach it to the role.

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. Add replication configuration to the *source* bucket.

   a. Although the Amazon S3 API requires replication configuration as XML, the AWS CLI requires that you specify the replication configuration as JSON. Save the following JSON in a file called `replication.json` to the local directory on your computer.

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket"
      }
    }
  ]
}
```

   b. Update the JSON by providing values for the *destination-bucket* and *IAM-role-ARN*. Save the changes.

   c. Run the following command to add the replication configuration to your source bucket. Be sure to provide the *source* bucket name.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

To retrieve the replication configuration, use the `get-bucket-replication` command.

```
$ aws s3api get-bucket-replication \
--bucket source \
--profile acctA
```

6. Test the setup in the Amazon S3 console:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

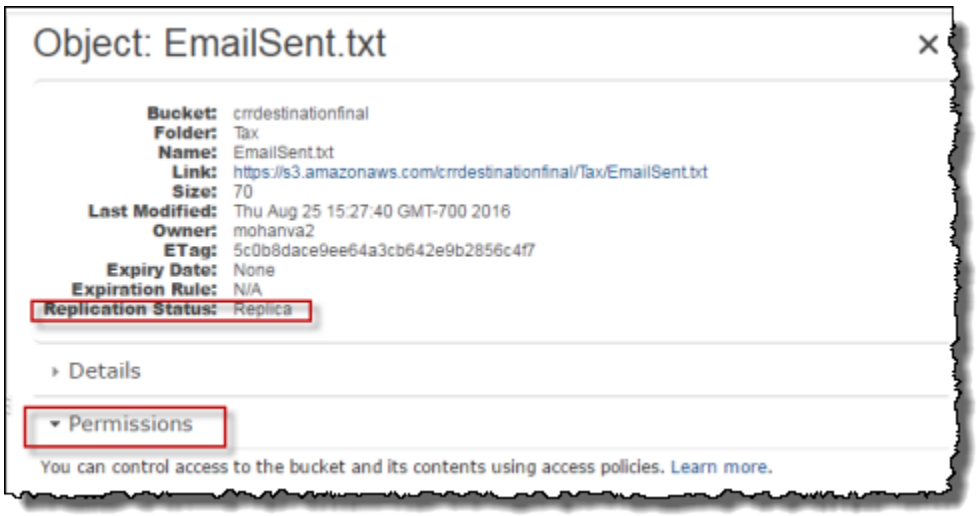
   b. In the *source* bucket, create a folder named `Tax`.

   c. Add sample objects to the `Tax` folder in the *source* bucket.

   > **Note**
   >
   > The amount of time it takes for Amazon S3 to replicate an object depends on the size of the object. For information about how to see the status of replication, see Getting replication status information.

   In the *destination* bucket, verify the following:

   - That Amazon S3 replicated the objects.
   - In object **properties**, that the **Replication Status** is set to `Replica` (identifying this as a replica object).
   - In object **properties**, that the permission section shows no permissions. This means that the replica is still owned by the *source* bucket owner, and the *destination* bucket owner has no permission on the object replica. You can add optional configuration to tell Amazon S3 to change the replica ownership. For an example, see How to change the replica owner.




### Using the AWS SDKs

Use the following code examples to add a replication configuration to a bucket with the AWS SDK for Java and AWS SDK for .NET, respectively.

Java

The following example adds a replication configuration to a bucket and then retrieves and verifies the configuration. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import
 com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateRoleRequest;
import com.amazonaws.services.identitymanagement.model.PutRolePolicyRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.CreateBucketRequest;
```

```
import com.amazonaws.services.s3.model.DeleteMarkerReplication;
import com.amazonaws.services.s3.model.DeleteMarkerReplicationStatus;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.replication.ReplicationFilter;
import com.amazonaws.services.s3.model.replication.ReplicationFilterPredicate;
import com.amazonaws.services.s3.model.replication.ReplicationPrefixPredicate;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CrossRegionReplication {

        public static void main(String[] args) throws IOException {
                Regions clientRegion = Regions.DEFAULT_REGION;
                String accountId = "*** Account ID ***";
                String roleName = "*** Role name ***";
                String sourceBucketName = "*** Source bucket name ***";
                String destBucketName = "*** Destination bucket name ***";
                String prefix = "Tax/";

                String roleARN = String.format("arn:aws:iam::%s:%s", accountId,
 roleName);
                String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

                AmazonS3 s3Client = AmazonS3Client.builder()
                                .withCredentials(new ProfileCredentialsProvider())
                                .withRegion(clientRegion)
                                .build();

                createBucket(s3Client, clientRegion, sourceBucketName);
                createBucket(s3Client, clientRegion, destBucketName);
                assignRole(roleName, clientRegion, sourceBucketName,
 destBucketName);

                try {

                        // Create the replication rule.
```

```
                        List<ReplicationFilterPredicate> andOperands = new
ArrayList<ReplicationFilterPredicate>();
                        andOperands.add(new ReplicationPrefixPredicate(prefix));

                        Map<String, ReplicationRule> replicationRules = new
HashMap<String, ReplicationRule>();
                        replicationRules.put("ReplicationRule1",
                                        new ReplicationRule()
                                                        .withPriority(0)

.withStatus(ReplicationRuleStatus.Enabled)

.withDeleteMarkerReplication(
                                                                        new
DeleteMarkerReplication().withStatus(

   DeleteMarkerReplicationStatus.DISABLED))
                                                        .withFilter(new
ReplicationFilter().withPredicate(
                                                                        new
ReplicationPrefixPredicate(prefix)))
                                                        .withDestinationConfig(new
ReplicationDestinationConfig()

.withBucketARN(destinationBucketARN)

.withStorageClass(StorageClass.Standard)));

                        // Save the replication rule to the source bucket.
                        s3Client.setBucketReplicationConfiguration(sourceBucketName,
                                        new BucketReplicationConfiguration()
                                                        .withRoleARN(roleARN)

.withRules(replicationRules));

                        // Retrieve the replication configuration and verify that
the configuration
                        // matches the rule we just set.
                        BucketReplicationConfiguration replicationConfig = s3Client

.getBucketReplicationConfiguration(sourceBucketName);
                        ReplicationRule rule =
replicationConfig.getRule("ReplicationRule1");
                        System.out.println("Retrieved destination bucket ARN: "
```

```
                                        +
rule.getDestinationConfig().getBucketARN());
                        System.out.println("Retrieved priority: " +
rule.getPriority());
                        System.out.println("Retrieved source-bucket replication rule
status: " + rule.getStatus());
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }

        private static void createBucket(AmazonS3 s3Client, Regions region, String
bucketName) {
                CreateBucketRequest request = new CreateBucketRequest(bucketName,
region.getName());
                s3Client.createBucket(request);
                BucketVersioningConfiguration configuration = new
BucketVersioningConfiguration()
                                .withStatus(BucketVersioningConfiguration.ENABLED);

                SetBucketVersioningConfigurationRequest enableVersioningRequest =
new SetBucketVersioningConfigurationRequest(
                                bucketName, configuration);
                s3Client.setBucketVersioningConfiguration(enableVersioningRequest);

        }

        private static void assignRole(String roleName, Regions region, String
sourceBucket, String destinationBucket) {
                AmazonIdentityManagement iamClient =
AmazonIdentityManagementClientBuilder.standard()
                                .withRegion(region)
                                .withCredentials(new ProfileCredentialsProvider())
                                .build();
                StringBuilder trustPolicy = new StringBuilder();
                trustPolicy.append("{\\r\\n   ");
```

```
                trustPolicy.append("\\\"Version\\\":\\\"2012-10-17\\\",\\r\\n   ");
                trustPolicy.append("\\\"Statement\\\":[\\r\\n      {\\r\\n
 ");
                trustPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Principal\\\":{\\r\\n            ");
                trustPolicy.append("\\\"Service\\\":\\\"s3.amazonaws.com\\\"\\r\\n
      },\\r\\n         ");
                trustPolicy.append("\\\"Action\\\":\\\"sts:AssumeRole\\\"\\r\\n
  }\\r\\n   ]\\r\\n}");

                CreateRoleRequest createRoleRequest = new CreateRoleRequest()
                                .withRoleName(roleName)

 .withAssumeRolePolicyDocument(trustPolicy.toString());

                iamClient.createRole(createRoleRequest);

                StringBuilder permissionPolicy = new StringBuilder();
                permissionPolicy.append(
                                "{\\r\\n   \\\"Version\\\":\\\"2012-10-17\\\",\\r\\n
   \\\"Statement\\\":[\\r\\n      {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n             ");
                permissionPolicy.append("\\\"s3:GetObjectVersionForReplication\\\",\
\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:GetObjectVersionAcl\\\"\\r\\n         ],\\r\
\n         \\\"Resource\\\":[\\r\\n            ");
                permissionPolicy.append("\\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("/*\\\"\\r\\n         ]\\r\\n      },\\r\\n
     {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:ListBucket\\\",\\r\\n            \\
\"s3:GetReplicationConfiguration\\\"\\r\\n         ");
                permissionPolicy.append("],\\r\\n         \\\"Resource\\\":[\\r\\n
          \\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("\\r\\n         ");
                permissionPolicy
```

```
                                .append("]\\r\\n      },\\r\\n      {\\r\\n
 \\\"Effect\\\":\\\"Allow\\\",\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Action\\\":[\\r\\n            \\
\"s3:ReplicateObject\\\",\\r\\n            ");
                permissionPolicy
                                .append("\\\"s3:ReplicateDelete\\\",\\r\\n
  \\\"s3:ReplicateTags\\\",\\r\\n            ");
                permissionPolicy.append("\\\"s3:GetObjectVersionTagging\\\"\\r\\n\\r
\\n         ],\\r\\n         ");
                permissionPolicy.append("\\\"Resource\\\":\\\"arn:aws:s3:::");
                permissionPolicy.append(destinationBucket);
                permissionPolicy.append("/*\\\"\\r\\n      }\\r\\n   ]\\r\\n}");

                PutRolePolicyRequest putRolePolicyRequest = new
 PutRolePolicyRequest()
                                .withRoleName(roleName)
                                .withPolicyDocument(permissionPolicy.toString())
                                .withPolicyName("crrRolePolicy");

                iamClient.putRolePolicy(putRolePolicyRequest);

        }
}
```

C#

The following AWS SDK for .NET code example adds a replication configuration to a bucket and then retrieves it. To use this code, provide the names for your buckets and the Amazon Resource Name (ARN) for your IAM role. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
```

```
   {
       private const string sourceBucket = "*** source bucket ***";
       // Bucket ARN example - arn:aws:s3:::destinationbucket
       private const string destinationBucketArn = "*** destination bucket ARN
***";
       private const string roleArn = "*** IAM Role ARN ***";
       // Specify your bucket region (an example region is shown).
       private static readonly RegionEndpoint sourceBucketRegion =
RegionEndpoint.USWest2;
       private static IAmazonS3 s3Client;
       public static void Main()
       {
           s3Client = new AmazonS3Client(sourceBucketRegion);
           EnableReplicationAsync().Wait();
       }
       static async Task EnableReplicationAsync()
       {
           try
           {
               ReplicationConfiguration replConfig = new ReplicationConfiguration
               {
                   Role = roleArn,
                   Rules =
                       {
                           new ReplicationRule
                           {
                               Prefix = "Tax",
                               Status = ReplicationRuleStatus.Enabled,
                               Destination = new ReplicationDestination
                               {
                                   BucketArn = destinationBucketArn
                               }
                           }
                       }
               };

               PutBucketReplicationRequest putRequest = new
PutBucketReplicationRequest
               {
                   BucketName = sourceBucket,
                   Configuration = replConfig
               };
```

```
                PutBucketReplicationResponse putResponse = await
 s3Client.PutBucketReplicationAsync(putRequest);

                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
 client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

### Configuring replication when source and destination buckets are owned by different accounts

Setting up replication when *source* and *destination* buckets are owned by different AWS accounts is similar to setting replication when both buckets are owned by the same account. The only difference is that the *destination* bucket owner must grant the *source* bucket owner permission to replicate objects by adding a bucket policy.

For more information about configuring replication using server-side encryption with AWS Key Management Service in cross-account scenarios, see Granting additional permissions for cross-account scenarios.

**To configure replication when the source and destination buckets are owned by different AWS accounts**

1. In this example, you create *source* and *destination* buckets in two different AWS accounts. You need to have two credential profiles set for the AWS CLI (in this example, we use `acctA` and `acctB` for profile names). For more information about setting credential profiles, see Named Profiles in the *AWS Command Line Interface User Guide*.
2. Follow the step-by-step instructions in Configuring for buckets in the same account with the following changes:
   - For all AWS CLI commands related to *source* bucket activities (for creating the *source* bucket, enabling versioning, and creating the IAM role), use the `acctA` profile. Use the `acctB` profile to create the *destination* bucket.
   - Make sure that the permissions policy specifies the *source* and *destination* buckets that you created for this example.
3. In the console, add the following bucket policy on the *destination* bucket to allow the owner of the *source* bucket to replicate objects. Be sure to edit the policy by providing the AWS account ID of the *source* bucket owner and the *destination* bucket name.

> **Note**
>
> To use the following example, replace the *user input placeholders* with your own information. Replace *DOC-EXAMPLE-BUCKET* with your destination bucket name. Replace *source-bucket-acct-ID:role/service-role/source-acct-IAM-role* with the role you are using for this replication configuration.

If you created the IAM service role manually, set the role path as `role/service-role/`, as shown in the below policy example. For more information, see IAM ARNs in the *IAM User Guide*.

```
{
   "Version":"2012-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"Set-permissions-for-objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-acct-IAM-role"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      },
      {
         "Sid":"Set permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-acct-IAM-role"
         },
         "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      }
   ]
}
```

Choose the bucket and add the bucket policy. For instructions, see Adding a bucket policy by using the Amazon S3 console.

In replication, the owner of the source object owns the replica by default. When source and destination buckets are owned by different AWS accounts, you can add optional configuration settings to change replica ownership to the AWS account that owns the destination buckets. This includes granting the `ObjectOwnerOverrideToBucketOwner` permission. For more information, see Changing the replica owner.

### Changing the replica owner

In replication, the owner of the source object also owns the replica by default. When source and destination buckets are owned by different AWS accounts and you want to change replica ownership to the AWS account that owns the destination buckets, you can add optional configuration settings to change replica ownership to the AWS account that owns the destination buckets. You might do this, for example, to restrict access to object replicas. This is referred to as the *owner override* option of the replication configuration. For more information about the owner override option, see Adding the owner override option to the replication configuration. For information about setting the replication configuration, see Replicating objects overview.

To configure the owner override, you do the following:

- Add the owner override option to the replication configuration to tell Amazon S3 to change replica ownership.
- Grant Amazon S3 permissions to change replica ownership.
- Add permission in the destination buckets policy to allow changing replica ownership. This allows the owner of the destination buckets to accept the ownership of object replicas.

For more information, see Adding the owner override option to the replication configuration. For a working example with step-by-step instructions, see How to change the replica owner.

### Bucket owner enforced setting for Object Ownership

When you use Amazon S3 replication and the source and destination buckets are owned by different AWS accounts, the bucket owner of the destination bucket can disable ACLs (with the bucket owner enforced setting for Object Ownership) to change replica ownership to the AWS account that owns the destination bucket. This setting mimics the existing owner override behavior without the need of `s3:ObjectOwnerOverrideToBucketOwner` permission. This means that all objects that are replicated to the destination bucket with the bucket owner enforced setting are owned by the destination bucket owner. For more information about Object Ownership, see Controlling ownership of objects and disabling ACLs for your bucket.

### Adding the owner override option to the replication configuration

> **Warning**
>
> Add the owner override option only when the source and destination buckets are owned by different AWS accounts. Amazon S3 doesn't check if the buckets are owned by same or

different accounts. If you add the owner override when both buckets are owned by same AWS account, Amazon S3 applies the owner override. It grants full permissions to the owner of the destination bucket and doesn't replicate subsequent updates to the source object access control list (ACL). The replica owner can directly change the ACL associated with a replica with a PUT ACL request, but not through replication.

To specify the owner override option, add the following to each `Destination` element:

- The `AccessControlTranslation` element, which tells Amazon S3 to change replica ownership
- The `Account` element, which specifies the AWS account of the destination bucket owner

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

The following example replication configuration tells Amazon S3 to replicate objects that have the `Tax` key prefix to the destination bucket and change ownership of the replicas.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
```

```
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

### Granting Amazon S3 permission to change replica ownership

Grant Amazon S3 permissions to change replica ownership by adding permission for the `s3:ObjectOwnerOverrideToBucketOwner` action in the permissions policy associated with the IAM role. This is the IAM role that you specified in the replication configuration that allows Amazon S3 to assume and replicate objects on your behalf.

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

### Adding permission in the destination bucket policy to allow changing replica ownership

The owner of the destination bucket must grant the owner of the source bucket permission to change replica ownership. The owner of the destination bucket grants the owner of the source bucket permission for the `s3:ObjectOwnerOverrideToBucketOwner` action. This allows the destination bucket owner to accept ownership of the object replicas. The following example bucket policy statement shows how to do this.

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
```

```
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

### Additional considerations

When you configure the ownership override option, the following considerations apply:

- By default, the owner of the source object also owns the replica. Amazon S3 replicates the object version and the ACL associated with it.

  If you add the owner override, Amazon S3 replicates only the object version, not the ACL. In addition, Amazon S3 doesn't replicate subsequent changes to the source object ACL. Amazon S3 sets the ACL on the replica that grants full control to the destination bucket owner.
- When you update a replication configuration to enable, or disable, the owner override, the following occurs.

  - If you add the owner override option to the replication configuration:

    When Amazon S3 replicates an object version, it discards the ACL that is associated with the source object. Instead, it sets the ACL on the replica, giving full control to the owner of the destination bucket. It doesn't replicate subsequent changes to the source object ACL. However, this ACL change doesn't apply to object versions that were replicated before you set the owner override option. ACL updates on source objects that were replicated before the owner override was set continue to be replicated (because the object and its replicas continue to have the same owner).
  - If you remove the owner override option from the replication configuration:

    Amazon S3 replicates new objects that appear in the source bucket and the associated ACLs to the destination buckets. For objects that were replicated before you removed the owner override, Amazon S3 doesn't replicate the ACLs because the object ownership change that Amazon S3 made remains in effect. That is, ACLs put on the object version that were replicated when the owner override was set continue to be not replicated.

### How to change the replica owner

When the *source* and *destination* buckets in a replication configuration are owned by different AWS accounts, you can tell Amazon S3 to change replica ownership to the AWS account that owns

the *destination* bucket. This example explains how to use the Amazon S3 console and the AWS CLI to change replica ownership. For more information, see Changing the replica owner.

> **Note**
>
> When you use S3 replication and the source and destination buckets are owned by different AWS accounts, the bucket owner of the destination bucket can disable ACLs (with the bucket owner enforced setting for Object Ownership) to change replica ownership to the AWS account that owns the destination bucket. This setting mimics the existing owner override behavior without the need of `s3:ObjectOwnerOverrideToBucketOwner` permission. This means that all objects that are replicated to the destination bucket with the bucket owner enforced setting are owned by the destination bucket owner. For more information about Object Ownership, see Controlling ownership of objects and disabling ACLs for your bucket.

For more information about configuring replication using sever-side encryption with AWS Key Management Service in cross-account scenarios, see Granting additional permissions for cross-account scenarios.

**Using the S3 console**

For step-by-step instructions, see Configuring replication for source and destination buckets owned by the same account. This topic provides instructions for setting replication configuration when buckets are owned by same and different AWS accounts.

**Using the AWS CLI**

To change replica ownership using the AWS CLI, you create buckets, enable versioning on the buckets, create an IAM role that gives Amazon S3 permission to replicate objects, and add the replication configuration to the source bucket. In the replication configuration you direct Amazon S3 to change replica owner. You also test the setup.

**To change replica ownership when source and destination buckets are owned by different AWS accounts (AWS CLI)**

1. In this example, you create the *source* and *destination* buckets in two different AWS accounts. Configure the AWS CLI with two named profiles. This example uses profiles named `acctA` and `acctB`, respectively. For more information about setting credential profiles, see Named Profiles in the *AWS Command Line Interface User Guide.*

> **Important**
>
> The profiles you use for this exercise must have the necessary permissions. For example, in the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if the profile you use has the `iam:PassRole` permission. If you use administrator user credentials to create a named profile then you can perform all the tasks. For more information, see Granting a User Permissions to Pass a Role to an AWS Service in the *IAM User Guide*.

You will need to make sure these profiles have necessary permissions. For example, the replication configuration includes an IAM role that Amazon S3 can assume. The named profile you use to attach such configuration to a bucket can do so only if it has the `iam:PassRole` permission. If you specify administrator user credentials when creating these named profiles, they have all the permissions. For more information, see Granting a User Permissions to Pass a Role to an AWS Service in the *IAM User Guide*.

2. Create the *source* bucket and enable versioning. This example creates the *source* bucket in the US East (N. Virginia) (us-east-1) Region.

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. Create a *destination* bucket and enable versioning. This example creates the *destination* bucket in the US West (Oregon) (us-west-2) Region. Use an AWS account profile different from the one you used for the *source* bucket.

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
```

```
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctB
```

4. You must add permissions to your *destination* bucket policy to allow changing the replica ownership.

   a. Save the following policy to *destination-bucket-policy*.json.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:ReplicateTags",
        "s3:GetObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::destination/*"
      ]
    }
  ]
}
```

   b. Put the above policy to *destination* bucket:

```
aws s3api put-bucket-policy --region $ {destination_region} --
bucket $ {destination} --policy file://destination_bucket_policy.json
```

5. Create an IAM role. You specify this role in the replication configuration that you add to the *source* bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create a role.
   - Attach a permissions policy to the role.

   a. Create an IAM role.

      i. Copy the following trust policy and save it to a file named `s3-role-trust-policy.json` in the current directory on your local computer. This policy grants Amazon S3 permissions to assume the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. Run the following AWS CLI command to create a role.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

   b. Attach a permissions policy to the role.

      i. Copy the following permissions policy and save it to a file named `s3-role-perm-pol-changeowner.json` in the current directory on your local computer. This policy grants permissions for various Amazon S3 bucket and object actions. In the following steps, you create an IAM role and attach this policy to the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::destination/*"
      }
   ]
}
```

ii. To create a policy and attach it to the role, run the following command.

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
```

```
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. Add a replication configuration to your source bucket.

   a. The AWS CLI requires specifying the replication configuration as JSON. Save the following JSON in a file named `replication.json` in the current directory on your local computer. In the configuration, the addition of `AccessControlTranslation` to indicate change in replica ownership.

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::destination",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

   b. Edit the JSON by providing values for the *`destination`* bucket owner account ID and *`IAM-role-ARN`*. Save the changes.

   c. To add the replication configuration to the source bucket, run the following command. Provide the *`source`* bucket name.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
```

```
--profile acctA
```

7. Check replica ownership in the Amazon S3 console.

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. Add objects to the *source* bucket. Verify that the *destination* bucket contains the object replicas and that the ownership of the replicas has changed to the AWS account that owns the *destination* bucket.

### Using the AWS SDKs

For a code example to add replication configuration, see Using the AWS SDKs. You need to modify the replication configuration appropriately. For conceptual information, see Changing the replica owner.

### Meeting compliance requirements using S3 Replication Time Control (S3 RTC)

S3 Replication Time Control (S3 RTC) helps you meet compliance or business requirements for data replication and provides visibility into Amazon S3 replication times. S3 RTC replicates most objects that you upload to Amazon S3 in seconds, and 99.99 percent of those objects within 15 minutes.

S3 RTC by default includes S3 Replication metrics and Amazon S3 Event Notifications, which you can use to monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, and the maximum replication time. You can enable replication metrics independently of S3 RTC. For more information, see Monitoring progress with replication metrics. Additionally, S3 RTC provides `OperationMissedThreshold` and `OperationReplicatedAfterThreshold` events that notify the bucket owner if object replication exceeds or replicates after the 15-minute threshold.

With S3 RTC, Amazon S3 events can notify you in the rare instance when objects do not replicate within 15 minutes and when those objects replicate after the 15 minute threshold. Amazon S3 events are available through Amazon SQS, Amazon SNS, or AWS Lambda. For more information, see the section called “Amazon S3 Event Notifications”.

### Topics

- S3 Replication Time Control
- Replication metrics with S3 RTC
- Using Amazon S3 event notifications to track replication objects

- Best practices and guidelines for S3 RTC
- Enabling S3 Replication Time Control (S3 RTC)

**S3 Replication Time Control**

You can start using S3 Replication Time Control (S3 RTC) with a new or existing replication rule. You can choose to apply your replication rule to an entire S3 bucket, or to Amazon S3 objects with a specific prefix or tag. When you enable S3 RTC, replication metrics are also enabled on your replication rule.

If you are using the latest version of the replication configuration (that is, you specify the `Filter` element in a replication configuration rule), Amazon S3 does not replicate the delete marker by default. However you can add delete marker replication to non-tag-based rules.

> **Note**
>
> Replication metrics are billed at the same rate as Amazon CloudWatch custom metrics. For information, see Amazon CloudWatch pricing.

For more information about creating a rule with S3 RTC, see Enabling S3 Replication Time Control (S3 RTC).

**Replication metrics with S3 RTC**

Replication rules with S3 Replication Time Control (S3 RTC) enabled publishes replication metrics. With replication metrics, you can monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, the maximum replication time to the destination Region, and the total number of operations that failed replication. You can then monitor each dataset that you replicate separately.

Replication metrics are available within 15 minutes of enabling S3 RTC. Replication metrics are available through the Amazon S3 console, the Amazon S3 API, the AWS SDKs, the AWS Command Line Interface (AWS CLI), and Amazon CloudWatch. For more information, see Monitoring metrics with Amazon CloudWatch.

For more information about finding replication metrics via the Amazon S3 console, see Viewing replication metrics by using the Amazon S3 console.

### Using Amazon S3 event notifications to track replication objects

You can track replication time for objects that did not replicate within 15 minutes by monitoring specific event notifications that S3 Replication Time Control (S3 RTC) publishes. These events are published when an object that was eligible for replication using S3 RTC didn't replicate within 15 minutes, and when that object replicates after the 15 minute threshold.

Replication events are available within 15 minutes of enabling S3 RTC. Amazon S3 events are available through Amazon SQS, Amazon SNS, or AWS Lambda. For more information, see Amazon S3 Event Notifications.

### Best practices and guidelines for S3 RTC

When replicating data in Amazon S3 using S3 Replication Time Control (S3 RTC), follow these best practice guidelines to optimize replication performance for your workloads.

#### Topics

- Amazon S3 Replication and request rate performance guidelines
- Estimating your replication request rates
- Exceeding S3 RTC data transfer rate limits
- AWS KMS encrypted object replication request rates

### Amazon S3 Replication and request rate performance guidelines

When uploading and retrieving storage from Amazon S3, your applications can achieve thousands of transactions per second in request performance. For example, an application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in an S3 bucket, including the requests that S3 replication makes on your behalf. There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by parallelizing reads. For example, if you create 10 prefixes in an S3 bucket to parallelize reads, you could scale your read performance to 55,000 read requests per second.

Amazon S3 automatically scales in response to sustained request rates above these guidelines, or sustained request rates concurrent with LIST requests. While Amazon S3 is internally optimizing for the new request rate, you might receive HTTP 503 request responses temporarily until the optimization is complete. This might occur with increases in request per second rates, or when you first enable S3 RTC. During these periods, your replication latency might increase. The S3

RTC service level agreement (SLA) doesn't apply to time periods when Amazon S3 performance guidelines on requests per second are exceeded.

The S3 RTC SLA also doesn't apply during time periods where your replication data transfer rate exceeds the default 1 Gbps limit. If you expect your replication transfer rate to exceed 1 Gbps, you can contact AWS Support Center or use Service Quotas to request an increase in your limit.

**Estimating your replication request rates**

Your total request rate including the requests that Amazon S3 replication makes on your behalf should be within the Amazon S3 request rate guidelines for both the replication source and destination buckets. For each object replicated, Amazon S3 replication makes up to five GET/HEAD requests and one PUT request to the source bucket, and one PUT request to each destination bucket.

For example, if you expect to replicate 100 objects per second, Amazon S3 replication might perform an additional 100 PUT requests on your behalf for a total of 200 PUTs per second to the source S3 bucket. Amazon S3 replication also might perform up to 500 GET/HEAD (5 GET/HEAD requests for each object replicated.)

> **Note**
>
> You incur costs for only one PUT request per object replicated. For more information, see the pricing information in the Amazon S3 FAQ on replication.

**Exceeding S3 RTC data transfer rate limits**

If you expect your S3 Replication Time Control data transfer rate to exceed the default 1 Gbps limit, contact AWS Support Center or use Service Quotas to request an increase in your limit.

**AWS KMS encrypted object replication request rates**

When you replicate objects encrypted with server-side encryption (SSE-KMS) using Amazon S3 replication, AWS Key Management Service (AWS KMS) requests per second limits apply. AWS KMS might reject an otherwise valid request because your request rate exceeds the limit for the number of requests per second. When a request is throttled, AWS KMS returns a `ThrottlingException` error. The AWS KMS request rate limit applies to requests you make directly and to requests made by Amazon S3 replication on your behalf.

For example, if you expect to replicate 1,000 objects per second, you can subtract 2,000 requests from your AWS KMS request rate limit. The resulting request rate per second is available for your AWS KMS workloads excluding replication. You can use AWS KMS request metrics in Amazon CloudWatch to monitor the total AWS KMS request rate on your AWS account.

**Enabling S3 Replication Time Control (S3 RTC)**

S3 Replication Time Control (S3 RTC) helps you meet compliance or business requirements for data replication and provides visibility into Amazon S3 replication times. S3 RTC replicates most objects that you upload to Amazon S3 in seconds, and 99.99 percent of those objects within 15 minutes.

With S3 RTC, you can monitor the total number and size of objects that are pending replication, and the maximum replication time to the destination Region. Replication metrics are available through the AWS Management Console and Amazon CloudWatch User Guide. For more information, see the section called “S3 Replication metrics in CloudWatch ”.

**Using the S3 console**

For step-by-step instructions, see Configuring replication for source and destination buckets owned by the same account. This topic provides instructions for enabling S3 RTC in your replication configuration when buckets are owned by same and different AWS accounts.

**Using the AWS CLI**

To use the AWS CLI to replicate objects with S3 RTC enabled, you create buckets, enable versioning on the buckets, create an IAM role that gives Amazon S3 permission to replicate objects, and add the replication configuration to the source bucket. The replication configuration needs to have S3 Replication Time Control (S3 RTC) enabled.

**To replicate with S3 RTC enabled (AWS CLI)**

- The following example sets `ReplicationTime` and `Metric`, and adds replication configuration to the source bucket.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
```

```
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::destination",
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes` and `ReplicationTime:Time:Minutes` can only have 15 as a valid value.

### Using the AWS SDK for Java

The following Java example adds replication configuration with S3 Replication Time Control (S3 RTC).

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
```

```
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;

public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
      .rules(
          ReplicationRule
            .builder()
            .status("Enabled")
            .priority(1)
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
```

```
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
        .role("role_arn")
        .build();

    // Put replication configuration
    PutBucketReplicationRequest putBucketReplicationRequest =
 PutBucketReplicationRequest
      .builder()
      .bucket("source_bucket")
      .replicationConfiguration(replicationConfig)
      .build();

    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

For more information, see Meeting compliance requirements using S3 Replication Time Control (S3 RTC).

**Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS)**

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API

response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

There are some special considerations when you're replicating objects that have been encrypted by using server-side encryption. Amazon S3 supports the following types of server-side encryption:

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)

For more information about server-side encryption, see the section called "Server-side encryption".

This topic explains the permissions that you need to direct Amazon S3 to replicate objects that have been encrypted by using server-side encryption. This topic also provides additional configuration elements that you can add and example AWS Identity and Access Management (IAM) policies that grant the necessary permissions for replicating encrypted objects.

For an example with step-by-step instructions, see Enabling replication for encrypted objects. For information about creating a replication configuration, see Replicating objects overview.

**Note**

You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in *AWS Key Management Service Developer Guide*.

**Topics**

- How default bucket encryption affects replication
- Replicating objects encrypted with SSE-C
- Replicating objects encrypted with SSE-S3, SSE-KMS, or DSSE-KMS
- Enabling replication for encrypted objects

### How default bucket encryption affects replication

When you enable default encryption for a replication destination bucket, the following encryption behavior applies:

- If objects in the source bucket are not encrypted, the replica objects in the destination bucket are encrypted by using the default encryption settings of the destination bucket. As a result, the entity tags (ETags) of the source objects differ from the ETags of the replica objects. If you have applications that use ETags, you must update those applications to account for this difference.
- If objects in the source bucket are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), the replica objects in the destination bucket use the same type of encryption as the source objects. The default encryption settings of the destination bucket are not used.

### Replicating objects encrypted with SSE-C

By using server-side encryption with customer-provided keys (SSE-C), you can manage your own proprietary encryption keys. With SSE-C, you manage the keys while Amazon S3 manages the encryption and decryption process. You must provide an encryption key as part of your request, but you don't need to write any code to perform object encryption or decryption. When you upload an object, Amazon S3 encrypts the object by using the key that you provided. Amazon S3 then purges that key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. For more information, see the section called “Customer-provided encryption keys (SSE-C)”.

S3 Replication supports objects that are encrypted with SSE-C. You can configure SSE-C object replication in the Amazon S3 console or with the AWS SDKs, the same way that you configure replication for unencrypted objects. There aren't additional SSE-C permissions beyond what are currently required for replication.

S3 Replication automatically replicates newly uploaded SSE-C encrypted objects if they are eligible, as specified in your S3 Replication configuration. To replicate existing objects in your buckets, use S3 Batch Replication. For more information about replicating objects, see the section called “Setting up live replication” and the section called “Replicating existing objects”.

There are no additional charges for replicating SSE-C objects. For details about replication pricing, see the Amazon S3 pricing page.

### Replicating objects encrypted with SSE-S3, SSE-KMS, or DSSE-KMS

By default, Amazon S3 doesn't replicate objects that are encrypted with SSE-KMS or DSSE-KMS. This section explains the additional configuration elements that you can add to direct Amazon S3 to replicate these objects.

For an example with step-by-step instructions, see Enabling replication for encrypted objects. For information about creating a replication configuration, see Replicating objects overview.

#### Specifying additional information in the replication configuration

In the replication configuration, you do the following:

- In the `Destination` element in your replication configuration, add the ID of the symmetric AWS KMS customer managed key that you want Amazon S3 to use to encrypt object replicas, as shown in the following example replication configuration.
- Explicitly opt in by enabling replication of objects encrypted by using KMS keys (SSE-KMS or DSSE-KMS). To opt in, add the `SourceSelectionCriteria` element, as shown in the following example replication configuration.

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 Region as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

> **Important**
>
> The KMS key must have been created in the same AWS Region as the destination buckets. The KMS key *must* be valid. The `PutBucketReplication` API operation doesn't check the validity of KMS keys. If you use a KMS key that isn't valid, you will receive the HTTP `200 OK` status code in response, but replication fails.

The following example shows a replication configuration that includes optional configuration elements. This replication configuration has one rule. The rule applies to objects with the `Tax` key prefix. Amazon S3 uses the specified AWS KMS key ID to encrypt these object replicas.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 Region as the destination bucket. (S3 uses this key to encrypt object replicas.)</
ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

### Granting additional permissions for the IAM role

To replicate objects that are encrypted at rest by using SSE-S3, SSE-KMS, or DSSE-KMS, grant the following additional permissions to the AWS Identity and Access Management (IAM) role that you specify in the replication configuration. You grant these permissions by updating the permissions policy that's associated with the IAM role.

- **`s3:GetObjectVersionForReplication` action for source objects** – This action allows Amazon S3 to replicate both unencrypted objects and objects created with server-side encryption by using SSE-S3, SSE-KMS, or DSSE-KMS.

  > **Note**
  >
  > We recommend that you use the `s3:GetObjectVersionForReplication` action instead of the `s3:GetObjectVersion` action because `s3:GetObjectVersionForReplication` provides Amazon S3 with only the minimum permissions necessary for replication. In addition, the `s3:GetObjectVersion` action allows replication of unencrypted and SSE-S3-encrypted objects, but not of objects that are encrypted by using KMS keys (SSE-KMS or DSSE-KMS).

- **`kms:Decrypt` and `kms:Encrypt` AWS KMS actions for the KMS keys**
  - You must grant `kms:Decrypt` permissions for the AWS KMS key that's used to decrypt the source object.
  - You must grant `kms:Encrypt` permissions for the AWS KMS key that's used to encrypt the object replica.
- **`kms:GenerateDataKey` action for replicating plaintext objects** – If you're replicating plaintext objects to a bucket with SSE-KMS or DSSE-KMS encryption enabled by default, you must include the `kms:GenerateDataKey` permission for the destination encryption context and the KMS key in the IAM policy.

We recommend that you restrict these permissions only to the destination buckets and objects by using AWS KMS condition keys. The AWS account that owns the IAM role must have permissions for the `kms:Encrypt` and `kms:Decrypt` actions for the KMS keys that are listed in the policy. If the KMS keys are owned by another AWS account, the owner of the KMS keys must grant these permissions to the AWS account that owns the IAM role. For more information about managing access to these KMS keys, see Using IAM Policies with AWS KMS in the *AWS Key Management Service Developer Guide*.

### S3 Bucket Keys and replication

To use replication with an S3 Bucket Key, the AWS KMS key policy for the KMS key that's used to encrypt the object replica must include the `kms:Decrypt` permission for the calling principal. The call to `kms:Decrypt` verifies the integrity of the S3 Bucket Key before using it. For more information, see Using an S3 Bucket Key with replication.

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket's Amazon Resource Name (ARN), not the object's ARN (for example, `arn:aws:s3:::bucket_ARN`). You must update your IAM policies to use the bucket ARN for the encryption context:

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

For more information, see Encryption context (`x-amz-server-side-encryption-context`) (in the "Using the REST API" section) and Changes to note before enabling an S3 Bucket Key.

### Example policies – Using SSE-S3 and SSE-KMS with replication

The following example IAM policies show statements for using SSE-S3 and SSE-KMS with replication.

### Example – Using SSE-KMS with separate destination buckets

The following example policy shows statements for using SSE-KMS with separate destination buckets.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action": ["kms:Decrypt"],
         "Effect": "Allow",
         "Condition": {
             "StringLike": {
                  "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                      "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                  ]
             }
```

```
            },
            "Resource": [
               "List of AWS KMS key ARNs that are used to encrypt source objects."
            ]
         },
         {
            "Action": ["kms:Encrypt"],
            "Effect": "Allow",
            "Condition": {
               "StringLike": {
                  "kms:ViaService": "s3.destination-bucket-1-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                     "arn:aws:s3:::amzn-s3-demo-destination-bucket1/key-prefix1*"
                  ]
            }
            },
            "Resource": [
               "AWS KMS key ARNs (in the same AWS Region as destination bucket 1). Used to
 encrypt object replicas created in destination bucket 1."
            ]
         },
         {
            "Action": ["kms:Encrypt"],
            "Effect": "Allow",
            "Condition": {
               "StringLike": {
                  "kms:ViaService": "s3.destination-bucket-2-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                     "arn:aws:s3:::amzn-s3-demo-destination-bucket2/key-prefix1*"
                  ]
            }
            },
            "Resource": [
               "AWS KMS key ARNs (in the same AWS Region as destination bucket 2). Used to
 encrypt object replicas created in destination bucket 2."
            ]
         }
      ]
}
```

### Example – Replicating objects created with SSE-S3 and SSE-KMS

The following is a complete IAM policy that grants the necessary permissions to replicate unencrypted objects, objects created with SSE-S3, and objects created with SSE-KMS.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
```

```
                "kms:EncryptionContext:aws:s3:arn":[
                   "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                ]
             }
          },
          "Resource":[
            "List of the AWS KMS key ARNs that are used to encrypt source objects."
          ]
       },
       {
          "Action":[
             "kms:Encrypt"
          ],
          "Effect":"Allow",
          "Condition":{
             "StringLike":{
                "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
                "kms:EncryptionContext:aws:s3:arn":[
                   "arn:aws:s3:::amzn-s3-demo-destination-bucket/prefix1*"
                ]
             }
          },
          "Resource":[
             "AWS KMS key ARNs (in the same AWS Region as the destination bucket) to use
 for encrypting object replicas"
          ]
       }
    ]
}
```

### Example – Replicating objects with S3 Bucket Keys

The following is a complete IAM policy that grants the necessary permissions to replicate objects with S3 Bucket Keys.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
```

```
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-source-bucket"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
```

```
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS Region as the destination bucket) to use
 for encrypting object replicas"
         ]
      }
   ]
}
```

### Granting additional permissions for cross-account scenarios

In a cross-account scenario, where the source and destination buckets are owned by different AWS accounts, you can use a KMS key to encrypt object replicas. However, the KMS key owner must grant the source bucket owner permission to use the KMS key.

> **Note**
>
> If you need to replicate SSE-KMS data cross-account, then your replication rule must specify a customer managed key from AWS KMS for the destination account. AWS managed keys don't allow cross-account use, and therefore can't be used to perform cross-account replication.

### To grant the source bucket owner permission to use the KMS key (AWS KMS console)

1. Sign in to the AWS Management Console and open the AWS KMS console at https://console.aws.amazon.com/kms.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. To view the keys in your account that you create and manage, in the navigation pane choose **Customer managed keys**.
4. Choose the KMS key.

5. Under the **General configuration** section, choose the **Key policy** tab.
6. Scroll down to **Other AWS accounts**.
7. Choose **Add other AWS accounts**.

   The **Other AWS accounts** dialog box appears.
8. In the dialog box, choose **Add another AWS account**. For **arn:aws:iam::**, enter the source bucket account ID.
9. Choose **Save changes**.

**To grant the source bucket owner permission to use the KMS key (AWS CLI)**

- For information about the `put-key-policy` AWS Command Line Interface (AWS CLI) command, see put-key-policy in the *AWS CLI Command Reference*. For information about the underlying `PutKeyPolicy` API operation, see PutKeyPolicy in the AWS Key Management Service API Reference.

**AWS KMS transaction quota considerations**

When you add many new objects with AWS KMS encryption after enabling Cross-Region Replication (CRR), you might experience throttling (HTTP `503 Service Unavailable` errors). Throttling occurs when the number of AWS KMS transactions per second exceeds the current quota. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

To request a quota increase, use Service Quotas. For more information, see Requesting a quota increase. If Service Quotas isn't supported in your Region, open an AWS Support case.

**Enabling replication for encrypted objects**

By default, Amazon S3 doesn't replicate objects that are encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). To replicate objects encrypted with SSE-KMS or DSS-KMS, you must modify the bucket replication configuration to tell Amazon S3 to replicate these objects. This example explains how to use the Amazon S3 console and the AWS Command Line Interface (AWS CLI) to change the bucket replication configuration to enable replicating encrypted objects.

For more information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

**Note**

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket's Amazon Resource Name (ARN), not the object's ARN. You must update your IAM policies to use the bucket ARN for the encryption context. For more information, see S3 Bucket Keys and replication.

**Note**

You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in *AWS Key Management Service Developer Guide*.

### Using the S3 console

For step-by-step instructions, see Configuring replication for source and destination buckets owned by the same account. This topic provides instructions for setting a replication configuration when the buckets are owned by the same and different AWS accounts.

### Using the AWS CLI

To replicate encrypted objects with the AWS CLI, you do the following:

- Create source and destination buckets and enable versioning on these buckets.
- Create an AWS Identity and Access Management (IAM) service role that gives Amazon S3 permission to replicate objects. The IAM role's permissions include the necessary permissions to replicate the encrypted objects.
- Add a replication configuration to the source bucket. The replication configuration provides information related to replicating objects that are encrypted by using KMS keys.
- Add encrypted objects to the source bucket.
- Test the setup to confirm that your encrypted objects are being replicated to the destination bucket.

The following procedures walk you through this process.

**To replicate server-side encrypted objects (AWS CLI)**

1. In this example, you create both the *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-bucket* buckets in the same AWS account. You also set a credentials profile for the AWS CLI. This example uses the profile name *acctA*.

   For more information about setting credential profiles, see Named Profiles in the AWS Command Line Interface User Guide. To use the commands in this example, replace the *user input placeholders* with your own information.

2. Use the following commands to create the *DOC-EXAMPLE-SOURCE-BUCKET* bucket and enable versioning on it. The following example commands create the *DOC-EXAMPLE-SOURCE-BUCKET* bucket in the US East (N. Virginia) (`us-east-1`) Region.

```
aws s3api create-bucket \
--bucket DOC-EXAMPLE-SOURCE-BUCKET \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-SOURCE-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. Use the following commands to create the *DOC-EXAMPLE-DESTINATION-BUCKET* bucket and enable versioning on it. The following example commands create the *DOC-EXAMPLE-DESTINATION-BUCKET* bucket in the US West (Oregon) (`us-west-2`) Region.

   **Note**

   To set up a replication configuration when both *DOC-EXAMPLE-SOURCE-BUCKET* and *DOC-EXAMPLE-DESTINATION-BUCKET* buckets are in the same AWS account, you use the same profile. In this example, we use *acctA*. To configure replication when the buckets are owned by different AWS accounts, you specify different profiles for each.

```
aws s3api create-bucket \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--region us-west-2 \
```

```
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. Next, you create an IAM service role. You will specify this role in the replication configuration that you add to the *DOC-EXAMPLE-SOURCE-BUCKET* bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create a service role.
   - Attach a permissions policy to the role.

   a. To create an IAM service role, do the following:

      i. Copy the following trust policy and save it to a file called `s3-role-trust-policy-kmsobj.json` in the current directory on your local computer. This policy grants Amazon S3 service principal permissions to assume the role so that Amazon S3 can perform tasks on your behalf.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. Use the following command to create the role:

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
```

```
--profile acctA
```

b. Next, you attach a permissions policy to the role. This policy grants permissions for various Amazon S3 bucket and object actions.

i. Copy the following permissions policy and save it to a file named `s3-role-permissions-policykmsobj.json` in the current directory on your local computer. You will create an IAM role and attach the policy to it later.

> **Important**
>
> In the permissions policy, you specify the AWS KMS key IDs that will be used for encryption of the *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-bucket* buckets. You must create two separate KMS keys for the *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-bucket* buckets. AWS KMS keys are not shared outside the AWS Region in which they were created.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket",
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
```

```
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLikeIfExists":{
              "s3:x-amz-server-side-encryption":[
                 "aws:kms",
                 "AES256",
                 "aws:kms:dsse"
              ],
              "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                 "AWS KMS key IDs(in ARN format) to use for encrypting
object replicas"
              ]
           }
        },
        "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
     },
     {
        "Action":[
           "kms:Decrypt"
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLike":{
              "kms:ViaService":"s3.us-east-1.amazonaws.com",
              "kms:EncryptionContext:aws:s3:arn":[
                 "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
              ]
           }
        },
        "Resource":[
           "AWS KMS key IDs(in ARN format) used to encrypt source
objects."
        ]
     },
     {
        "Action":[
           "kms:Encrypt"
        ],
        "Effect":"Allow",
        "Condition":{
           "StringLike":{
              "kms:ViaService":"s3.us-west-2.amazonaws.com",
              "kms:EncryptionContext:aws:s3:arn":[
```

```
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
             ]
          }
       },
       "Resource":[
          "AWS KMS key IDs(in ARN format) to use for encrypting object
 replicas"
       ]
    }
  ]
}
```

ii. Create a policy and attach it to the role.

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. Next, add the following replication configuration to the *amzn-s3-demo-source-bucket* bucket. It tells Amazon S3 to replicate objects with the `Tax/` prefix to the *amzn-s3-demo-destination-bucket* bucket.

> **Important**
>
> In the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if you have the `iam:PassRole` permission. The profile that you specify in the CLI command must have this permission. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

```
<ReplicationConfiguration>
 <Role>IAM-Role-ARN</Role>
 <Rule>
   <Priority>1</Priority>
   <DeleteMarkerReplication>
      <Status>Disabled</Status>
   </DeleteMarkerReplication>
   <Filter>
      <Prefix>Tax</Prefix>
```

```
    </Filter>
    <Status>Enabled</Status>
    <SourceSelectionCriteria>
      <SseKmsEncryptedObjects>
        <Status>Enabled</Status>
      </SseKmsEncryptedObjects>
    </SourceSelectionCriteria>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
      <EncryptionConfiguration>
        <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
      </EncryptionConfiguration>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

To add a replication configuration to the *amzn-s3-demo-source-bucket* bucket, do the following:

a. The AWS CLI requires you to specify the replication configuration as JSON. Save the following JSON in a file (`replication.json`) in the current directory on your local computer.

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
```

```
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. Edit the JSON to provide values for the *amzn-s3-demo-destination-bucket* bucket, *AWS KMS key IDs (in ARN format)*, and *IAM-role-ARN*. Save the changes.

c. Use the following command to add the replication configuration to your *amzn-s3-demo-source-bucket* bucket. Be sure to provide the *amzn-s3-demo-source-bucket* bucket name.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

6. Test the configuration to verify that encrypted objects are replicated. In the Amazon S3 console, do the following:

a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

b. In the *amzn-s3-demo-source-bucket* bucket, create a folder named `Tax`.

c. Add sample objects to the folder. Be sure to choose the encryption option and specify your KMS key to encrypt the objects.

d. Verify that the *amzn-s3-demo-destination-bucket* bucket contains the object replicas and that they are encrypted by using the KMS key that you specified in the configuration. For more information, see the section called “Getting replication status”.

### Using the AWS SDKs

For a code example that shows how to add a replication configuration, see Using the AWS SDKs. You must modify the replication configuration appropriately.

For conceptual information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS).

**Replicating metadata changes with Amazon S3 replica modification sync**

Amazon S3 replica modification sync can help you keep object metadata such as tags, ACLs, and Object Lock settings replicated between replicas and source objects. By default, Amazon S3 replicates metadata from the source objects to the replicas only. When replica modification sync is enabled, Amazon S3 replicates metadata changes made to the replica copies back to the source object, making the replication bidirectional.

**Enabling replica modification sync**

You can use Amazon S3 replica modification sync with new or existing replication rules. You can apply it to an entire S3 bucket or to Amazon S3 objects that have a specific prefix.

To enable replica modification sync using the Amazon S3 console, see Examples for configuring live replication. This topic provides instructions for enabling replica modification sync in your replication configuration when buckets are owned by the same or different AWS accounts.

To enable replica modification sync using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the bucket containing the replicas with `ReplicaModifications` enabled. To set up two-way replication, create a replication rule from the source bucket (*amzn-s3-demo-bucket1*) to the bucket containing the replicas (*amzn-s3-demo-bucket2*). Then, create a second replication rule from the bucket containing the replicas (*amzn-s3-demo-bucket2*) to the source bucket (*amzn-s3-demo-bucket1*). Buckets can be in the same, or in different, AWS Regions.

> **Note**
>
> You must enable replica modification sync on both buckets to replicate replica metadata changes like object access control lists (ACLs), object tags, or Object Lock settings on the replicated objects. Like all replication rules, these rules can either be applied to the entire Amazon S3 bucket or a subset of Amazon S3 objects filtered by prefix or object tags.

In the following example configuration, Amazon S3 replicates metadata changes under the prefix *Tax* to the bucket *amzn-s3-demo-bucket*, which would contain the source objects.

```
{
```

```
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

For full instructions on creating replication rules using the AWS CLI, see Configuring replication for source and destination buckets owned by the same account.

**Replicating delete markers between buckets**

By default, when S3 Replication is enabled and an object is deleted in the source bucket, Amazon S3 adds a delete marker in the source bucket only. This action protects data from malicious deletions.

If you have *delete marker replication* enabled, these markers are copied to the destination buckets, and Amazon S3 behaves as if the object was deleted in both source and destination buckets. For more information about how delete markers work, see Working with delete markers.

> **Note**
>
> Delete marker replication is not supported for tag-based replication rules. Delete marker replication also does not adhere to the 15-minute SLA granted when using S3 Replication Time Control.

If you are not using the latest replication configuration version, delete operations will affect replication differently. For more information, see How delete operations affect replication.

### Enabling delete marker replication

You can start using delete marker replication with a new or existing replication rule. You can apply it to an entire S3 bucket or to Amazon S3 objects that have a specific prefix.

> **Note**
>
> When you enable delete marker replication and your bucket has an S3 Lifecycle expiration rule, the delete markers added by the S3 Lifecycle expiration rule won't be replicated to the destination bucket.

To enable delete marker replication using the Amazon S3 console, see Using the S3 console. This topic provides instructions for enabling delete marker replication in your replication configuration when buckets are owned by the same or different AWS accounts.

To enable delete marker replication using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the source bucket with `DeleteMarkerReplication` enabled.

In the following example configuration, delete markers are replicated to the destination bucket *DOC-EXAMPLE-BUCKET* for objects under the prefix *Tax*.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

For full instructions on creating replication rules through the AWS CLI, see Configuring replication for source and destination buckets owned by the same account in the Replication walkthroughs section.

## Managing or pausing live replication

Live replication is the automatic, asynchronous copying of objects across buckets in the same or different AWS Regions. After you set up your replication configuration, Amazon S3 replicates newly created objects and object updates from a source bucket to one or more specified destination buckets.

You use the Amazon S3 console to add replication rules to the source bucket. Replication rules define the source bucket objects to replicate and the destination bucket or buckets where the replicated objects are stored. For more information about replication, see Replicating objects overview.

You can manage replication rules on the **Replication** page. You can add, view, enable, disable, or delete replication rules. You can also change the priority of your replication rules. For information about adding replication rules to a bucket, see Using the S3 console.

**To manage the replication rules for an S3 bucket by using the Amazon S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. On the **General purpose buckets** tab, choose the name of the bucket that you want.
4. Choose the **Management** tab, and then scroll down to **Replication rules**.
5. You can change your replication rules in the following ways:
   - To enable or disable a replication rule, choose the option button to the left of the rule. On the **Actions** menu, choose **Enable rule** or **Disable rule**. You can also disable, enable, or delete all the rules in the bucket from the **Actions** menu.

     **Note**

     If you disable a replication rule and then later re-enable the rule, any new or changed objects that weren't replicated while the rule was disabled are *not* automatically replicated when the rule is re-enabled. To replicate those objects,

you must use S3 Batch Replication. For more information, see the section called "Replicating existing objects".

- To change the priority of a rule, choose the option button to the left of the rule, and then choose **Edit rule**.

  You set rule priorities to avoid conflicts caused by objects that are included in the scope of more than one rule. In the case of overlapping rules, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority. For more information about rule priority, see Replication configuration.

### Pausing or stopping replication

To temporarily pause replication and have it automatically resume later, you can use the `aws:s3:bucket-pause-replication` action in AWS Fault Injection Service. For more information, see aws:s3:bucket-pause-replication and Pause S3 Replication in the *AWS Fault Injection Service User Guide*.

To stop replication in Amazon S3, we recommend disabling your replication rules. If you disable a replication rule and then later re-enable the rule, any new or changed objects that weren't replicated while the rule was disabled are *not* automatically replicated when the rule is re-enabled. To replicate those objects, you must use S3 Batch Replication. For more information, see the section called "Replicating existing objects".

Replication will also stop if you remove the AWS Identity and Access Management (IAM) role, AWS Key Management Service (AWS KMS) permissions, or the bucket policy permissions that grant Amazon S3 the required permissions. However, we don't recommend these approaches because they cause replication to fail. Amazon S3 reports the replication status for affected objects as `FAILED`. If permissions are later restored, objects marked as `FAILED` are *not* automatically replicated. To replicate those objects, you must use S3 Batch Replication.

## Monitoring progress with replication metrics and S3 Event Notifications

S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

S3 Replication metrics are turned on automatically when you enable S3 Replication Time Control (S3 RTC). You can also enable S3 Replication metrics independently of S3 RTC while creating

or editing a rule. S3 RTC includes other features, such as a service level agreement (SLA) and notifications for missed thresholds. For more information, see Meeting compliance requirements using S3 Replication Time Control (S3 RTC).

The bytes pending, operations pending, and replication-latency metrics apply only to new objects that are replicated with S3 Cross-Region Replication (S3 CRR) or S3 Same-Region Replication (S3 SRR). The operations failing replication metric tracks both new objects that are replicated with S3 CRR or S3 SRR and existing objects that are replicated with S3 Batch Replication. To assist in troubleshooting any configuration issues, you can also set up Amazon S3 Event Notifications to receive replication-failure events.

When enabled, S3 Replication metrics publish the following metrics to Amazon CloudWatch:

- **Bytes Pending Replication** – The total number of bytes of objects that are pending replication for a given replication rule.
- **Replication Latency** – The maximum number of seconds by which the replication destination buckets are behind the source bucket for a given replication rule.
- **Operations Pending Replication** – The number of operations that are pending replication for a given replication rule. This metric tracks operations related to objects, delete markers, tags, access control lists (ACLs), and S3 Object Lock.
- **Operations Failed Replication** – The number of operations that failed replication for a given replication rule. This metric tracks operations related to objects, delete markers, tags, ACLs, and Object Lock. Unlike the other replication metrics, this metric applies both to new objects that are replicated with S3 CRR or S3 SRR and to existing objects that are replicated with S3 Batch Replication.

> **Note**
>
> **Operations Failed Replication** tracks S3 Replication failures aggregated at a per-minute interval. To identify the specific objects that have failed replication and their failure reasons, subscribe to the `OperationFailedReplication` event in Amazon S3 Event Notifications. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.

If a job fails to run at all, metrics are not sent to Amazon CloudWatch. For example, your job won't run if you don't have the necessary permissions to run an S3 Batch Replication job, or if the tags or prefix in your replication configuration don't match.

**Topics**



## Enabling S3 Replication metrics

You can start using S3 Replication metrics with a new or existing replication rule. You can choose to apply your replication rule to an entire S3 bucket, or to Amazon S3 objects with a specific prefix or tag.

This topic provides instructions for enabling S3 Replication metrics in your replication configuration when the source and destination buckets are owned by the same or different AWS accounts.

To enable replication metrics by using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the source bucket with `Metrics` enabled. In this example configuration, objects under the prefix *`Tax`* are replicated to the destination bucket *`DOC-EXAMPLE-BUCKET`*, and metrics are generated for those objects.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "Metrics": {
                    "Status": "Enabled"
                }
```

```
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

For full instructions on creating replication rules, see Configuring replication for source and destination buckets owned by the same account.

For more information about viewing replication metrics in the S3 console, see Viewing replication metrics by using the Amazon S3 console.

> **Note**
>
> S3 Replication metrics are billed at the same rate as Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch pricing.

## Receiving replication failure events with Amazon S3 Event Notifications

S3 Event Notifications can notify you in instances when objects don't replicate to their destination AWS Region. Amazon S3 events are available through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see the section called “Amazon S3 Event Notifications”.

For a list of failure codes captured by S3 Event Notifications, see Amazon S3 replication failure reasons.

## Viewing replication metrics with S3 Storage Lens

To get detailed metrics for S3 Replication, including replication rule count metrics, you can use Amazon S3 Storage Lens. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to protect your data. For a complete list of metrics, see the S3 Storage Lens metrics glossary.

## Viewing replication metrics by using the Amazon S3 console

There are three types of Amazon CloudWatch metrics for Amazon S3: storage metrics, request metrics, and replication metrics. S3 Replication metrics are turned on automatically when you

enable replication with S3 Replication Time Control (S3 RTC) by using the AWS Management Console or the Amazon S3 API. You can also enable S3 Replication metrics independently of S3 RTC while creating or editing a rule.

Replication metrics track the rule IDs of the replication configuration. A replication rule ID can be specific to a prefix, a tag, or a combination of both.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

**Prerequisites**

Enable a replication rule that has S3 Replication metrics.

**To view replication metrics**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**. In the **Buckets** list, choose the name of the bucket that contains the objects that you want replication metrics for.
3. Choose the **Metrics** tab.
4. Under **Replication metrics**, choose **Replication rules**.
5. Choose **Display charts**.

   Amazon S3 displays **Replication latency (in seconds)**, **Bytes pending replication**, **Operations pending replication**, and **Operations failed replication** charts.

You can then view the replication metrics **Replication latency (in seconds)**, **Operations pending replication**, **Bytes pending replication**, and **Operations failed replication** for the rules that you selected. If you're using S3 Replication Time Control, Amazon CloudWatch begins reporting replication metrics 15 minutes after you enable S3 RTC on the respective replication rule. You can view replication metrics on the Amazon S3 console or the CloudWatch console. For more information, see Replication metrics with S3 RTC.

> **Note**
>
> You can also view detailed metrics for S3 Replication in the Amazon S3 console by using Amazon S3 Storage Lens. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more

information, see Using S3 Storage Lens to protect your data. For a complete list of metrics, see the S3 Storage Lens metrics glossary.

## Amazon S3 replication failure reasons

The following table lists Amazon S3 Replication failure reasons. You can view these reasons by receiving the **failureReason** event with Amazon S3 Event Notifications. You can receive S3 Event Notifications through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see Amazon S3 Event Notifications.

You can also view these failure reasons in an S3 Batch Replication completion report. For more information, see Batch Replication completion report.

| Replication failure reason | Description |
| --- | --- |
| `AssumeRoleNotPermitted` | Amazon S3 can't assume the AWS Identity and Access Management (IAM) role that's specified in the replication configuration or in the Batch Operations job. |
| `DstBucketInvalidRegion` | The destination bucket is not in the same AWS Region as specified by the Batch Operations job. This error is specific to Batch Replication. |
| `DstBucketNotFound` | Amazon S3 is unable to find the destinati on bucket that's specified in the replication configuration. |
| `DstBucketObjectLockConfigMissing` | To replicate objects from a source bucket with Object Lock enabled, the destination bucket must also have Object Lock enabled. This error indicates that Object Lock might not be enabled in the destination bucket. For more information, see Object Lock considera tions. |

| Replication failure reason | Description |
| --- | --- |
| `DstBucketUnversioned` | Versioning is not enabled for the S3 destination bucket. To replicate objects with S3 Replication, enable versioning for the destination bucket. |
| `DstDelObjNotPermitted` | Amazon S3 is unable to replicate delete markers to the destination bucket. The `s3:ReplicateDelete` permission might be missing for the destination bucket. |
| `DstKmsKeyInvalidState` | The AWS Key Management Service (AWS KMS) key for the destination bucket isn't in a valid state. Review and enable the required AWS KMS key. For more information about managing AWS KMS keys, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*. |
| `DstKmsKeyNotFound` | The AWS KMS key that's configured for the destination bucket in the replication configuration doesn't exist. |
| `DstMultipartCompleteNotPermitted` | Amazon S3 is unable to complete multipart uploads of objects in the destination bucket. The `s3:ReplicateObject` permission might be missing for the destination bucket. |
| `DstMultipartInitNotPermitted` | Amazon S3 is unable to initiate multipart uploads of objects to the destination bucket. The `s3:ReplicateObject` permission might be missing for the destination bucket. |
| `DstMultipartPartUploadNotPermitted` | Amazon S3 is unable to upload multipart objects to the destination bucket. The `s3:ReplicateObject` permission might be missing for the destination bucket. |

| Replication failure reason | Description |
| --- | --- |
| DstObjectHardDeleted | S3 Batch Replication does not support re-replicating objects deleted with the version ID of the object from the destination bucket. This error is specific to Batch Replication. |
| DstPutAclNotPermitted | Amazon S3 is unable to replicate object access control lists (ACLs) to the destination bucket. The s3:ReplicateObject permission might be missing for the destination bucket. |
| DstPutLegalHoldNotPermitted | Amazon S3 is unable to put an Object Lock legal hold on the destination objects when it is replicating immutable objects. The s3:PutObjectLegalHold permission might be missing for the destination bucket. For more information, see Legal holds. |
| DstPutObjectNotPermitted | Amazon S3 is unable to replicate objects to the destination bucket. The s3:ReplicateObject or s3:ObjectOwnerOverrideToBucketOwner permissions might be missing for the destination bucket. |
| DstPutTaggingNotPermitted | Amazon S3 is unable to replicate object tags to the destination bucket. The s3:ReplicateObject permission might be missing for the destination bucket. |
| DstVersionNotFound | Amazon S3 is unable to find the required object version in the destination bucket for which metadata needs to be replicated. |

| Replication failure reason | Description |
| --- | --- |
| `InitiateReplicationNotPermitted` | Amazon S3 is unable to initiate replication on objects. The `s3:InitiateReplication` permission might be missing for the Batch Operations job. This error is specific to Batch Replication. |
| `SrcBucketInvalidRegion` | The source bucket is not in the same AWS Region as specified by the Batch Operations job. This error is specific to Batch Replication. |
| `SrcBucketNotFound` | Amazon S3 is unable to find the source bucket. |
| `SrcBucketReplicationConfigMissing` | Amazon S3 couldn't find a replication configuration for the source bucket. |
| `SrcGetAclNotPermitted` | Amazon S3 is unable to access the object in the source bucket for replication. The `s3:GetObjectVersionAcl` permission might be missing for the source bucket object.<br><br>The objects in the source bucket must be owned by the bucket owner. If ACLs are enabled, then verify if Object Ownership is set to Bucket owner preferred or Object writer. If Object Ownership is set to Bucket owner preferred, then the source bucket objects must have the `bucket-owner-full-control` ACL for the bucket owner to become the object owner. The source account can take ownership of all objects in their bucket by setting Object Ownership to Bucket owner enforced and disabling ACLs. |

| Replication failure reason | Description |
| --- | --- |
| SrcGetLegalHoldNotPermitted | Amazon S3 is unable to access the S3 Object Lock legal hold information. |
| SrcGetObjectNotPermitted | Amazon S3 is unable to access the object in the source bucket for replication. The s3:GetObjectVersionForReplication permission might be missing for the source bucket. |
| SrcGetRetentionNotPermitted | Amazon S3 is unable to access the S3 Object Lock retention period information. |
| SrcGetTaggingNotPermitted | Amazon S3 is unable to access object tag information from the source bucket. The s3:GetObjectVersionTagging permission might be missing for the source bucket. |
| SrcHeadObjectNotPermitted | Amazon S3 is unable to retrieve object metadata from the source bucket. The s3:GetObjectVersionForReplication permission might be missing for the source bucket. |
| SrcKeyNotFound | Amazon S3 is unable to find the source object key to replicate. Source object may have been deleted before replication was complete. |
| SrcKmsKeyInvalidState | The AWS KMS key for the source bucket isn't in a valid state. Review and enable the required AWS KMS key. For more information about managing AWS KMS keys, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*. |

| Replication failure reason | Description |
| --- | --- |
| SrcObjectNotEligible | Some objects aren't eligible for replication. This may be due to the object's storage class or the object tags don't match the replicati on configuration. |
| SrcObjectNotFound | Source object does not exist. |
| SrcReplicationNotPending | Amazon S3 has already replicated this object. This object is no longer pending replication. |
| SrcVersionNotFound | Amazon S3 is unable to find the source object version to replicate. Source object version may have been deleted before replication was complete. |

**Related topics**

Setting up permissions for live replication

Troubleshooting replication

## Getting replication status information

Replication status can help you determine the current state of an object being replicated. The replication status of a source object will return either PENDING, COMPLETED, or FAILED. The replication status of a replica will return REPLICA.

**Topics**

- Replication status overview
- Replication status if replicating to multiple destination buckets
- Replication status if Amazon S3 replica modification sync is enabled
- Finding replication status

### Replication status overview

In replication, you have a source bucket on which you configure replication and destination where Amazon S3 replicates objects. When you request an object (using GET object) or object metadata (using HEAD object) from these buckets, Amazon S3 returns the `x-amz-replication-status` header in the response:

- When you request an object from the source bucket, Amazon S3 returns the `x-amz-replication-status` header if the object in your request is eligible for replication.

  For example, suppose that you specify the object prefix `TaxDocs` in your replication configuration to tell Amazon S3 to replicate only objects with the key name prefix `TaxDocs`. Any objects that you upload that have this key name prefix—for example, `TaxDocs/document1.pdf`—will be replicated. For object requests with this key name prefix, Amazon S3 returns the `x-amz-replication-status` header with one of the following values for the object's replication status: `PENDING`, `COMPLETED`, or `FAILED`.

  > **Note**
  >
  > If object replication fails after you upload an object, you can't retry replication. You must upload the object again. Objects transition to a `FAILED` state for issues such as missing replication role permissions, AWS KMS permissions, or bucket permissions. For temporary failures, such as if a bucket or Region is unavailable, replication status will not transition to `FAILED`, but will remain `PENDING`. After the resource is back online, S3 will resume replicating those objects.

- When you request an object from a destination bucket, if the object in your request is a replica that Amazon S3 created, Amazon S3 returns the `x-amz-replication-status` header with the value `REPLICA`.

> **Note**
>
> Before deleting an object from a source bucket that has replication enabled, check the object's replication status to ensure that the object has been replicated.
> If lifecycle configuration is enabled on the source bucket, Amazon S3 suspends lifecycle actions until it marks the objects status as either `COMPLETED` or `FAILED`.

### Replication status if replicating to multiple destination buckets

When you replicate objects to multiple destination buckets, the `x-amz-replication-status` header acts differently. The header of the source object only returns a value of `COMPLETED` when replication is successful to all destinations. The header remains at the `PENDING` value until replication has completed for all destinations. If one or more destinations fail replication, the header returns `FAILED`.

### Replication status if Amazon S3 replica modification sync is enabled

When your replication rules enable Amazon S3 replica modification sync, replicas can report statuses other than `REPLICA`. If metadata changes are in the process of replicating, the `x-amz-replication-status` header returns `PENDING`. If replica modification sync fails to replicate metadata, the header returns `FAILED`. If metadata is replicated correctly, the replicas return the header `REPLICA`.

### Finding replication status

To get the replication status of the objects in a bucket, you can use the Amazon S3 Inventory tool. Amazon S3 sends a CSV file to the destination bucket that you specify in the inventory configuration. You can also use Amazon Athena to query the replication status in the inventory report. For more information about Amazon S3 Inventory, see Amazon S3 Inventory.

You can also find the object replication status using the console, the AWS Command Line Interface (AWS CLI), or the AWS SDK.

### Using the S3 console

In the S3 console, you can view the replication status for an object on the object **Details** page under **Object management overview**.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name.
3. In the **Objects** list, choose the object name.
4. Under the **Properties** tab find **Object management overview**, here you can see the **Replication status**.

### Using the AWS CLI

Use the `head-object` command to retrieve object metadata, as follows.

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

The command returns object metadata, including the `ReplicationStatus` as shown in the following example response.

```
{
   "AcceptRanges":"bytes",
   "ContentType":"image/jpeg",
   "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
   "ContentLength":3191,
   "ReplicationStatus":"COMPLETED",
   "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
   "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
   "Metadata":{

   }
}
```

### Using the AWS SDKs

The following code fragments get replication status with the AWS SDK for Java and AWS SDK for .NET, respectively.

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
```

```
            BucketName = sourceBucket,
            Key        = objectKey
        };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

## Replicating existing objects with S3 Batch Replication

By using S3 Batch Replication, you can replicate the following types of objects:

- Objects that existed before a replication configuration was in place
- Objects that have previously been replicated
- Objects that have failed replication

You can replicate these objects on demand by using a Batch Operations job. S3 Batch Replication differs from live replication, which continuously and automatically replicates new objects across Amazon S3 buckets.

To get started with Batch Replication, you can:

- **Initiate Batch Replication for a new replication rule or destination** – You can create a one-time Batch Replication job when you're creating the first rule in a new replication configuration or when you're adding a new destination to an existing configuration through the Amazon S3 console.
- **Initiate Batch Replication for an existing replication configuration** – You can create a new Batch Replication job by using S3 Batch Operations through the Amazon S3 console, the AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 REST API.

When the Batch Replication job finishes, you receive a completion report. For more information about how to use the report to examine the job, see Tracking job status and completion reports.

### S3 Batch Replication considerations

- Your source bucket must have an existing replication configuration. To enable replication, see Setting up live replication and Examples for configuring live replication.

- If you have S3 Lifecycle configured for your bucket, we recommend disabling your lifecycle rules while the Batch Replication job is active. Doing so helps ensure parity between the source and destination buckets. Otherwise, these buckets could diverge, and the destination bucket won't be an exact replica of the source bucket. For example, consider the following scenario:
  - Your source bucket has multiple versions of an object and a delete marker on that object.
  - Your source and destination buckets have a lifecycle configuration to remove expired delete markers.

  In this scenario, Batch Replication might replicate the delete marker to the destination bucket before replicating the object versions. This behavior could result in your lifecycle configuration marking the delete marker as expired and the delete marker being removed from the destination bucket before the object versions are replicated.
- The AWS Identity and Access Management (IAM) role that you specify to run the Batch Operations job must have the necessary permissions to perform the underlying Batch Replication operation. For more information about creating IAM roles, see Configuring IAM policies for Batch Replication.
- Batch Replication requires a manifest, which can be generated by Amazon S3. The generated manifest must be stored in the same AWS Region as the source bucket. If you choose not to generate the manifest, you can supply an Amazon S3 Inventory report or CSV file that contains the objects that you want to replicate.
- Batch Replication doesn't support re-replicating objects that were deleted with the version ID of the object from the destination bucket. To re-replicate these objects, you can copy the source objects in place with a Batch Copy job. Copying those objects in place creates new versions of the objects in the source bucket and automatically initiates replication to the destination bucket. Deleting and recreating the destination bucket doesn't initiate replication.

  For more information about Batch Copy, see Examples that use Batch Operations to copy objects.
- If you're using a replication rule on the S3 bucket, make sure to update your replication configuration by granting the IAM role that's attached to the replication rule the proper permissions to replicate objects. This IAM role must have the necessary permissions to perform replication on both the source and destination buckets.
- If you submit multiple Batch Replication jobs for the same bucket within a short time frame, Amazon S3 will run those jobs concurrently.
- If you submit multiple Batch Replication jobs for two different buckets, be aware that Amazon S3 might not run all jobs concurrently. If you exceed the number of Batch Replication jobs that can run at one time on your account, Amazon S3 will pause the lower priority jobs to work on the

higher priority ones. After the higher priority items have been completed, any paused jobs will become active again.

- Batch Replication isn't supported for objects that are stored in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.
- To batch replicate S3 Intelligent-Tiering objects that are stored in the Archive Access or Deep Archive Access storage tiers, you must first initiate a restore request and wait until the objects are moved to the Frequent Access tier.

## Specifying a manifest for a Batch Replication job

A manifest is an Amazon S3 object that contains the object keys that you want Amazon S3 to act upon. If you want to create a Batch Replication job, you must supply either a user-generated manifest or have Amazon S3 generate a manifest based on your replication configuration.

If you supply a user-generated manifest, it must be in the form of an Amazon S3 Inventory report or a CSV file. If the objects in your manifest are in a versioned bucket, you must specify the version IDs for the objects. Only the object with the version ID that's specified in the manifest will be replicated. To learn more about specifying a manifest, see Specifying a manifest.

If you choose to have Amazon S3 generate a manifest file on your behalf, the objects listed will use the same source bucket, prefix, and tags as all your replication configurations of the source bucket. With a generated manifest, Amazon S3 will replicate all eligible versions of your objects.

> **Note**
>
> If you choose to have Amazon S3 generate the manifest, the manifest must be stored in the same AWS Region as the source bucket.

## Filters for a Batch Replication job

When creating your Batch Replication job, you can optionally specify additional filters, such as the object creation date and replication status, to reduce the scope of the job.

You can filter objects to replicate based on the `ObjectReplicationStatuses` value, by providing one or more of the following values:

- `"NONE"` – Indicates that Amazon S3 has never attempted to replicate the object before.

- `"FAILED"` – Indicates that Amazon S3 has attempted, but failed, to replicate the object before.
- `"COMPLETED"` – Indicates that Amazon S3 has successfully replicated the object before.
- `"REPLICA"` – Indicates that this is a replica object that Amazon S3 has replicated from another source.

For more information about replication statuses, see Getting replication status information.

If you don't filter your Batch Replication job, Batch Operations will attempt to replicate all objects (no matter their `ObjectReplicationStatus`) in your manifest that match the rules in your replication configuration, except for certain objects that aren't replicated by default. For more information, see the section called “What isn't replicated with replication configurations?”

Depending on your goal, you might set `ObjectReplicationStatuses` to one or more of the following values:

- To replicate only existing objects that have never been replicated, only include `"NONE"`.
- To retry replicating only objects that previously failed to replicate, only include `"FAILED"`.
- To both replicate existing objects and retry replicating objects that previously failed to replicate, include both `"NONE"` and `"FAILED"`.
- To backfill a destination bucket with objects that have been replicated to another destination, include `"COMPLETED"`.
- To replicate objects that were previously replicated, include `"REPLICA"`.

## Batch Replication completion report

When you create a Batch Replication job, you can request a CSV completion report. This report shows objects, replication success or failure codes, outputs, and descriptions. For more information about job tracking and completion reports, see Completion reports.

For a list of replication failure codes and descriptions, see Amazon S3 replication failure reasons.

For information about troubleshooting Batch Replication, see Batch Replication errors.

## Getting started with Batch Replication

To learn more about how to use Batch Replication, see Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication.

## Configuring IAM policies for Batch Replication

Because S3 Batch Replication is a type of Batch Operations job, you must create a Batch Operations AWS Identity and Access Management (IAM) role to grant Amazon S3 permissions to perform actions on your behalf. You also must attach a Batch Replication IAM policy to the Batch Operations IAM role. The following example creates an IAM role that gives Batch Operations permission to initiate a Batch Replication job.

### Create IAM role and policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Under **Access management**, choose **Roles**.
3. Choose **Create Role**.
4. Choose **AWS service** as the type of trusted entity, **Amazon S3** as the service, and **S3 Batch Operations** as the use case.
5. Choose **Next: Permissions**.
6. Choose **Create Policy**.
7. Choose **JSON** and insert one of the following policies based on your manifest.

> **Note**
>
> Different permission are needed if you are generating a manifest or supplying one. For more information see, Specifying a manifest for a Batch Replication job.

#### Policy if using and storing an S3 generated manifest

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
```

```
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

### Policy if using a user supplied manifest

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
```

```
            ],
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::*** replication source bucket ***/*"
            ]
         },
               {
            "Action":[
               "s3:GetObject",
               "s3:GetObjectVersion"
            ],
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::*** manifest bucket ***/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:PutObject"
            ],
            "Resource":[
               "arn:aws:s3:::*** completion report bucket ****/*"
            ]
         }
      ]
}
```

8. Choose **Next: Tags**.
9. Choose **Next: Review**.
10. Choose a name for the policy and choose **Create policy**.
11. Attach this policy to your role and choose **Next: Tags**.
12. Choose **Next: Review**.
13. Choose a name for the role and choose **Create role**.

### Verify trust policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Under **Access management**, choose **Roles**, and select your newly created role.

3. Under **Trust relationships** tab, choose **Edit trust relationship**.
4. Verify this role is using the following trust policy:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

## Create a Batch Replication job for a first replication rule or new destination

When you create the first rule in a new replication configuration or add a new destination to an existing configuration through the AWS Management Console, you can optionally create a Batch Replication job.

To use Batch Replication for an existing configuration without adding a new destination see, Create a Batch Replication job for existing replication rules.

### Using Batch Replication for a new replication rule or destination through the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to replicate.
3. To create a new replication rule or edit an existing rule, choose **Management**, and scroll down to **Replication rules**:
   - To create a new replication rule, choose **Create replication rule**.

> **Note**
>
> For examples on how to set up a basic replication rule see, Examples for configuring live replication.

- To edit an existing replication rule, select the rule, and then choose **Edit rule**.

4. Create your new replication rule or edit the destination for your existing replication rule, and choose **Save**.

   After you create the first rule in a new replication configuration or edit an existing configuration to add a new destination, a **Replicate existing objects?** dialog appears, giving you the option to create a Batch Replication job.

5. If you want to run this job now, choose **Yes, replicate existing objects**.

   If you want to run this job at a later time, choose **No, do not replicate existing objects**.

6. Create your S3 Batch Replication job. The S3 Batch Replication job has several settings:

   **Job run option**

   If you want the S3 Batch Replication job to run immediately, you can choose **Job runs automatically when ready**. If you want to run the job at a later time, choose **Job waits to be run when ready**.

   If you choose **Job runs automatically when ready**, you will not be able to create and save a Batch Operations manifest. To save the Batch Operations manifest, choose **Job waits to be run when ready**.

   **Batch Operations manifest**

   The manifest is a list of all of the objects that you want to run the specified action on. You can choose to save the Batch Operations manifest. Similar to S3 Inventory files, the manifest will be saved as a CSV file and stored in a bucket. To learn more about Batch Operations manifests, see Specifying a manifest.

   **Completion report**

   S3 Batch Operations execute one task for each object specified in the manifest. Completion reports provide an easy way to view the results of your tasks in a consolidated format with

no additional setup required. You can request a completion report for all tasks or only for failed tasks. To learn more about completion reports, see Completion reports.

**Permissions**

One of the most common causes of replication failures is insufficient permissions in the provided AWS Identity and Access Management (IAM) role. For information about creating this role see, Configuring IAM policies for Batch Replication.

7. Choose **Create Batch Operations job**.

## Create a Batch Replication job for existing replication rules

You can configure S3 Batch Replication for an existing replication configuration by using the AWS SDKs, AWS Command Line Interface (AWS CLI), or the Amazon S3 console. For an overview of Batch Replication see, Replicating existing objects with S3 Batch Replication.

As a prerequisite, you must create a Batch Operations AWS Identity and Access Management (IAM) role to grant Amazon S3 permissions to perform actions on your behalf, see Configuring IAM policies for Batch Replication.

When the Batch Replication job finishes, you receive a completion report. For more information about how to use the report to examine the job, see Tracking job status and completion reports.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Batch Operations** on the navigation pane of the Amazon S3 console.
3. Choose **Create job**.
4. Choose the **Region** where you want to create your job.
5. Select the **Manifest format**. This example will show how to create a manifest based on an existing S3 replication configuration.

> **Note**
>
> The manifest is a list of all of the objects that you want to run the specified action on. To learn more about Batch Operations manifests, see Specifying a manifest. If you have a manifest prepared, choose **S3 inventory report (manifest.json)** or **CSV**. If

the objects in your manifest are in a versioned bucket, you should specify the version IDs for the objects. For more information about creating a manifest see, Specifying a manifest.

6. To create a manifest based on your replication configuration, choose **Create manifest using S3 Replication configuration**. Then chose the source bucket of your replication configuration.
7. (Optional) You may include additional filters such as object creation date and replication status. For examples on how to filter by replication status see, Specifying a manifest for a Batch Replication job.
8. To save a manifest, select **Save Batch Operations manifest**.

   a. If you choose to generate and save a manifest, you must choose either **Bucket in this account** or **Bucket in another AWS account**. Specify the bucket name in the text box.

   > **Note**
   >
   > The generated manifest must be stored in the same AWS Region as the source bucket.

   b. Choose the **Encryption type**.
9. (Optional) Provide a **Description**.
10. Adjust the **Priority** of the job if needed. Higher numbers indicate higher priority. Amazon S3 attempts to run higher priority jobs before lower priority jobs. For more information about job priority, see Assigning job priority.
11. (Optional) Generate a completion report. To generate select **Generate completion report**.

    If you choose to generate a completion report, you must choose either to report **Failed tasks only** or **All tasks**, and provide a destination bucket for the report.
12. Select a valid IAM role.

    > **Note**
    >
    > For more information about creating a IAM role, see Configuring IAM policies for Batch Replication.

13. (Optional) Add job tags to the Batch Replication job.
14. Choose **Next**.

15. Review your job configuration and select **Create job**.

### Using the AWS CLI with an S3 manifest

The following example creates an S3 Batch Replication job using an S3 generated manifest for the AWS account *111122223333*. This example will try to replicate existing objects and objects that previously failed to replicate. For information about filtering by replication status see, Specifying a manifest for a Batch Replication job.

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", "SourceBucket": "arn:aws:s3:::*** replication source bucket
 ***", "EnableManifestOutput": false, "Filter": {"EligibleForReplication": true,
 "ObjectReplicationStatuses": ["NONE","FAILED"]}}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> The job must be initiated from the same AWS Region replication source bucket. The IAM role `role/batch-Replication-IAM-policy` was previously created. See Configuring IAM policies for Batch Replication.

After you have successfully initiated a Batch Replication job, you receive the job ID as the response. You can monitor this job using the following command.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

### Using the AWS CLI with a user-provided manifest

The following example creates an S3 Batch Replication job using a user-defined manifest for AWS account *111122223333*. If the objects in your manifest are in a versioned bucket, you must specify the version IDs for the objects. Only the object with the version ID specified in the manifest will be replicated. For more information about creating a manifest see, Specifying a manifest.

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key","VersionId"]},"Location":{"ObjectArn":"arn:aws:s3:::*** completion
 report bucket ****/manifest.csv","ETag":"Manifest Etag"}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

**Note**

The job must be initiated from the same AWS Region replication source bucket. The IAM role `role/batch-Replication-IAM-policy` was previously created. See Configuring IAM policies for Batch Replication.

After you have successfully initiated a Batch Replication job, you receive the job ID as the response. You can monitor this job using the following command.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

## Categorizing your storage using tags

Use object tagging to categorize storage. Each tag is a key-value pair.

You can add tags to new objects when you upload them, or you can add them to existing objects.

- You can associate up to 10 tags with an object. Tags that are associated with an object must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Amazon S3 object tags are internally represented in UTF-16. Note that in UTF-16, characters consume either 1 or 2 character positions.
- The key and values are case sensitive.
- For more information about tag restrictions, see User-defined tag restrictions in the *AWS Billing and Cost Management User Guide*. For basic tag restrictions, see Tag restrictions in the *Amazon EC2 User Guide*.

### Examples

Consider the following tagging examples:

#### Example PHI information

Suppose that an object contains protected health information (PHI) data. You might tag the object using the following key-value pair.

```
PHI=True
```

or

```
Classification=PHI
```

#### Example Project files

Suppose that you store project files in your S3 bucket. You might tag these objects with a key named `Project` and a value, as shown following.

```
Project=Blue
```

#### Example Multiple tags

You can add multiple tags to an object, as shown following.

```
Project=x
Classification=confidential
```

#### Key name prefixes and tags

Object key name prefixes also enable you to categorize storage. However, prefix-based categorization is one-dimensional. Consider the following object key names:

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

These key names have the prefixes `photos/`, `project/projectx/`, and `project/projecty/`. These prefixes enable one-dimensional categorization. That is, everything under a prefix is one category. For example, the prefix `project/projectx` identifies all documents related to project x.

With tagging, you now have another dimension. If you want photo1 in project x category, you can tag the object accordingly.

**Additional benefits**

In addition to data classification, tagging offers benefits such as the following:

- Object tags enable fine-grained access control of permissions. For example, you could grant a user permissions to read-only objects with specific tags.
- Object tags enable fine-grained object lifecycle management in which you can specify a tag-based filter, in addition to a key name prefix, in a lifecycle rule.
- When using Amazon S3 analytics, you can configure filters to group objects together for analysis by object tags, by key name prefix, or by both prefix and tags.
- You can also customize Amazon CloudWatch metrics to display information by specific tag filters. The following sections provide details.

> **Important**
>
> It is acceptable to use tags to label objects containing confidential data, such as personally identifiable information (PII) or protected health information (PHI). However, the tags themselves shouldn't contain any confidential information.

**Adding object tag sets to multiple Amazon S3 object with a single request**

To add object tag sets to more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

The S3 Batch Operations feature tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or REST API. For more information, see the section called “Batch Operations basics”.

For more information about object tags, see Managing object tags.

# API operations related to object tagging

Amazon S3 supports the following API operations that are specifically for object tagging:

### Object API operations

- PUT Object tagging – Replaces tags on an object. You specify tags in the request body. There are two distinct scenarios of object tag management using this API.
  - Object has no tags – Using this API you can add a set of tags to an object (the object has no prior tags).
  - Object has a set of existing tags – To modify the existing tag set, you must first retrieve the existing tag set, modify it on the client side, and then use this API to replace the tag set.

    > **Note**
    >
    > If you send this request with an empty tag set, Amazon S3 deletes the existing tag set on the object. If you use this method, you will be charged for a Tier 1 Request (PUT). For more information, see Amazon S3 Pricing.
    > The DELETE Object tagging request is preferred because it achieves the same result without incurring charges.

- GET Object tagging – Returns the tag set associated with an object. Amazon S3 returns object tags in the response body.
- DELETE Object tagging – Deletes the tag set associated with an object.

### Other API operations that support tagging

- PUT Object and Initiate Multipart Upload– You can specify tags when you create objects. You specify tags using the `x-amz-tagging` request header.
- GET Object – Instead of returning the tag set, Amazon S3 returns the object tag count in the `x-amz-tag-count` header (only if the requester has permissions to read tags) because the header response size is limited to 8 K bytes. If you want to view the tags, you make another request for the GET Object tagging API operation.
- POST Object – You can specify tags in your POST request.

As long as the tags in your request don't exceed the 8 K byte HTTP request header size limit, you can use the `PUT Object` API to create objects with tags. If the tags you specify exceed the header size limit, you can use this POST method in which you include the tags in the body.

PUT Object - Copy – You can specify the `x-amz-tagging-directive` in your request to direct Amazon S3 to either copy (default behavior) the tags or replace tags by a new set of tags provided in the request.

Note the following:

- S3 Object Tagging is strongly consistent. For more information, see Amazon S3 data consistency model.

## Additional configurations

This section explains how object tagging relates to other configurations.

### Object tagging and lifecycle management

In bucket lifecycle configuration, you can specify a filter to select a subset of objects to which the rule applies. You can specify a filter based on the key name prefixes, object tags, or both.

Suppose that you store photos (raw and the finished format) in your Amazon S3 bucket. You might tag these objects as shown following.

```
phototype=raw
or
phototype=finished
```

You might consider archiving the raw photos to S3 Glacier sometime after they are created. You can configure a lifecycle rule with a filter that identifies the subset of objects with the key name prefix (`photos/`) that have a specific tag (`phototype=raw`).

For more information, see Managing your storage lifecycle.

### Object tagging and replication

If you configured Replication on your bucket, Amazon S3 replicates tags, provided you grant Amazon S3 permission to read the tags. For more information, see Setting up live replication.

### Object tagging event notifications

You can set up an Amazon S3 event notification to receive notice when an object tag is added or deleted from an object. The `s3:ObjectTagging:Put` event type notifies you when a tag is PUT on an object or when an existing tag is updated. The `s3:ObjectTagging:Delete` event type notifies you when a tag is removed from an object. For more information, see Enabling event notifications.

For more information about object tagging, see the following topics:

**Topics**

- Tagging and access control policies
- Managing object tags

## Tagging and access control policies

You can also use permissions policies (bucket and user policies) to manage permissions related to object tagging. For policy actions see the following topics:

- Object operations
- Bucket operations

Object tags enable fine-grained access control for managing permissions. You can grant conditional permissions based on object tags. Amazon S3 supports the following condition keys that you can use to grant conditional permissions based on object tags:

- `s3:ExistingObjectTag/<tag-key>` – Use this condition key to verify that an existing object tag has the specific tag key and value.

  > **Note**
  >
  > When granting permissions for the `PUT Object` and `DELETE Object` operations, this condition key is not supported. That is, you cannot create a policy to grant or deny a user permissions to delete or overwrite an object based on its existing tags.

- `s3:RequestObjectTagKeys` – Use this condition key to restrict the tag keys that you want to allow on objects. This is useful when adding tags to objects using the PutObjectTagging and PutObject, and POST object requests.

- `s3:RequestObjectTag/<tag-key>` – Use this condition key to restrict the tag keys and values that you want to allow on objects. This is useful when adding tags to objects using the PutObjectTagging and PutObject, and POST Bucket requests.

For a complete list of Amazon S3 service-specific condition keys, see Bucket policy examples using condition keys. The following permissions policies illustrate how object tagging enables fine grained access permissions management.

**Example 1: Allow a user to read only the objects that have a specific tag and key value**

The following permissions policy limits a user to only reading objects that have the `environment:production` tag key and value. This policy uses the `s3:ExistingObjectTag` condition key to specify the tag key and value.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

**Example 2: Restrict which object tag keys that users can add**

The following permissions policy grants a user permissions to perform the `s3:PutObjectTagging` action, which allows user to add tags to an existing object. The condition uses the `s3:RequestObjectTagKeys` condition key to specify the allowed tag keys, such as `Owner` or `CreationDate`. For more information, see Creating a condition that tests multiple key values in the *IAM User Guide*.

The policy ensures that every tag key specified in the request is an authorized tag key. The `ForAnyValue` qualifier in the condition ensures that at least one of the specified keys must be present in the request.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

**Example 3: Require a specific tag key and value when allowing users to add object tags**

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition requires the user to include a specific tag key (such as *`Project`*) with the value set to *`X`*.

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
```

```
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

## Managing object tags

This section explains how you can manage object tags using the AWS SDKs for Java and .NET or the Amazon S3 console.

Object tagging gives you a way to categorize storage. Each tag is a key-value pair that adheres to the following rules:

- You can associate up to 10 tags with an object. Tags that are associated with an object must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Amazon S3 object tags are internally represented in UTF-16. Note that in UTF-16, characters consume either 1 or 2 character positions.
- The key and values are case sensitive.

For more information about object tags, see Categorizing your storage using tags. For more information about tag restrictions, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

### Using the S3 console

#### To add tags to an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to add tags to.

   You can also optionally navigate to a folder.
3. In the **Objects** list, select the checkbox next to the names of the objects that you want to add tags to.
4. In the **Actions** menu, choose **Edit tags**.
5. Review the objects listed, and choose **Add tags**.
6. Each object tag is a key-value pair. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**.

   You can enter up to 10 tags for an object.
7. Choose **Save changes**.

   Amazon S3 adds the tags to the specified objects.

For more information, see also Viewing object properties in the Amazon S3 console and Uploading objects in this guide.

**Using the AWS SDKs**

Java

The following example shows how to use the AWS SDK for Java to set tags for a new object and retrieve or replace tags for an existing object. For more information about object tagging, see Categorizing your storage using tags. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.util.ArrayList;
```

```
import java.util.List;

public class ManagingObjectTags {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** File path ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create an object, add two new tags, and upload the object to Amazon
 S3.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
 new File(filePath));
            List<Tag> tags = new ArrayList<Tag>();
            tags.add(new Tag("Tag 1", "This is tag 1"));
            tags.add(new Tag("Tag 2", "This is tag 2"));
            putRequest.setTagging(new ObjectTagging(tags));
            PutObjectResult putResult = s3Client.putObject(putRequest);

            // Retrieve the object's tags.
            GetObjectTaggingRequest getTaggingRequest = new
 GetObjectTaggingRequest(bucketName, keyName);
            GetObjectTaggingResult getTagsResult =
 s3Client.getObjectTagging(getTaggingRequest);

            // Replace the object's tags with two new tags.
            List<Tag> newTags = new ArrayList<Tag>();
            newTags.add(new Tag("Tag 3", "This is tag 3"));
            newTags.add(new Tag("Tag 4", "This is tag 4"));
            s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName,
 keyName, new ObjectTagging(newTags)));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following example shows how to use the AWS SDK for .NET to set the tags for a new object and retrieve or replace the tags for an existing object. For more information about object tagging, see Categorizing your storage using tags.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
```

```
            try
            {
                // 1. Put an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>{
                        new Tag { Key = "Keyx1", Value = "Value1"},
                        new Tag { Key = "Keyx2", Value = "Value2" }
                    }
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags.Tagging[i].Key, objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.TagSet = new List<Tag>{
                    new Tag { Key = "Key3", Value = "Value3"},
                    new Tag { Key = "Key4", Value = "Value4" }
                };


                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
```

```
                    Tagging = newTagSet
                };
                PutObjectTaggingResponse response2 = await
 client.PutObjectTaggingAsync(putObjTagsRequest);

                // 4. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest2 = new
 GetObjectTaggingRequest();
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
 client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
 objectTags2.Tagging[i].Key, objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

# Using cost allocation S3 bucket tags

To track the storage cost or other criteria for individual projects or groups of projects, label your Amazon S3 buckets using cost allocation tags. A *cost allocation tag* is a key-value pair that you associate with an S3 bucket. After you activate cost allocation tags, AWS uses the tags to organize your resource costs on your cost allocation report. Cost allocation tags can only be used to label

buckets. For information about tags used for labeling objects, see Categorizing your storage using tags.

The *cost allocation report* lists the AWS usage for your account by product category and linked account user. The report contains the same line items as the detailed billing report (see Understanding your AWS billing and usage reports for Amazon S3) and additional columns for your tag keys.

AWS provides two types of cost allocation tags, an AWS-generated tag and user-defined tags. AWS defines, creates, and applies the AWS-generated `createdBy` tag for you after an Amazon S3 CreateBucket event. You define, create, and apply *user-defined* tags to your S3 bucket.

You must activate both types of tags separately in the Billing and Cost Management console before they can appear in your billing reports. For more information about AWS-generated tags, see AWS-Generated Cost Allocation Tags.

- To create tags in the console, see Viewing the properties for an S3 bucket.
- To create tags using the Amazon S3 API, see PUT Bucket tagging in the *Amazon Simple Storage Service API Reference*.
- To create tags using the AWS CLI, see put-bucket-tagging in the AWS CLI Command Reference.
- For more information about activating tags, see Using cost allocation tags in the *AWS Billing User Guide*.

**User-defined cost allocation tags**

A user-defined cost allocation tag has the following components:

- The tag key. The tag key is the name of the tag. For example, in the tag project/Trinity, project is the key. The tag key is a case-sensitive string that can contain 1 to 128 Unicode characters.

- The tag value. The tag value is a required string. For example, in the tag project/Trinity, Trinity is the value. The tag value is a case-sensitive string that can contain from 0 to 256 Unicode characters.

For details on the allowed characters for user-defined tags and other restrictions, see User-Defined Tag Restrictions in the *AWS Billing User Guide*. For more information about user-defined tags, see User-Defined Cost Allocation Tags in the *AWS Billing User Guide*.

### S3 bucket tags

Each S3 bucket has a tag set. A *tag set* contains all of the tags that are assigned to that bucket. A tag set can contain as many as 50 tags, or it can be empty. Keys must be unique within a tag set, but values in a tag set don't have to be unique. For example, you can have the same value in tag sets named project/Trinity and cost-center/Trinity.

Within a bucket, if you add a tag that has the same key as an existing tag, the new value overwrites the old value.

AWS doesn't apply any semantic meaning to your tags. We interpret tags strictly as character strings.

To add, list, edit, or delete tags, you can use the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 API.

## More Info

- Using Cost Allocation Tags in the *AWS Billing User Guide*
- Understanding your AWS billing and usage reports for Amazon S3
- AWS Billing reports for Amazon S3

# Billing and usage reporting for Amazon S3

**Important**

On May 13, 2024, we started deploying a change to eliminate charges for unauthorized requests that aren't initiated by the bucket owner. After the deployment of this change is completed, bucket owners will never incur request or bandwidth charges for requests that return `AccessDenied` (HTTP `403 Forbidden`) errors when these requests are initiated from outside of their individual AWS account or AWS organization. For more information on a full list of HTTP 3XX and 4XX status codes that won't be billed, see Billing for Amazon S3 error responses. This billing change requires no updates to your applications and applies to all S3 buckets. When deployment of this change is completed in all AWS Regions, we'll update our documentation.

When using Amazon S3, you don't have to pay any upfront fees or commit to how much content you'll store. Like other AWS services, you pay as you go and pay only for what you use.

AWS provides the following reports for Amazon S3:

- **Billing reports** – Multiple reports that provide high-level views of all of the activity for the AWS services that you're using, including Amazon S3. AWS always bills the owner of the S3 bucket for Amazon S3 fees, unless the bucket was created as a Requester Pays bucket. For more information about Requester Pays, see Using Requester Pays buckets for storage transfers and usage. For more information about billing reports, see AWS Billing reports for Amazon S3.
- **Usage report** – A summary of activity for a specific service, aggregated by hour, day, or month. You can choose which usage type and operation to include. You can also choose how the data is aggregated. For more information, see AWS usage report for Amazon S3.

The following topics provide information about billing and usage reporting for Amazon S3.

**Topics**



## AWS Billing reports for Amazon S3

Your monthly bill from AWS separates your usage information and cost by AWS service and function. There are several AWS Billing reports available: the monthly report, the cost allocation report, and detailed billing reports. For information about how to see your billing reports, see Viewing Your Bill in the *AWS Billing User Guide*.

To track your AWS usage and provide estimated charges associated with your account, you can set up AWS Cost and Usage Reports. For more information, see What are AWSCost and Usage Reports? in the *AWS Data Exports Guide*.

You can also download a usage report that gives more detail about your Amazon S3 storage usage than the billing reports. For more information, see AWS usage report for Amazon S3.

The following table lists the charges associated with Amazon S3 usage.

### Amazon S3 usage charges

| Charge | Comments |
| --- | --- |
| Storage | You pay for storing objects in your S3 buckets. The rate you're charged depends on your objects' size, how long you stored the objects during the month, and the storage class. Amazon S3 offers the following storage classes: S3 Standard, S3 Express One Zone, S3 Intelligent-Tiering, S3 Standard-IA (IA for infrequent access), S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or Reduced Redundancy Storage (RRS). For more information about storage classes, see Using Amazon S3 storage classes.<br><br>Be aware that if you have S3 Versioning enabled, you're charged for each version of an object that is retained. For more information about versioning, see How S3 Versioning works. |
| Monitoring and automation | You pay a monthly monitoring and automation fee per object stored in the S3 Intelligent-Tiering storage class to monitor access patterns and move objects between access tiers in S3 Intelligent-Tiering. |
| Requests | You pay for requests, for example, GET requests, made against your S3 buckets and objects. This includes lifecycle requests. The rates for requests depend on what kind of request you're making. For information about request pricing, see Amazon S3 Pricing. |

| Charge | Comments |
| --- | --- |
| Retrievals | You pay for retrieving objects that are stored in S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive storage. |
| Early deletes | If you delete an object stored in S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage before the minimum storage commitment has passed, you pay an early deletion fee for that object. |
| Storage management | You pay for the storage management features (Amazon S3 Inventory, analytics, and object tagging) that are enabled on your account's buckets. |
| Bandwidth | You pay for all bandwidth into and out of Amazon S3, except for the following:<br>• Data transferred in from the internet<br>• Data transferred out to an Amazon Elastic Compute Cloud (Amazon EC2) instance, when the instance is in the same AWS Region as the S3 bucket<br>• Data transferred out to Amazon CloudFront (CloudFront)<br>You also pay a fee for any data transferred by using Amazon S3 Transfer Acceleration. |

For detailed information about Amazon S3 usage charges for storage, data transfer, and services, see Amazon S3 Pricing and the Amazon S3 FAQs.

For information about understanding the codes and abbreviations used in the billing and usage reports for Amazon S3, see Understanding your AWS billing and usage reports for Amazon S3.

### More info

- AWS usage report for Amazon S3
- Using cost allocation S3 bucket tags
- AWS Billing and Cost Management
- Amazon S3 Pricing

## AWS usage report for Amazon S3

When you download a usage report, you can choose to aggregate usage data by hour, day, or month. The Amazon S3 usage report lists operations by usage type and AWS Region. For more detailed reports about your Amazon S3 storage usage, download dynamically generated AWS usage reports. You can choose which usage type, operation, and time period to include. You can also choose how the data is aggregated. For more information about usage reports, see AWS Usage Reportin the *AWS Data Exports User Guide*.

The Amazon S3 usage report includes the following information:

- **Service** – Amazon S3
- **Operation** – The operation performed on your bucket or object. For a detailed explanation of Amazon S3 operations, see Tracking Operations in Your Usage Reports.
- **UsageType** – One of the following values:
  - A code that identifies the type of storage
  - A code that identifies the type of request
  - A code that identifies the type of retrieval
  - A code that identifies the type of data transfer
  - A code that identifies early deletions from S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-Infrequent Access (S3 One Zone-IA), S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage
  - `StorageObjectCount` – The count of objects stored within a given bucket

For a detailed explanation of Amazon S3 usage types, see Understanding your AWS billing and usage reports for Amazon S3.

- **Resource** – The name of the bucket associated with the listed usage.
- **StartTime** – Start time of the day that the usage applies to, in Coordinated Universal Time (UTC).
- **EndTime** – End time of the day that the usage applies to, in Coordinated Universal Time (UTC).
- **UsageValue** – One of the following volume values. The typical unit of measurement for data is gigabytes (GB). However, depending on the service and the report, terabytes (TB) might appear instead.
  - The number of requests during the specified time period
  - The amount of data transferred
  - The amount of data stored in a given hour
  - The amount of data associated with restorations from S3 Standard-IA, S3 One Zone-IA, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage

> **Tip**
>
> For detailed information about every request that Amazon S3 receives for your objects, turn on server access logging for your buckets. For more information, see Logging requests with server access logging.

You can download a usage report as an XML or a comma-separated values (CSV) file. The following is an example CSV usage report opened in a spreadsheet application.

| Service | Operation | UsageType | Resource | StartTime | EndTime | UsageValue |
|---|---|---|---|---|---|---|
| AmazonS3 | HeadBucket | USW2-C3DataTransfer-Out-Bytes | admin-created3 | 6/1/2017 0:00 | 7/1/2017 0:00 | 15309 |
| AmazonS3 | PutObject | USW2-C3DataTransfer-In-Bytes | admin-created3 | 6/1/2017 0:00 | 7/1/2017 0:00 | 19062 |
| AmazonS3 | HeadBucket | USW2-Requests-Tier2 | admin-created3 | 6/1/2017 0:00 | 7/1/2017 0:00 | 68 |
| AmazonS3 | PutObjectForRepl | USW1-Requests-SIA-Tier1 | ca-example-bucket | 6/1/2017 0:00 | 7/1/2017 0:00 | 178294 |
| AmazonS3 | PutObjectForRepl | USW1-USW2-AWS-In-Bytes | ca-example-bucket | 6/1/2017 0:00 | 7/1/2017 0:00 | 387929083 |
| AmazonS3 | GetObjectForRepl | USW2-Requests-NoCharge | admin-created3 | 6/1/2017 0:00 | 7/1/2017 0:00 | 108 |
| AmazonS3 | GetObjectForRepl | USW2-USW1-AWS-Out-Bytes | my-test-bucket-bash | 6/1/2017 0:00 | 7/1/2017 0:00 | 387910021 |

For more information, see Understanding your AWS billing and usage reports for Amazon S3.

## Downloading the AWS Usage Report

You can download a usage report as an XML or a CSV file.

**To download the usage report**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the title bar, choose your username or account ID, and then choose **Billing and Cost Management**.
3. In the navigation pane, choose **Cost and usage reports**.
4. Under AWS Usage Report, choose **Create a Usage Report**.
5. On the **Download usage report** page, choose the following settings:
   - **Services** – Choose **Amazon Simple Storage Service**.
   - **Usage Types** – For a detailed explanation of Amazon S3 usage types, see Understanding your AWS billing and usage reports for Amazon S3.
   - **Operation** – For a detailed explanation of Amazon S3 operations, see Tracking Operations in Your Usage Reports.
   - **Time Period** – The time period that you want the report to cover.
   - **Report Granularity** – Whether you want the report to include subtotals by the hour, by the day, or by the month.
6. Choose **Download**, choose the download format (**XML Report** or **CSV Report**), and then follow the prompts to open or save the report.

### More info

- Understanding your AWS billing and usage reports for Amazon S3
- AWS Billing reports for Amazon S3

## Understanding your AWS billing and usage reports for Amazon S3

**Important**

On May 13, 2024, we started deploying a change to eliminate charges for unauthorized requests that aren't initiated by the bucket owner. After the deployment of this change is completed, bucket owners will never incur request or bandwidth charges for requests that return `AccessDenied` (HTTP `403 Forbidden`) errors when these requests are initiated from outside of their individual AWS account or AWS organization. For more information

on a full list of HTTP 3XX and 4XX status codes that won't be billed, see Billing for Amazon S3 error responses. This billing change requires no updates to your applications and applies to all S3 buckets. When deployment of this change is completed in all AWS Regions, we'll update our documentation.

Amazon S3 billing and usage reports use codes and abbreviations. For usage types in the table that follows, replace *region*, *region1*, and *region2* with abbreviations from this list:

- **APE1:** Asia Pacific (Hong Kong)
- **APN1:** Asia Pacific (Tokyo)
- **APN2:** Asia Pacific (Seoul)
- **APN3:** Asia Pacific (Osaka)
- **APS1:** Asia Pacific (Singapore)
- **APS2:** Asia Pacific (Sydney)
- **APS3:** Asia Pacific (Mumbai)
- **APS4:** Asia Pacific (Jakarta)
- **APS5:** Asia Pacific (Hyderabad)
- **APS6:** Asia Pacific (Melbourne)
- **CAN1:** Canada (Central)
- **CAN2:** Canada West (Calgary)
- **CNN1:** China (Beijing)
- **CNW1:** China (Ningxia)
- **AFS1:** Africa (Cape Town)
- **EUC2:** Europe (Zurich)
- **EUN1:** Europe (Stockholm)
- **EUS2:** Europe (Spain)
- **EUC1:** Europe (Frankfurt)
- **EU:** Europe (Ireland)
- **EUS1:** Europe (Milan)
- **EUW2:** Europe (London)
- **EUW3:** Europe (Paris)

- **ILC1:** Israel (Tel Aviv)
- **MEC1:** Middle East (UAE)
- **MES1:** Middle East (Bahrain)
- **SAE1:** South America (São Paulo)
- **UGW1:** AWS GovCloud (US-West)
- **UGE1:** AWS GovCloud (US-East)
- **USE1 (or no prefix):** US East (N. Virginia)
- **USE2:** US East (Ohio)
- **USW1:** US West (N. California)
- **USW2:** US West (Oregon)

For S3 Multi-Region Access Points usage types in the table that follows, replace *`regiongroup1`* and *`regiongroup2`* with abbreviations from this list:

- **AP:** Asia Pacific
- **AU:** Australia
- **EU:** Europe
- **IN:** India
- **NA:** North America
- **SA:** South America

Region groups are geographical groupings of several AWS Regions. For more information, see Regions and Availability Zones. For information about pricing by AWS Region, see Amazon S3 Pricing.

The first column in the following table lists usage types that appear in your billing and usage reports. The typical unit of measurement for data is gigabytes (GB). However, depending on the service and the report, terabytes (TB) might appear instead.

### Usage Types

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region1*-*region2*-AWS-In-ABytes | GB | Hourly | The amount of accelerated data transferred to *region1* from *region2* |
| *region1*-*region2*-AWS-In-ABytes-T1 | GB | Hourly | The amount of T1 accelerated data transferred to *region1* from *region2*, where T1 refers to CloudFront requests to points of presence (POPs) in the United States, Europe, and Japan |
| *region1*-*region2*-AWS-In-ABytes-T2 | GB | Hourly | The amount of T2 accelerated data transferred to *region1* from *region2*, where T2 refers to CloudFront requests to POPs in all other AWS edge locations |
| *region1*-*region2*-AWS-In-Bytes | GB | Hourly | The amount of data transferred to *region1* from *region2* |
| *region1*-*region2*-AWS-Out-ABytes | GB | Hourly | The amount of accelerated data transferred from *region1* to *region2* |
| *region1*-*region2*-AWS-Out-ABytes-T1 | GB | Hourly | The amount of T1 accelerated data transferred from |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| | | | *region1* to *region2*, where T1 refers to CloudFront requests to POPs in the United States, Europe, and Japan |
| *region1*-*region2*-AWS-Out-ABytes-T2 | GB | Hourly | The amount of T2 accelerated data transferred from *region1* to *region2*, where T2 refers to CloudFront requests to POPs in all other AWS edge locations |
| *region1*-*region2*-AWS-Out-Bytes | GB | Hourly | The amount of data transferred from *region1* to *region2* |
| *region*-BatchOperations-Jobs | Count | Hourly | The number of S3 Batch Operations jobs performed |
| *region*-BatchOperations-Objects | Count | Hourly | The number of object operations performed by S3 Batch Operations |
| *region*-Bulk-Retrieval-Bytes | GB | Hourly | The amount of data retrieved with Bulk S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive requests |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-BytesDeleted-GDA | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Glacier Deep Archive storage |
| *region*-BytesDeleted-GIR | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Glacier Instant Retrieval storage. |
| *region*-BytesDeleted-GLACIER | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Glacier Flexible Retrieval storage |
| *region*-BytesDeleted-INT | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Intelligent-Tiering storage |
| *region*-BytesDeleted-RRS | GB | Monthly | The amount of data deleted by a DeleteObject operation from Reduced Redundancy Storage (RRS) storage |
| *region*-BytesDeleted-SIA | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Standard-IA storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-BytesDeleted-STANDARD | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 Standard storage |
| *region*-BytesDeleted-ZIA | GB | Monthly | The amount of data deleted by a DeleteObject operation from S3 One Zone-IA storage |
| *region*-C3DataTransfer-In-Bytes | GB | Hourly | The amount of data transferred into Amazon S3 from Amazon EC2 within the same AWS Region |
| *region*-C3DataTransfer-Out-Bytes | GB | Hourly | The amount of data transferred from Amazon S3 to Amazon EC2 within the same AWS Region |
| *region*-CloudFront-In-Bytes | GB | Hourly | The amount of data transferred into an AWS Region from a CloudFront distribution |
| *region*-CloudFront-Out-Bytes | GB | Hourly | The amount of data transferred from an AWS Region to a CloudFront distribution |
| *region*-DataTransfer-In-Bytes | GB | Hourly | The amount of data transferred into Amazon S3 from the internet |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-DataTransfer-Out-Bytes | GB | Hourly | The amount of data transferred from Amazon S3 to the internet[1] |
| *region*-DataTransfer-Regional-Bytes | GB | Hourly | The amount of data transferred from Amazon S3 to AWS resources within the same AWS Region |
| *region*-EarlyDelete-ByteHrs | GB-Hours | Hourly | Prorated storage usage for objects deleted from, S3 Glacier Flexible Retrieval storage before the 90-day minimum commitment ended[2] |
| *region*-EarlyDelete-GDA | GB-Hours | Hourly | Prorated storage usage for objects deleted from S3 Glacier Deep Archive storage before the 180-day minimum commitment ended [2] |
| *region*-EarlyDelete-GIR | GB-Hours | Hourly | Prorated storage usage for objects deleted from S3 Glacier Instant Retrieval before the 90-day minimum commitment ended. |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-EarlyDelete-GIR-SmObjects | GB-Hours | Hourly | Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 Glacier Instant Retrieval before the 90-day minimum commitment ended. |
| *region*-EarlyDelete-SIA | GB-Hours | Hourly | Prorated storage usage for objects deleted from S3 Standard-IA before the 30-day minimum commitment ended[3] |
| *region*-EarlyDelete-SIA-SmObjects | GB-Hours | Hourly | Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 Standard-IA before the 30-day minimum commitment ended[3] |
| *region*-EarlyDelete-ZIA | GB-Hours | Hourly | Prorated storage usage for objects deleted from S3 One Zone-IA before the 30-day minimum commitment ended[3] |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-EarlyDelete-ZIA-SmObjects | GB-Hours | Hourly | Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 One Zone-IA before the 30-day minimum commitment ended[3] |
| *region*-Expedited-Retrieval-Bytes | GB | Hourly | The amount of data retrieved with Expedited S3 Glacier Flexible Retrieval requests |
| *region*-Inventory-Objects Listed | Objects | Hourly | The number of objects listed for an object group (objects are grouped by bucket or prefix) with an inventory list |
| *region*-Monitoring-Automation-INT | Objects | Hourly | The number of unique objects monitored and auto-tiered in the S3 Intelligent-Tiering storage class |
| *region*-MRAP-Out-Bytes | GB | Hourly | The amount of data transferred through an S3 Multi-Region Access Points endpoint out of buckets in a Region (MRAP data routing pricing). |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-MRAP-In-Bytes | GB | Hourly | The amount of data transferred through an S3 Multi-Region Access Points endpoint out of buckets in a Region (MRAP data routing pricing). |
| *regiongroup1-regiongroup2-*-MRAP-Out-Bytes | GB | Hourly | The amount of data transferred through an S3 Multi-Region Access Points endpoint from a bucket in *regiongroup1* to a client in *regiongroup2* located outside of the AWS network. |
| *regiongroup1-regiongroup2-*-MRAP-In-Bytes | GB | Hourly | The amount of data transferred through an S3 Multi-Region Access Points endpoint to a bucket in *regiongroup1* from a client in *regiongroup2* located outside of the AWS network. |
| *region*-OverwriteBytes-Copy-GDA | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Glacier Deep Archive storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-OverwriteBytes-Copy-GIR | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Glacier Instant Retrieval storage. |
| *region*-OverwriteBytes-Copy-GLACIER | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Glacier Flexible Retrieval storage |
| *region*-OverwriteBytes-Copy-INT | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Intelligent-Tiering storage |
| *region*-OverwriteBytes-Copy-RRS | GB | Monthly | The amount of data overwritten by a CopyObject operation from Reduced Redundancy Storage (RRS) storage |
| *region*-OverwriteBytes-Copy-SIA | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Standard-IA storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-OverwriteBytes-Copy-STANDARD | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 Standard storage |
| *region*-OverwriteBytes-Copy-ZIA | GB | Monthly | The amount of data overwritten by a CopyObject operation from S3 One Zone-IA storage |
| *region*-OverwriteBytes-Put-GDA | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Glacier Deep Archive storage |
| *region*-OverwriteBytes-Put-GIR | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Glacier Instant Retrieval storage. |
| *region*-OverwriteBytes-Put-GLACIER | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Glacier Flexible Retrieval storage |
| *region*-OverwriteBytes-Put-INT | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Intelligent-Tiering storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-OverwriteBytes-Put-RRS | GB | Monthly | The amount of data overwritten by a PutObject operation from Reduced Redundancy Storage (RRS) storage |
| *region*-OverwriteBytes-Put-SIA | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Standard-IA storage |
| *region*-OverwriteBytes-Put-STANDARD | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 Standard storage |
| *region*-OverwriteBytes-Put-ZIA | GB | Monthly | The amount of data overwritten by a PutObject operation from S3 One Zone-IA storage |

| Usage Type | Units | Granularity | Description |
| --- | --- | --- | --- |
| *region1*-*region2*-S3RTC-In-Bytes | GB | Monthly | The amount of data transferred for S3 Replication Time Control (S3 RTC) from *region2* to *region1* by the PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , and WriteACLReplTime operations |
| *region1*-*region2*-S3RTC-Out-Bytes | GB | Monthly | The amount of data transferred for S3 Replication Time Control (S3 RTC) from *region1* to *region2* by the PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , and WriteACLReplTime operations |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Requests-GDA-Tier1 | Count | Hourly | The number of PUT, COPY, POST, CreateMultipartUpload , UploadPart , or CompleteMultipartUpload requests on S3 Glacier Deep Archive objects [6] |
| *region*-Requests-GDA-Tier2 | Count | Hourly | The number of GET and HEAD requests on S3 Glacier Deep Archive objects |
| *region*-Requests-GDA-Tier3 | Count | Hourly | The number of S3 Glacier Deep Archive standard restore requests |
| *region*-Requests-GDA-Tier5 | Count | Hourly | The number of Bulk S3 Glacier Deep Archive restore requests |
| *region*-Requests-GIR-Tier1 | Count | Hourly | The number of PUT, COPY, or POST requests on S3 Glacier Instant Retrieval objects. |
| *region*-Requests-GIR-Tier2 | Count | Hourly | The number of GET and all other non-S3 Glacier Instant Retrieval-Tier1 requests on S3 Glacier Instant Retrieval objects. |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Requests-GLACIER-Tier1 | Count | Hourly | The number of PUT, COPY, POST, CreateMultipartUpload , UploadPart , or CompleteMultipartUpload requests on S3 Glacier Flexible Retrieval objects [6] |
| *region*-Requests-GLACIER-Tier2 | Count | Hourly | The number of GET and all other requests not listed on S3 Glacier Flexible Retrieval objects |
| *region*-Requests-INT-Tier1 | Count | Hourly | The number of PUT, COPY, or POST requests on S3 Intelligent-Tiering objects |
| *region*-Requests-INT-Tier2 | Count | Hourly | The number of GET and all other non-Tier1 requests for S3 Intelligent-Tiering objects |
| *region*-Requests-SIA-Tier1 | Count | Hourly | The number of PUT, COPY, or POST requests on S3 Standard-IA objects |
| *region*-Requests-SIA-Tier2 | Count | Hourly | The number of GET and all other non-S3 Glacier Instant Retrieval -Tier1 requests on S3 Standard-IA objects |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Requests-Tier1 | Count | Hourly | The number of PUT, COPY, or POST requests for S3 Standard, RRS, and tags, plus LIST requests for all buckets and objects |
| *region*-Requests-Tier2 | Count | Hourly | The number of GET and all other non-Tier1 requests |
| *region*-Requests-Tier3 | Count | Hourly | The number of lifecycle requests to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive and standard S3 Glacier Flexible Retrieval restore requests |
| *region*-Requests-Tier4 | Count | Hourly | The number of lifecycle transitions to S3 Glacier Instant Retrieval, S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage |
| *region*-Requests-Tier5 | Count | Hourly | The number of Bulk S3 Glacier Flexible Retrieval restore requests |
| *region*-Requests-Tier6 | Count | Hourly | The number of Expedited S3 Glacier Flexible Retrieval restore requests |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Requests-Tier8 | Count | Hourly | The number of S3 Access Grants requests |
| *region*-Requests-XZ-Tier1 | Count | Hourly | The number of PUT or COPY requests on S3 Express One Zone objects |
| *region*-Requests-XZ-Tier2 | Count | Hourly | The number of GET and all other non-S3 Express One Zone-Tier1 requests on S3 Express One Zone objects |
| *region*-Requests-ZIA-Tier1 | Count | Hourly | The number of PUT, COPY, or POST requests on S3 One Zone-IA objects |
| *region*-Requests-ZIA-Tier2 | Count | Hourly | The number of GET and all other non-S3 One Zone-IA-Tier1 requests on S3 One Zone-IA objects |
| *region*-Retrieval-GIR | GB | Hourly | The amount of data retrieved from S3 Glacier Instant Retrieval storage. |
| *region*-Retrieval-SIA | GB | Hourly | The amount of data retrieved from S3 Standard-IA storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Retrieval-XZ | GB | Hourly | The portion of the data that exceeds 512 KB in a given retrieval request (PUT or COPY) with S3 Express One Zone storage |
| *region*-Retrieval-ZIA | GB | Hourly | The amount of data retrieved from S3 One Zone-IA storage |
| *region*-S3DSSE-In-Bytes | GB | Monthly | The amount of data dual-encrypted by Amazon S3 |
| *region*-S3DSSE-Out-Bytes | GB | Monthly | The amount of dual-encrypted data decrypted by Amazon S3 |
| *region*-S3G-DataTransfer-In-Bytes | GB | Hourly | The amount of data transferred into Amazon S3 to restore objects from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage |
| *region*-S3G-DataTransfer-Out-Bytes | GB | Hourly | The amount of data transferred from Amazon S3 to transition objects to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Select-Returned-Bytes | GB | Hourly | The amount of data returned with Select requests from S3 Standard storage |
| *region*-Select-Returned-GIR-Bytes | GB | Hourly | The amount of data returned with Select requests from S3 Glacier Instant Retrieval storage. |
| *region*-Select-Returned-INT-Bytes | GB | Hourly | The amount of data returned with Select requests from S3 Intelligent-Tiering storage |
| *region*-Select-Returned-SIA-Bytes | GB | Hourly | The amount of data returned with Select requests from S3 Standard-IA storage |
| *region*-Select-Returned-ZIA-Bytes | GB | Hourly | The amount of data returned with Select requests from S3 One Zone-IA storage |
| *region*-Select-Scanned-Bytes | GB | Hourly | The amount of data scanned with Select requests from S3 Standard storage |
| *region*-Select-Scanned-GIR-Bytes | GB | Hourly | The amount of data scanned with Select requests from S3 Glacier Instant Retrieval storage. |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-Select-Scanned-INT-Bytes | GB | Hourly | The amount of data scanned with Select requests from S3 Intelligent-Tiering storage |
| *region*-Select-Scanned-SIA-Bytes | GB | Hourly | The amount of data scanned with Select requests from S3 Standard-IA storage |
| *region*-Select-Scanned-ZIA-Bytes | GB | Hourly | The amount of data scanned with Select requests from S3 One Zone-IA storage |
| *region*-Standard-Retrieval-Bytes | GB | Hourly | The amount of data retrieved with standard S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive requests |
| *region*-StorageAnalytics-ObjCount | Objects | Hourly | The number of unique objects monitored in each Storage Class Analysis configuration. |
| *region*-StorageLens-ObjCount | Objects | Daily | The number of unique objects in each S3 Storage Lens dashboard that are tracked by S3 Storage Lens advanced metrics and recommendations. |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-StorageLensFreeTier-ObjCount | Objects | Daily | The number of unique objects in each S3 Storage Lens dashboard that are tracked by S3 Storage Lens usage metrics. |
| StorageObjectCount | Count | Daily | The number of objects stored within a given bucket |
| *region*-TagStorage-TagHrs | Tag-Hours | Daily | The total of tags on all objects in the bucket reported by hour |
| *region*-TimedStorage-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Standard storage |
| *region*-TimedStorage-GDA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Glacier Deep Archive storage |
| *region*-TimedStorage-GDA-Staging | GB-Month | Daily | The number of GB-months that data was stored in S3 Glacier Deep Archive staging storage |
| *region*-TimedStorage-GIR-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Glacier Instant Retrieval storage. |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-TimedStorage-GIR-SmObjects | GB-Month | Daily | The number of GB-months that small objects (smaller than 128 KB) were stored in S3 Glacier Instant Retrieval storage. |
| *region*-TimedStorage-GlacierByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Glacier Flexible Retrieval storage |
| *region*-TimedStorage-GlacierStaging | GB-Month | Daily | The number of GB-months that data was stored in S3 Glacier Flexible Retrieval staging storage |
| *region*-TimedStorage-INT-FA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in the Frequent Access tier of S3 Intelligent-Tiering storage[5] |
| *region*-TimedStorage-INT-IA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in the Infrequent Access tier of S3 Intelligent-Tiering storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-TimedStorage-INT-AA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in the Archive Access tier of S3 Intelligent-Tiering storage |
| *region*-TimedStorage-INT-AIA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in the Archive Instant Access tier of S3 Intelligent-Tiering storage |
| *region*-TimedStorage-INT-DAA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in the Deep Archive Access tier of S3 Intelligent-Tiering storage |
| *region*-TimedStorage-RRS-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in Reduced Redundancy Storage (RRS) storage |
| *region*-TimedStorage-SIA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Standard-IA storage |

| Usage Type | Units | Granularity | Description |
|---|---|---|---|
| *region*-TimedStorage-SIA-SmObjects | GB-Month | Daily | The number of GB-months that small objects (smaller than 128 KB) were stored in S3 Standard-IA storage[4] |
| *region*-TimedStorage-XZ-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 Express One Zone storage |
| *region*-TimedStorage-ZIA-ByteHrs | GB-Month | Daily | The number of GB-months that data was stored in S3 One Zone-IA storage |
| *region*-TimedStorage-ZIA-SmObjects | GB-Month | Daily | The number of GB-months that small objects (smaller than 128 KB) were stored in S3 One Zone-IA storage |
| *region*-Upload-XZ | GB | Hourly | The amount of data that exceeds 512 KB in a given upload request (PUT or COPY) with S3 Express One Zone |

**Notes**

1. If you terminate a transfer before completion, the amount of data that is transferred might exceed the amount of data that your application receives. This discrepancy can occur because a transfer termination request cannot be executed instantaneously, and some amount of data

might be in transit, pending execution of the termination request. This data in transit is billed as data transferred "out."

2. When objects that are archived to the S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage class are deleted, overwritten, or transitioned to a different storage class before the minimum storage commitment has passed, which is 90 days for S3 Glacier Instant Retrieval and S3 Glacier Flexible Retrieval, or 180 days for S3 Glacier Deep Archive, there is a prorated charge per gigabyte for the remaining days.
3. For objects that are in S3 Standard-IA or S3 One Zone-IA storage, when they are deleted, overwritten, or transitioned to a different storage class before 30 days, there is a prorated charge per gigabyte for the remaining days.
4. For small objects (smaller than 128 KB) that are in S3 Standard-IA or S3 One Zone-IA storage, when they are deleted, overwritten, or transitioned to a different storage class before 30 days, there is a prorated charge per gigabyte for the remaining days.
5. There is no minimum billable object size for objects in the S3 Intelligent-Tiering storage class. Objects that are smaller than 128 KB are not monitored or eligible for auto-tiering. Smaller objects are always stored in the S3 Intelligent-Tiering Frequent Access tier.
6. When you initiate a `CreateMultipartUpload`, `UploadPart`, or `UploadPartCopy` request to either the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, requests are billed at S3 Standard request rates until you complete the multipart upload. After the upload is completed, the single `CompleteMultipartUpload` request is billed at the PUT rate for the destination S3 Glacier storage. In-progress multipart upload parts for a PUT to the S3 Glacier Flexible Retrieval storage class are billed as S3 Glacier Flexible Retrieval Staging Storage at S3 Standard storage rates until the upload is completed. Similarly, in-progress multipart upload parts for a PUT to the S3 Glacier Deep Archive storage class are billed as S3 Glacier Deep Archive Staging Storage at S3 Standard storage rates until the upload is completed.
7. S3 Express One Zone applies a flat per-request charge for request sizes up to 512 KB. An additional per GB charge is applied for PUT requests and GET requests for the portion of request greater than 512 KB.
8. For information about supported features for S3 Express One Zone storage class, see Amazon S3 features not supported by S3 Express One Zone.
9. Usage types with units that are billed in GB are calculated in bytes in the usage reports.
10. A GB-Month is derived by taking the total number of GB-hours, aggregating these over the course of a month, and then dividing by the number of hours in that month. To learn more see, Frequently Asked Questions: How will I be charged and billed for my use of Amazon S3?

**Note**

In general, S3 bucket owners are billed for requests with HTTP `200 OK` successful responses and HTTP 4XX client error responses. Bucket owners aren't billed for HTTP 5XX server error responses, such as HTTP `503 Slow Down` errors. For more information on S3 error codes under HTTP 3XX and 4XX status codes that aren't billed, see Billing for Amazon S3 error responses. For more information about billing charges if your bucket is configured as a Requester Pays bucket, see How Requester Pays charges work.

### Tracking Operations in Your Usage Reports

Operations describe the action taken on your AWS object or bucket by the specified usage type. Operations are indicated by self-explanatory codes, such as `PutObject` or `ListBucket`. To see which actions on your bucket generated a specific type of usage, use these codes. When you create a usage report, you can choose to include **All Operations**, or a specific operation, for example, `GetObject`, to report on.

### More info

- AWS usage report for Amazon S3
- AWS Billing reports for Amazon S3
- Amazon S3 Pricing
- Amazon S3 FAQs

## Billing for Amazon S3 error responses

**Important**

On May 13, 2024, we started deploying a change to eliminate charges for unauthorized requests that aren't initiated by the bucket owner. After the deployment of this change is completed, bucket owners will never incur request or bandwidth charges for requests that return `AccessDenied` (HTTP `403 Forbidden`) errors when these requests are initiated from outside of their individual AWS account or AWS organization. The current page shows a full list of HTTP 3XX and 4XX status codes that won't be billed. This billing change

requires no updates to your applications and applies to all S3 buckets. When deployment of this change is completed in all AWS Regions, we'll update our documentation.

In general, S3 bucket owners are billed for requests with HTTP `200 OK` successful responses and HTTP `4XX` client error responses. Bucket owners aren't billed for HTTP `5XX` server error responses, such as HTTP `503 Slow Down` errors. For more information about billing charges if your bucket is configured as a Requester Pays bucket, see How Requester Pays charges work.

The following table lists specific error codes under HTTP `3XX` and `4XX` status codes that aren't billed. For buckets configured with website hosting, applicable request and other charges will still apply when S3 returns a custom error document or for custom redirects.

**Note**

For `AccessDenied` (HTTP `403 Forbidden`), S3 doesn't charge the bucket owner when the request is initiated outside of the bucket owner's individual AWS account or the bucket owner's AWS organization.

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| 301 Moved Permanently | PermanentRedirect | The bucket that you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint. | |
| | PermanentRedirectControlError | The API operation you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint. | |

| HTTP status code | Error code | Description of error code |
|---|---|---|
| 307 Temporary Redirect | TemporaryRedirect | You are being redirected to the bucket while the Domain Name System (DNS) server is being updated. |
| 400 Bad Request | AuthorizationHeaderMalformed | The authorization header that you provided is not valid. |
| | AuthorizationQueryParametersError | The authorization query parameters that you provided are not valid. |
| | ExpiredToken | The provided token has expired. |
| | IllegalLocationConstraintException | You are trying to access a bucket from a different Region than where the bucket exists. To avoid this error, use the `--region` option. For example: `aws s3 cp awsexample.txt s3://amzn-s3-demo-bucket/ --region ap-east-1`. |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | InvalidArgument | This error might occur for the following reasons:<br>• The specified argument was not valid.<br>• The request was missing a required header.<br>• The specified argument was incomplete or in the wrong format.<br>• The specified argument must have a length greater than or equal to 3. | |
| | InvalidDigest | The Content-MD5 or checksum value that you specified is not valid. | |
| | InvalidEncryptionAlgorithmError | The encryption request that you specified is not valid. The valid value is AES256. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | InvalidRequest | This error might occur for the following reasons:<br>• The request is using the wrong signature version. Use `AWS4-HMAC-SHA256` (Signature Version 4).<br>• An access point can be created only for an existing bucket.<br>• The access point is not in a state where it can be deleted.<br>• An access point can be listed only for an existing bucket.<br>• The next token is not valid.<br>• At least one action must be specified in a lifecycle rule.<br>• At least one lifecycle rule must be specified.<br>• | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | | The number of lifecycle rules must not exceed the allowed limit of 1000 rules.<br>• The range for the `MaxResults` parameter is not valid.<br>• SOAP requests must be made over an HTTPS connection.<br>• Amazon S3 Transfer Acceleration is not supported for buckets with non-DNS compliant names.<br>• Amazon S3 Transfer Acceleration is not supported for buckets with periods (.) in their names.<br>• The Amazon S3 Transfer Acceleration endpoint supports only virtual style requests.<br>• Amazon S3 Transfer Acceleration is not | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | | configured on this bucket.<br>• Amazon S3 Transfer Acceleration is disabled on this bucket.<br>• Amazon S3 Transfer Acceleration is not supported on this bucket. For assistance, contact AWS Support.<br>• Amazon S3 Transfer Acceleration cannot be enabled on this bucket. For assistance, contact AWS Support.<br>• Conflicting values provided in HTTP headers and query par ameters.<br>• Conflicting values provided in HTTP headers and POST form fields.<br>• CopyObject request made on objects larger than 5GB in size. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | InvalidSOAPRequest | The SOAP request body is not valid. | |
| | InvalidStorageClass | The storage class that you specified is not valid. | |
| | InvalidTag | Your request contains tag input that is not valid. For example, your request might contain duplicate keys, keys or values that are too long, or system tags. | |
| | InvalidToken | The provided token is malformed or otherwise not valid. | |
| | InvalidURI | The specified URI couldn't be parsed. | |
| | KeyTooLongError | Your key is too long. | |
| | MalformedACLError | The ACL that you provided was not well formed or did not validate against our published schema. | |
| | MalformedPOSTRequest | The body of your POST request is not well-formed multipart/form-data. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | MalformedXML | The XML that you provided was not well formed or did not validate against our published schema. | |
| | MaxPostPreDataLengthExceededError | Your POST request fields preceding the upload file were too large. | |
| | MetadataTooLarge | Your metadata headers exceed the maximum allowed metadata size. | |
| | MissingRequestBodyError | You sent an empty XML document as a request. | |
| | MissingSecurityHeader | Your request is missing a required header. | |
| | NoLoggingStatusForKey | There is no such thing as a logging status subresource for a key. | |
| | RequestHeaderSectionTooLarge | The request header and query parameters used to make the request exceed the maximum allowed sizes | |
| | UnexpectedContent | This request contains unsupported content. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | UserKeyMustBeSpecified | The bucket POST request must contain the specified field name. If it is specified, check the order of the fields. | |
| | IncorrectEndpoint | The specified bucket exists in another Region. Direct requests to the correct endpoint. | |
| 403 Forbidden | RequestTimeTooSkewed | The difference between the request time and the server's time is too large. | |
| | SignatureDoesNotMatch | The request signature that the server calculated does not match the signature that you provided. Check your AWS secret access key and signing method. For more information, see REST Authentication and SOAP Authentication. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | NotSignedUp | Your account is not signed up for the Amazon S3 service. You must sign up before you can use Amazon S3. You can sign up at the following URL: https://aws.amazon.com/s3 | |
| | InvalidSecurity | The provided security credentials are not valid. | |
| | InvalidPayer | All access to this object has been disabled. For further assistance, see Contact Us. | |
| | InvalidAccessKeyId | The AWS access key ID that you provided does not exist in our records. | |
| | AccountProblem | There is a problem with your AWS account that prevents the operation from completing s uccessfully. For further assistance, see Contact Us. | |

| HTTP status code | Error code | Description of error code | |
|---|---|---|---|
| | UnauthorizedAccessError | Applicable in China Regions only. Returned when a request is made to a bucket that doesn't have an ICP license. For more information, see ICP Recordal. | |
| 404 Not Found | NoSuchUpload | The specified multipart upload does not exist. The upload ID might not be valid, or the multipart upload might have been aborted or completed. | |
| | NoSuchWebsiteConfiguration | The specified bucket does not have a website configuration. | |
| 405 Method Not Allowed | MethodNotAllowed | The specified method is not allowed against this resource. | |
| 409 Conflict | BucketAlreadyExists | The requested bucket name is not available. The bucket namespace is shared by all users of the system. Specify a different name and try again. | |

| HTTP status code | Error code | Description of error code |
|---|---|---|
| | InvalidBucketState | The request is not valid for the current state of the bucket. |
| | OperationAborted | A conflicting conditional operation is currently in progress against this resource. Try again. |
| 411 Length Required | MissingContentLength | You must provide the Content-Length HTTP header. |
| 412 Precondition Failed | RequestIsNotMultiPartContent | A bucket POST request must be of the enclosure-type multipart/form-data. |

# Filtering and retrieving data using Amazon S3 Select

**Important**

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

With Amazon S3 Select, you can use structured query language (SQL) statements to filter the contents of an Amazon S3 object and retrieve only the subset of data that you need. By using Amazon S3 Select to filter this data, you can reduce the amount of data that Amazon S3 transfers, which reduces the cost and latency to retrieve this data.

Amazon S3 Select only allows you to query one object at a time. It works on an object stored in CSV, JSON, or Apache Parquet format. It also works with an object that is compressed with GZIP

or BZIP2 (for CSV and JSON objects only), and a server-side encrypted object. You can specify the format of the results as either CSV or JSON, and you can determine how the records in the result are delimited.

You pass SQL expressions to Amazon S3 in the request. Amazon S3 Select supports a subset of SQL. For more information about the SQL elements that are supported by Amazon S3 Select, see SQL reference for Amazon S3 Select.

You can perform SQL queries by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the `SelectObjectContent` REST API operation, or the AWS SDKs.

> **Note**
>
> The Amazon S3 console limits the amount of data returned to 40 MB. To retrieve more data, use the AWS CLI or the API.

## Requirements and limits

The following are requirements for using Amazon S3 Select:

- You must have `s3:GetObject` permission for the object you are querying.
- If the object you are querying is encrypted with server-side encryption with customer-provided keys (SSE-C), you must use `https`, and you must provide the encryption key in the request.

The following limits apply when using Amazon S3 Select:

- S3 Select can query only one object per request.
- The maximum length of a SQL expression is 256 KB.
- The maximum length of a record in the input or result is 1 MB.
- Amazon S3 Select can only emit nested data by using the JSON output format.
- You cannot query an object stored in the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or Reduced Redundancy Storage (RRS) storage classes. You also cannot query an object stored in the S3 Intelligent-Tiering Archive Access tier or the S3 Intelligent-Tiering Deep Archive Access tier. For more information about storage classes, see Using Amazon S3 storage classes.

Additional limitations apply when using Amazon S3 Select with a Parquet object:

- Amazon S3 Select supports only columnar compression using GZIP or Snappy. Amazon S3 Select doesn't support whole-object compression for a Parquet object.
- Amazon S3 Select doesn't support Parquet output. You must specify the output format as CSV or JSON.
- The maximum uncompressed row group size is 512 MB.
- You must use the data types that are specified in the object's schema.
- Selecting on a repeated field returns only the last value.

## Constructing a request

When you construct a request, you provide details of the object that is being queried by using an `InputSerialization` object. You provide details of how the results are to be returned by using an `OutputSerialization` object. You also include the SQL expression that Amazon S3 uses to filter the request.

For more information about constructing an Amazon S3 Select request, see SelectObjectContent in the *Amazon Simple Storage Service API Reference*. You can also see one of the SDK code examples in the following sections.

### Requests using scan ranges

With Amazon S3 Select, you can scan a subset of an object by specifying a range of bytes to query. This capability lets you parallelize scanning the whole object by splitting the work into separate Amazon S3 Select requests for a series of non-overlapping scan ranges.

Scan ranges don't need to be aligned with record boundaries. An Amazon S3 Select scan range request runs across the byte range that you specify. A record that starts within the specified scan range but extends beyond that scan range will be processed by the query. For example, the following shows an Amazon S3 object that contains a series of records in a line-delimited CSV format:

```
A,B
C,D
D,E
E,F
G,H
I,J
```

Suppose that you're using the Amazon S3 Select `ScanRange` parameter and *Start* at (Byte) 1 and *End* at (Byte) 4. So the scan range would start at "`,`" and scan until the end of the record starting at `C`. Your scan range request will return the result `C, D` because that is the end of the record.

Amazon S3 Select scan range requests support Parquet, CSV (without quoted delimiters), or JSON objects (in `LINES` mode only). CSV and JSON objects must be uncompressed. For line-based CSV and JSON objects, when a scan range is specified as part of the Amazon S3 Select request, all records that start within the scan range are processed. For Parquet objects, all of the row groups that start within the scan range requested are processed.

Amazon S3 Select scan range requests are available to use with the AWS CLI, Amazon S3 API, and AWS SDKs. You can use the `ScanRange` parameter in the Amazon S3 Select request for this feature. For more information, see SelectObjectContent in the *Amazon Simple Storage Service API Reference*.

## Errors

Amazon S3 Select returns an error code and associated error message when an issue is encountered while attempting to run a query. For a list of error codes and descriptions, see the List of SELECT Object Content Error Codes section of the *Error Responses* page in the *Amazon Simple Storage Service API Reference*.

For more information about Amazon S3 Select, see the following topics.

**Topics**

- Examples of using Amazon S3 Select on an object
- SQL reference for Amazon S3 Select

## Examples of using Amazon S3 Select on an object

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

You can use S3 Select to select content from one object by using the Amazon S3 console, the REST API, and the AWS SDKs.

For more information about supported SQL functions for S3 Select, see SQL functions.

### Using the S3 console

### To select content from an object in the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the bucket that contains the object that you want to select content from, and then choose the name of the object.
4. Choose **Object actions**, and choose **Query with S3 Select**.
5. Configure **Input settings**, based on the format of your input data.
6. Configure **Output settings**, based on the format of the output that you want to receive.
7. To extract records from the chosen object, under **SQL query**, enter the SELECT SQL commands. For more information on how to write SQL commands, see SQL reference for Amazon S3 Select.
8. After you enter SQL queries, choose **Run SQL query**. Then, under **Query results**, you can see the results of your SQL queries.

### Using the REST API

You can use the AWS SDKs to select content from an object. However, if your application requires it, you can send REST requests directly. For more information about the request and response format, see SelectObjectContent.

### Using the AWS SDKs

You can use Amazon S3 Select to select some of the content of an object by using the `selectObjectContent` method. If this method is successful, it returns the results of the SQL expression.

Java

The following Java code returns the value of the first column for each record that is stored in an object that contains data stored in CSV format. It also requests `Progress` and `Stats` messages to be returned. You must provide a valid bucket name and an object that contains data in CSV format.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;

/**
 * This example shows how to query data from S3Select and consume the response in
 the form of an
 * InputStream of records and write it to a file.
 */

public class RecordInputStreamExample {

    private static final String BUCKET_NAME = "${my-s3-bucket}";
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-
path}";
    private static final String QUERY = "select s._1 from S3Object s";

    public static void main(String[] args) throws Exception {
        final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();
```

```
        SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,
CSV_OBJECT_KEY, QUERY);
        final AtomicBoolean isResultComplete = new AtomicBoolean(false);

        try (OutputStream fileOutputStream = new FileOutputStream(new File
(S3_SELECT_RESULTS_PATH));
             SelectObjectContentResult result =
s3Client.selectObjectContent(request)) {
            InputStream resultInputStream =
result.getPayload().getRecordsInputStream(
                    new SelectObjectContentEventVisitor() {
                        @Override
                        public void visit(SelectObjectContentEvent.StatsEvent event)
                        {
                            System.out.println(
                                    "Received Stats, Bytes Scanned: " +
event.getDetails().getBytesScanned()
                                            +  " Bytes Processed: " +
event.getDetails().getBytesProcessed());
                        }

                        /*
                         * An End Event informs that the request has finished
successfully.
                         */
                        @Override
                        public void visit(SelectObjectContentEvent.EndEvent event)
                        {
                            isResultComplete.set(true);
                            System.out.println("Received End Event. Result is
complete.");
                        }
                    }
            );

            copy(resultInputStream, fileOutputStream);
        }

        /*
         * The End Event indicates all matching records have been transmitted.
         * If the End Event is not received, the results may be incomplete.
         */
        if (!isResultComplete.get()) {
```

```
            throw new Exception("S3 Select request was incomplete as End Event was
 not received.");
        }
    }

    private static SelectObjectContentRequest generateBaseCSVRequest(String bucket,
 String key, String query) {
        SelectObjectContentRequest request = new SelectObjectContentRequest();
        request.setBucketName(bucket);
        request.setKey(key);
        request.setExpression(query);
        request.setExpressionType(ExpressionType.SQL);

        InputSerialization inputSerialization = new InputSerialization();
        inputSerialization.setCsv(new CSVInput());
        inputSerialization.setCompressionType(CompressionType.NONE);
        request.setInputSerialization(inputSerialization);

        OutputSerialization outputSerialization = new OutputSerialization();
        outputSerialization.setCsv(new CSVOutput());
        request.setOutputSerialization(outputSerialization);

        return request;
    }
}
```

JavaScript

For a JavaScript example that uses the AWS SDK for JavaScript with the S3 `SelectObjectContent` API operation to select records from JSON and CSV files that are stored in Amazon S3, see the blog post Introducing support for Amazon S3 Select in the AWS SDK for JavaScript.

Python

For a Python example of using SQL queries to search through data that was loaded to Amazon S3 as a comma-separated value (CSV) file by using S3 Select, see the blog post Querying data without servers or databases using Amazon S3 Select.

# SQL reference for Amazon S3 Select

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

This reference contains a description of the structured query language (SQL) elements that are supported by Amazon S3 Select.

**Topics**



## SELECT command


> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more


Amazon S3 Select supports only the `SELECT` SQL command. The following ANSI standard clauses are supported for `SELECT`:

- `SELECT` list
- `FROM` clause
- `WHERE` clause
- `LIMIT` clause

> **Note**
>
> Amazon S3 Select queries currently do not support subqueries or joins.

### SELECT list

The SELECT list names the columns, functions, and expressions that you want the query to return. The list represents the output of the query.

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

The first form of SELECT with the * (asterisk) returns every row that passed the WHERE clause, as-is. The second form of SELECT creates a row with user-defined output scalar expressions ***projection1*** and ***projection2*** for each column.

### FROM clause

Amazon S3 Select supports the following forms of the FROM clause:

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

In each form of the FROM clause, `table_name` is the `S3Object` that's being queried. Users coming from traditional relational databases can think of this as a database schema that contains multiple views over a table.

Following standard SQL, the FROM clause creates rows that are filtered in the WHERE clause and projected in the SELECT list.

For JSON objects that are stored in Amazon S3 Select, you can also use the following forms of the FROM clause:

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

Using this form of the FROM clause, you can select from arrays or objects within a JSON object. You can specify path by using one of the following forms:

- By name (in an object): .*name* or ['*name*']
- By index (in an array): [*index*]
- By wildcard character (in an object): .*
- By wildcard character (in an array): [*]

**Note**

- This form of the FROM clause works only with JSON objects.
- Wildcard characters always emit at least one record. If no record matches, then Amazon S3 Select emits the value MISSING. During output serialization (after the query finishes running), Amazon S3 Select replaces MISSING values with empty records.
- Aggregate functions (AVG, COUNT, MAX, MIN, and SUM) skip MISSING values.
- If you don't provide an alias when using a wildcard character, you can refer to the row by using the last element in the path. For example, you could select all prices from a list of books by using the query SELECT price FROM S3Object[*].books[*].price. If the path ends in a wildcard character instead of a name, then you can use the value _1 to refer to the row. For example, instead of SELECT price FROM S3Object[*].books[*].price, you could use the query SELECT _1.price FROM S3Object[*].books[*].
- Amazon S3 Select always treats a JSON document as an array of root-level values. Thus, even if the JSON object that you are querying has only one root element, the FROM clause must begin with S3Object[*]. However, for compatibility reasons, Amazon S3 Select allows you to omit the wildcard character if you don't include a path. Thus, the complete clause FROM S3Object is equivalent to FROM S3Object[*] as S3Object. If you include a path, you must also use the wildcard character. So, FROM S3Object and FROM S3Object[*].*path* are both valid clauses, but FROM S3Object.*path* is not.

**Example**

**Examples:**

*Example #1*

This example shows results when using the following dataset and query:

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select produces each result for the following reasons:

- `{"id":"id-1"}` – `S3Object[0].Rules[0].id` produced a match.
- `{}` – `S3Object[0].Rules[1].id` did not match a record, so Amazon S3 Select emitted `MISSING`, which was then changed to an empty record during output serialization and returned.
- `{"id":"id-2"}` – `S3Object[0].Rules[2].id` produced a match.
- `{}` – `S3Object[1]` did not match on `Rules`, so Amazon S3 Select emitted `MISSING`, which was then changed to an empty record during output serialization and returned.

If you don't want Amazon S3 Select to return empty records when it doesn't find a match, you can test for the value `MISSING`. The following query returns the same results as the previous query, but with the empty values omitted:

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

*Example #2*

This example shows results when using the following dataset and queries:

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

### WHERE clause

The WHERE clause follows this syntax:

```
WHERE condition
```

The WHERE clause filters rows based on the *condition*. A condition is an expression that has a Boolean result. Only rows for which the condition evaluates to TRUE are returned in the result.

### LIMIT clause

The LIMIT clause follows this syntax:

```
LIMIT number
```

The LIMIT clause limits the number of records that you want the query to return based on *number*.

### Attribute access

The SELECT and WHERE clauses can refer to record data by using one of the methods in the following sections, depending on whether the file that is being queried is in CSV or JSON format.

### CSV

- **Column Numbers** – You can refer to the *Nth* column of a row with the column name *_N*, where *N* is the column position. The position count starts at 1. For example, the first column is named _1 and the second column is named _2.

  You can refer to a column as *_N* or *alias*.*_N*. For example, _2 and `myAlias._2` are both valid ways to refer to a column in the `SELECT` list and `WHERE` clause.

- **Column Headers** – For objects in CSV format that have a header row, the headers are available to the `SELECT` list and `WHERE` clause. In particular, as in traditional SQL, within `SELECT` and `WHERE` clause expressions, you can refer to the columns by *alias*.*column_name* or *column_name*.

### JSON

- **Document** – You can access JSON document fields as *alias*.*name*. You can also access nested fields, for example, *alias*.*name1*.*name2*.*name3*.
- **List** – You can access elements in a JSON list by using zero-based indexes with the `[]` operator. For example, you can access the second element of a list as *alias*`[1]`. You can combine accessing list elements with fields, for example, *alias*.*name1*.*name2*`[1]`.*name3*.
- **Examples:** Consider this JSON object as a sample dataset:

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

*Example #1*

The following query returns these results:

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

*Example #2*

The following query returns these results:

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

### Case sensitivity of header and attribute names

With Amazon S3 Select, you can use double quotation marks to indicate that column headers (for CSV objects) and attributes (for JSON objects) are case sensitive. Without double quotation marks, object headers and attributes are case insensitive. An error is thrown in cases of ambiguity.

The following examples are either 1) Amazon S3 objects in CSV format with the specified column headers, and with `FileHeaderInfo` set to `"Use"` for the query request; or 2) Amazon S3 objects in JSON format with the specified attributes.

*Example #1:* The object being queried has the header or attribute `NAME`.

- The following expression successfully returns values from the object. Because there are no quotation marks, the query is case insensitive.

  ```
  SELECT s.name from S3Object s
  ```

- The following expression results in a 400 error `MissingHeaderName`. Because there are quotation marks, the query is case sensitive.

  ```
  SELECT s."name" from S3Object s
  ```

*Example #2:* The Amazon S3 object being queried has one header or attribute with `NAME` and another header or attribute with `name`.

- The following expression results in a 400 error `AmbiguousFieldName`. Because there are no quotation marks, the query is case insensitive, but there are two matches, so the error is thrown.

  ```
  SELECT s.name from S3Object s
  ```

- The following expression successfully returns values from the object. Because there are quotation marks, the query is case sensitive, so there is no ambiguity.

```
SELECT s."NAME" from S3Object s
```

### Using reserved keywords as user-defined terms

Amazon S3 Select has a set of reserved keywords that are needed to run the SQL expressions used to query object content. Reserved keywords include function names, data types, operators, and so on. In some cases, user-defined terms, such as the column headers (for CSV files) or attributes (for JSON objects), might clash with a reserved keyword. When this happens, you must use double quotation marks to indicate that you are intentionally using a user-defined term that clashes with a reserved keyword. Otherwise a 400 parse error will result.

For the full list of reserved keywords, see Reserved keywords.

The following example is either 1) an Amazon S3 object in CSV format with the specified column headers, with `FileHeaderInfo` set to `"Use"` for the query request, or 2) an Amazon S3 object in JSON format with the specified attributes.

*Example:* The object being queried has a header or attribute named `CAST`, which is a reserved keyword.

- The following expression successfully returns values from the object. Because quotation marks are used in the query, S3 Select uses the user-defined header or attribute.

```
SELECT s."CAST" from S3Object s
```

- The following expression results in a 400 parse error. Because no quotation marks are used in the query, `CAST` clashes with a reserved keyword.

```
SELECT s.CAST from S3Object s
```

### Scalar expressions

Within the `WHERE` clause and the `SELECT` list, you can have SQL *scalar expressions*, which are expressions that return scalar values. They have the following form:

- ***literal***

An SQL literal.

- ***column_reference***

  A reference to a column in the form *column_name* or *alias*.*column_name*.

- ***unary_op expression***

  In this case, ***unary_op*** is an SQL unary operator.

- ***expression binary_op expression***

  In this case, ***binary_op*** is an SQL binary operator.

- ***func_name***

  In this case, ***func_name*** is the name of the scalar function to invoke.

- ***expression*** [ NOT ] BETWEEN ***expression*** AND ***expression***
- ***expression*** LIKE ***expression*** [ ESCAPE ***expression*** ]

## Data types

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports several primitive data types.

### Data type conversions

The general rule is to follow the CAST function if it's defined. If CAST is not defined, then all input data is treated as a string. In that case, you must cast your input data into the relevant data types when necessary.

For more information about the CAST function, see CAST.

### Supported data types

Amazon S3 Select supports the following set of primitive data types.

| Name | Description | Examples |
| --- | --- | --- |
| `bool` | A Boolean value, either TRUE or FALSE. | `FALSE` |
| `int`, `integer` | An 8-byte signed integer in the range -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. | `100000` |
| `string` | A UTF8-encoded variable-length string. The default limit is 1 character. The maximum character limit is 2,147,483,647. | `'xyz'` |
| `float` | An 8-byte floating point number. | `CAST(0.456 AS FLOAT)` |
| `decimal`, `numeric` | A base-10 number, with a maximum precision of 38 (that is, the maximum number of significant digits), and with a scale within the range of $-2^{31}$ to $2^{31}-1$ (that is, the base-10 exponent).<br><br>**Note**<br>Amazon S3 Select ignores scale and precision when you provide both at the same time. | `123.456` |
| `timestamp` | Timestamps represent a specific moment in time, always include a local offset, and are capable of arbitrary precision.<br><br>In the text format, timestamps follow the W3C note on date and time formats, but they must end with the literal T if the timestamps are not at least whole-day precision. Fractional seconds are allowed, with at least one digit of precision, and an unlimited maximum. Local-time offsets can be represented as either hour:minute offsets from UTC, or as the literal Z to denote a local time of UTC. Local-time offsets are required on timestamps with time and are not allowed on date values. | `CAST('2007-04-05T14:30Z' AS TIMESTAMP)` |

### Supported Parquet types

Amazon S3 Select supports the following Parquet types.

- DATE
- DECIMAL
- ENUM
- INT(8)
- INT(16)
- INT(32)
- INT(64)
- LIST

> **Note**
>
> For LIST Parquet type output, Amazon S3 Select supports only JSON format. However, if the query limits the data to simple values, the LIST Parquet type can also be queried in CSV format.

- STRING
- TIMESTAMP supported precision (MILLIS/MICROS/NANOS)

> **Note**
>
> Timestamps saved as an INT(96) are unsupported.
> Because of the range of the INT(64) type, timestamps that are using the NANOS unit can represent only values between 1677-09-21 00:12:43 and 2262-04-11 23:47:16. Values outside of this range cannot be represented with the NANOS unit.

### Mapping of Parquet types to supported data types in Amazon S3 Select

| Parquet types | Supported data types |
| --- | --- |
| DATE | timestamp |

| Parquet types | Supported data types |
| --- | --- |
| DECIMAL | decimal, numeric |
| ENUM | string |
| INT(8) | int, integer |
| INT(16) | int, integer |
| INT(32) | int, integer |
| INT(64) | decimal, numeric |
| LIST | Each Parquet type in list is mapped to the corresponding data type. |
| STRING | string |
| TIMESTAMP | timestamp |

## Operators

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following operators.

### Logical operators

- AND
- NOT

- OR

### Comparison operators

- <
- >
- <=
- >=
- =
- <>
- !=
- BETWEEN
- IN – For example: `IN ('a', 'b', 'c')`

### Pattern-matching operators

- LIKE
- _ (Matches any character)
- % (Matches any sequence of characters)

### Unitary operators

- IS NULL
- IS NOT NULL

### Math operators

Addition, subtraction, multiplication, division, and modulo are supported, as follows:

- +
- -
- *

- /
- %

### Operator precedence

The following table shows the operators' precedence in decreasing order.

| Operator or element | Associativity | Required |
| --- | --- | --- |
| - | right | unary minus |
| *, /, % | left | multiplication, division, modulo |
| +, - | left | addition, subtraction |
| IN | | set membership |
| BETWEEN | | range containment |
| LIKE | | string pattern matching |
| <> | | less than, greater than |
| = | right | equality, assignment |
| NOT | right | logical negation |
| AND | left | logical conjunction |

| Operator or element | Associativity | Required |
| --- | --- | --- |
| OR | left | logical disjunction |

## Reserved keywords

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

The following is the list of reserved keywords for Amazon S3 Select. These keywords include function names, data types, operators, and so on, that are needed to run the SQL expressions that are used to query object content.

```
absolute
action
add
all
allocate
alter
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
bit_length
blob
bool
```

```
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
declare
default
deferrable
```

```
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
in
indicator
initially
```

```
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
module
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
of
on
only
```

```
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
sql
sqlcode
sqlerror
```

```
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
write
year
```

zone

## SQL functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following SQL functions.

### Topics



### Aggregate functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following aggregate functions.

| Function | Argument type | Return type |
| --- | --- | --- |
| `AVG(`*`expressio`*<br>*`n`* `)` | `INT`, `FLOAT`, `DECIMAL` | `DECIMAL` for an `INT` argument, `FLOAT` for a floating-point |

| Function | Argument type | Return type |
| --- | --- | --- |
| | | argument; otherwise the same as the argument data type. |
| COUNT | - | INT |
| MAX(*expression*) | INT, DECIMAL | Same as the argument type. |
| MIN(*expression*) | INT, DECIMAL | Same as the argument type. |
| SUM(*expression*) | INT, FLOAT, DOUBLE, DECIMAL | INT for an INT argument, FLOAT for a floating-point argument; otherwise, the same as the argument data type. |

### SUM example

To aggregate the total object sizes of a folder in an S3 Inventory report, use a SUM expression.

The following S3 Inventory report is a CSV file that's compressed with GZIP. There are three columns.

- The first column is the name of the S3 bucket (*DOC-EXAMPLE-BUCKET*) that the S3 Inventory report is for.
- The second column is the object key name that uniquely identifies the object in the bucket.

The *example-folder/* value in the first row is for the folder *example-folder*. In Amazon S3, when you create a folder in your bucket, S3 creates a 0-byte object with a key that's set to the folder name that you provided.

The *example-folder/object1* value in the second row is for the object *object1* in the folder *example-folder*.

The *example-folder/object2* value in the third row is for the object *object2* in the folder *example-folder*.

For more information about S3 folders, see Organizing objects in the Amazon S3 console by using folders.

- The third column is the object size in bytes.

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

To use a SUM expression to calculate the total size of the folder *example-folder*, run the SQL query with Amazon S3 Select.

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

Query Result:

```
3581291
```

### Conditional functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following conditional functions.

**Topics**

- CASE
- COALESCE
- NULLIF

### CASE

The CASE expression is a conditional expression, similar to `if/then/else` statements found in other languages. CASE is used to specify a result when there are multiple conditions. There are two types of CASE expressions: simple and searched.

In simple CASE expressions, an expression is compared with a value. When a match is found, the specified action in the THEN clause is applied. If no match is found, the action in the ELSE clause is applied.

In searched CASE expressions, each CASE is evaluated based on a Boolean expression, and the CASE statement returns the first matching CASE. If no matching CASE is found among the WHEN clauses, the action in the ELSE clause is returned.

**Syntax**

> **Note**
>
> Currently, Amazon S3 Select doesn't support ORDER BY or queries that contain new lines. Make sure that you use queries with no line breaks.

The following is a simple CASE statement that's used to match conditions:

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

The following is a searched CASE statement that's used to evaluate each condition:

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

### Examples

> **Note**
>
> If you use the Amazon S3 console to run the following examples and your CSV file contains a header row, choose **Exclude the first line of CSV data**.

**Example 1:** Use a simple CASE expression to replace `New York City` with `Big Apple` in a query. Replace all other city names with `other`.

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

Query result:

```
venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

**Example 2:** Use a searched CASE expression to assign group numbers based on the `pricepaid` value for individual ticket sales:

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

Query result:

```
pricepaid |  case
-----------+---------
12624.00 | group 2
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

### COALESCE

COALESCE evaluates the arguments in order and returns the first non-unknown value, that is, the first non-null or non-missing value. This function does not propagate null and missing values.

#### Syntax

```
COALESCE ( expression, expression, ... )
```

#### Parameters

*expression*

The target expression that the function operates on.

#### Examples

```
COALESCE(1)                -- 1
COALESCE(null)             -- null
COALESCE(null, null)       -- null
COALESCE(missing)          -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)          -- 1
COALESCE(null, null, 1)    -- 1
COALESCE(null, 'string')   -- 'string'
COALESCE(missing, 1)       -- 1
```

### NULLIF

Given two expressions, NULLIF returns NULL if the two expressions evaluate to the same value; otherwise, NULLIF returns the result of evaluating the first expression.

#### Syntax

```
NULLIF ( expression1, expression2 )
```

#### Parameters

*expression1*, *expression2*

The target expressions that the function operates on.

### Examples

```
NULLIF(1, 1)             -- null
NULLIF(1, 2)             -- 1
NULLIF(1.0, 1)           -- null
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null
```

## Conversion functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following conversion function.

### Topics

- CAST

## CAST

The CAST function converts an entity, such as an expression that evaluates to a single value, from one type to another.

### Syntax

```
CAST ( expression AS data_type )
```

### Parameters

*expression*

A combination of one or more values, operators, and SQL functions that evaluate to a value.

*data_type*

The target data type, such as INT, to cast the expression to. For a list of supported data types, see Data types.

### Examples

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

### Date functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following date functions.

### Topics

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

### DATE_ADD

Given a date part, a quantity, and a timestamp, DATE_ADD returns an updated timestamp by altering the date part by the quantity.

### Syntax

```
DATE_ADD( date_part, quantity, timestamp )
```

**Parameters**

*date_part*

Specifies which part of the date to modify. This can be one of the following:

- year
- month
- day
- hour
- minute
- second

*quantity*

The value to apply to the updated timestamp. Positive values for *quantity* add to the timestamp's date_part, and negative values subtract.

*timestamp*

The target timestamp that the function operates on.

**Examples**

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to
 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision
 as necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to
 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

**DATE_DIFF**

Given a date part and two valid timestamps, DATE_DIFF returns the difference in date parts. The return value is a negative integer when the *date_part* value of *timestamp1* is greater than the *date_part* value of *timestamp2*. The return value is a positive integer when the *date_part* value of *timestamp1* is less than the *date_part* value of *timestamp2*.

### Syntax

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

### Parameters

*date_part*

Specifies which part of the timestamps to compare. For the definition of date_part, see DATE_ADD.

*timestamp1*

The first timestamp to compare.

*timestamp2*

The second timestamp to compare.

### Examples

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)            -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                     -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                       -- 12
DATE_DIFF(month, `2011T`, `2010T`)                       -- -12
DATE_DIFF(day, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

## EXTRACT

Given a date part and a timestamp, EXTRACT returns the timestamp's date part value.

### Syntax

```
EXTRACT( date_part FROM timestamp )
```

### Parameters

*date_part*

Specifies which part of the timestamps to extract. This can be one of the following:

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

*timestamp*

The target timestamp that the function operates on.

### Examples

```
EXTRACT(YEAR FROM `2010-01-01T`)                                -- 2010
EXTRACT(MONTH FROM `2010T`)                                     -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                                  -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)                  -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)                -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)         -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)       -- 8
```

## TO_STRING

Given a timestamp and a format pattern, TO_STRING returns a string representation of the timestamp in the given format.

### Syntax

```
TO_STRING ( timestamp time_format_pattern )
```

### Parameters

*timestamp*

The target timestamp that the function operates on.

*time_format_pattern*

A string that has the following special character interpretations:

| Format | Example | Description |
|---|---|---|
| yy | 69 | 2-digit year |
| y | 1969 | 4-digit year |
| yyyy | 1969 | Zero-padded 4-digit year |
| M | 1 | Month of year |
| MM | 01 | Zero-padded month of yea r |
| MMM | Jan | Abbreviated month year n ame |
| MMMM | January | Full month of year name |
| MMMMM | J | Month of year first letter (NOTE: This format is not valid for use with the TO_TIMEST AMP function.) |
| d | 2 | Day of month (1-31) |

| Format | Example | Description |
| --- | --- | --- |
| dd | 02 | Zero-padded day of month (01-31) |
| a | AM | AM or PM of day |
| h | 3 | Hour of day (1-12) |
| hh | 03 | Zero-padded hour of day (01-12) |
| H | 3 | Hour of day (0-23) |
| HH | 03 | Zero-padded hour of day (00-23) |
| m | 4 | Minute of hour (0-59) |
| mm | 04 | Zero-padded minute of hour (00-59) |
| s | 5 | Second of minute (0-59) |
| ss | 05 | Zero-padded second of minute (00-59) |

| Format | Example | Description |
| --- | --- | --- |
| S | 0 | Fraction of a second (precision: 0.1, range: 0.0-0.9) |
| SS | 6 | Fraction of a second (precision: 0.01, range: 0.0-0.99) |
| SSS | 60 | Fraction of a second (precision: 0.001, range: 0.0-0.999) |
| ... | ... | ... |
| SSSSSSSSS | 60000000 | Fraction of a second (maximum precision: 1 nanosecond, range: 0.0-0.999999999) |
| n | 60000000 | Nano of a second |
| X | +07 or Z | Offset in hours, or Z if the offset is 0 |

| Format | Example | Description |
| --- | --- | --- |
| XX or XXXX | +0700 or Z | Offset in hours and minutes, or Z if the offset is 0 |
| XXX or XXXXX | +07:00 or Z | Offset in hours and minutes, or Z if the offset is 0 |
| x | 7 | Offset in hours |
| xx or xxxx | 700 | Offset in hours and mi nutes |
| xxx or xxxxx | +07:00 | Offset in hours and mi nutes |

### Examples

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                   -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                        -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                        -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')               -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')           --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')     --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')   --
 "1969-07-20T20:18:00+0800"
```

```
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX')  --
 "1969-07-20T20:18:00+08:00"
```

### TO_TIMESTAMP

Given a string, TO_TIMESTAMP converts it to a timestamp. TO_TIMESTAMP is the inverse operation of TO_STRING.

#### Syntax

```
TO_TIMESTAMP ( string )
```

#### Parameters

*string*

The target string that the function operates on.

#### Examples

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

### UTCNOW

UTCNOW returns the current time in UTC as a timestamp.

#### Syntax

```
UTCNOW()
```

#### Parameters

UTCNOW takes no parameters.

#### Examples

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

## String functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following string functions.

### Topics



### CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH (or CHARACTER_LENGTH) counts the number of characters in the specified string.

> **Note**
>
> CHAR_LENGTH and CHARACTER_LENGTH are synonyms.

### Syntax

```
CHAR_LENGTH ( string )
```

### Parameters

*string*

The target string that the function operates on.

### Examples

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

### LOWER

Given a string, LOWER converts all uppercase characters to lowercase characters. Any non-uppercased characters remain unchanged.

### Syntax

```
LOWER ( string )
```

### Parameters

*string*

The target string that the function operates on.

### Examples

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

### SUBSTRING

Given a string, a start index, and optionally a length, SUBSTRING returns the substring from the start index up to the end of the string, or up to the length provided.

> **Note**
>
> The first character of the input string has an index position of 1.
>
> - If `start` is < 1, with no length specified, then the index position is set to 1.
> - If `start` is < 1, with a length specified, then the index position is set to `start + length -1`.
> - If `start + length -1` < 0, then an empty string is returned.
> - If `start + length -1` > = 0, then the substring starting at index position 1 with the length `start + length - 1` is returned.

### Syntax

```
SUBSTRING( string FROM start [ FOR length ] )
```

### Parameters

*string*

The target string that the function operates on.

*start*

The start position of the string.

*length*

The length of the substring to return. If not present, proceed to the end of the string.

### Examples

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

### TRIM

Trims leading or trailing characters from a string. The default character to remove is a space (' ').

### Syntax

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

### Parameters

*string*

The target string that the function operates on.

`LEADING | TRAILING | BOTH`

This parameter indicates whether to trim leading or trailing characters, or both leading and trailing characters.

*`remove_chars`*

The set of characters to remove. *`remove_chars`* can be a string with a length > 1. This function returns the string with any character from *`remove_chars`* found at the beginning or end of the string that was removed.

### Examples

```
TRIM('       foobar         ')               -- 'foobar'
TRIM('      \tfoobar\t          ')            -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

## UPPER

Given a string, UPPER converts all lowercase characters to uppercase characters. Any non-lowercased characters remain unchanged.

### Syntax

```
UPPER ( string )
```

### Parameters

*`string`*

The target string that the function operates on.

### Examples

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

# Performing large-scale batch operations on Amazon S3 objects

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can perform a single operation on lists of Amazon S3 objects that you specify. A single job can perform a specified operation on billions of objects containing exabytes of data. Amazon S3 tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, and serverless experience. You can use S3 Batch Operations through the AWS Management Console, AWS CLI, Amazon SDKs, or REST API.

Use S3 Batch Operations to copy objects and set object tags or access control lists (ACLs). You can also initiate object restores from S3 Glacier Flexible Retrieval or invoke an AWS Lambda function to perform custom actions using your objects. You can perform these operations on a custom list of objects, or you can use an Amazon S3 Inventory report to easily generate lists of objects. Amazon S3 Batch Operations use the same Amazon S3 APIs that you already use with Amazon S3, so you'll find the interface familiar.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets. For more information about using Batch Operations with S3 Express One Zone and directory buckets, see Using Batch Operations with S3 Express One Zone.

## S3 Batch Operations basics

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can run a single operation or action on lists of Amazon S3 objects that you specify.

### Terminology

This section uses the terms *jobs*, *operations*, and *tasks*, which are defined as follows:

#### Job

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on the objects listed in the manifest. After you provide

this information and request that the job begin, the job performs the operation for each object in the manifest.

**Operation**

The operation is the type of API action, such as copying objects, that you want the Batch Operations job to run. Each job performs a single type of operation across all objects that are specified in the manifest.

**Task**

A task is the unit of execution for a job. A task represents a single call to an Amazon S3 or AWS Lambda API operation to perform the job's operation on a single object. Over the course of a job's lifetime, S3 Batch Operations create one task for each object specified in the manifest.

### How an S3 Batch Operations job works

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on a list of objects. To create a job, you give S3 Batch Operations a list of objects and specify the action to perform on those objects.

For information about the operations that S3 Batch Operations supports, see Operations supported by S3 Batch Operations.

A batch job performs a specified operation on every object that is included in its *manifest*. A manifest lists the objects that you want a batch job to process and it is stored as an object in a bucket. You can use a comma-separated values (CSV)-formatted Amazon S3 Inventory report as a manifest, which makes it easy to create large lists of objects located in a bucket. You can also specify a manifest in a simple CSV format that enables you to perform batch operations on a customized list of objects contained within a single bucket.

After you create a job, Amazon S3 processes the list of objects in the manifest and runs the specified operation against each object. While a job is running, you can monitor its progress programmatically or through the Amazon S3 console. You can also configure a job to generate a completion report when it finishes. The completion report describes the results of each task that was performed by the job. For more information about monitoring jobs, see Managing S3 Batch Operations jobs.

## S3 Batch Operations tutorial

The following tutorial presents complete end-to-end procedures for some Batch Operations tasks.

- Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

## Granting permissions for Amazon S3 Batch Operations

Before creating and running S3 Batch Operations jobs, you must grant required permissions. To create an Amazon S3 Batch Operations job, the `s3:CreateJob` user permission is required. The same entity that creates the job must also have the `iam:PassRole` permission to pass the AWS Identity and Access Management (IAM) role that is specified for the job to Batch Operations.

For general information about specifying IAM resources, see IAM JSON policy, Resource elements in the *IAM User Guide*. The following sections provide information about creating an IAM role and attaching policies.

**Topics**

- Creating an S3 Batch Operations IAM role
- Attaching permissions policies

### Creating an S3 Batch Operations IAM role

Amazon S3 must have permissions to perform S3 Batch Operations on your behalf. You grant these permissions through an AWS Identity and Access Management (IAM) role. This section provides examples of the trust and permissions policies you use when creating an IAM role. For more information, see IAM roles in the *IAM User Guide*. For examples, see Controlling permissions for S3 Batch Operations using job tags and Copying objects using S3 Batch Operations.

In your IAM policies, you can also use condition keys to filter access permissions for S3 Batch Operations jobs. For more information and a complete list of Amazon S3 specific condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

**Trust policy**

To allow the S3 Batch Operations service principal to assume the IAM role, attach the following trust policy to the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
```

```
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

## Attaching permissions policies

Depending on the type of operations, you can attach one of the following policies.

Before you configure permissions, note the following:

- Regardless of the operation, Amazon S3 needs permissions to read your manifest object from your S3 bucket and optionally write a report to your bucket. Therefore, all of the following policies include these permissions.
- For Amazon S3 Inventory report manifests, S3 Batch Operations requires permission to read the manifest.json object and all associated CSV data files.
- Version-specific permissions such as `s3:GetObjectVersion` are only required when you are specifying the version ID of the objects.
- If you are running S3 Batch Operations on encrypted objects, the IAM role must also have access to the AWS KMS keys used to encrypt them.
- If you submit an inventory report manifest encrypted with AWS KMS, your IAM policy must include the permissions `"kms:Decrypt"` and `"kms:GenerateDataKey"` for the manifest.json object and all associated CSV data files.
- If the Batch Operations job generates a manifest in a bucket that has ACLs enabled and is in a different AWS account, you must grant the `s3:PutObjectAcl` permission in the IAM policy of the IAM role configured for the batch job. If you do not include this permission, the batch job fails with the error `Error occurred when preparing manifest: Failed to write manifest`.

### Copy objects: PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DestinationBucket/*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::SourceBucket",
                "arn:aws:s3:::SourceBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

### Replace object tagging: PutObjectTagging

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

### Delete object tagging: DeleteObjectTagging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

### Replace access control list: PutObjectAcl

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

### Restore objects: RestoreObject

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
```

```
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

### Apply Object Lock retention: PutObjectRetention

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

### Apply Object Lock legal hold: PutObjectLegalHold

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

**Replicate existing objects: InitiateReplication with an S3 generated manifest**

Use this policy if using and storing an S3 generated manifest. For more information about using Batch Operations for replicate existing objects see, [Replicating existing objects with S3 Batch Replication](#).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
```

```
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

**Replicate existing objects: InitiateReplication with a user manifest**

Use this policy if using a user supplied manifest. For more information about using Batch Operations for replicate existing objects see, Replicating existing objects with S3 Batch Replication.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
```

```
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*"
         ]
      }
   ]
}
```

# Creating an S3 Batch Operations job

With Amazon S3 Batch Operations, you can perform large-scale batch operations on a list of specific Amazon S3 objects. This section describes the information that you need to create an S3 Batch Operations job and the results of a `CreateJob` request. It also provides instructions for creating a Batch Operations job by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

When you create an S3 Batch Operations job, you can request a completion report for all tasks or only for failed tasks. As long as at least one task has been invoked successfully, S3 Batch Operations generates a report for jobs that have been completed, have failed, or have been canceled. For more information, see Examples: S3 Batch Operations completion reports.

**Topics**



## Batch Operations job request elements

To create an S3 Batch Operations job, you must provide the following information:

**Operation**

Specify the operation that you want S3 Batch Operations to run against the objects in the manifest. Each operation type accepts parameters that are specific to that operation. With

Batch Operations, you can perform an operation in bulk, with the same results as if you performed that operation one-by-one on each object.

**Manifest**

The *manifest* is a list of all of the objects that you want S3 Batch Operations to run the specified operation on. You can use the following methods to specify a manifest for a Batch Operations job:

- Manually create your own customized, CSV-formatted object list.
- Choose an existing CSV-formatted Amazon S3 Inventory report.
- Direct Batch Operations to generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for batch replication jobs that you create in the Amazon S3 console, or for any job type that you create by using the AWS CLI, AWS SDKs, or Amazon S3 REST API.

> **Note**
>
> - Regardless of how you specify your manifest, the list itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from, or save generated manifests to directory buckets. Objects described within the manifest, however, can be stored in directory buckets. For more information, see Directory buckets.
> - If the objects in your manifest are in a versioned bucket, specifying the version IDs for the objects directs Batch Operations to perform the operation on a specific version. If no version IDs are specified, Batch Operations performs the operation on the latest version of the objects. If your manifest includes a version ID field, you must provide a version ID for all objects in the manifest.

For more information, see Specifying a manifest.

**Priority**

Use job priorities to indicate the relative priority of this job to others running in your account. A higher number indicates higher priority.

Job priorities only have meaning relative to the priorities that are set for other jobs in the same account and Region. You can choose whatever numbering system works for you. For

example, you might want to assign all **Restore** (`RestoreObject`) jobs a priority of 1, all **Copy** (`CopyObject`) jobs a priority of 2, and all **Replace access control lists (ACLs)** (`PutObjectAcl`) jobs a priority of 3.

S3 Batch Operations prioritizes jobs according to priority numbers, but strict ordering isn't guaranteed. Therefore, don't use job priorities to ensure that any one job starts or finishes before any other job. If you must ensure strict ordering, wait until one job has finished before starting the next.

**RoleArn**

Specify an AWS Identity and Access Management (IAM) role to run the job. The IAM role that you use must have sufficient permissions to perform the operation that is specified in the job. For example, to run a `CopyObject` job, the IAM role must have the `s3:GetObject` permission for the source bucket and the `s3:PutObject` permission for the destination bucket. The role also needs permissions to read the manifest and write the job-completion report.

For more information about IAM roles, see IAM roles in the *IAM User Guide*.

For more information about Amazon S3 permissions, see Policy actions for Amazon S3.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

**Report**

Specify whether you want S3 Batch Operations to generate a completion report. If you request a job-completion report, you must also provide the parameters for the report in this element. The necessary information includes:

- The bucket where you want to store the report

> **Note**
>
> The report must be stored in a general purpose bucket. Batch Operations can't save reports to directory buckets. For more information, see Directory buckets.

- The format of the report
- Whether you want the report to include the details of all tasks or only failed tasks
- An optional prefix string

> **Note**
>
> Completion reports are always encrypted with Amazon S3 managed keys (SSE-S3).

**Tags (optional)**

You can label and control access to your S3 Batch Operations jobs by adding *tags*. You can use tags to identify who is responsible for a Batch Operations job, or to control how users interact with Batch Operations jobs. The presence of job tags can grant or limit a user's ability to cancel a job, activate a job in the confirmation state, or change a job's priority level. For example, you could grant a user permission to invoke the `CreateJob` operation, provided that the job is created with the tag `"Department=Finance"`.

You can create jobs with tags attached to them, and you can add tags to jobs after you create them.

For more information, see the section called “Using tags”.

**Description (optional)**

To track and monitor your job, you can also provide a description of up to 256 characters. Amazon S3 includes this description whenever it returns information about a job or displays job details on the Amazon S3 console. You can then easily sort and filter jobs according to the descriptions that you assigned. Descriptions don't need to be unique, so you can use descriptions as categories (for example, "Weekly Log Copy Jobs") to help you track groups of similar jobs.

## Specifying a manifest

A manifest is an Amazon S3 object that contains the object keys that you want Amazon S3 to act upon. You can supply a manifest in one of the following ways:

- Create a new manifest file manually.

- Use an existing manifest.
- Direct Batch Operations to generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for batch replication jobs that you create in the Amazon S3 console, or for any job type that you create by using the AWS CLI, AWS SDKs, or Amazon S3 REST API.

> **Note**
>
> Regardless of how you specify your manifest, the list itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from, or save generated manifests to directory buckets. Objects described within the manifest, however, can be stored in directory buckets. For more information, see Directory buckets.

### Creating a manifest file

To create a manifest file manually, you specify the manifest object key, ETag (entity tag), and optional version ID in a CSV-formatted list. The contents of the manifest must be URL-encoded.

By default, Amazon S3 automatically uses server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt a manifest that's uploaded to an Amazon S3 bucket. Manifests that use server-side encryption with customer-provided keys (SSE-C) are not supported. Manifests that use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) are supported only when you're using CSV-formatted inventory reports. Using a manually created manifest with AWS KMS is not supported.

Your manifest must contain the bucket name, object key, and optionally, the object version for each object. Any other fields in the manifest are not used by S3 Batch Operations.

> **Note**
>
> If the objects in your manifest are in a versioned bucket, specifying the version IDs for the objects directs Batch Operations to perform the operation on a specific version. If no version IDs are specified, Batch Operations performs the operation on the latest version of the objects. If your manifest includes a version ID field, you must provide a version ID for all objects in the manifest.

The following is an example manifest in CSV format without version IDs.

```
Examplebucket,objectkey1
Examplebucket,objectkey2
Examplebucket,objectkey3
Examplebucket,photos/jpgs/objectkey4
Examplebucket,photos/jpgs/newjersey/objectkey5
Examplebucket,object%20key%20with%20spaces
```

The following is an example manifest in CSV format that includes version IDs.

```
Examplebucket,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
Examplebucket,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
Examplebucket,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
Examplebucket,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
Examplebucket,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
Examplebucket,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

### Specifying an existing manifest file

You can specify a manifest file for a create job request by using one of the following two formats:

- **Amazon S3 Inventory report** – Must be a CSV-formatted Amazon S3 Inventory report. You must specify the `manifest.json` file that is associated with the inventory report. For more information about inventory reports, see Amazon S3 Inventory. If the inventory report includes version IDs, S3 Batch Operations operates on the specific object versions.

  > **Note**
  >
  > - S3 Batch Operations supports CSV *inventory reports* that are encrypted with SSE-KMS.
  > - If you submit an inventory report manifest that's encrypted with SSE-KMS, your IAM policy must include the permissions `"kms:Decrypt"` and `"kms:GenerateDataKey"` for the `manifest.json` object and all associated CSV data files.

- **CSV file** – Each row in the file must include the bucket name, object key, and optionally, the object version. Object keys must be URL-encoded, as shown in the following examples. The manifest must either include version IDs for all objects or omit version IDs for all objects. For more information about the CSV manifest format, see JobManifestSpec in the *Amazon Simple Storage Service API Reference*.

> **Note**
>
> S3 Batch Operations doesn't support CSV *manifest files* that are encrypted with SSE-KMS.

> **Important**
>
> When you're using a manually created manifest and a versioned bucket, we recommend that you specify the version IDs for the objects. When you create a job, S3 Batch Operations parses the entire manifest before running the job. However, it doesn't take a "snapshot" of the state of the bucket.
> Because manifests can contain billions of objects, jobs might take a long time to run, which can affect which version of an object that the job acts upon. Suppose that you overwrite an object with a new version while a job is running and you didn't specify a version ID for that object. In this case, Amazon S3 performs the operation on the latest version of the object, not on the version that existed when you created the job. The only way to avoid this behavior is to specify version IDs for the objects that are listed in the manifest.

### Generating a manifest automatically

You can direct Amazon S3 to generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for batch replication jobs that you create in the Amazon S3 console, or for any job type that you create by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information about Batch Replication, see Replicating existing objects with S3 Batch Replication.

To generate a manifest automatically, you specify the following elements as part of your job creation request:

- Information about the bucket that contains your source objects, including the bucket owner and Amazon Resource Name (ARN)
- Information about the manifest output, including a flag to create a manifest file, the output bucket owner, the ARN, the prefix, the file format, and the encryption type
- Optional criteria to filter objects by creation date, key name, size, storage class, and tags

### Object filter criteria

To filter the list of objects to be included in an automatically generated manifest, you can specify the following criteria. For more information, see JobManifestGeneratorFilter in the *Amazon S3 API Reference*.

**CreatedAfter**

If provided, the generated manifest includes only source bucket objects that were created after this time.

**CreatedBefore**

If provided, the generated manifest includes only source bucket objects that were created before this time.

**EligibleForReplication**

If provided, the generated manifest includes objects only if they are eligible for replication according to the replication configuration on the source bucket.

**KeyNameConstraint**

If provided, the generated manifest includes only source bucket objects whose object keys match the string constraints specified for **MatchAnySubstring**, **MatchAnyPrefix**, and **MatchAnySuffix**.

**MatchAnySubstring** – If provided, the generated manifest includes objects if the specified string appears anywhere within the object key string.

**MatchAnyPrefix** – If provided, the generated manifest includes objects if the specified string appears at the start of the object key string.

**MatchAnySuffix** – If provided, the generated manifest includes objects if the specified string appears at the end of the object key string.

**MatchAnyStorageClass**

If provided, the generated manifest includes only source bucket objects that are stored with the specified storage class.

**ObjectReplicationStatuses**

If provided, the generated manifest includes only source bucket objects that have one of the specified replication statuses.

**ObjectSizeGreaterThanBytes**

If provided, the generated manifest includes only source bucket objects whose file size is greater than the specified number of bytes.

**ObjectSizeLessThanBytes**

If provided, the generated manifest includes only source bucket objects whose file size is less than the specified number of bytes.

> **Note**
>
> You can't clone most jobs that have automatically generated manifests. Batch replication jobs can be cloned, except when they use the `KeyNameConstraint`, `MatchAnyStorageClass`, `ObjectSizeGreaterThanBytes`, or `ObjectSizeLessThanBytes` manifest filter criteria.

The syntax for specifying manifest criteria varies depending on the method that you use to create your job. For examples, see Creating a job.

### Creating a job

You can create S3 Batch Operations jobs by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

For more information about creating a job request, see Batch Operations job request elements.

### Prerequisites

Before you create a Batch Operations job, confirm that you have configured the relevant permissions. For more information, see Granting permissions for Amazon S3 Batch Operations.

### Using the S3 console

### To create a batch job

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create your job.

> **Note**
>
> For copy operations, you must create the job in the same Region as the destination bucket. For all other operations, you must create the job in the same Region as the objects in the manifest.

3. Choose **Batch Operations** on the left navigation pane of the Amazon S3 console.
4. Choose **Create job**.
5. View the **AWS Region** where you want to create your job.
6. Under **Manifest format**, choose the type of manifest object to use.

   - If you choose **S3 inventory report**, enter the path to the manifest.json object that Amazon S3 generated as part of the CSV-formatted Inventory report, and optionally the version ID for the manifest object if you want to use a version other than the most recent.
   - If you choose **CSV**, enter the path to a CSV-formatted manifest object. The manifest object must follow the format described in the console. You can optionally include the version ID for the manifest object if you want to use a version other than the most recent.

   > **Note**
   >
   > The Amazon S3 console supports automatic manifest generation for batch replication jobs only. For all other job types, if you want Amazon S3 to generate a manifest automatically based on filter criteria that you specify, you must configure your job using the AWS CLI, AWS SDKs, or Amazon S3 REST API.

7. Choose **Next**.
8. Under **Operation**, choose the operation that you want to perform on all objects listed in the manifest. Fill out the information for the operation you chose and then choose **Next**.
9. Fill out the information for **Configure additional options** and then choose **Next**.
10. For **Review**, verify the settings. If you need to make changes, choose **Previous**. Otherwise, choose **Create Job**.

**Using the AWS CLI**

Specify manifest

The following example shows how to create an S3 Batch Operations `S3PutObjectTagging` job that acts on objects that are listed in an existing manifest file.

**To create a Batch Operations `S3PutObjectTagging` job**

1. Use the following commands to create an AWS Identity and Access Management (IAM) role, and then create an IAM policy to assign the relevant permissions. The following role and policy grant Amazon S3 permission to add object tags, which you will need when you create the job in a subsequent step.

   a. Use the following example command to create an IAM role for Batch Operations to use. To use this example command, replace *`S3BatchJobRole`* with the name that you want to give to the role.

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   Record the role's Amazon Resource Name (ARN). You will need the ARN when you create a job.

   b. Use the following example command to create an IAM policy with the necessary permissions and attach it to the IAM role that you created in the previous step. For more information about the necessary permissions, see Granting permissions for Amazon S3 Batch Operations.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

To use this example command, replace the *user input placeholders* as follows:

- Replace *S3BatchJobRole* with the name of your IAM role. Make sure that this name matches the name that you used earlier.
- Replace *PutObjectTaggingBatchJobPolicy* with the name that you want to give your IAM policy.
- Replace *amzn-s3-demo-destination-bucket* with the name of the bucket that contains the objects that you want to apply tags to.
- Replace *DOC-EXAMPLE-MANIFEST-BUCKET* with the name of the bucket that contains the manifest.
- Replace *DOC-EXAMPLE-REPORT-BUCKET* with the name of the bucket where you want the completion report to be delivered to.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*"
      ]
    }
  ]
}'
```

2. Use the following example command to create an `S3PutObjectTagging` job.

   The `manifest.csv` file provides a list of bucket and object key values. The job applies the specified tags to the objects that are identified in the manifest. The `ETag` is the ETag of the `manifest.csv` object, which you can get from the Amazon S3 console. This request specifies the `no-confirmation-required` parameter, so that you can run the job without having to confirm it with the `update-job-status` command. For more information, see create-job in the *AWS CLI Command Reference*.

   To use this example command, replace the *`user input placeholders`* with your own information. Replace *`IAM-role`* with the ARN of the IAM role that you created earlier.

```
aws s3control create-job \
    --region us-west-2 \
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
```

```
{"ObjectArn":"arn:aws:s3:::my_manifests/
manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-
BUCKET","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
    --description "job description" \
    --no-confirmation-required
```

In response, Amazon S3 returns a job ID (for example, `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`). You will need the job ID to identify, monitor, and modify the job.

Generate manifest

The following example shows how to create an S3 Batch Operations `S3DeleteObjectTagging` job that automatically generates a manifest based on your object filter criteria. This criteria includes the creation date, key name, size, storage class, and tags.

**To create a Batch Operations `S3DeleteObjectTagging` job**

1. Use the following commands to create an AWS Identity and Access Management (IAM) role, and then create an IAM policy to assign permissions. The following role and policy grant Amazon S3 permission to delete object tags, which you will need when you create the job in a subsequent step.

   a. Use the following example command to create an IAM role for Batch Operations to use. To use this example command, replace *S3BatchJobRole* with the name that you want to give to the role.

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
```

```
        },
        "Action":"sts:AssumeRole"
      }
   ]
}'
```

Record the role's Amazon Resource Name (ARN). You will need the ARN when you create a job.

b. Use the following example command to create an IAM policy with the necessary permissions and attach it to the IAM role that you created in the previous step. For more information about the necessary permissions, see Granting permissions for Amazon S3 Batch Operations.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

To use this example command, replace the *user input placeholders* as follows:

- Replace *S3BatchJobRole* with the name of your IAM role. Make sure that this name matches the name that you used earlier.
- Replace *DeleteObjectTaggingBatchJobPolicy* with the name that you want to give your IAM policy.
- Replace *amzn-s3-demo-destination-bucket* with the name of the bucket that contains the objects that you want to apply tags to.
- Replace *DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET* with the name of the bucket where you want to save the manifest.
- Replace *DOC-EXAMPLE-REPORT-BUCKET* with the name of the bucket where you want the completion report to be delivered to.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
```

```
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    }
  ]
}'
```

2. Use the following example command to create the `S3DeleteObjectTagging` job.

   In this example, the values in the `--report` section specify the bucket, prefix, format, and scope of the job report that will be generated. The `--manifest -generator` section specifies information about the source bucket that contains the objects the job will act upon, information about the manifest output list that will be generated for the job, and filter criteria to narrow the scope of objects to be included in the manifest by creation date, name constraints, size, and storage class. The command also specifies the job's priority, IAM role, and AWS Region.

   For more information, see create-job in the *AWS CLI Command Reference*.

   To use this example command, replace the *`user input placeholders`* with your own information. Replace *`IAM-role`* with the ARN of the IAM role that you created earlier.

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
```

```
            "MatchAnyPrefix": [
              "prefix"
            ],
            "MatchAnySuffix": [
              "suffix"
            ]
          },
          "ObjectSizeGreaterThanBytes": 100,
          "ObjectSizeLessThanBytes": 200,
          "MatchAnyStorageClass": [
            "STANDARD",
            "STANDARD_IA"
          ]
        }
      }
    }' \
    --priority 2 \
    --role-arn IAM-role \
    --region us-east-1
```

In response, Amazon S3 returns a job ID (for example, `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`). You will need this job ID to identify, monitor, or modify the job.

### Using the AWS SDK for Java

Specify manifest

The following example shows how to create an S3 Batch Operations `S3PutObjectTagging` job that acts on objects that are listed in an existing manifest file. To use this example, replace the *`user input placeholders`* with your own information.

**Example**

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
```

```
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.*;

import java.util.UUID;
import java.util.ArrayList;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateJob {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String iamRoleArn = "IAM Role ARN";
        String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
        String uuid = UUID.randomUUID().toString();

        ArrayList tagSet = new ArrayList<S3Tag>();
        tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));


        try {
            JobOperation jobOperation = new JobOperation()
                    .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                            .withTagSet(tagSet)
                    );

            JobManifest manifest = new JobManifest()
                    .withSpec(new JobManifestSpec()
                            .withFormat("S3BatchOperations_CSV_20180820")
                            .withFields(new String[]{
                                    "Bucket", "Key"
                            }))
                    .withLocation(new JobManifestLocation()
                            .withObjectArn("arn:aws:s3:::my_manifests/manifest.csv")
                            .withETag("60e460c9d1046e73f7dde5043ac3ae85"));
            JobReport jobReport = new JobReport()
                    .withBucket(reportBucketName)
                    .withPrefix("reports")
                    .withFormat("Report_CSV_20180820")
                    .withEnabled(true)
                    .withReportScope("AllTasks");

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
```

```
                    .build();

            s3ControlClient.createJob(new CreateJobRequest()
                    .withAccountId(accountId)
                    .withOperation(jobOperation)
                    .withManifest(manifest)
                    .withReport(jobReport)
                    .withPriority(42)
                    .withRoleArn(iamRoleArn)
                    .withClientRequestToken(uuid)
                    .withDescription("job description")
                    .withConfirmationRequired(false)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Generate manifest

The following example shows how to create an S3 Batch Operations `s3PutObjectCopy` job that automatically generates a manifest based on object filter criteria, including the creation date, key name, and size. To use this example, replace the *`user input placeholders`* with your own information.

**Example**

```
        package aws.example.s3control;

    import com.amazonaws.AmazonServiceException;
    import com.amazonaws.SdkClientException;
    import com.amazonaws.auth.profile.ProfileCredentialsProvider;
    import com.amazonaws.services.s3control.AWSS3Control;
    import com.amazonaws.services.s3control.AWSS3ControlClient;
```

```
    import com.amazonaws.services.s3control.model.CreateJobRequest;
    import com.amazonaws.services.s3control.model.CreateJobResult;
    import com.amazonaws.services.s3control.model.JobManifestGenerator;
    import com.amazonaws.services.s3control.model.JobManifestGeneratorFilter;
    import com.amazonaws.services.s3control.model.JobOperation;
    import com.amazonaws.services.s3control.model.JobReport;
    import com.amazonaws.services.s3control.model.KeyNameConstraint;
    import com.amazonaws.services.s3control.model.S3JobManifestGenerator;
    import com.amazonaws.services.s3control.model.S3ManifestOutputLocation;
    import com.amazonaws.services.s3control.model.S3SetObjectTaggingOperation;
    import com.amazonaws.services.s3control.model.S3Tag;

    import java.time.Instant;
    import java.util.Date;
    import java.util.UUID;
    import java.util.ArrayList;

    import static com.amazonaws.regions.Regions.US_WEST_2;

    public class test {
        public static void main(String[] args) {
            String accountId = "012345678901";
            String iamRoleArn = "arn:aws:iam::012345678901:role/ROLE";
            String sourceBucketName = "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET";
            String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
            String manifestOutputBucketName = "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-
OUTPUT-BUCKET";
            String uuid = UUID.randomUUID().toString();
            long minimumObjectSize = 100L;

            ArrayList<S3Tag> tagSet = new ArrayList<>();
            tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));

            ArrayList<String> prefixes = new ArrayList<>();
            prefixes.add("s3KeyStartsWith");

            try {
                JobOperation jobOperation = new JobOperation()
                        .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                                .withTagSet(tagSet)
                        );
                S3ManifestOutputLocation manifestOutputLocation = new
 S3ManifestOutputLocation()
                        .withBucket(manifestOutputBucketName)
```

```
                        .withManifestPrefix("manifests")
                        .withExpectedManifestBucketOwner(accountId)
                        .withManifestFormat("S3InventoryReport_CSV_20211130");

                JobManifestGeneratorFilter jobManifestGeneratorFilter = new
JobManifestGeneratorFilter()
                        .withEligibleForReplication(true)
                        .withKeyNameConstraint(
                                new KeyNameConstraint()
                                        .withMatchAnyPrefix(prefixes))
                        .withCreatedBefore(Date.from(Instant.now()))
                        .withObjectSizeGreaterThanBytes(minimumObjectSize);

                S3JobManifestGenerator s3JobManifestGenerator = new
S3JobManifestGenerator()
                        .withEnableManifestOutput(true)
                        .withManifestOutputLocation(manifestOutputLocation)
                        .withFilter(jobManifestGeneratorFilter)
                        .withSourceBucket(sourceBucketName);

                JobManifestGenerator jobManifestGenerator = new
JobManifestGenerator()
                        .withS3JobManifestGenerator(s3JobManifestGenerator);

                JobReport jobReport = new JobReport()
                        .withBucket(reportBucketName)
                        .withPrefix("reports")
                        .withFormat("Report_CSV_20180820")
                        .withEnabled(true)
                        .withReportScope("AllTasks");

                AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                        .withCredentials(new ProfileCredentialsProvider())
                        .withRegion(US_WEST_2)
                        .build();

                CreateJobResult createJobResult = s3ControlClient.createJob(new
CreateJobRequest()
                        .withAccountId(accountId)
                        .withOperation(jobOperation)
                        .withManifestGenerator(jobManifestGenerator)
                        .withReport(jobReport)
                        .withPriority(42)
                        .withRoleArn(iamRoleArn)
```

```
                        .withClientRequestToken(uuid)
                        .withDescription("job description")
                        .withConfirmationRequired(true)
                );

                System.out.println("Created job " + createJobResult.getJobId());

            } catch (AmazonServiceException e) {
                // The call was transmitted successfully, but Amazon S3 couldn't
process
                // it and returned an error response.
                e.printStackTrace();
            } catch (SdkClientException e) {
                // Amazon S3 couldn't be contacted for a response, or the client
                // couldn't parse the response from Amazon S3.
                e.printStackTrace();
            }
        }
    }
```

### Using the REST API

You can use the REST API to create a Batch Operations job. For more information, see CreateJob in the *Amazon Simple Storage Service API Reference*.

### Job responses

If the `CreateJob` request succeeds, Amazon S3 returns a job ID. The job ID is a unique identifier that Amazon S3 generates automatically so that you can identify your Batch Operations job and monitor its status.

When you create a job through the AWS CLI, AWS SDKs, or REST API, you can set S3 Batch Operations to begin processing the job automatically. The job runs as soon as it's ready instead of waiting behind higher-priority jobs.

When you create a job through the Amazon S3 console, you must review the job details and confirm that you want to run the job before Batch Operations can begin to process it. If a job remains in the suspended state for over 30 days, it will fail.

# Operations supported by S3 Batch Operations

S3 Batch Operations supports several different operations. The topics in this section describe each of these operations.

## Copy objects

The **Copy** operation copies each object that is specified in the manifest. You can copy objects to a bucket in the same AWS Region or to a bucket in a different Region. S3 Batch Operations supports most options available through Amazon S3 for copying objects. These options include setting object metadata, setting permissions, and changing an object's storage class.

You can also use the Copy operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. For more information, see Encrypting objects with Amazon S3 Batch Operations.

When you copy objects, you can change the checksum algorithm used to calculate the checksum of the object. If objects don't have an additional checksum calculated, you can also add one by specifying the checksum algorithm for Amazon S3 to use. For more information, see Checking object integrity.

For more information about copying objects in Amazon S3 and the required and optional parameters, see Copying, moving, and renaming objects in this guide and CopyObject in the *Amazon Simple Storage Service API Reference*.

### Restrictions and limitations

- All source objects must be in one bucket.
- All destination objects must be in one bucket.
- You must have read permissions for the source bucket and write permissions for the destination bucket.
- Objects to be copied can be up to 5 GB in size.
- If you try to copy objects from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive classes to the S3 Standard storage class, you need to first restore these objects. For more information, see Restoring an archived object.
- Copy jobs must be created in the destination Region, which is the Region you intend to copy the objects to.

- All Copy options are supported except for conditional checks on ETags and server-side encryption with customer-provided encryption keys (SSE-C).
- If the buckets are unversioned, you will overwrite objects with the same key names.
- Objects are not necessarily copied in the same order as they appear in the manifest. For versioned buckets, if preserving current/non-current version order is important, you should copy all noncurrent versions first. Then, after the first job is complete, copy the current versions in a subsequent job.
- Copying objects to the Reduced Redundancy Storage (RRS) class is not supported.

### Copying objects using S3 Batch Operations

You can use S3 Batch Operations to create a PUT copy job to copy objects within the same account or to a different destination account. The following sections contain examples of how to store and use a manifest that is in a different account. In the first section, you can use Amazon S3 Inventory to deliver the inventory report to the destination account for use during job creation or, you can use a comma-separated values (CSV) manifest in the source or destination account, as shown in the second example. The third example shows how to use the Copy operation to activate S3 Bucket Key encryption on existing objects.

#### Copy Operation Examples

- Using an inventory report delivered to the destination account to copy objects across AWS accounts
- Using a CSV manifest to copy objects across AWS accounts
- Using S3 Batch Operations to encrypt objects with S3 Bucket Keys

#### Using an inventory report delivered to the destination account to copy objects across AWS accounts

You can use Amazon S3 Inventory to create an inventory report and use the report to create a list of objects to copy with S3 Batch Operations. For more information about using a CSV manifest in the source or destination account, see the section called “Using a CSV manifest to copy objects across AWS accounts”.

Amazon S3 Inventory generates inventories of the objects in a bucket. The resulting list is published to an output file. The bucket that is inventoried is called the *source bucket*, and the bucket where the inventory report file is stored is called the *destination bucket*.

The Amazon S3 Inventory report can be configured to be delivered to another AWS account. This allows S3 Batch Operations to read the inventory report when the job is created in the destination account.

For more information about Amazon S3 Inventory source and destination buckets, see Source and destination buckets.

The easiest way to set up an inventory is by using the AWS Management Console, but you can also use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

The following console procedure contains the high-level steps for setting up permissions for an S3 Batch Operations job. In this procedure, you copy objects from a source account to a destination account, with the inventory report stored in the destination account.

**To set up Amazon S3 Inventory for source and destination buckets owned by different accounts**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose a destination bucket to store the inventory report in.

   Decide on a destination manifest bucket for storing the inventory report. In this procedure, the *destination account* is the account that owns both the destination manifest bucket and the bucket that the objects are copied to.
3. Configure an inventory to list the objects in a source bucket and publish the list to the destination manifest bucket.

   Configure an inventory list for a source bucket. When you do this, you specify the destination bucket where you want the list to be stored. The inventory report for the source bucket is published to the destination bucket. In this procedure, the *source account* is the account that owns the source bucket.

   For information about how to use the console to configure an inventory or how to encrypt an inventory list file, see Configuring Amazon S3 Inventory.

   Choose **CSV** for the output format.

   When you enter information for the destination bucket, choose **Buckets in another account**. Then enter the name of the destination manifest bucket. Optionally, you can enter the account ID of the destination account.

After the inventory configuration is saved, the console displays a message similar to the following:

Amazon S3 could not create a bucket policy on the destination bucket. Ask the destination bucket owner to add the following bucket policy to allow Amazon S3 to place data in that bucket.

The console then displays a bucket policy that you can use for the destination bucket.

4. Copy the destination bucket policy that appears on the console.
5. In the destination account, add the copied bucket policy to the destination manifest bucket where the inventory report is stored.
6. Create a role in the destination account that is based on the S3 Batch Operations trust policy. For more information about the trust policy, see Trust policy.

   For more information about creating a role, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

   Enter a name for the role (the example role uses the name `BatchOperationsDestinationRoleCOPY`). Choose the **S3** service, and then choose the **S3 bucket Batch Operations** use case, which applies the trust policy to the role.

   Then choose **Create policy** to attach the following policy to the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
```

```
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectDestinationManifestBucket/*"
      ]
    }
  ]
}
```

The role uses the policy to grant `batchoperations.s3.amazonaws.com` permission to read the manifest in the destination bucket. It also grants permissions to GET objects, access control lists (ACLs), tags, and versions in the source object bucket. And it grants permissions to PUT objects, ACLs, tags, and versions into the destination object bucket.

7. In the source account, create a bucket policy for the source bucket that grants the role that you created in the previous step to GET objects, ACLs, tags, and versions in the source bucket. This step allows S3 Batch Operations to get objects from the source bucket through the trusted role.

   The following is an example of the bucket policy for the source account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
```

```
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

8. After the inventory report is available, create an S3 Batch Operations PUT object copy job in the destination account, choosing the inventory report from the destination manifest bucket. You need the ARN for the role that you created in the destination account.

   For general information about creating a job, see Creating an S3 Batch Operations job.

   For information about creating a job using the console, see Creating an S3 Batch Operations job.

### Using a CSV manifest to copy objects across AWS accounts

You can use a CSV manifest that's stored in the source account to copy objects across AWS accounts with S3 Batch Operations. To use an S3 Inventory report as a manifest, see the section called "Using an inventory report to copy objects across AWS accounts".

For an example of the CSV format for manifest files, see the section called "Creating a manifest file".

The following procedure shows how to set up permissions when using an S3 Batch Operations job to copy objects from a source account to a destination account with a CSV manifest file that's stored in the source account.

**To use a CSV manifest to copy objects across AWS accounts**

1. Create a role in the destination account that is based on the S3 Batch Operations trust policy. In this procedure, the *destination account* is the account that the objects are being copied to.

   For more information about the trust policy, see Trust policy.

   For more information about creating a role, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

   If you create the role by using the console, enter a name for the role (the example role uses the name `BatchOperationsDestinationRoleCOPY`). Choose the **S3** service, and then choose the **S3 bucket Batch Operations** use case, which applies the trust policy to the role.

Then choose **Create policy** to attach the following policy to the role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectSourceManifestBucket/*"
      ]
    }
  ]
}
```

Using the policy, the role grants `batchoperations.s3.amazonaws.com` permission to read the manifest in the source manifest bucket. It grants permissions to GET objects, access control lists (ACLs), tags, and versions in the source object bucket. It also grants permissions to PUT objects, ACLs, tags, and versions into the destination object bucket.

2. In the source account, create a bucket policy for the bucket that contains the manifest to grant the role that you created in the previous step permissions to GET objects and versions in the source manifest bucket.

   This step allows S3 Batch Operations to read the manifest by using the trusted role. Apply the bucket policy to the bucket that contains the manifest.

The following is an example of the bucket policy to apply to the source manifest bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceManfiestRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::DestinationAccountNumber:user/ConsoleUserCreatingJob",

 "arn:aws:iam::DestinationAccountNumber:role/BatchOperationsDestinationRoleCOPY"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceManifestBucket/*"
    }
  ]
}
```

This policy also grants permissions to allow a console user who is creating a job in the destination account the same permissions in the source manifest bucket through the same bucket policy.

3. In the source account, create a bucket policy for the source bucket that grants the role that you created permissions to GET objects, ACLs, tags, and versions in the source object bucket. S3 Batch Operations can then get objects from the source bucket through the trusted role.

   The following is an example of the bucket policy for the bucket that contains the source objects:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS":
 "arn:aws:iam::DestinationAccountNumber:role/BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

4. Create an S3 Batch Operations job in the destination account. You need the Amazon Resource Name (ARN) for the role that you created in the destination account. For more information about creating a job, see Creating an S3 Batch Operations job.

### Using S3 Batch Operations to encrypt objects with S3 Bucket Keys

In this section, you use the Amazon S3 Batch Operations Copy operation to identify and activate S3 Bucket Keys encryption on existing objects. For more information about S3 Bucket Keys, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys and Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects.

Topics covered in this example include the following:

#### Topics

**Prerequisites**

To follow along with the steps in this procedure, you must have an AWS account and at least one S3 bucket to hold your working files and encrypted results. You might also find much of the existing S3 Batch Operations documentation useful, including the following topics:

- S3 Batch Operations basics
- Creating an S3 Batch Operations job
- Operations supported by S3 Batch Operations
- Managing S3 Batch Operations jobs

**Step 1: Get your list of objects using Amazon S3 Inventory**

To get started, identify the S3 bucket that contains the objects to encrypt, and get a list of its contents. An Amazon S3 Inventory report is the most convenient and affordable way to do this. The report provides the list of the objects in a bucket along with associated metadata. The **source bucket** refers to the inventoried bucket, and the **destination bucket** refers to the bucket where you store the inventory report file. For more information about Amazon S3 Inventory source and destination buckets, see Amazon S3 Inventory.

The easiest way to set up an inventory is by using the AWS Management Console. But you can also use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs. Before following these steps, be sure to sign in to the console and open the Amazon S3 console at https://console.aws.amazon.com/s3/. If you encounter permission denied errors, add a bucket policy to your destination bucket. For more information, see Grant permissions for S3 Inventory and S3 analytics.

**To get a list of objects using S3 Inventory**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and choose a bucket that contains objects to encrypt.
3. On the **Management** tab, navigate to the **Inventory configurations** section, and choose **Create inventory configuration**.
4. Give your new inventory a name, enter the name of the destination S3 bucket, and optionally create a destination prefix for Amazon S3 to assign objects in that bucket.
5. For **Output format**, choose **CSV**.

6. (Optional) In the **Additional fields – *optional*** section, choose **Encryption** and any other report fields that interest you. Set the frequency for report deliveries to **Daily** so that the first report is delivered to your bucket sooner.
7. Choose **Create** to save your configuration.

Amazon S3 can take up to 48 hours to deliver the first report, so check back when the first report arrives. After you receive your first report, proceed to the next section to filter your S3 Inventory report's contents. If you no longer want to receive inventory reports for this bucket, delete your S3 Inventory configuration. Otherwise, S3 delivers reports on a daily or weekly schedule.

An inventory list isn't a single point-in-time view of all objects. Inventory lists are a rolling snapshot of bucket items, which are eventually consistent (for example, the list might not include recently added or deleted objects). Combining S3 Inventory and S3 Batch Operations works best when you work with static objects, or with an object set that you created two or more days ago. To work with more recent data, use the ListObjectsV2 (GET Bucket) API operation to build your list of objects manually. If needed, repeat the process for the next few days or until your inventory report shows the desired status for all keys.

**Step 2: Filter your object list with S3 Select**

After you receive your S3 Inventory report, you can filter the report's contents to list only the objects that aren't encrypted with S3 Bucket Keys. If you want all your bucket's objects encrypted with S3 Bucket Keys, you can ignore this step. However, filtering your S3 Inventory report at this stage saves you the time and expense of re-encrypting objects that you previously encrypted.

Although the following steps show how to filter using Amazon S3 Select, you can also use Amazon Athena. To decide which tool to use, look at your S3 Inventory report's `manifest.json` file. This file lists the number of data files that are associated with that report. If the number is large, use Amazon Athena because it runs across multiple S3 objects, whereas S3 Select works on one object at a time. For more information about using Amazon S3 and Athena together, see Querying Amazon S3 Inventory with Amazon Athena and Using Athena in the blog post Encrypting objects with Amazon S3 Batch Operations.

**To filter your S3 Inventory report using S3 Select**

1. Open the `manifest.json` file from your inventory report and look at the `fileSchema` section of the JSON. This informs the query that you run on the data.

The following JSON is an example `manifest.json` file for a CSV-formatted inventory on a bucket with versioning enabled. Depending on how you configured your inventory report, your manifest might look different.

```
  {
    "sourceBucket": "batchoperationsdemo",
    "destinationBucket": "arn:aws:s3:::testbucket",
    "version": "2021-05-22",
    "creationTimestamp": "1558656000000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 BucketKeyStatus",
    "files": [
      {
        "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
f053-4c16-8c75-6100f8892202.csv.gz",
        "size": 72691,
        "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
      }
    ]
  }
```

If versioning isn't activated on the bucket, or if you choose to run the report for the latest versions, the `fileSchema` is `Bucket`, `Key`, and `BucketKeyStatus`.

If versioning *is* activated, depending on how you set up the inventory report, the `fileSchema` might include the following: `Bucket`, `Key`, `VersionId`, `IsLatest`, `IsDeleteMarker`, `BucketKeyStatus`. So pay attention to columns 1, 2, 3, and 6 when you run your query.

S3 Batch Operations needs the bucket, key, and version ID as inputs to perform the job, in addition to the field to search by, which is `BucketKeyStatus`. You don't need the version ID field, but it helps to specify it when you operate on a versioned bucket. For more information, see Working with objects in a versioning-enabled bucket.

2. Locate the data files for the inventory report. The `manifest.json` object lists the data files under **files**.
3. After you locate and select the data file in the S3 console, choose **Actions**, and then choose **Query with S3 Select**.
4. Keep the preset **CSV**, **Comma**, and **GZIP** fields selected, and choose **Next**.

5. To review your inventory report's format before proceeding, choose **Show file preview**.
6. Enter the columns to reference in the SQL expression field, and choose **Run SQL**. The following expression returns columns 1–3 for all objects without an S3 Bucket Key configured.

   ```
   select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
   ```

   The following are example results.

   ```
   batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
   batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
   batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
   batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
   batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
   batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
   batchoperationsdemo,200907200065HQ
%7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
   batchoperationsdemo,200907200076HQ
%7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
   batchoperationsdemo,201103190002HQ
%7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
   ```

7. Download the results, save them into a CSV format, and upload them to Amazon S3 as your list of objects for the S3 Batch Operations job.
8. If you have multiple manifest files, run **Query with S3 Select** on those also. Depending on the size of the results, you could combine the lists and run a single S3 Batch Operations job or run each list as a separate job.

   Consider the price of running each S3 Batch Operations job when you decide the number of jobs to run.

### Step 3: Set up and run your S3 Batch Operations job

Now that you have your filtered CSV lists of S3 objects, you can begin the S3 Batch Operations job to encrypt the objects with S3 Bucket Keys.

A *job* refers collectively to the list (manifest) of objects provided, the operation performed, and the specified parameters. The easiest way to encrypt this set of objects is by using the PUT copy operation and specifying the same destination prefix as the objects listed in the manifest. This either overwrites the existing objects in an unversioned bucket or, with versioning turned on, creates a newer, encrypted version of the objects.

As part of copying the objects, specify that Amazon S3 should encrypt the object with SSE-KMS encryption and S3. This job copies the objects, so all your objects show an updated creation date upon completion, regardless of when you originally added them to S3. Also specify the other properties for your set of objects as part of the S3 Batch Operations job, including object tags and storage class.

**Substeps**

- Set up your IAM policy
- Set up your Batch Operations IAM role
- Turn on S3 Bucket Keys for an existing bucket
- Create your Batch Operations job
- Run your Batch Operations job
- Things to note

**Set up your IAM policy**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policy**, and then choose **Create Policy**.
3. Choose the **JSON** tab. Choose **Edit policy** and add the example IAM policy that appears in the following code block.

   After copying the policy example into your IAM Console, replace the following:

   a. Replace *SOURCE_BUCKET_FOR_COPY* with the name of your source bucket.

   b. Replace *DESTINATION_BUCKET_FOR_COPY* with the name of your destination bucket.

   c. Replace *MANIFEST_KEY* with the name of your manifest object.

   d. Replace *REPORT_BUCKET* with the name of the bucket where you want to save reports.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
```

```
            "s3:PutObjectTagging",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionTagging",
            "s3:PutObjectVersionAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:GetObjectTagging",
            "s3:GetObjectVersion",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
        ],
        "Resource": [
            "arn:aws:s3:::SOURCE_BUCKET_FOR_COPY/*",
            "arn:aws:s3:::DESTINATION_BUCKET_FOR_COPY/*"
        ]
    },
    {
        "Sid": "ReadManifest",
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
        ],
        "Resource": "arn:aws:s3:::MANIFEST_KEY"
    },
    {
        "Sid": "WriteReport",
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::REPORT_BUCKET/*"
    }
  ]
}
```

4. Choose **Next: Tags**.
5. Add any tags that you want (optional), and choose **Next: Review**.
6. Add a policy name, optionally add a description, and choose **Create policy**.
7. Choose **Review policy** and **Save changes**.

8. With your S3 Batch Operations policy now complete, the console returns you to the IAM **Policies** page. Filter on the policy name, choose the button to the left of the policy name, choose **Policy actions**, and choose **Attach**.

   To attach the newly created policy to an IAM role, select the appropriate users, groups, or roles in your account and choose **Attach policy**. This takes you back to the IAM console.

### Set up your Batch Operations IAM role

1. On the IAM Console, in the navigation pane, choose **Roles**, and then choose **Create role**.
2. Choose **AWS service**, **S3**, and **S3 Batch Operations**. Then choose **Next: Permissions**.
3. Start entering the name of the IAM **policy** that you just created. Select the check box by the policy name when it appears, and choose **Next: Tags**.
4. (Optional) Add tags or keep the key and value fields blank for this exercise. Choose **Next: Review**.
5. Enter a role name, and accept the default description or add your own. Choose **Create role**.
6. Ensure that the user creating the job has the permissions in the following example.

   Replace *{ACCOUNT-ID}* with your AWS account ID and *{IAM_ROLE_NAME}* with the name that you plan to apply to the IAM role that you will create in the Batch Operations job creation step later. For more information, see Granting permissions for Amazon S3 Batch Operations.

```
{
"Sid": "AddIamPermissions",
"Effect": "Allow",
"Action": [
"iam:GetRole",
"iam:PassRole"
],
"Resource": "arn:aws:iam::ACCOUNT-ID:role/IAM_ROLE_NAME"
}
```

### Turn on S3 Bucket Keys for an existing bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket that you want to turn on an S3 Bucket Key for.
3. Choose **Properties**.

4. Under **Default encryption**, choose **Edit**.
5. Under **Encryption type**, you can choose between **Amazon S3 managed keys (SSE-S3)** and **AWS Key Management Service key (SSE-KMS)**.
6. If you chose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify the AWS KMS key through one of the following options.

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**. From the list of available keys, choose a symmetric encryption KMS key in the same Region as your bucket. Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.
7. Under **Bucket Key**, choose **Enable**, and then choose **Save changes**.

Now that an S3 Bucket Key is turned on at the bucket level, objects that are uploaded, modified, or copied into this bucket will inherit this encryption configuration by default. This includes objects that are copied by using Amazon S3 Batch Operations.

**Create your Batch Operations job**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Batch Operations**, and then choose **Create Job**.
3. Choose the **Region** where you store your objects, and choose **CSV** as the manifest type.
4. Enter the path or navigate to the CSV manifest file that you created earlier from S3 Select (or Athena) results. If your manifest contains version IDs, select that box. Choose **Next**.
5. Choose the **Copy** operation, and choose the copy destination bucket. You can keep server-side encryption disabled. As long as the bucket destination has S3 Bucket Keys enabled, the copy operation applies S3 Bucket Keys at the destination bucket.
6. (Optional) Choose a storage class and the other parameters as desired. The parameters that you specify in this step apply to all operations performed on the objects that are listed in the manifest. Choose **Next**.
7. To configure server-side encryption, follow these steps:

   a. Under **Server-side encryption**, choose one of the following:

- To keep the bucket settings for default server-side encryption of objects when storing them in Amazon S3, choose **Do not specify an encryption key**. As long as the bucket destination has S3 Bucket Keys enabled, the copy operation applies an S3 Bucket Key at the destination bucket.

  > **Note**
  >
  > If the bucket policy for the specified destination requires objects to be encrypted before storing them in Amazon S3, you must specify an encryption key. Otherwise, copying objects to the destination will fail.

- To encrypt objects before storing them in Amazon S3, choose **Specify an encryption key**.

b. Under **Encryption settings**, if you choose **Specify an encryption key**, you must choose either **Use destination bucket settings for default encryption** or **Override destination bucket settings for default encryption**.

c. If you choose **Override destination bucket settings for default encryption**, you must configure the following encryption settings.

   i. Under **Encryption type**, you must choose either **Amazon S3 managed keys (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**. SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt each object. SSE-KMS provides you with more control over your key. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3) and Using server-side encryption with AWS KMS keys (SSE-KMS).

   ii. If you choose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify your AWS KMS key through one of the following options.

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose a symmetric encryption KMS key in the same Region as your bucket. Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

iii. Under **Bucket Key**, choose **Enable**. The copy operation applies an S3 Bucket Key at the destination bucket.

8. Give your job a description (or keep the default), set its priority level, choose a report type, and specify the **Path to completion report destination**.
9. In the **Permissions** section, be sure to choose the Batch Operations IAM role that you defined earlier. Choose **Next**.
10. Under **Review**, verify the settings. If you want to make changes, choose **Previous**. After confirming the Batch Operations settings, choose **Create job**.

    For more information, see Creating an S3 Batch Operations job.

### Run your Batch Operations job

The setup wizard automatically returns you to the S3 Batch Operations section of the Amazon S3 console. Your new job transitions from the **New** state to the **Preparing** state as S3 begins the process. During the Preparing state, S3 reads the job's manifest, checks it for errors, and calculates the number of objects.

1. Choose the refresh button in the Amazon S3 console to check progress. Depending on the size of the manifest, reading can take minutes or hours.
2. After S3 finishes reading the job's manifest, the job moves to the **Awaiting your confirmation** state. Choose the option button to the left of the Job ID, and choose **Run job**.
3. Check the settings for the job, and choose **Run job** in the bottom-right corner.

   After the job begins running, you can choose the refresh button to check progress through the console dashboard view or by selecting the specific job.
4. When the job is complete, you can view the **Successful** and **Failed** object counts to confirm that everything performed as expected. If you enabled job reports, check your job report for the exact cause of any failed operations.

   You can also perform these steps by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information about tracking job status and completion reports, see Tracking job status and completion reports.

### Things to note

Consider the following issues when you use S3 Batch Operations to encrypt objects with S3 Bucket Keys:

- You will be charged for S3 Batch Operations jobs, objects, and requests in addition to any charges associated with the operation that S3 Batch Operations performs on your behalf, including data transfers, requests, and other charges. For more information, see Amazon S3 pricing.
- If you use a versioned bucket, each S3 Batch Operations job performed creates new encrypted versions of your objects. It also maintains the previous versions without an S3 Bucket Key configured. To delete the old versions, set up an S3 Lifecycle expiration policy for noncurrent versions as described in Lifecycle configuration elements.
- The copy operation creates new objects with new creation dates, which can affect lifecycle actions like archiving. If you copy all objects in your bucket, all the new copies have identical or similar creation dates. To further identify these objects and create different lifecycle rules for various data subsets, consider using object tags.

### Summary

In this section, you sorted existing objects to filter out already encrypted data. Then you applied the S3 Bucket Key feature on unencrypted objects by using S3 Batch Operations to copy existing data to a bucket with an S3 Bucket Key activated. This process can save you time and money while allowing you to complete operations such as encrypting all existing objects.

For more information about S3 Batch Operations, see Performing large-scale batch operations on Amazon S3 objects.

For examples that show the copy operation with tags using the AWS CLI and AWS SDK for Java, see Creating a Batch Operations job with job tags used for labeling.

## Invoke AWS Lambda function

The **Invoke AWS Lambda function** initiates AWS Lambda functions to perform custom actions on objects that are listed in a manifest. This section describes how to create a Lambda function to use with S3 Batch Operations and how to create a job to invoke the function. The S3 Batch Operations job uses the `LambdaInvoke` operation to run a Lambda function on every object listed in a manifest.

You can work with S3 Batch Operations for Lambda using the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST APIs. For more information about using Lambda, see Getting Started with AWS Lambda in the *AWS Lambda Developer Guide*.

The following sections explain how you can get started using S3 Batch Operations with Lambda.

**Topics**



### Using Lambda with Amazon S3 batch operations

When using S3 Batch Operations with AWS Lambda, you must create new Lambda functions specifically for use with S3 Batch Operations. You can't reuse existing Amazon S3 event-based functions with S3 Batch Operations. Event functions can only receive messages; they don't return messages. The Lambda functions that are used with S3 Batch Operations must accept and return messages. For more information about using Lambda with Amazon S3 events, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

You create an S3 Batch Operations job that invokes your Lambda function. The job runs the same Lambda function on all of the objects listed in your manifest. You can control what versions of your Lambda function to use while processing the objects in your manifest. S3 Batch Operations support unqualified Amazon Resource Names (ARNs), aliases, and specific versions. For more information, see Introduction to AWS Lambda Versioning in the *AWS Lambda Developer Guide*.

If you provide the S3 Batch Operations job with a function ARN that uses an alias or the `$LATEST` qualifier, and you update the version that either of those points to, S3 Batch Operations starts calling the new version of your Lambda function. This can be useful when you want to update functionality part of the way through a large job. If you don't want S3 Batch Operations to change the version that is used, provide the specific version in the `FunctionARN` parameter when you create your job.

### Using Lambda and Amazon S3 batch operations with directory buckets

Directory buckets are a type of Amazon S3 bucket that is designed for workloads or performance-critical applications that require consistent single-digit millisecond latency. For more information, see Directory buckets.

There are special requirements for using Amazon S3 batch operations to invoke Lambda functions that act on directory buckets. For example, you must structure your Lambda request using an updated JSON schema, and specify `InvocationSchemaVersion` **`2.0`** when you create the job. This updated schema allows you to specify optional key-value pairs for `UserArguments`, which you can use to modify certain parameters of existing Lambda functions. For more information, see Automate object processing in Amazon S3 directory buckets with S3 Batch Operations and AWS Lambda in the *AWS Storage Blog*.

### Response and result codes

S3 Batch Operations invokes the Lambda function with one or more keys, each of which has a `TaskID` associated with it. S3 Batch Operations expects a per-key result code from Lambda functions. Any task IDs sent in the request which are not returned with a per-key result code will be be given the result code from the `treatMissingKeysAs` field. `treatMissingKeysAs` is an optional request field and defaults to `TemporaryFailure`. The following table contains the other possible result codes and values for the `treatMissingKeysAs` field.

| Response code | Description |
| --- | --- |
| `Succeeded` | The task completed normally. If you requested a job completion report, the task's result string is included in the report. |
| `TemporaryFailure` | The task suffered a temporary failure and will be redriven before the job completes. The result string is ignored. If this is the final redrive, the error message is included in the final report. |
| `PermanentFailure` | The task suffered a permanent failure. If you requested a job-completion report, the task is marked as `Failed` and includes the error |

| Response code | Description |
| --- | --- |
| | message string. Result strings from failed tasks are ignored. |

### Creating a Lambda function to use with S3 Batch Operations

This section provides example AWS Identity and Access Management (IAM) permissions that you must use with your Lambda function. It also contains an example Lambda function to use with S3 Batch Operations. If you have never created a Lambda function before, see Tutorial: Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

You must create Lambda functions specifically for use with S3 Batch Operations. You can't reuse existing Amazon S3 event-based Lambda functions. This is because Lambda functions that are used for S3 Batch Operations must accept and return special data fields.

**Important**

AWS Lambda functions written in Java accept either RequestHandler or RequestStreamHandler handler interfaces. However, to support S3 Batch Operations request and response format, AWS Lambda requires the `RequestStreamHandler` interface for custom serialization and deserialization of a request and response.
This interface allows Lambda to pass an InputStream and OutputStream to the Java `handleRequest` method.
Be sure to use the `RequestStreamHandler` interface when using Lambda functions with S3 Batch Operations. If you use a `RequestHandler` interface, the batch job will fail with "Invalid JSON returned in Lambda payload" in the completion report.
For more information, see Handler interfaces in the *AWS Lambda User Guide*.

### Example IAM permissions

The following are examples of the IAM permissions that are necessary to use a Lambda function with S3 Batch Operations.

### Example — S3 Batch Operations trust policy

The following is an example of the trust policy that you can use for the Batch Operations IAM role. This IAM role is specified when you create the job and gives Batch Operations permission to assume the IAM role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "batchoperations.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

### Example — Lambda IAM policy

The following is an example of an IAM policy that gives S3 Batch Operations permission to invoke the Lambda function and read the input manifest.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

### Example request and response

This section provides request and response examples for the Lambda function.

### Example Request

The following is a JSON example of a request for the Lambda function.

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:awsexamplebucket1"
        }
    ]
}
```

### Example Response

The following is a JSON example of a response for the Lambda function.

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
  ]
}
```

### Example Lambda function for S3 Batch Operations

The following example Python Lambda removes a delete marker from a versioned object.

As the example shows, keys from S3 Batch Operations are URL encoded. To use Amazon S3 with other AWS services, it's important that you URL decode the key that is passed from S3 Batch Operations.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
```

```
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
```

```
                    )
                    logger.info(result_string)
                else:
                    raise
        else:
            raise ValueError(
                f"The x-amz-delete-marker header is either not "
                f"present or is not 'true'."
            )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

### Creating an S3 Batch Operations job that invokes a Lambda function

When creating an S3 Batch Operations job to invoke a Lambda function, you must provide the following:

- The ARN of your Lambda function (which might include the function alias or a specific version number)
- An IAM role with permission to invoke the function
- The action parameter `LambdaInvokeFunction`

For more information about creating an S3 Batch Operations job, see Creating an S3 Batch Operations job and Operations supported by S3 Batch Operations.

The following example creates an S3 Batch Operations job that invokes a Lambda function using the AWS CLI.

```
aws s3control create-job
    --account-id <AccountID>
    --operation  '{"LambdaInvoke": { "FunctionArn":
 "arn:aws:lambda:Region:AccountID:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::ManifestLocation","ETag":"ManifestETag"}}'
    --report
 '{"Bucket":"arn:aws:s3:::awsexamplebucket1","Format":"Report_CSV_20180820","Enabled":true,"Pre
    --priority 2
    --role-arn arn:aws:iam::AccountID:role/BatchOperationsRole
    --region Region
    --description “Lambda Function"
```

**Providing task-level information in Lambda manifests**

When you use AWS Lambda functions with S3 Batch Operations, you might want additional data to accompany each task/key that is operated on. For example, you might want to have both a source object key and new object key provided. Your Lambda function could then copy the source key to a new S3 bucket under a new name. By default, Amazon S3 batch operations let you specify only the destination bucket and a list of source keys in the input manifest to your job. The following describes how you can include additional data in your manifest so that you can run more complex Lambda functions.

To specify per-key parameters in your S3 Batch Operations manifest to use in your Lambda function's code, use the following URL-encoded JSON format. The `key` field is passed to your Lambda function as if it were an Amazon S3 object key. But it can be interpreted by the Lambda function to contain other values or multiple keys, as shown following.

> **Note**
>
> The maximum number of characters for the `key` field in the manifest is 1,024.

**Example — manifest substituting the "Amazon S3 keys" with JSON strings**

The URL-encoded version must be provided to S3 Batch Operations.

```
my-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
my-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
my-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

**Example — manifest URL-encoded**

This URL-encoded version must be provided to S3 Batch Operations. The non-URL-encoded version does not work.

```
my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A
%20%22newObject3Key%22%7D
```

**Example — Lambda function with manifest format writing results to the job report**

This URL-encoded manifest example contains pipe-delimited object keys for the following Lambda function to parse.

```
my-bucket,object1key%7Clower
my-bucket,object2key%7Cupper
my-bucket,object3key%7Creverse
my-bucket,object4key%7Cdelete
```

This Lambda function shows how to parse a pipe-delimited task that is encoded into the S3 Batch Operations manifest. The task indicates which revision operation is applied to the specified object.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")
```

```
def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)

    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
```

```
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

### Learning from S3 Batch Operations tutorial

The following tutorial presents complete end-to-end procedures for some Batch Operations tasks with Lambda.

- Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

## Replace all object tags

The **Replace all object tags** operation replaces the Amazon S3 object tags on every object listed in the manifest. An Amazon S3 object tag is a key-value pair of strings that you can use to store metadata about an object.

To create a Replace all object tags job, you provide a set of tags that you want to apply. S3 Batch Operations applies the same set of tags to every object. The tag set that you provide replaces whatever tag sets are already associated with the objects in the manifest. S3 Batch Operations does not support adding tags to objects while leaving the existing tags in place.

If the objects in your manifest are in a versioned bucket, you can apply the tag set to specific versions of every object. You do this by specifying a version ID for every object in the manifest. If you don't include a version ID for any object, then S3 Batch Operations applies the tag set to the latest version of every object.

### Restrictions and limitations

- The AWS Identity and Access Management (IAM) role that you specify to run the Batch Operations job must have permissions to perform the underlying Amazon S3 Replace all object tags operation. For more information about the permissions required, see PutObjectTagging in the *Amazon Simple Storage Service API Reference*.
- S3 Batch Operations uses the Amazon S3 PutObjectTagging operation to apply tags to each object in the manifest. All restrictions and limitations that apply to the underlying operation also apply to S3 Batch Operations jobs.

For more information about using the console to create jobs, see Creating an S3 batch operations job.

For more information about object tagging, see Categorizing your storage using tags in this guide, and see PutObjectTagging, GetObjectTagging, and DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.

## Delete all object tags

The **Delete all object tags** operation removes all Amazon S3 object tag sets currently associated with the objects that are listed in the manifest. S3 Batch Operations does not support deleting tags from objects while keeping other tags in place.

If the objects in your manifest are in a versioned bucket, you can remove the tag sets from a specific version of an object. Do this by specifying a version ID for every object in the manifest. If you don't include a version ID for an object, S3 Batch Operations removes the tag set from the latest version of every object.

For more information about Batch Operations manifests, see Specifying a manifest.

> **Warning**
>
> Running this job removes all object tag sets on every object listed in the manifest.

### Restrictions and limitations

- The AWS Identity and Access Management (IAM) role that you specify to run the job must have permissions to perform the underlying Amazon S3 Delete object tagging operation. For more information, see DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.
- S3 Batch Operations uses the Amazon S3 DeleteObjectTagging operation to remove the tag sets from every object in the manifest. All restrictions and limitations that apply to the underlying operation also apply to S3 Batch Operations jobs.

For more information about creating jobs, see Creating an S3 Batch Operations job.

For more details about object tagging, see Replace all object tags in this guide, and PutObjectTagging, GetObjectTagging, and DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.

## Replace access control list

The **Replace access control list (ACL)** operation replaces the Amazon S3 access control lists (ACLs) for every object that is listed in the manifest. Using ACLs, you can define who can access an object and what actions they can perform.

S3 Batch Operations support custom ACLs that you define and canned ACLs that Amazon S3 provides with a predefined set of access permissions.

If the objects in your manifest are in a versioned bucket, you can apply the ACLs to specific versions of every object. You do this by specifying a version ID for every object in the manifest. If you don't include a version ID for any object, then S3 Batch Operations applies the ACL to the latest version of the object.

For more information about ACLs in Amazon S3, Access control list (ACL) overview.

### S3 Block Public Access

If you want to limit public access to all objects in a bucket, you should use Amazon S3 Block Public Access instead of S3 Batch Operations. Block Public Access can limit public access on a per-bucket or account-wide basis with a single, simple operation that takes effect quickly. This makes it a better choice when your goal is to control public access to all objects in a bucket or account. Use S3 Batch Operations when you need to apply a customized ACL to every object in the manifest. For more information about S3 Block Public Access, see Blocking public access to your Amazon S3 storage.

### S3 Object Ownership

If the objects in the manifest are in a bucket that uses the bucket owner enforced setting for Object Ownership, the **Replace access control list (ACL)** operation can only specify object ACLs that grant full control to the bucket owner. The operation can't grant object ACL permissions to other AWS accounts or groups. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

### Restrictions and limitations

- The role that you specify to run the Replace access control list job must have permissions to perform the underlying Amazon S3 `PutObjectAcl` operation. For more information about the permissions required, see PutObjectAcl in the *Amazon Simple Storage Service API Reference*.

- S3 Batch Operations uses the Amazon S3 `PutObjectAcl` operation to apply the specified ACL to every object in the manifest. Therefore, all restrictions and limitations that apply to the underlying PutObjectAcl operation also apply to S3 Batch Operations Replace access control list jobs.

## Restore objects with Batch Operations

The **Restore** operation initiates restore requests for the archived Amazon S3 objects that are listed in your manifest. The following archived objects must be restored before they can be accessed in real time:

- Objects archived in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes
- Objects archived through the S3 Intelligent-Tiering storage class in the Archive Access or Deep Archive Access tiers

Using an S3 Initiate Restore Object operation in your S3 Batch Operations job results in a restore request for every object that is specified in the manifest.

> **Important**
>
> The S3 Initiate Restore Object job only *initiates* the request to restore objects. S3 Batch Operations reports the job as complete for each object after the request is initiated for that object. Amazon S3 doesn't update the job or otherwise notify you when the objects have been restored. However, you can use S3 Event Notifications to receive notifications when the objects are available in Amazon S3. For more information, see Amazon S3 Event Notifications.

When you create an S3 Initiate Restore Object job, the following arguments are available:

**ExpirationInDays**

This argument specifies how long the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive object remains available in Amazon S3. Initiate Restore Object jobs that target S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive objects require that you set `ExpirationInDays` to 1 or greater.

> **Important**
>
> Do not set `ExpirationInDays` when creating S3 Initiate Restore Object operation jobs that target S3 Intelligent-Tiering Archive Access and Deep Archive Access tier objects. Objects in S3 Intelligent-Tiering archive access tiers are not subject to restore expiration, so specifying `ExpirationInDays` results in a restore request failure.

**GlacierJobTier**

Amazon S3 can restore objects by using one of three different retrieval tiers: `EXPEDITED`, `STANDARD`, and `BULK`. However, the S3 Batch Operations feature supports only the `STANDARD` retrieval tiers. For more information about the differences between the retrieval tiers, see Archive retrieval options.

For more information about the pricing for each tier, see the **Requests & data retrievals** section on the Amazon S3 pricing page.

**Differences when restoring from S3 Glacier and S3 Intelligent-Tiering**

Restoring archived files from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes differs from restoring files from the S3 Intelligent-Tiering storage class in the Archive Access or Deep Archive Access tiers.

- When you restore from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, a temporary *copy* of the object is created. Amazon S3 deletes this copy after the value that you specified in the `ExpirationInDays` argument has elapsed. After this temporary copy is deleted, you must submit an additional restore request to access the object.
- When restoring archived S3 Intelligent-Tiering objects, do *not* specify the `ExpirationInDays` argument. When you restore an object from the S3 Intelligent-Tiering Archive Access or Deep Archive Access tiers, the object transitions back into the S3 Intelligent-Tiering Frequent Access tier. After a minimum of 90 consecutive days of no access, the object automatically transitions into the Archive Access tier. After a minimum of 180 consecutive days of no access, the object automatically moves into the Deep Archive Access tier.
- Batch Operations jobs can operate either on S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage class objects *or* on S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tier objects. Batch Operations can't operate on both types of archived objects in the same job. To restore objects of both types, you *must* create separate Batch Operations jobs.

### Overlapping restores

If your S3 Initiate Restore Object job tries to restore an object that is already in the process of being restored, S3 Batch Operations proceeds as follows.

The restore operation succeeds for the object if either of the following conditions is true:

- Compared to the restoration request already in progress, this job's `ExpirationInDays` value is the same and its `GlacierJobTier` value is faster.
- The previous restoration request has already been completed, and the object is currently available. In this case, Batch Operations updates the expiration date of the restored object to match the `ExpirationInDays` value that's specified in the in-progress restoration request.

The restore operation fails for the object if any of the following conditions are true:

- The restoration request already in progress has not yet been completed, and the restoration duration for this job (specified by the `ExpirationInDays` value) is different from the restoration duration that is specified in the in-progress restoration request.
- The restoration tier for this job (specified by the `GlacierJobTier` value) is the same or slower than the restoration tier that is specified in the in-progress restoration request.

### Limitations

S3 Initiate Restore Object jobs have the following limitations:

- You must create the job in the same Region as the archived objects.
- S3 Batch Operations does not support the `EXPEDITED` retrieval tier.

For more information about restoring objects, see Restoring an archived object.

## S3 Object Lock retention

The **Object Lock retention** operation allows you to apply retention dates for your objects using either *governance* mode or *compliance* mode. These retention modes apply different levels of protection. You can apply either retention mode to any object version. Retention dates, like legal holds, prevent an object from being overwritten or deleted. Amazon S3 stores the *retain until date* specified in the object's metadata and protects the specified version of the object version until the retention period expires.

You can use S3 Batch Operations with Object Lock to manage retention dates of many Amazon S3 objects at once. You specify the list of target objects in your manifest and submit it to Batch Operations for completion. For more information, see S3 Object Lock the section called "Retention periods".

Your S3 Batch Operations job with retention dates runs *until completion, until cancellation, or until a failure state* is reached. You should use S3 Batch Operations and S3 Object Lock retention when you want to add, change, or remove the retention date for many objects with a single request.

Batch Operations verifies that Object Lock is enabled on your bucket before processing any keys in the manifest. To perform the operations and validation, Batch Operations needs `s3:GetBucketObjectLockConfiguration` and `s3:PutObjectRetention` permissions in an IAM role to allow Batch Operations to call Object Lock on your behalf. For more information, see the section called "Object Lock considerations".

For information about using this operation with the REST API, see `S3PutObjectRetention` in the CreateJob operation in the *Amazon Simple Storage Service API Reference*.

For an AWS Command Line Interface example of using this operation, see the section called "Use Batch Operations with Object Lock retention". For an AWS SDK for Java example, see the section called "Use Batch Operations with Object Lock retention".

**Restrictions and limitations**

- S3 Batch Operations does not make any bucket level changes.
- Versioning and S3 Object Lock must be configured on the bucket where the job is performed.
- All objects listed in the manifest must be in the same bucket.
- The operation works on the latest version of the object unless a version is explicitly specified in the manifest.
- You need `s3:PutObjectRetention` permission in your IAM role to use this.
- `s3:GetBucketObjectLockConfiguration` IAM permission is required to confirm that Object Lock is enabled for the S3 bucket.
- You can only extend the retention period of objects with `COMPLIANCE` mode retention dates applied, and it cannot be shortened.

## S3 Object Lock legal hold

The **Object Lock legal hold** operation enables you to place a legal hold on an object version. Like setting a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated retention period and remains in effect until removed.

You can use S3 Batch Operations with Object Lock to add legal holds to *many* Amazon S3 objects at once. You can do this by listing the target objects in your manifest and submitting that list to Batch Operations. Your S3 Batch Operations job with Object Lock legal hold runs until completion, until cancellation, or until a failure state is reached.

S3 Batch Operations verifies that Object Lock is enabled on your S3 bucket before processing any keys in the manifest. To perform the object operations and bucket level validation, S3 Batch Operations needs `s3:PutObjectLegalHold` and `s3:GetBucketObjectLockConfiguration` in an IAM role allowing S3 Batch Operations to call S3 Object Lock on your behalf.

When you create the S3 Batch Operations job to remove the legal hold, you just need to specify *Off* as the legal hold status. For more information, see the section called “Object Lock considerations”.

For information about how to use this operation with the REST API, see `S3PutObjectLegalHold` in the CreateJob operation in the *Amazon Simple Storage Service API Reference*.

For an example use of this operation, see Using the AWS SDK for Java.

**Restrictions and limitations**

- S3 Batch Operations does not make any bucket level changes.
- All objects listed in the manifest must be in the same bucket.
- Versioning and S3 Object Lock must be configured on the bucket where the job is performed.
- The operation works on the latest version of the object unless a version is explicitly specified in the manifest.
- `s3:PutObjectLegalHold` permission is required in your IAM role to add or remove legal hold from objects.
- `s3:GetBucketObjectLockConfiguration` IAM permission is required to confirm that S3 Object Lock is enabled for the S3 bucket.

- Copy objects

- Invoke AWS Lambda function
- Replace all object tags
- Delete all object tags
- Replace access control list
- Restore objects with Batch Operations
- S3 Object Lock retention
- S3 Object Lock legal hold
- Replicating existing objects with S3 Batch Replication

## Managing S3 Batch Operations jobs

Amazon S3 provides a robust set of tools to help you manage your S3 Batch Operations jobs after you create them. This section describes the operations that you can use to manage and track your jobs using the AWS Management Console, AWS CLI, AWS SDKs, or REST API.

**Topics**

- Using the Amazon S3 console to manage your S3 Batch Operations jobs
- Listing jobs
- Viewing job details
- Assigning job priority

### Using the Amazon S3 console to manage your S3 Batch Operations jobs

Using the console, you can manage your S3 Batch Operations jobs. For example, you can:

- View active and queued jobs
- Change a job's priority
- Confirm and run a job
- Clone a job
- Cancel a job

**To manage Batch Operations using the console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose the specific job that you would like to manage.

## Listing jobs

You can retrieve a list of your S3 Batch Operations jobs. The list includes jobs that haven't yet finished and jobs that finished within the last 90 days. The job list includes information for each job, such as its ID, description, priority, current status, and the number of tasks that have succeeded and failed. You can filter your job list by status. When you retrieve a job list through the console, you can also search your jobs by description or ID and filter them by AWS Region.

**Get a list of Active and Complete jobs**

The following AWS CLI example gets a list of `Active` and `Complete` jobs.

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id acct-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

For more information and examples, see list-jobs in the *AWS CLI Command Reference*.

## Viewing job details

If you want more information about a job than you can retrieve by listing jobs, you can view all of the details for a single job. You can view details for jobs that haven't yet finished or jobs that finished within the last 90 days. In addition to the information returned in a job list, a single job's details include other items, such as:

- The operation parameters
- Details about the manifest
- Information about the completion report (if you configured one when you created the job)
- The Amazon Resource Name (ARN) of the user role that you assigned to run the job

By viewing an individual job's details, you can access a job's entire configuration. To view a job's details, you can use the Amazon S3 console or the AWS Command Line Interface (AWS CLI).

**Get an S3 Batch Operations job description in the Amazon S3 console**

**To view a Batch Operations job description by using the console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose the job ID of the specific job to view its details.

**Get an S3 Batch Operations job description in the AWS CLI**

The following example gets the description of an S3 Batch Operations job by using the AWS CLI. To use the following example command, replace the *`user input placeholders`* with your own information.

```
aws s3control describe-job \
--region us-west-2 \
--account-id acct-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

For more information and examples, see describe-job in the *AWS CLI Command Reference*.

## Assigning job priority

You can assign each job a numeric priority, which can be any positive integer. S3 Batch Operations prioritize jobs according to the assigned priority. Jobs with a higher priority (or a higher numeric value for the priority parameter) are evaluated first. Priority is determined in descending order. For example, a job queue with a priority value of 10 is given scheduling preference over a job queue with a priority value of 1.

You can change a job's priority while it is running. If you submit a new job with a higher priority while a job is running, the lower-priority job can pause to allow the higher-priority job to run.

Changing job priority does not affect job processing speed.

> **Note**
>
> S3 Batch Operations honor job priorities on a best-effort basis. Although jobs with higher priorities generally take precedence over jobs with lower priorities, Amazon S3 does not guarantee strict ordering of jobs.

### Using the S3 console

#### How to update job priority in the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Select the specific job that you would like to manage.
4. Choose **Action**. In the dropdown list, choose **Update priority**.

### Using the AWS CLI

The following example updates the job priority using the AWS CLI. A higher number indicates a higher execution priority.

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id acct-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

### Using the AWS SDK for Java

The following example updates the priority of an S3 Batch Operations job using the AWS SDK for Java.

For more information about job priority, see Assigning job priority.

#### Example

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobPriorityRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobPriority {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobPriority(new UpdateJobPriorityRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withPriority(98));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Tracking job status and completion reports

With S3 Batch Operations, you can view and update job status, add notifications and logging, track job failures, and generate completion reports.

**Topics**



## Job statuses

After you create and run a job, it progresses through a series of statuses. The following table describes the statuses and the possible transitions between them.

| Status | Description | Transitions |
| --- | --- | --- |
| `New` | A job begins in the `New` state when you create it. | A job automatically moves to the `Preparing` state when Amazon S3 begins processing the manifest object. |
| `Preparing` | Amazon S3 is processing the manifest object and other job parameters to set up and run the job. | A job automatically moves to the `Ready` state after Amazon S3 finishes processing the manifest and other parameters. It is then ready to begin running the specified operation on the objects listed in the manifest.<br><br>If the job requires confirmation before running, such as when you create a job using the Amazon S3 console, then the job transitions from `Preparing` to |

| Status | Description | Transitions |
| --- | --- | --- |
| | | Suspended . It remains in the Suspended state until you confirm that you want to run it. |
| Suspended | The job requires confirmation, but you have not yet confirmed that you want to run it. Only jobs that you create using the Amazon S3 console require confirmation. A job that is created using the console enters the Suspended state immediately after Preparing . After you confirm that you want to run the job and the job becomes Ready, it never returns to the Suspended state. | After you confirm that you want to run the job, its status changes to Ready. |
| Ready | Amazon S3 is ready to begin running the requested object operations. | A job automatically moves to Active when Amazon S3 begins to run it. The amount of time that a job remains in the Ready state depends on whether you have higher-priority jobs running already and how long those jobs take to complete. |

| Status | Description | Transitions |
| --- | --- | --- |
| `Active` | Amazon S3 is performing the requested operation on the objects listed in the manifest. While a job is `Active`, you can monitor its progress using the Amazon S3 console or the `DescribeJob` operation through the REST API, AWS CLI, or AWS SDKs. | A job moves out of the `Active` state when it is no longer running operations on objects. This can happen automatically, such as when a job completes successfully or fails. Or it can occur as a result of user actions, such as canceling a job. The state that the job moves to depends on the reason for the transition. |
| `Pausing` | The job is transitioning to `Paused` from another state. | A job automatically moves to `Paused` when the `Pausing` stage is finished. |
| `Paused` | A job can become `Paused` if you submit another job with a higher priority while the current job is running. | A `Paused` job automatic ally returns to `Active` after any higher-priority jobs that are blocking the job's' execution complete, fail, or are suspended. |
| `Complete` | The job has finished performing the requested operation on all objects in the manifest. The operation might have succeeded or failed for every object. If you configured the job to generate a completion report, the report is available as soon as the job is `Complete`. | `Complete` is a terminal state. Once a job reaches `Complete`, it does not transition to any other state. |

| Status | Description | Transitions |
| --- | --- | --- |
| `Cancelling` | The job is transitioning to the `Cancelled` state. | A job automatically moves to `Cancelled` when the `Cancelling` stage is finished. |
| `Cancelled` | You requested that the job be canceled, and S3 Batch Operations has successfully cancelled the job. The job will not submit any new requests to Amazon S3. | `Cancelled` is a terminal state. After a job reaches `Cancelled`, it will not transition to any other state. |
| `Failing` | The job is transitioning to the `Failed` state. | A job automatically moves to `Failed` once the `Failing` stage is finished. |
| `Failed` | The job has failed and is no longer running. For more information about job failures, see Tracking job failure. | `Failed` is a terminal state. After a job reaches `Failed`, it will not transition to any other state. |

## Updating job status

The following AWS CLI and SDK for Java examples update the status of a Batch Operations job. For more information about using the S3 console to manage Batch Operations jobs, see Using the Amazon S3 console to manage your S3 Batch Operations jobs.

### Using the AWS CLI

- If you didn't specify the `--no-confirmation-required` parameter in the previous `create-job` example, the job remains in a suspended state until you confirm the job by setting its status to `Ready`. Amazon S3 then makes the job eligible for execution.

```
aws s3control update-job-status \
    --region us-west-2 \
```

```
    --account-id 181572960644 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- Cancel the job by setting the job status to `Cancelled`.

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 181572960644 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

### Using the AWS SDK for Java

The following example updates the status of an S3 Batch Operations job using the AWS SDK for Java.

For more information about job status, see Tracking job status and completion reports.

### Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobStatus {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";
```

```
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withRequestedJobStatus("Ready"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Notifications and logging

In addition to requesting completion reports, you can also capture, review, and audit Batch Operations activity using AWS CloudTrail. Because Batch Operations use existing Amazon S3 APIs to perform tasks, those tasks also emit the same events that they would if you called them directly. Thus, you can track and record the progress of your job and all of its tasks using the same notification, logging, and auditing tools and processes that you already use with Amazon S3. For more information, see the examples in the following sections.

> **Note**
>
> Amazon S3 Batch Operations generates both management and data events in CloudTrail during job execution. The volume of these events scale with the number of keys in each job's manifest. Refer to the CloudTrail pricing page for details, which includes examples of how pricing changes depending on the number of trails you have configured in your account. To learn how to configure and log events to fit your needs, see Create your first trail in the *AWS CloudTrail User Guide*.

For more information about Amazon S3 events, see Amazon S3 Event Notifications.

## Tracking job failure

If an S3 Batch Operations job encounters a problem that prevents it from running successfully, such as not being able to read the specified manifest, the job fails. When a job fails, it generates one or more failure codes or failure reasons. S3 Batch Operations store the failure codes and reasons with the job so that you can view them by requesting the job's details. If you requested a completion report for the job, the failure codes and reasons also appear there.

To prevent jobs from running a large number of unsuccessful operations, Amazon S3 imposes a task-failure threshold on every Batch Operations job. When a job has run at least 1,000 tasks, Amazon S3 monitors the task failure rate. At any point, if the failure rate (the number of tasks that have failed as a proportion of the total number of tasks that have run) exceeds 50 percent, the job fails. If your job fails because it exceeded the task-failure threshold, you can identify the cause of the failures. For example, you might have accidentally included some objects in the manifest that don't exist in the specified bucket. After fixing the errors, you can resubmit the job.

> **Note**
>
> S3 Batch Operations operate asynchronously and the tasks don't necessarily run in the order that the objects are listed in the manifest. Therefore, you can't use the manifest ordering to determine which objects' tasks succeeded and which ones failed. Instead, you can examine the job's completion report (if you requested one) or view your AWS CloudTrail event logs to help determine the source of the failures.

## Completion reports

When you create a job, you can request a completion report. As long as S3 Batch Operations successfully invoke at least one task, Amazon S3 generates a completion report after it finishes running tasks, fails, or is canceled. You can configure the completion report to include all tasks or only failed tasks.

The completion report includes the job configuration and status and information for each task, including the object key and version, status, error codes, and descriptions of any errors. Completion reports provide an easy way to view the results of your tasks in a consolidated format with no additional setup required. Completion reports are encrypted with Amazon S3 managed keys (SSE-S3). For an example of a completion report, see Examples: S3 Batch Operations completion reports.

If you don't configure a completion report, you can still monitor and audit your job and its tasks using CloudTrail and Amazon CloudWatch. For more information, see the following section.

**Topics**



## Examples: Tracking an S3 Batch Operations job in Amazon EventBridge through AWS CloudTrail

Amazon S3 Batch Operations job activity is recorded as events in AWS CloudTrail. You can create a custom rule in Amazon EventBridge and send these events to the target notification resource of your choice, such as Amazon Simple Notification Service (Amazon SNS).

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. Amazon CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes that you make in either CloudWatch or EventBridge appear in each console. For more information, see the *Amazon EventBridge User Guide*.

**Tracking Examples**



### S3 Batch Operations events recorded in CloudTrail

When a Batch Operations job is created, it is recorded as a `JobCreated` event in CloudTrail. As the job runs, it changes state during processing, and other `JobStatusChanged` events are recorded in CloudTrail. You can view these events on the CloudTrail console. For more information about CloudTrail, see the *AWS CloudTrail User Guide*.

> **Note**
>
> Only S3 Batch Operations job `status-change` events are recorded in CloudTrail.

### Example S3 Batch Operations job completion event recorded by CloudTrail

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

### EventBridge rule for tracking S3 Batch Operations job events

The following example shows how to create a rule in Amazon EventBridge to capture S3 Batch Operations events recorded by AWS CloudTrail to a target of your choice.

To do this, you create a rule by following all the steps in Creating EventBridge rules that react to events. You paste the following S3 Batch Operations custom event pattern policy where applicable, and choose the target service of your choice.

### S3 Batch Operations custom event pattern policy

```
{
```

```
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

The following examples are two Batch Operations events that were sent to Amazon Simple Queue Service (Amazon SQS) from an EventBridge event rule. A Batch Operations job goes through many different states while processing (`New`, `Preparing`, `Active`, etc.), so you can expect to receive several messages for each job.

**Example JobCreated sample event**

```
{
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
```

```
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
        "serviceEventDetails": {
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

### Example JobStatusChanged job completion event

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
```

```
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```

## Examples: S3 Batch Operations completion reports

When you create an S3 Batch Operations job, you can request a completion report for all tasks or just for failed tasks. As long as at least one task has been invoked successfully, S3 Batch Operations generates a report for jobs that have completed, failed, or been canceled.

The completion report contains additional information for each task, including the object key name and version, status, error codes, and descriptions of any errors. The description of errors for each failed task can be used to diagnose issues that occur during job creation, such as permissions.

> **Note**
>
> Completion reports are always encrypted with Amazon S3 managed keys (SSE-S3).

### Example top-level manifest result file

The top-level `manifest.json` file contains the locations of each succeeded report and (if the job had any failures) the location of failed reports, as shown in the following example.

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
```

```
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

### Example failed tasks reports

Failed tasks reports contain the following information for all *failed* tasks:

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

The following example report shows a case in which the AWS Lambda function timed out, causing failures to exceed the failure threshold. It was then marked as a `PermanentFailure`.

```
awsexamplebucket1,image_14975,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-
abcf-379dc749c452 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15897,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-
b511-29232fde5fe4 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_14819,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-
c7f18827f551 Task timed out after 3.00 seconds""}"
```

```
awsexamplebucket1,image_15930,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-
cbf027b7957e Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17644,,failed,200,PermanentFailure,"Lambda
 returned function error: {""errorMessage"":""2019-04-05T17:35:46.025Z
 10a764e4-2b26-4d8c-9056-1e1072b4723f Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17398,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-
aee8-4d02f8c0235c Task timed out after 3.00 seconds""}"
```

**Example succeeded tasks report**

Succeeded tasks reports contain the following for the *successful* tasks:

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

In the following example, the Lambda function successfully copied the Amazon S3 object to another bucket. The returned Amazon S3 response is passed back to S3 Batch Operations and is then written into the final completion report.

```
awsexamplebucket1,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type':
 'application/xml'}}}"
awsexamplebucket1,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
```

```
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2':
 'GiCZNYr8LHd/Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-
amz-copy-source-version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3',
 'x-amz-request-id': '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT',
 'content-type': 'application/xml'}}}"
awsexamplebucket1,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type':
 'application/xml'}}}"
```

## Controlling access and labeling jobs using tags

You can label and control access to your S3 Batch Operations jobs by adding *tags*. Tags can be used to identify who is responsible for a Batch Operations job. The presence of job tags can grant or limit a user's ability to cancel a job, activate a job in the confirmation state, or change a job's priority level. You can create jobs with tags attached to them, and you can add tags to jobs after they are created. Each tag is a key-value pair that can be included when you create the job or updated later.

> **Warning**
>
> Job tags should not contain any confidential information or personal data.

Consider the following tagging example: Suppose that you want your Finance department to create a Batch Operations job. You could write an AWS Identity and Access Management (IAM) policy that allows a user to invoke `CreateJob`, provided that the job is created with the `Department` tag assigned the value `Finance`. Furthermore, you could attach that policy to all users who are members of the Finance department.

Continuing with this example, you could write a policy that allows a user to update the priority of any job that has the desired tags, or cancel any job that has those tags. For more information, see the section called "Controlling permissions".

You can add tags to new S3 Batch Operations jobs when you create them, or you can add them to existing jobs.

Note the following tag restrictions:

- You can associate up to 50 tags with a job as long as they have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length.
- The key and values are case sensitive.

For more information about tag restrictions, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

## API operations related to S3 Batch Operations job tagging

Amazon S3 supports the following API operations that are specific to S3 Batch Operations job tagging:

- GetJobTagging — Returns the tag set associated with a Batch Operations job.
- PutJobTagging — Replaces the tag set associated with a job. There are two distinct scenarios for S3 Batch Operations job tag management using this API action:
  - Job has no tags — You can add a set of tags to a job (the job has no prior tags).
  - Job has a set of existing tags — To modify the existing tag set, you can either replace the existing tag set entirely, or make changes within the existing tag set by retrieving the existing tag set using GetJobTagging, modify that tag set, and use this API action to replace the tag set with the one you have modified.

> **Note**
>
> If you send this request with an empty tag set, S3 Batch Operations deletes the existing tag set on the object. If you use this method, you are charged for a Tier 1 Request (PUT). For more information, see Amazon S3 pricing.

> To delete existing tags for your Batch Operations job, the `DeleteJobTagging` action is preferred because it achieves the same result without incurring charges.

- DeleteJobTagging — Deletes the tag set associated with a Batch Operations job.

## Creating a Batch Operations job with job tags used for labeling

You can label and control access to your S3 Batch Operations jobs by adding *tags*. Tags can be used to identify who is responsible for a Batch Operations job. You can create jobs with tags attached to them, and you can add tags to jobs after they are created. For more information, see the section called “Using tags”.

### Using the AWS CLI

The following AWS CLI example creates an S3 Batch Operations `S3PutObjectCopy` job using job tags as labels for the job.

1. Select the action or `OPERATION` that you want the Batch Operations job to perform, and choose your `TargetResource`.

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::destination-bucket"
  }
}
EOF
```

2. Identify the job `TAGS` that you want for the job. In this case, you apply two tags, `department` and `FiscalYear`, with the values `Marketing` and `2020` respectively.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
```

```
]
EOF
```

3. Specify the `MANIFEST` for the Batch Operations job.

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::example-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. Configure the `REPORT` for the Batch Operations job.

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::example-report-bucket",
  "Format": "Example_Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. Run the`create-job` action to create your Batch Operations job with inputs set in the preceding steps.

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
```

```
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

### Using the AWS SDK for Java

### Example

The following example creates an S3 Batch Operations job with tags using the AWS SDK for Java.

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::example-manifest-bucket/
manifests/10_manifest.csv";
    final String manifestObjectVersionId = "example-5dc7a8bfb90808fc5d546218";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new
 JobManifestSpec().withFormat(JobManifestFormat.S3InventoryReport_CSV_20161130);

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::example-report-bucket";
    final String jobReportPrefix = "example-job-reports";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final String lambdaFunctionArn = "arn:aws:lambda:us-
west-2:123456789012:function:example-function";

    final JobOperation jobOperation = new JobOperation()
```

```
            .withLambdaInvoke(new
 LambdaInvokeOperation().withFunctionArn(lambdaFunctionArn));

    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final String roleArn = "arn:aws:iam::123456789012:role/example-batch-operations-
role";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Test lambda job")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withTags(departmentTag, fiscalYearTag)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

## Deleting the tags from an S3 Batch Operations job

You can use these examples to delete the tags from a Batch Operations job.

### Using the AWS CLI

The following example deletes the tags from a Batch Operations job using the AWS CLI.

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

### Delete the job tags of a Batch Operations job

#### Example

The following example deletes the tags of an S3 Batch Operations job using the AWS SDK for Java.

```
public void deleteJobTagging(final AWSS3ControlClient awss3ControlClient,
                             final String jobId) {
    final DeleteJobTaggingRequest deleteJobTaggingRequest = new
 DeleteJobTaggingRequest()
            .withJobId(jobId);

    final DeleteJobTaggingResult deleteJobTaggingResult =
                awss3ControlClient.deleteJobTagging(deleteJobTaggingRequest);
}
```

## Putting job tags for an existing S3 Batch Operations job

You can use PutJobTagging to add job tags to your existing S3 Batch Operations jobs. For more information, see the following examples.

### Using the AWS CLI

The following is an example of using `s3control put-job-tagging` to add job tags to your S3 Batch Operations job using the AWS CLI.

> **Note**
>
> If you send this request with an empty tag set, S3 Batch Operations deletes the existing tag set on the object. Also, if you use this method, you are charged for a Tier 1 Request (PUT). For more information, see Amazon S3 pricing.
> To delete existing tags for your Batch Operations job, the `DeleteJobTagging` action is preferred because it achieves the same result without incurring charges.

1. Identify the job TAGS that you want for the job. In this case, you apply two tags, `department` and `FiscalYear`, with the values `Marketing` and `2020` respectively.

   ```
   read -d '' TAGS <<EOF
   [
   ```

```
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

2. Run the `put-job-tagging` action with the required parameters.

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

### Using the AWS SDK for Java

### Example

The following example puts the tags of an S3 Batch Operations job using the AWS SDK for Java.

```
public void putJobTagging(final AWSS3ControlClient awss3ControlClient,
                          final String jobId) {
    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final PutJobTaggingRequest putJobTaggingRequest = new PutJobTaggingRequest()
            .withJobId(jobId)
            .withTags(departmentTag, fiscalYearTag);

    final PutJobTaggingResult putJobTaggingResult =
 awss3ControlClient.putJobTagging(putJobTaggingRequest);
}
```

## Getting the tags of a S3 Batch Operations job

You can use `GetJobTagging` to return the tags of an S3 Batch Operations job. For more information, see the following examples.

### Using the AWS CLI

The following example gets the tags of a Batch Operations job using the AWS CLI.

```
aws \
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

### Using the AWS SDK for Java

### Example

The following example gets the tags of an S3 Batch Operations job using the AWS SDK for Java.

```
public List<S3Tag> getJobTagging(final AWSS3ControlClient awss3ControlClient,
                                 final String jobId) {
    final GetJobTaggingRequest getJobTaggingRequest = new GetJobTaggingRequest()
            .withJobId(jobId);

    final GetJobTaggingResult getJobTaggingResult =
            awss3ControlClient.getJobTagging(getJobTaggingRequest);

    final List<S3Tag> tags = getJobTaggingResult.getTags();

    return tags;
}
```

## Controlling permissions for S3 Batch Operations using job tags

To help you manage your S3 Batch Operations jobs, you can add *job tags*. With job tags, you can control access to your Batch Operations jobs and enforce that tags be applied when any job is created.

You can apply up to 50 job tags to each Batch Operations job. This allows you to set very granular policies restricting the set of users that can edit the job. Job tags can grant or limit a user's ability

to cancel a job, activate a job in the confirmation state, or change a job's priority level. In addition, you can enforce that tags be applied to all new jobs, and specify the allowed key-value pairs for the tags. You can express all of these conditions using the same IAM policy language. For more information, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

The following example shows how you can use S3 Batch Operations job tags to grant users permission to create and edit only the jobs that are run within a specific *department* (for example, the *Finance* or *Compliance* department). You can also assign jobs based on the stage of *development* that they are related to, such as *QA* or *Production*.

In this example, you use S3 Batch Operations job tags in AWS Identity and Access Management (IAM) policies to grant users permission to create and edit only the jobs being run within their department. You assign jobs based on the stage of development that they are related to, such as *QA* or *Production*.

This example uses the following departments, with each using Batch Operations in different ways:

- Finance
- Compliance
- Business Intelligence
- Engineering

**Topics**

- Controlling access by assigning tags to users and resources
- Tagging Batch Operations jobs by stage and enforcing limits on job priority

### Controlling access by assigning tags to users and resources

In this scenario, the administrators are using attribute-based access control (ABAC). ABAC is an IAM authorization strategy that defines permissions by attaching tags to both users and AWS resources.

Users and jobs are assigned one of the following department tags:

**Key : Value**

- `department : Finance`

- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> Job tag keys and values are case sensitive.

Using the ABAC access control strategy, you grant a user in the Finance department permission to create and manage S3 Batch Operations jobs within their department by associating the tag `department=Finance` with their user.

Furthermore, you can attach a managed policy to the IAM user that allows any user in their company to create or modify S3 Batch Operations jobs within their respective departments.

The policy in this example includes three policy statements:

- The first statement in the policy allows the user to create a Batch Operations job provided that the job creation request includes a job tag that matches their respective department. This is expressed using the `"${aws:PrincipalTag/department}"` syntax, which is replaced by the user's department tag at policy evaluation time. The condition is satisfied when the value provided for the department tag in the request (`"aws:RequestTag/department"`) matches the user's department.
- The second statement in the policy allows users to change the priority of jobs or update a job's status provided that the job the user is updating matches the user's department.
- The third statement allows a user to update a Batch Operations job's tags at any time via a `PutJobTagging` request as long as (1) their department tag is preserved and (2) the job they're updating is within their department.

```
{
      "Version": "2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
```

```
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        }
    ]
}
```

**Tagging Batch Operations jobs by stage and enforcing limits on job priority**

All S3 Batch Operations jobs have a numeric priority, which Amazon S3 uses to decide in what order to run the jobs. For this example, you restrict the maximum priority that most users can assign to jobs, with higher priority ranges reserved for a limited set of privileged users, as follows:

- QA stage priority range (low): 1-100

- Production stage priority range (high): 1-300

To do this, introduce a new tag set representing the *stage* of the job:

**Key : Value**

- `stage : QA`
- `stage : Production`

**Creating and updating low-priority jobs within a department**

This policy introduces two new restrictions on S3 Batch Operations job creation and update, in addition to the department-based restriction:

- It allows users to create or update jobs in their department with a new condition that requires the job to include the tag `stage=QA`.
- It allows users to create or update a job's priority up to a new maximum priority of 100.

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
```

```
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

### Creating and updating high-priority jobs within a department

A small number of users might require the ability to create high priority jobs in either *QA* or *Production*. To support this need, you create a managed policy that's adapted from the low-priority policy in the previous section.

This policy does the following:

- Allows users to create or update jobs in their department with either the tag `stage=QA` or `stage=Production`.
- Allows users to create or update a job's priority up to a maximum of 300.

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                      "s3:UpdateJobStatus"
            ],
                "Resource": "*",
```

```
                "Condition": {
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
             }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
             }
        },
          {
                "Effect": "Allow",
                "Action": "s3:PutJobTagging",
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ],
                            "aws:ResourceTag/stage": [
```

```
                                "QA",
                                "Production"
                        ]
                    }
                }
            }
        ]
}
```

## Managing S3 Object Lock using S3 Batch Operations

With S3 Object Lock, you can place a legal hold on an object version. Like setting a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated retention period and remains in effect until removed. For more information, see S3 Object Lock legal hold.

For information about using S3 Batch Operations with Object Lock to add legal holds to *many* Amazon S3 objects at once, see the following sections.

**Topics**

- Enabling S3 Object Lock using S3 Batch Operations
- Setting Object Lock retention using Batch Operations
- Using S3 Batch Operations with S3 Object Lock retention compliance mode
- Use S3 Batch Operations with S3 Object Lock retention governance mode
- Using S3 Batch Operations to turn off S3 Object Lock legal hold

### Enabling S3 Object Lock using S3 Batch Operations

You can use S3 Batch Operations with S3 Object Lock to manage retention or enable a legal hold for many Amazon S3 objects at once. You specify the list of target objects in your manifest and submit it to Batch Operations for completion. For more information, see the section called “Object Lock retention” and the section called “Object Lock legal hold”.

The following examples show how to create an IAM role with S3 Batch Operations permissions and update the role permissions to create jobs that enable Object Lock. In the examples, replace any variable values with those that suit your needs. You must also have a CSV manifest identifying the objects for your S3 Batch Operations job. For more information, see the section called “Specifying a manifest”.

### Using the AWS CLI

1. Create an IAM role and assign S3 Batch Operations permissions to run.

   This step is required for all S3 Batch Operations jobs.

```
export AWS_PROFILE='aws-user'

read -d '' bops_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name bops-objectlock --assume-role-policy-document
 "${bops_trust_policy}"
```

2. Set up S3 Batch Operations with S3 Object Lock to run.

   In this step, you allow the role to do the following:

   a. Run Object Lock on the S3 bucket that contains the target objects that you want Batch Operations to run on.

   b. Read the S3 bucket where the manifest CSV file and the objects are located.

   c. Write the results of the S3 Batch Operations job to the reporting bucket.

```
read -d '' bops_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ReportBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name object-lock-
permissions --policy-document "${bops_permissions}"
```

### Using the AWS SDK for Java

The following examples show how to create an IAM role with S3 Batch Operations permissions, and update the role permissions to create jobs that enable object lock using the AWS SDK for Java. In the code, replace any variable values with those that suit your needs. You must also have a CSV manifest identifying the objects for your S3 Batch Operations job. For more information, see the section called “Specifying a manifest”.

You perform the following steps:

1. Create an IAM role and assign S3 Batch Operations permissions to run. This step is required for all S3 Batch Operations jobs.
2. Set up S3 Batch Operations with S3 Object Lock to run.

   You allow the role to do the following:

   1. Run Object Lock on the S3 bucket that contains the target objects that you want Batch Operations to run on.
   2. Read the S3 bucket where the manifest CSV file and the objects are located.
   3. Write the results of the S3 Batch Operations job to the reporting bucket.

```
public void createObjectLockRole() {
    final String roleName = "bops-object-lock";

    final String trustPolicy = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Principal\": { " +
            "        \"Service\": [" +
            "          \"batchoperations.s3.amazonaws.com\"" +
            "        ]" +
            "      }, " +
            "      \"Action\": \"sts:AssumeRole\" " +
            "    } " +
            "  ]" +
            "}";

    final String bopsPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": \"s3:GetBucketObjectLockConfiguration\"," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket\"" +
            "            ]" +
            "        }," +
```

```
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:GetObject\"," +
            "                \"s3:GetObjectVersion\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObject\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ReportBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withAssumeRolePolicyDocument(bopsPermissions)
            .withRoleName(roleName);

    final CreateRoleResult createRoleResult = iam.createRole(createRoleRequest);

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bopsPermissions)
            .withPolicyName("bops-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

## Setting Object Lock retention using Batch Operations

The following example allows the rule to set S3 Object Lock retention for your objects in the manifest bucket.

You update the role to include `s3:PutObjectRetention` permissions so that you can run Object Lock retention on the objects in your bucket.

### Using the AWS CLI

```
export AWS_PROFILE='aws-user'

read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name retention-permissions
 --policy-document "${retention_permissions}"
```

### Using the AWS SDK for Java

```
public void allowPutObjectRetention() {
    final String roleName = "bops-object-lock";

    final String retentionPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
```

```
            "                \"s3:PutObjectRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(retentionPermissions)
            .withPolicyName("retention-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

## Using S3 Batch Operations with S3 Object Lock retention compliance mode

The following example builds on the previous examples of creating a trust policy and setting S3 Batch Operations and S3 Object Lock configuration permissions on your objects. This example sets the retention mode to `COMPLIANCE` and the `retain until date` to January 1, 2025. It creates a job that targets objects in the manifest bucket and reports the results in the reports bucket that you identified.

### Using the AWS CLI

#### Example Set mention compliance across multiple objects

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-01T00:00:00",
```

```
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Set compliance retain-until to 1 Jul 2030";
```

### Example Extend the COMPLIANCE mode's `retain until date` to January 15, 2025

The following example extends the COMPLIANCE mode's `retain until date` to January 15, 2025.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
```

```
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

### Using the AWS SDK for Java

### Example Set the retention mode to COMPLIANCE and the retain until date to January 1, 2025.

```
public String createComplianceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
```

```
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date janFirst = format.parse("01/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(janFirst)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Set compliance retain-until to 1 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

### Example Extending the `COMPLIANCE` mode's `retain until date`

The following example extends the `COMPLIANCE` mode's `retain until date` to January 15, 2025.

```
public String createExtendComplianceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";
```

```
    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan15th = format.parse("15/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(jan15th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Extend compliance retention to 15 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
```

```
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

## Use S3 Batch Operations with S3 Object Lock retention governance mode

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to apply S3 Object Lock retention governance with the `retain until date` of January 30, 2025, across multiple objects. It creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

### Using the AWS CLI

#### Example Apply S3 Object Lock retention governance across multiple objects with the retain until date of January 30, 2025

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
```

```
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

### Example Bypass retention governance across multiple objects

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to bypass retention governance across multiple objects and creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name bypass-governance-
permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
    "Retention": {
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
```

```
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::REPORT_BUCKET",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/bops-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Remove governance retention";
```

### Using the AWS SDK for Java

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to apply S3 Object Lock retention governance with the `retain until date` set to January 30, 2025 across multiple objects. It creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

**Example Apply S3 Object Lock retention governance across multiple objects with the retain until date of January 30, 2025**

```
public String createGovernanceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";
```

```
    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/governance-objects";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.GOVERNANCE)
                            .withRetainUntilDate(jan30th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Put governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
```

```
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

### Example Bypass retention governance across multiple objects

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to bypass retention governance across multiple objects and creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

```
public void allowBypassGovernance() {
    final String roleName = "bops-object-lock";

    final String bypassGovernancePermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:BypassGovernanceRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bypassGovernancePermissions)
            .withPolicyName("bypass-governance-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

```
public String createRemoveGovernanceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/bops-governance";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Remove governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
```

```
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

## Using S3 Batch Operations to turn off S3 Object Lock legal hold

The following example builds on the previous examples of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to disable Object Lock legal hold on objects using Batch Operations.

The example first updates the role to grant `s3:PutObjectLegalHold` permissions, creates a Batch Operations job that turns off (removes) legal hold from the objects identified in the manifest, and then reports on it.

### Using the AWS CLI

#### Example Updates the role to grant `s3:PutObjectLegalHold` permissions

```
export AWS_PROFILE='aws-user'

read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]

EOF
```

```
aws iam put-role-policy --role-name bops-objectlock --policy-name legal-hold-
permissions --policy-document "${legal_hold_permissions}"
```

### Example Turn off legal hold

The following example turns off legal hold.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/legalhold-object-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-bops",
```

```
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

### Using the AWS SDK for Java

### Example Updates the role to grant `s3:PutObjectLegalHold` permissions

```
public void allowPutObjectLegalHold() {
    final String roleName = "bops-object-lock";

    final String legalHoldPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectLegalHold\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(legalHoldPermissions)
            .withPolicyName("legal-hold-permissions")
```

```
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

### Example Turn off legal hold

Use the example below if you want to turn off legal hold.

```
public String createLegalHoldOffJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/legalhold-object-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/legalhold-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectLegalHold(new S3SetObjectLegalHoldOperation()
                    .withLegalHold(new S3ObjectLockLegalHold()
                            .withStatus(S3ObjectLockLegalHoldStatus.OFF)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
```

```
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Turn off legal hold")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

## S3 Batch Operations tutorial

The following tutorial presents complete end-to-end procedures for some Batch Operations tasks.

- Tutorial: Batch-transcoding videos with S3 Batch Operations, AWS Lambda, and AWS Elemental MediaConvert

# Monitoring Amazon S3

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. We recommend collecting monitoring data from all of the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs. Before you start monitoring Amazon S3, create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

For more information about logging and monitoring in Amazon S3, see the following topics.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**

# Monitoring tools

AWS provides various tools that you can use to monitor Amazon S3. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

## Automated monitoring tools

You can use the following automated monitoring tools to watch Amazon S3 and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state. The state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.
- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

## Manual monitoring tools

Another important part of monitoring Amazon S3 involves manually monitoring those items that the CloudWatch alarms don't cover. The Amazon S3, CloudWatch, Trusted Advisor, and other AWS Management Console dashboards provide an at-a-glance view of the state of your AWS environment. You might want to enable *server access logging*, which tracks requests for access to your bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. For more information, see Logging requests with server access logging.

- The Amazon S3 dashboard shows the following:
  - Your buckets and the objects and properties they contain
- The CloudWatch home page shows the following:
  - Current alarms and status

- Graphs of alarms and resources
- Service health status

  In addition, you can use CloudWatch to do the following:

- Create customized dashboards to monitor the services you care about.
- Graph metric data to troubleshoot issues and discover trends.
- Search and browse all your AWS resource metrics.
- Create and edit alarms to be notified of problems.

- AWS Trusted Advisor can help you monitor your AWS resources to improve performance, reliability, security, and cost effectiveness. Four Trusted Advisor checks are available to all users; more than 50 checks are available to users with a Business or Enterprise support plan. For more information, see AWS Trusted Advisor.

  Trusted Advisor has these checks that relate to Amazon S3:

  - Checks of the logging configuration of Amazon S3 buckets.
  - Security checks for Amazon S3 buckets that have open access permissions.
  - Fault tolerance checks for Amazon S3 buckets that do not have versioning enabled, or have versioning suspended.

## Logging options for Amazon S3

You can record the actions that are taken by users, roles, or AWS services on Amazon S3 resources and maintain log records for auditing and compliance purposes. To do this, you can use server-access logging, AWS CloudTrail logging, or a combination of both. We recommend that you use CloudTrail for logging bucket-level and object-level actions for your Amazon S3 resources. For more information about each option, see the following sections:

- Logging requests with server access logging
- Logging Amazon S3 API calls using AWS CloudTrail

The following table lists the key properties of CloudTrail logs and Amazon S3 server-access logs. To make sure that CloudTrail meets your security requirements, review the table and notes.

| Log properties | AWS CloudTrail | Amazon S3 server logs |
| --- | --- | --- |
| Can be forwarded to other systems (Amazon CloudWatch Logs, Amazon CloudWatch Events) | Yes | No |
| Deliver logs to more than one destination (for example, send the same logs to two different buckets) | Yes | No |
| Turn on logs for a subset of objects (prefix) | Yes | No |
| Cross-account log delivery (target and source bucket owned by different accounts) | Yes | No |
| Integrity validation of log file by using digital signature or hashing | Yes | No |
| Default or choice of encryption for log files | Yes | No |
| Object operations (by using Amazon S3 APIs) | Yes | Yes |
| Bucket operations (by using Amazon S3 APIs) | Yes | Yes |
| Searchable UI for logs | Yes | No |
| Fields for Object Lock parameters, Amazon S3 Select properties for log records | Yes | No |

| Log properties | AWS CloudTrail | Amazon S3 server logs |
|---|---|---|
| Fields for `Object Size`, `Total Time`, `Turn-Around Time`, and `HTTP Referer` for log records | No | Yes |
| Lifecycle transitions, expirations, restores | No | Yes |
| Logging of keys in a batch delete operation | No | Yes |
| Authentication failures[1] | No | Yes |
| Accounts where logs get delivered | Bucket owner[2], and requester | Bucket owner only |
| **Performance and Cost** | **AWS CloudTrail** | **Amazon S3 Server Logs** |
| Price | Management events (first delivery) are free; data events incur a fee, in addition to storage of logs | No other cost in addition to storage of logs |
| Speed of log delivery | Data events every 5 minutes; management events every 15 minutes | Within a few hours |
| Log format | JSON | Log file with space-separated, newline-delimited records |

**Notes**

1. CloudTrail does not deliver logs for requests that fail authentication (in which the provided credentials are not valid). However, it does include logs for requests in which authorization fails (`AccessDenied`) and requests that are made by anonymous users.

2. The S3 bucket owner receives CloudTrail logs when the account does not have full access to the object in the request. For more information, see Amazon S3 object-level actions in cross-account scenarios.
3. S3 does not support delivery of CloudTrail logs or server access logs to the requester or the bucket owner for VPC endpoint requests when the VPC endpoint policy denies them or for requests that fail before the VPC policy is evaluated.

# Logging Amazon S3 API calls using AWS CloudTrail

Amazon S3 is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. The calls captured include calls from the Amazon S3 console and code calls to the Amazon S3 API operations. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

**CloudTrail trails**

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region

trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

**CloudTrail Lake event data stores**

*CloudTrail Lake* lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 Lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

## Using CloudTrail logs with Amazon S3 server access logs and CloudWatch Logs

AWS CloudTrail logs provide a record of actions taken by a user, role, or an AWS service in Amazon S3, while Amazon S3 server access logs provide detailed records for the requests that are made to an S3 bucket. For more information about how the different logs work, and their properties, performance, and costs, see the section called “Logging options”.

You can use AWS CloudTrail logs together with server access logs for Amazon S3. CloudTrail logs provide you with detailed API tracking for Amazon S3 bucket-level and object-level operations. Server access logs for Amazon S3 provide you with visibility into object-level operations on your data in Amazon S3. For more information about server access logs, see Logging requests with server access logging.

You can also use CloudTrail logs together with Amazon CloudWatch for Amazon S3. CloudTrail integration with CloudWatch Logs delivers S3 bucket-level API activity captured by CloudTrail to a CloudWatch log stream in the CloudWatch log group that you specify. You can create CloudWatch alarms for monitoring specific API activity and receive email notifications when the specific API activity occurs. For more information about CloudWatch alarms for monitoring specific API activity, see the AWS CloudTrail User Guide. For more information about using CloudWatch with Amazon S3, see Monitoring metrics with Amazon CloudWatch.

**Note**

S3 does not support delivery of CloudTrail logs to the requester or the bucket owner for VPC endpoint requests when the VPC endpoint policy denies them.

## CloudTrail tracking with Amazon S3 SOAP API calls

CloudTrail tracks Amazon S3 SOAP API calls. Amazon S3 SOAP support over HTTP is deprecated, but it is still available over HTTPS. For more information about Amazon S3 SOAP support, see Appendix a: Using the SOAP API.

**Important**

Newer Amazon S3 features are not supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

### Amazon S3 SOAP actions tracked by CloudTrail logging

| SOAP API name | API event name used in CloudTrail log |
| --- | --- |
| ListAllMyBuckets | `ListBuckets` |
| CreateBucket | `CreateBucket` |

| SOAP API name | API event name used in CloudTrail log |
| --- | --- |
| DeleteBucket | DeleteBucket |
| GetBucketAccessControlPolicy | GetBucketAcl |
| SetBucketAccessControlPolicy | PutBucketAcl |
| GetBucketLoggingStatus | GetBucketLogging |
| SetBucketLoggingStatus | PutBucketLogging |

For more information about CloudTrail and Amazon S3, see the following topics:

**Topics**

- Amazon S3 CloudTrail events
- CloudTrail log file entries for Amazon S3 and S3 on Outposts
- Enabling CloudTrail event logging for S3 buckets and objects
- Identifying Amazon S3 requests using CloudTrail

## Amazon S3 CloudTrail events

**Important**

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

This section provides information about the events that S3 logs to CloudTrail.

## Amazon S3 data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can log data events for the Amazon S3 resource types by using the CloudTrail console, AWS CLI, or CloudTrail API operations. For more information about how to log data events, see Logging data events with the AWS Management Console and Logging data events with the AWS Command Line Interface in the *AWS CloudTrail User Guide*.

The following table lists the Amazon S3 resource types for which you can log data events. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the AWS CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| **S3** | `AWS::S3::Object` | • AbortMultipartUpload<br>• CompleteMultipartUpload<br>• CopyObject<br>• CreateMultipartUpload<br>• DeleteObject<br>• DeleteObjectTagging<br>• DeleteObjects<br>• GetObject<br>• GetObjectAcl<br>• GetObjectAttributes<br>• GetObjectLegalHold |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| | | • GetObjectRetention<br>• GetObjectTagging<br>• GetObjectTorrent<br>• HeadObject<br>• ListMultipartUploads<br>• ListObjectVersions<br>• ListObjects<br>• ListParts<br>• PutObject<br>• PutObjectAcl<br>• PutObjectLegalHold<br>• PutObjectRetention<br>• PutObjectTagging<br>• RestoreObject<br>• SelectObjectContent<br>• UploadPart<br>• UploadPartCopy |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| **S3 Express One Zone** | `AWS::S3Express::Object` | • AbortMultipartUpload<br>• CompleteMultipartUpload<br>• CreateSession<br>• CopyObject<br>• CreateMultipartUpload<br>• DeleteObject<br>• DeleteObjects<br>• GetObject<br>• GetObjectAttributes<br>• HeadBucket<br>• HeadObject<br>• ListObjectsV2<br>• ListParts<br>• PutObject<br>• UploadPart<br>• UploadPartCopy |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| **S3 Access Point** | `AWS::S3::Access Point` | • AbortMultipartUpload<br>• CompleteMultipartUpload<br>• CopyObject (same-region copies only)<br>• CreateMultipartUpload<br>• DeleteObject<br>• DeleteObjectTagging<br>• GetBucketAcl<br>• GetBucketCors<br>• GetBucketLocation<br>• GetBucketNotificationConfiguration<br>• GetBucketPolicy<br>• GetObject<br>• GetObjectAcl<br>• GetObjectAttributes<br>• GetObjectLegalHold<br>• GetObjectRetention<br>• GetObjectTagging<br>• HeadBucket<br>• HeadObject<br>• ListMultipartUploads<br>• ListObjects<br>• ListObjectsV2<br>• ListObjectVersions<br>• ListParts<br>• Presign<br>• PutObject |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| | | • PutObjectLegalHold<br>• PutObjectRetention<br>• PutObjectAcl<br>• PutObjectTagging<br>• RestoreObject<br>• UploadPart<br>• UploadPartCopy (same-region copies only) |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| **S3 Object Lambda** | `AWS::S3ObjectLambda::AccessPoint` | • AbortMultipartUpload<br>• CompleteMultipartUpload<br>• CopyObject (same-region copies only)<br>• CreateMultipartUpload<br>• DeleteObject<br>• DeleteObjectTagging<br>• GetObject<br>• GetObjectAcl<br>• GetObjectLegalHold<br>• GetObjectRetention<br>• GetObjectTagging<br>• HeadObject<br>• ListMultipartUploads<br>• ListObjects<br>• ListObjectVersions<br>• ListParts<br>• PutObject<br>• PutObjectLegalHold<br>• PutObjectRetention<br>• PutObjectAcl<br>• PutObjectTagging<br>• RestoreObject<br>• UploadPart<br>• WriteGetObjectResponse |

| Data event type (console) | resources.type value | Data APIs logged to CloudTrail |
| --- | --- | --- |
| **S3 Outposts** | `AWS::S3Outposts::Object` | • AbortMultipartUpload<br>• CompleteMultipartUpload<br>• CopyObject (same-region copies only)<br>• CreateMultipartUpload<br>• DeleteObject<br>• DeleteObjectTagging<br>• GetObject<br>• GetObjectTagging<br>• HeadObject<br>• ListMultipartUploads<br>• ListObjects<br>• ListObjectsV2<br>• ListParts<br>• PutObject<br>• PutObjectTagging<br>• UploadPart<br>• UploadPartCopy |

You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.`ARN fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector in the *AWS CloudTrail API Reference*.

## Amazon S3 management events in CloudTrail

Amazon S3 logs all control plane operations as management events. For more information about S3 API operations, see the Amazon S3 API Reference.

## How CloudTrail captures requests made to Amazon S3

By default, CloudTrail logs S3 bucket-level API calls that were made in the last 90 days, but not log requests made to objects. Bucket-level calls include events such as `CreateBucket`, `DeleteBucket`, `PutBucketLifecycle`, `PutBucketPolicy`, and so on. You can see bucket-level events on the CloudTrail console. However, you can't view data events (Amazon S3 object-level calls) there—you must parse or query CloudTrail logs for them.

## Amazon S3 account-level actions tracked by CloudTrail logging

CloudTrail logs account-level actions. Amazon S3 records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

The tables in this section list the Amazon S3 account-level actions that are supported for logging by CloudTrail.

Amazon S3 account-level API actions tracked by CloudTrail logging appear as the following event names. The CloudTrail event names differ from the API action name. For example, DeletePublicAccessBlock is DeleteAccountPublicAccessBlock.

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

## Amazon S3 bucket-level actions that are tracked by CloudTrail logging

By default, CloudTrail logs bucket-level actions for general purpose buckets. Amazon S3 records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

This section lists the Amazon S3 bucket-level actions that are supported for logging by CloudTrail.

Amazon S3 bucket-level API actions tracked by CloudTrail logging appear as the following event names. In some cases, the CloudTrail event name differs from the API action name. For example, `PutBucketLifecycleConfiguration` is `PutBucketLifecycle`.

- CreateBucket
- DeleteBucket

- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration
- DeleteBucketLifecycle
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors
- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration
- GetBucketOwnershipControls
- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication

- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite
- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle
- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning
- PutBucketWebsite

In addition to these API operations, you can also use the OPTIONS object object-level action. This action is treated like a bucket-level action in CloudTrail logging because the action checks the CORS configuration of a bucket.

## Amazon S3 Express One Zone bucket-level (Regional API endpoint) actions tracked by CloudTrail logging

By default, CloudTrail logs bucket-level actions for directory buckets as management events. The `eventsource` for CloudTrail management events for S3 Express One Zone is `s3express.amazonaws.com`.

These following Regional endpoint API operations are logged to CloudTrail.

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets

For more information, see Logging with AWS CloudTrail for S3 Express One Zone

## Amazon S3 object-level actions in cross-account scenarios

The following are special use cases involving the object-level API calls in cross-account scenarios and how CloudTrail logs are reported. CloudTrail delivers logs to the requester (the account that made the API call), except in some access denied cases where log entries are redacted or omitted. When setting up cross-account access, consider the examples in this section.

> **Note**
>
> The examples assume that CloudTrail logs are appropriately configured.

### Example 1: CloudTrail delivers logs to the bucket owner

CloudTrail delivers logs to the bucket owner even if the bucket owner does not have permissions for the same object API operation. Consider the following cross-account scenario:

- Account A owns the bucket.
- Account B (the requester) tries to access an object in that bucket.
- Account C owns the object. Account C might or might not be the same account as Account A.

> **Note**
>
> CloudTrail always delivers object-level API logs to the requester (Account B). In addition, CloudTrail also delivers the same logs to the bucket owner (Account A) even when the bucket owner does not own the object (Account C) or have permissions for those same API operations on that object.

**Example 2: CloudTrail does not proliferate email addresses that are used in setting object ACLs**

Consider the following cross-account scenario:

- Account A owns the bucket.
- Account B (the requester) sends a request to set an object ACL grant by using an email address. For more information about ACLs, see Access control list (ACL) overview.

The requester gets the logs along with the email information. However, the bucket owner—if they are eligible to receive logs, as in example 1—gets the CloudTrail log reporting the event. However, the bucket owner doesn't get the ACL configuration information, specifically the grantee email address and the grant. The only information that the log tells the bucket owner is that an ACL API call was made by Account B.

## CloudTrail log file entries for Amazon S3 and S3 on Outposts

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log

files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

> **Note**
>
> To view CloudTrail log file examples for Amazon S3 Express One Zone, see CloudTrail log file examples for S3 Express One Zone.

For more information, see the following examples.

**Topics**

- Example: CloudTrail log file entry for Amazon S3
- Example: Amazon S3 on Outposts log file entries

## Example: CloudTrail log file entry for Amazon S3

The following example shows a CloudTrail log entry that demonstrates the GET Service, PutBucketAcl, and GetBucketVersioning actions.

```
{
    "Records": [
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
```

```
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
```

```
                    },
                   "Permission": "FULL_CONTROL"
                }
            },
            "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
            "Owner": {
                "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
            }
        },
        "host": [
            "s3.us-west-2.amazonaws.com"
        ],
        "acl": [
            ""
        ]
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "SigV4",
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "AuthenticationMethod": "AuthHeader"
    },
    "requestID": "BD8798EACDD16751",
    "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "s3.amazonaws.com"
  }
  },
  {
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
      },
    "eventTime": "2019-02-01T03:26:37Z",
```

```
      "eventSource": "s3.amazonaws.com",
      "eventName": "GetBucketVersioning",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "bucketName": "amzn-s3-demo-bucket1",
          "versioning": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```


## Example: Amazon S3 on Outposts log file entries

Amazon S3 on Outposts management events are available via AWS CloudTrail. For more information, see Logging Amazon S3 API calls using AWS CloudTrail. In addition, you can optionally enable logging for data events in AWS CloudTrail.

A *trail* is a configuration that enables delivery of events as log files to an S3 bucket in a Region that you specify. CloudTrail logs for your Outposts buckets include a new field, `edgeDeviceDetails`, which identifies the Outpost where the specified bucket is located.

Additional log fields include the requested action, the date and time of the action, and the request parameters. CloudTrail log files are not an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates a PutObject action on `s3-outposts`.


```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/yourUserName",
        "accountId": "222222222222",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "yourUserName"
      },
      "eventTime": "2020-11-30T15:44:33Z",
      "eventSource": "s3-outposts.amazonaws.com",
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "26.29.66.20",
      "userAgent": "aws-cli/1.18.39 Python/3.4.10 Darwin/18.7.0 botocore/1.15.39",
      "requestParameters": {
        "expires": "Wed, 21 Oct 2020 07:28:00 GMT",
        "Content-Language": "english",
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "ObjectCannedACL": "BucketOwnerFullControl",
        "x-amz-server-side-encryption": "Aes256",
        "Content-Encoding": "gzip",
        "Content-Length": "10",
        "Cache-Control": "no-cache",
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Disposition": "attachment",
        "Content-MD5": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "x-amz-storage-class": "Outposts",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "bucketName": "amzn-s3-demo-bucket1",
        "Key": "path/upload.sh"
      },
      "responseElements": {
```

```
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "x-amz-server-side-encryption": "Aes256",
        "x-amz-version-id": "001",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "ETag": "d41d8cd98f00b204e9800998ecf8427f"
      },
      "additionalEventData": {
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "bytesTransferredIn": 10,
        "x-amz-id-2": "29xXQBV20
+xOHKItvzY1suLv1i6A52E0zOX159fpfsItYd58JhXwKxXAXI4IQkp6",
        "SignatureVersion": "SigV4",
        "bytesTransferredOut": 20,
        "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "8E96D972160306FA",
      "eventID": "ee3b4e0c-ab12-459b-9998-0a5a6f2e4015",
      "readOnly": false,
      "resources": [
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Object",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/path/upload.sh"
        },
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Bucket",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "444455556666",
      "sharedEventID": "02759a4c-c040-4758-b84b-7cbaaf17747a",
      "edgeDeviceDetails": {
        "type": "outposts",
        "deviceId": "op-01ac5d28a6a232904"
      },
      "eventCategory": "Data"
    }
```

# Enabling CloudTrail event logging for S3 buckets and objects

You can use CloudTrail data events to get information about bucket and object-level requests in Amazon S3. To enable CloudTrail data events for all of your buckets or for a list of specific buckets, you must create a trail manually in CloudTrail.

**Note**

- The default setting for CloudTrail is to find only management events. Check to ensure that you have the data events enabled for your account.
- With an S3 bucket that is generating a high workload, you could quickly generate thousands of logs in a short amount of time. Be mindful of how long you choose to enable CloudTrail data events for a busy bucket.

CloudTrail stores Amazon S3 data event logs in an S3 bucket of your choosing. Consider using a bucket in a separate AWS account to better organize events from multiple buckets that you might own into a central place for easier querying and analysis. AWS Organizations helps you create an AWS account that is linked to the account that owns the bucket that you're monitoring. For more information, see What is AWS Organizations? in the *AWS Organizations User Guide*.

When you log data events for a trail in CloudTrail, you can choose to use advanced event selectors or basic event selectors to log data events for objects stored in general purpose buckets. To log data events for objects stored in directory buckets, you must use advanced event selectors. For more information, see Logging with AWS CloudTrail for S3 Express One Zone.

When you create a trail in the CloudTrail console using advanced event selectors, in the data events section, you can choose **Log all events** for the **Log selector template** to log all object-level events. When you create a trail in the CloudTrail console using basic event selectors, in the data events section, you can select the **Select all S3 buckets in your account** check box to log all object-level events.

**Note**

- It's a best practice to create a lifecycle configuration for your AWS CloudTrail data event bucket. Configure the lifecycle configuration to periodically remove log files after the period of time you believe you need to audit them. Doing so reduces the amount of

data that Athena analyzes for each query. For more information, see Setting a lifecycle configuration on a bucket.

- For more information about logging format, see Logging Amazon S3 API calls using AWS CloudTrail.
- For examples of how to query CloudTrail logs, see the *AWS Big Data Blog* post Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena.

## Enable logging for objects in a bucket using the console

You can use the Amazon S3 console to configure an AWS CloudTrail trail to log data events for objects in an S3 bucket. CloudTrail supports logging Amazon S3 object-level API operations such as `GetObject`, `DeleteObject`, and `PutObject`. These events are called *data events*.

By default, CloudTrail trails don't log data events, but you can configure trails to log data events for S3 buckets that you specify, or to log data events for all the Amazon S3 buckets in your AWS account. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

CloudTrail does not populate data events in the CloudTrail event history. Additionally, not all bucket-level actions are populated in the CloudTrail event history. For more information about the Amazon S3 bucket–level API actions tracked by CloudTrail logging, see Amazon S3 bucket-level actions that are tracked by CloudTrail logging. For more information about how to query CloudTrail logs, see the AWS Knowledge Center article about using Amazon CloudWatch Logs filter patterns and Amazon Athena to query CloudTrail logs.

To configure a trail to log data events for an S3 bucket, you can use either the AWS CloudTrail console or the Amazon S3 console. If you are configuring a trail to log data events for all the Amazon S3 buckets in your AWS account, it's easier to use the CloudTrail console. For information about using the CloudTrail console to configure a trail to log S3 data events, see Data events in the *AWS CloudTrail User Guide*.

**Important**

Additional charges apply for data events. For more information, see AWS CloudTrail pricing.

The following procedure shows how to use the Amazon S3 console to configure a CloudTrail trail to log data events for an S3 bucket.

**To enable CloudTrail data events logging for objects in an S3 general purpose bucket or in an S3 directory bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket.
3. Choose **Properties**.
4. Under **AWS CloudTrail data events**, choose **Configure in CloudTrail**.

   You can create a new CloudTrail trail or reuse an existing trail and configure Amazon S3 data events to be logged in your trail. For information about how to create trails in the CloudTrail console, see Creating and updating a trail with the console in the *AWS CloudTrail User Guide*. For information about how to configure Amazon S3 data event logging in the CloudTrail console, see Logging data events for Amazon S3 Objects in the *AWS CloudTrail User Guide*.

   > **Note**
   >
   > If you use the CloudTrail console or the Amazon S3 console to configure a trail to log data events for an S3 bucket, the Amazon S3 console shows that object-level logging is enabled for the bucket.

**To disable CloudTrail data events logging for objects in an S3 bucket**

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the left navigation pane, choose **Trails**.
3. Choose the name of the trail that you created to log events for your bucket.
4. On the details page for your trail, choose **Stop logging** in the upper-right corner.
5. In the dialog box that appears, choose **Stop logging**.

For information about enabling object-level logging when you create an S3 bucket, see Creating a bucket.

For more information about CloudTrail logging with S3 buckets, see the following topics:

- Viewing the properties for an S3 bucket
- Logging Amazon S3 API calls using AWS CloudTrail
- Working with CloudTrail Log Files in the *AWS CloudTrail User Guide*

## Identifying Amazon S3 requests using CloudTrail

In Amazon S3, you can identify requests using an AWS CloudTrail event log. AWS CloudTrail is the preferred way of identifying Amazon S3 requests, but if you are using Amazon S3 server access logs, see the section called "Identifying S3 requests".

**Topics**

- Identifying requests made to Amazon S3 in a CloudTrail log
- Identifying Amazon S3 Signature Version 2 requests by using CloudTrail
- Identifying access to S3 objects by using CloudTrail

### Identifying requests made to Amazon S3 in a CloudTrail log

After you set up CloudTrail to deliver events to a bucket, you should start to see objects go to your destination bucket on the Amazon S3 console. These are formatted as follows:

```
s3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd
```

Events logged by CloudTrail are stored as compressed, gzipped JSON objects in your S3 bucket. To efficiently find requests, you should use a service like Amazon Athena to index and query the CloudTrail logs.

For more information about CloudTrail and Athena, see Creating the table for AWS CloudTrail logs in Athena using partition projection in the *Amazon Athena User Guide*.

### Identifying Amazon S3 Signature Version 2 requests by using CloudTrail

You can use a CloudTrail event log to identify which API signature version was used to sign a request in Amazon S3. This capability is important because support for Signature Version 2 will be turned off (deprecated). After that, Amazon S3 will no longer accept requests that use Signature Version 2, and all requests must use *Signature Version 4* signing.

We *strongly* recommend that you use CloudTrail to help determine whether any of your workflows are using Signature Version 2 signing. Remediate them by upgrading your libraries and code to use Signature Version 4 instead to prevent any impact to your business.

For more information, see Announcement: AWS CloudTrail for Amazon S3 adds new fields for enhanced security auditing in AWS re:Post.

> **Note**
>
> CloudTrail events for Amazon S3 include the signature version in the request details under the key name of 'additionalEventData. To find the signature version on requests made for objects in Amazon S3 such as GET, PUT, and DELETE requests, you must enable CloudTrail data events. (This feature is turned off by default.)

AWS CloudTrail is the preferred method for identifying Signature Version 2 requests. If you're using Amazon S3 server-access logs, see Identifying Signature Version 2 requests by using Amazon S3 access logs.

**Topics**

- Athena query examples for identifying Amazon S3 Signature Version 2 requests
- Partitioning Signature Version 2 data

### Athena query examples for identifying Amazon S3 Signature Version 2 requests

#### Example — Select all Signature Version 2 events, and print only EventTime, S3_Action, Request_Parameters, Region, SourceIP, and UserAgent

In the following Athena query, replace *s3_cloudtrail_events_db.cloudtrail_table* with your Athena details, and increase or remove the limit as needed.

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

**Example — Select all requesters that are sending Signature Version 2 traffic**

```
SELECT useridentity.arn, Count(requestid) as RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

### Partitioning Signature Version 2 data

If you have a large amount of data to query, you can reduce the costs and running time of Athena by creating a partitioned table.

To do this, create a new table with partitions as follows.

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned(
     eventversion STRING,
     userIdentity STRUCT<
         type:STRING,
         principalid:STRING,
         arn:STRING,
         accountid:STRING,
         invokedby:STRING,
         accesskeyid:STRING,
         userName:STRING,
       sessioncontext:STRUCT<
                  attributes:STRUCT<
                  mfaauthenticated:STRING,
                  creationdate:STRING>,
                  sessionIssuer:STRUCT<
                  type:STRING,
                  principalId:STRING,
                  arn:STRING,
                  accountId:STRING,
                  userName:STRING>
              >
          >,
     eventTime STRING,
     eventSource STRING,
```

```
        eventName STRING,
        awsRegion STRING,
        sourceIpAddress STRING,
        userAgent STRING,
        errorCode STRING,
        errorMessage STRING,
        requestParameters STRING,
        responseElements STRING,
        additionalEventData STRING,
        requestId STRING,
        eventId STRING,
        resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
    STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/';
```

Then, create the partitions individually. You can't get results from dates that you haven't created.

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
```

```
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

You can then make the request based on these partitions, and you don't need to load the full bucket.

```
SELECT useridentity.arn,
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

## Identifying access to S3 objects by using CloudTrail

You can use your AWS CloudTrail event logs to identify Amazon S3 object access requests for data events such as `GetObject`, `DeleteObject`, and `PutObject`, and discover additional information about those requests.

The following example shows how to get all PUT object requests for Amazon S3 from an AWS CloudTrail event log.

### Topics

- Athena query examples for identifying Amazon S3 object access requests

#### Athena query examples for identifying Amazon S3 object access requests

In the following Athena query examples, replace *s3_cloudtrail_events_db.cloudtrail_table* with your Athena details, and modify the date range as needed.

**Example — Select all events that have PUT object access requests, and print only `EventTime`, `EventSource`, `SourceIP`, `UserAgent`, `BucketName`, `object`, and `UserARN`**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'PutObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

**Example — Select all events that have GET object access requests, and print only `EventTime`, `EventSource`, `SourceIP`, `UserAgent`, `BucketName`, `object`, and `UserARN`**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

**Example — Select all anonymous requester events to a bucket in a certain period and print only `EventTime`, `EventName`, `EventSource`, `SourceIP`, `UserAgent`, `BucketName`, `UserARN`, and `AccountID`**

```
SELECT
```

```
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

**Example — Identify all requests that required an ACL for authorization**

The following Amazon Athena query example shows how to identify all requests to your S3 buckets that required an access control list (ACL) for authorization. If the request required an ACL for authorization, the `aclRequired` value in `additionalEventData` is `Yes`. If no ACLs were required, `aclRequired` is not present. You can use this information to migrate those ACL permissions to the appropriate bucket policies. After you've created these bucket policies, you can disable ACLs for these buckets. For more information about disabling ACLs, see Prerequisites for disabling ACLs.

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

**Note**

- These query examples can also be useful for security monitoring. You can review the results for `PutObject` or `GetObject` calls from unexpected or unauthorized IP addresses or requesters and for identifying any anonymous requests to your buckets.
- This query only retrieves information from the time at which logging was enabled.

If you are using Amazon S3 server access logs, see Identifying object access requests by using Amazon S3 access logs.

# Logging requests with server access logging

Server access logging provides detailed records for the requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. This information can also help you learn about your customer base and understand your Amazon S3 bill.

**Note**

Server access logs don't record information about wrong-Region redirect errors for Regions that launched after March 20, 2019. Wrong-Region redirect errors occur when a request for an object or bucket is made outside the Region in which the bucket exists.

## How do I enable log delivery?

To enable log delivery, perform the following basic steps. For details, see Enabling Amazon S3 server access logging.

1. **Provide the name of the destination bucket** (also known as a *target bucket*). This bucket is where you want Amazon S3 to save the access logs as objects. Both the source and destination buckets must be in the same AWS Region and owned by the same account. The destination bucket must not have an S3 Object Lock default retention period configuration. The destination bucket must also not have Requester Pays enabled.

You can have logs delivered to any bucket that you own that is in the same Region as the source bucket, including the source bucket itself. But for simpler log management, we recommend that you save access logs in a different bucket.

When your source bucket and destination bucket are the same bucket, additional logs are created for the logs that are written to the bucket, which creates an infinite loop of logs. We do not recommend doing this because it could result in a small increase in your storage billing. In addition, the extra logs about logs might make it harder to find the log that you are looking for.

If you choose to save access logs in the source bucket, we recommend that you specify a destination prefix (also known as a *target prefix*) for all log object keys. When you specify a prefix, all the log object names begin with a common string, which makes the log objects easier to identify.

2. **(Optional) Assign a destination prefix to all Amazon S3 log object keys.** The destination prefix (also known as a *target prefix*) makes it simpler for you to locate the log objects. For example, if you specify the prefix value `logs/`, each log object that Amazon S3 creates begins with the `logs/` prefix in its key, for example:

   ```
   logs/2013-11-01-21-32-16-E568B2907131C0C0
   ```

   If you specify the prefix value `logs`, the log object appears as follows:

   ```
   logs2013-11-01-21-32-16-E568B2907131C0C0
   ```

   Prefixes are also useful to distinguish between source buckets when multiple buckets log to the same destination bucket.

   This prefix can also help when you delete the logs. For example, you can set a lifecycle configuration rule for Amazon S3 to delete objects with a specific prefix. For more information, see Deleting Amazon S3 log files.

3. **(Optional) Set permissions so that others can access the generated logs.** By default, only the bucket owner always has full access to the log objects. If your destination bucket uses the Bucket owner enforced setting for S3 Object Ownership to disable access control lists (ACLs), you can't grant permissions in destination grants (also known as *target grants*) that use ACLs. However, you can update your bucket policy for the destination bucket to grant access to others.

For more information, see Identity and Access Management for Amazon S3 and Permissions for log delivery.

4. **(Optional) Set a log object key format for the log files.** You have two options for the log object key format (also known as the *target object key format*):

   - **Non-date-based partitioning** – This is the original log object key format. If you choose this format, the log file key format appears as follows:

     ```
     [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     For example, if you specify `logs/` as the prefix, your log objects are named like this:

     ```
     logs/2013-11-01-21-32-16-E568B2907131C0C0
     ```

   - **Date-based partitioning** – If you choose date-based partitioning, you can choose the event time or delivery time for the log file as the date source used in the log format. This format makes it easier to query the logs.

     If you choose date-based partitioning, the log file key format appears as follows:

     ```
     [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/
     [DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     For example, if you specify `logs/` as the target prefix, your log objects are named like this:

     ```
     logs/123456789012/us-west-2/DOC-EXAMPLE-SOURCE-
     BUCKET/2023/03/01/2023-03-01-21-32-16-E568B2907131C0C0
     ```

     For delivery time delivery, the time in the log file names corresponds to the delivery time for the log files.

     For event time delivery, the year, month, and day correspond to the day on which the event occurred, and the hour, minutes and seconds are set to `00` in the key. The logs delivered in these log files are for a specific day only.

If you're configuring your logs through the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API, use `TargetObjectKeyFormat` to specify the log object key format. To specify non-date-based partitioning, use `SimplePrefix`. To specify data-based partitioning,

use PartitionedPrefix. If you use PartitionedPrefix, use PartitionDateSource to specify either EventTime or DeliveryTime.

For SimplePrefix, the log file key format appears as follows:

```
[TargetPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

For PartitionedPrefix with event time or delivery time, the log file key format appears as follows:

```
[TargetPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

## Log object key format

Amazon S3 uses the following object key formats for the log objects that it uploads in the destination bucket:

- **Non-date-based partitioning** – This is the original log object key format. If you choose this format, the log file key format appears as follows:

  ```
  [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

- **Date-based partitioning** – If you choose date-based partitioning, you can choose the event time or delivery time for the log file as the date source used in the log format. This format makes it easier to query the logs.

  If you choose date-based partitioning, the log file key format appears as follows:

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
  [YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

In the log object key, YYYY, MM, DD, hh, mm, and ss are the digits of the year, month, day, hour, minute, and seconds (respectively). These dates and times are in Coordinated Universal Time (UTC).

A log file delivered at a specific time can contain records written at any point before that time. There is no way to know whether all log records for a certain time interval have been delivered or not.

The `UniqueString` component of the key is there to prevent overwriting of files. It has no meaning, and log processing software should ignore it.

## How are logs delivered?

Amazon S3 periodically collects access log records, consolidates the records in log files, and then uploads log files to your destination bucket as log objects. If you enable logging on multiple source buckets that identify the same destination bucket, the destination bucket will have access logs for all those source buckets. However, each log object reports access log records for a specific source bucket.

Amazon S3 uses a special log delivery account to write server access logs. These writes are subject to the usual access control restrictions. We recommend that you update the bucket policy on the destination bucket to grant access to the logging service principal (`logging.s3.amazonaws.com`) for access log delivery. You can also grant access for access log delivery to the S3 log delivery group through your bucket access control list (ACL). However, granting access to the S3 log delivery group by using your bucket ACL is not recommended.

When you enable server access logging and grant access for access log delivery through your destination bucket policy, you must update the policy to allow `s3:PutObject` access for the logging service principal. If you use the Amazon S3 console to enable server access logging, the console automatically updates the destination bucket policy to grant these permissions to the logging service principal. For more information about granting permissions for server access log delivery, see Permissions for log delivery.

> **Note**
>
> S3 does not support delivery of CloudTrail logs or server access logs to the requester or the bucket owner for VPC endpoint requests when the VPC endpoint policy denies them or for requests that fail before the VPC policy is evaluated.

### Bucket owner enforced setting for S3 Object Ownership

If the destination bucket uses the Bucket owner enforced setting for Object Ownership, ACLs are disabled and no longer affect permissions. You must update the bucket policy on the destination bucket to grant access to the logging service principal. For more information about Object Ownership, see Grant access to the S3 log delivery group for server access logging.

## Best-effort server log delivery

Server access log records are delivered on a best-effort basis. Most requests for a bucket that is properly configured for logging result in a delivered log record. Most log records are delivered within a few hours of the time that they are recorded, but they can be delivered more frequently.

The completeness and timeliness of server logging is not guaranteed. The log record for a particular request might be delivered long after the request was actually processed, or *it might not be delivered at all*. It is possible that you might even see a duplication of a log record. The purpose of server logs is to give you an idea of the nature of traffic against your bucket. Although log records are rarely lost or duplicated, be aware that server logging is not meant to be a complete accounting of all requests.

Because of the best-effort nature of server logging, your usage reports might include one or more access requests that do not appear in a delivered server log. You can find these usage reports under **Cost & usage reports** in the AWS Billing and Cost Management console.

## Bucket logging status changes take effect over time

Changes to the logging status of a bucket take time to actually affect the delivery of log files. For example, if you enable logging for a bucket, some requests made in the following hour might be logged, and others might not. Suppose that you change the destination bucket for logging from bucket A to bucket B. For the next hour, some logs might continue to be delivered to bucket A, whereas others might be delivered to the new destination bucket B. In all cases, the new settings eventually take effect without any further action on your part.

For more information about logging and log files, see the following sections:

**Topics**

- Enabling Amazon S3 server access logging
- Amazon S3 server access log format
- Deleting Amazon S3 log files
- Using Amazon S3 server access logs to identify requests

## Enabling Amazon S3 server access logging

Server access logging provides detailed records for the requests that are made to an Amazon S3 bucket. Server access logs are useful for many applications. For example, access log information

can be useful in security and access audits. This information can also help you learn about your customer base and understand your Amazon S3 bill.

By default, Amazon S3 doesn't collect server access logs. When you enable logging, Amazon S3 delivers access logs for a source bucket to a destination bucket (also known as a *target bucket*) that you choose. The destination bucket must be in the same AWS Region and AWS account as the source bucket.

An access log record contains details about the requests that are made to a bucket. This information can include the request type, the resources that are specified in the request, and the time and date that the request was processed. For more information about logging basics, see Logging requests with server access logging.

**Important**

- There is no extra charge for enabling server access logging on an Amazon S3 bucket. However, any log files that the system delivers to you will accrue the usual charges for storage. (You can delete the log files at any time.) We do not assess data-transfer charges for log file delivery, but we do charge the normal data-transfer rate for accessing the log files.
- Your destination bucket should not have server access logging enabled. You can have logs delivered to any bucket that you own that is in the same Region as the source bucket, including the source bucket itself. However, delivering logs to the source bucket will cause an infinite loop of logs and is not recommended. For simpler log management, we recommend that you save access logs in a different bucket. For more information, see How do I enable log delivery?
- S3 buckets that have S3 Object Lock enabled can't be used as destination buckets for server access logs. Your destination bucket must not have a default retention period configuration.
- The destination bucket must not have Requester Pays enabled.
- You can use default bucket encryption on the destination bucket *only* if you use server-side encryption with Amazon S3 managed keys (SSE-S3), which uses the 256-bit Advanced Encryption Standard (AES-256). Default server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) is not supported.

You can enable or disable server access logging by using the Amazon S3 console, Amazon S3 API, the AWS Command Line Interface (AWS CLI), or AWS SDKs.

## Permissions for log delivery

Amazon S3 uses a special log delivery account to write server access logs. These writes are subject to the usual access control restrictions. For access log delivery, you must grant the logging service principal (`logging.s3.amazonaws.com`) access to your destination bucket.

To grant permissions to Amazon S3 for log delivery, you can use either a bucket policy or bucket access control lists (ACLs), depending on your destination bucket's S3 Object Ownership settings. However, we recommend that you use a bucket policy instead of ACLs.

### Bucket owner enforced setting for S3 Object Ownership

If the destination bucket uses the Bucket owner enforced setting for Object Ownership, ACLs are disabled and no longer affect permissions. In this case, you must update the bucket policy for the destination bucket to grant access to the logging service principal. You can't update your bucket ACL to grant access to the S3 log delivery group. You also can't include destination grants (also known as *target grants*) in your PutBucketLogging configuration.

For information about migrating existing bucket ACLs for access log delivery to a bucket policy, see Grant access to the S3 log delivery group for server access logging. For more information about Object Ownership, see Controlling ownership of objects and disabling ACLs for your bucket. When you create new buckets, ACLs are disabled by default.

### Granting access by using a bucket policy

To grant access by using the bucket policy on the destination bucket, update the bucket policy to grant the `s3:PutObject` permission to the logging service principal. If you use the Amazon S3 console to enable server access logging, the console automatically updates the bucket policy on the destination bucket to grant this permission to the logging service principal. If you enable server access logging programmatically, you must manually update the bucket policy for the destination bucket to grant access to the logging service principal.

For an example bucket policy that grants access to the logging service principal, see the section called “Grant permissions to the logging service principal by using a bucket policy”.

### Granting access by using bucket ACLs

You can alternately use bucket ACLs to grant access for access log delivery. You add a grant entry to the bucket ACL that grants `WRITE` and `READ_ACP` permissions to the S3 log delivery group. However, granting access to the S3 log delivery group by using bucket ACLs is not recommended. For more information, see Controlling ownership of objects and disabling ACLs for your bucket. For information about migrating existing bucket ACLs for access log delivery to a bucket policy, see Grant access to the S3 log delivery group for server access logging. For an example ACL that grants access to the logging service principal, see the section called "Grant permissions to the log delivery group by using a bucket ACL".

**Grant permissions to the logging service principal by using a bucket policy**

This example bucket policy grants the `s3:PutObject` permission to the logging service principal (`logging.s3.amazonaws.com`). To use this bucket policy, replace the *`user input placeholders`* with your own information. In the following policy, *`amzn-s3-demo-destination-bucket`* is the destination bucket where server access logs will be delivered, and *`amzn-s3-demo-source-bucket`* is the source bucket. *`EXAMPLE-LOGGING-PREFIX`* is the optional destination prefix (also known as a *target prefix*) that you want to use for your log objects. *`SOURCE-ACCOUNT-ID`* is the AWS account that owns the source bucket.

> **Note**
>
> If there are `Deny` statements in your bucket policy, make sure that they don't prevent Amazon S3 from delivering access logs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
```

```
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

### Grant permissions to the log delivery group by using a bucket ACL

> **Note**
>
> As a security best practice, Amazon S3 disables access control lists (ACLs) by default in all new buckets. For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

Although we do not recommend this approach, you can grant permissions to the log delivery group by using a bucket ACL. However, if the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination grants (also known as *target grants*) in your PutBucketLogging configuration. Instead, you must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

In the bucket ACL, the log delivery group is represented by the following URL:

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

To grant `WRITE` and `READ_ACP` (ACL read) permissions, add the following grants to the destination bucket ACL:

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
```

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
</Grant>
```

For examples of adding ACL grants programmatically, see Configuring ACLs.

**Important**

When you enable Amazon S3 server access logging by using AWS CloudFormation on a bucket and you're using ACLs to grant access to the S3 log delivery group, you must also add `"AccessControl": "LogDeliveryWrite"` to your CloudFormation template. Doing so is important because you can grant those permissions only by creating an ACL for the bucket, but you can't create custom ACLs for buckets in CloudFormation. You can use only canned ACLs with CloudFormation.

## To enable server access logging

To enable server access logging by using the Amazon S3 console, Amazon S3 REST API, AWS SDKs, and AWS CLI, use the following procedures.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable server access logging for.
3. Choose **Properties**.
4. In the **Server access logging** section, choose **Edit**.
5. Under **Server access logging**, choose **Enable**.
6. Under **Destination bucket**, specify a bucket and an optional prefix. If you specify a prefix, we recommend including a forward slash (/) after the prefix to make it easier to find your logs.

> **Note**
>
> Specifying a prefix with a slash (/) makes it simpler for you to locate the log objects. For example, if you specify the prefix value `logs/`, each log object that Amazon S3 creates begins with the `logs/` prefix in its key, as follows:
>
> ```
> logs/2013-11-01-21-32-16-E568B2907131C0C0
> ```
>
> If you specify the prefix value `logs`, the log object appears as follows:
>
> ```
> logs2013-11-01-21-32-16-E568B2907131C0C0
> ```

7. Under **Log object key format**, do one of the following:

   - To choose non-date-based partitioning, choose **[DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]**.
   - To choose date-based partitioning, choose **[DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]**, then choose **S3 event time** or **Log file delivery time**.

8. Choose **Save changes**.

   When you enable server access logging on a bucket, the console both enables logging on the source bucket and updates the bucket policy for the destination bucket to grant the `s3:PutObject` permission to the logging service principal (`logging.s3.amazonaws.com`). For more information about this bucket policy, see Grant permissions to the logging service principal by using a bucket policy.

   You can view the logs in the destination bucket. After you enable server access logging, it might take a few hours before the logs are delivered to the target bucket. For more information about how and when logs are delivered, see How are logs delivered?

For more information, see Viewing the properties for an S3 bucket.

### Using the REST API

To enable logging, you submit a PutBucketLogging request to add the logging configuration on the source bucket. The request specifies the destination bucket (also known as a *target bucket*) and, optionally, the prefix to be used with all log object keys.

The following example identifies *amzn-s3-demo-destination-bucket* as the destination bucket and *logs/* as the prefix.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The following example identifies *amzn-s3-demo-destination-bucket* as the destination bucket, *logs/* as the prefix, and `EventTime` as the log object key format.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The log objects are written and owned by the S3 log delivery account, and the bucket owner is granted full permissions on the log objects. You can optionally use destination grants (also known as *target grants*) to grant permissions to other users so that they can access the logs. For more information, see PutBucketLogging.

> **Note**
>
> If the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't use destination grants to grant permissions to other users. To grant permissions to

others, you can update the bucket policy on the destination bucket. For more information, see Permissions for log delivery.

To retrieve the logging configuration on a bucket, use the GetBucketLogging API operation.

To delete the logging configuration, you send a `PutBucketLogging` request with an empty `BucketLoggingStatus`:

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

To enable logging on a bucket, you can use either the Amazon S3 API or the AWS SDK wrapper libraries.

**Using the AWS SDKs**

The following examples enable logging on a bucket. You must create two buckets, a source bucket and a destination (target) bucket. The examples update the bucket ACL on the destination bucket first. They then grant the log delivery group the necessary permissions to write logs to the destination bucket, and then they enable logging on the source bucket.

These examples won't work on destination buckets that use the Bucket owner enforced setting for Object Ownership.

If the destination (target) bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination (target) grants in your PutBucketLogging configuration. You must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;
    using Microsoft.Extensions.Configuration;

    /// <summary>
    /// This example shows how to enable logging on an Amazon Simple Storage
    /// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
    /// this example. The first is the bucket for which you wish to enable
    /// logging, and the second is the location where you want to store the
    /// logs.
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);
```

```
                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
```

```
                                ""Principal"": { ""Service"":
 ""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
 ""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
 accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
 enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
```

```
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
                .Build();
        }
    }
```

- For API details, see PutBucketLogging in *AWS SDK for .NET API Reference*.

Java

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";

        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);

        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
```

```
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
 "\"\n" +
                "                }\n" +
                "            }\n" +
                "    ]\n" +
                "}";
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
    public void enableServerAccessLogging(String sourceBucketName, String
 targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
 TargetObjectKeyFormat.builder()

 .partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
```

```
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

### Using the AWS CLI

We recommend that you create a dedicated logging bucket in each AWS Region that you have S3 buckets in. Then have your Amazon S3 access logs delivered to that S3 bucket. For more information and examples, see put-bucket-logging in the *AWS CLI Reference*.

If the destination (target) bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination (target) grants in your PutBucketLogging configuration. You must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

#### Example — Enable access logs with five buckets across two Regions

In this example, you have the following five buckets:

- `1-amzn-s3-demo-bucket1-us-east-1`
- `2-amzn-s3-demo-bucket1-us-east-1`
- `3-amzn-s3-demo-bucket1-us-east-1`
- `1-amzn-s3-demo-bucket1-us-west-2`
- `2-amzn-s3-demo-bucket1-us-west-2`

> **Note**
>
> The final step of the following procedure provides example bash scripts that you can use to create your logging buckets and enable server access logging on these buckets. To use those scripts, you must create the `policy.json` and `logging.json` files, as described in the following procedure.

1. Create two logging destination buckets in the US West (Oregon) and US East (N. Virginia) Regions and give them the following names:

   - `amzn-s3-demo-bucket1-logs-us-east-1`
   - `amzn-s3-demo-bucket1-logs-us-west-2`

2. Later in these steps, you will enable server access logging as follows:

   - `1-amzn-s3-demo-bucket1-us-east-1` logs to the S3 bucket `amzn-s3-demo-bucket1-logs-us-east-1` with the prefix `1-amzn-s3-demo-bucket1-us-east-1`
   - `2-amzn-s3-demo-bucket1-us-east-1` logs to the S3 bucket `amzn-s3-demo-bucket1-logs-us-east-1` with the prefix `2-amzn-s3-demo-bucket1-us-east-1`
   - `3-amzn-s3-demo-bucket1-us-east-1` logs to the S3 bucket `amzn-s3-demo-bucket1-logs-us-east-1` with the prefix `3-amzn-s3-demo-bucket1-us-east-1`
   - `1-amzn-s3-demo-bucket1-us-west-2` logs to the S3 bucket `amzn-s3-demo-bucket1-logs-us-west-2` with the prefix `1-amzn-s3-demo-bucket1-us-west-2`
   - `2-amzn-s3-demo-bucket1-us-west-2` logs to the S3 bucket `amzn-s3-demo-bucket1-logs-us-west-2` with the prefix `2-amzn-s3-demo-bucket1-us-west-2`

3. For each destination logging bucket, grant permissions for server access log delivery by using a bucket ACL *or* a bucket policy:

   - **Update the bucket policy** (Recommended) – To grant permissions to the logging service principal, use the following `put-bucket-policy` command. Replace *`amzn-s3-demo-destination-bucket-logs`* with the name of your destination bucket.

     ```
     aws s3api put-bucket-policy --bucket amzn-s3-demo-destination-bucket-logs --
     policy file://policy.json
     ```

     `Policy.json` is a JSON document in the current folder that contains the following bucket policy. To use this bucket policy, replace the *`user input placeholders`* with your own information. In the following policy, *`amzn-s3-demo-destination-bucket-logs`* is the destination bucket where server access logs will be delivered, and *`amzn-s3-demo-source-bucket`* is the source bucket. *`SOURCE-ACCOUNT-ID`* is the AWS account that owns the source bucket.

     ```
     {
         "Version": "2012-10-17",
     ```

```
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket-logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

- **Update the bucket ACL** – To grant permissions to the S3 log delivery group, use the following `put-bucket-acl` command. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination (target) bucket.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-destination-bucket-logs  --
grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. Then, create a `logging.json` file that contains your logging configuration (based on one of the three examples that follow). After you create the `logging.json` file, you can apply the logging configuration by using the following `put-bucket-logging` command. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination (target) bucket.

```
aws s3api put-bucket-logging --bucket amzn-s3-demo-destination-bucket-logs --
bucket-logging-status file://logging.json
```

> **Note**
>
> Instead of using this `put-bucket-logging` command to apply the logging configuration on each destination bucket, you can use one of the bash scripts provided in the next step. To use those scripts, you must create the `policy.json` and `logging.json` files, as described in this procedure.

The `logging.json` file is a JSON document in the current folder that contains your logging configuration. If a destination bucket uses the Bucket owner enforced setting for Object Ownership, your logging configuration can't contain destination (target) grants. For more information, see Permissions for log delivery.

**Example – `logging.json` without destination (target) grants**

The following example `logging.json` file doesn't contain destination (target) grants. Therefore, you can apply this configuration to a destination (target) bucket that uses the Bucket owner enforced setting for Object Ownership.

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/"
     }
 }
```

**Example – `logging.json` with destination (target) grants**

The following example `logging.json` file contains destination (target) grants.

If the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't include destination (target) grants in your PutBucketLogging configuration. For more information, see Permissions for log delivery.

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetGrants": [
             {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "user@example.com"
                 },
                "Permission": "FULL_CONTROL"
             }
         ]
     }
 }
```

**Example – `logging.json` with the log object key format set to S3 event time**

The following `logging.json` file changes the log object key format to S3 event time. For more information about setting the log object key format, see the section called “How do I enable log delivery?”

```
{
  "LoggingEnabled": {
      "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
      "TargetPrefix": "amzn-s3-demo-destination-bucket/",
      "TargetObjectKeyFormat": {
          "PartitionedPrefix": {
              "PartitionDateSource": "EventTime"
          }
       }
   }
}
```

5. Use one of the following bash scripts to add access logging for all the buckets in your account. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination

(target) bucket, and replace *us-west-2* with the name of the Region that your buckets are located in.

> **Note**
>
> This script works only if all of your buckets are in the same Region. If you have buckets in multiple Regions, you must adjust the script.

**Example – Grant access with bucket policies and add logging for the buckets in your account**

```
loggingBucket='amzn-s3-demo-destination-bucket-logs'
region='us-west-2'


# Create the logging bucket.
aws s3 mb s3://$loggingBucket --region $region

aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json

# List the buckets in this account.
buckets="$(aws s3 ls | awk '{print $3}')"

# Put a bucket logging configuration on each bucket.
for bucket in $buckets
    do
      # This if statement excludes the logging bucket.
      if [ "$bucket" != "$loggingBucket" ] ; then
          continue;
      fi
      printf '{
        "LoggingEnabled": {
          "TargetBucket": "%s",
          "TargetPrefix": "%s/"
      }
    }' "$loggingBucket" "$bucket"  > logging.json
    aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
    echo "$bucket done"
done
```

```
  rm logging.json

  echo "Complete"
```

### Example – Grant access with bucket ACLs and add logging for the buckets in your account

```
  loggingBucket='amzn-s3-demo-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" != "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

### Verifying your server access logs setup

After you enable server access logging, complete the following steps:

- Access the destination bucket and verify that the log files are being delivered. After the access logs are set up, Amazon S3 immediately starts capturing requests and logging them. However, it might take a few hours before the logs are delivered to the destination bucket. For more information, see the section called "Bucket logging status changes take effect over time" and the section called "Best-effort server log delivery".

  You can also automatically verify log delivery by using Amazon S3 request metrics and setting up Amazon CloudWatch alarms for these metrics. For more information, see Monitoring metrics with Amazon CloudWatch.
- Verify that you are able to open and read the contents of the log files.

For server access logging troubleshooting information, see Troubleshoot server access logging.

## Amazon S3 server access log format

Server access logging provides detailed records for the requests that are made to an Amazon S3 bucket. You can use server access logs for the following purposes:

- Performing security and access audits
- Learning about your customer base
- Understanding your Amazon S3 bill

This section describes the format and other details about Amazon S3 server access log files.

Server access log files consist of a sequence of newline-delimited log records. Each log record represents one request and consists of space-delimited fields.

The following is an example log consisting of five log records.

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
```

```
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-demo-bucket1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /amzn-s3-demo-bucket1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /amzn-s3-demo-bucket1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /amzn-s3-demo-bucket1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
 ECDHE-RSA-AES128-SHA AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com
 TLSV1.2 - Yes
```

**Note**

Any field can be set to `-` to indicate that the data was unknown or unavailable, or that the field was not applicable to this request.

### Topics

- Log record fields
- Additional logging for copy operations

- Custom access log information
- Programming considerations for extensible server access log format

## Log record fields

The following list describes the log record fields.

**Bucket Owner**

The canonical user ID of the owner of the source bucket. The canonical user ID is another form of the AWS account ID. For more information about the canonical user ID, see AWS account identifiers in the *AWS General Reference*. For information about how to find the canonical user ID for your account, see Finding the canonical user ID for your AWS account.

**Example entry**

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

**Bucket**

The name of the bucket that the request was processed against. If the system receives a malformed request and cannot determine the bucket, the request will not appear in any server access log.

**Example entry**

```

```

**Time**

The time at which the request was received; these dates and times are in Coordinated Universal Time (UTC). The format, using `strftime()` terminology, is as follows: `[%d/%b/%Y:%H:%M:%S %z]`

**Example entry**

```
[06/Feb/2019:00:00:38 +0000]
```

**Remote IP**

The apparent IP address of the requester. Intermediate proxies and firewalls might obscure the actual IP address of the machine that's making the request.

**Example entry**

```
192.0.2.3
```

**Requester**

The canonical user ID of the requester, or a `-` for unauthenticated requests. If the requester was an IAM user, this field returns the requester's IAM user name along with the AWS account root user that the IAM user belongs to. This identifier is the same one used for access control purposes.

**Example entry**

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

If the requester is using an assumed role, this field returns the assumed IAM role.

**Example entry**

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

**Request ID**

A string generated by Amazon S3 to uniquely identify each request.

**Example entry**

```
3E57427F33A59F07
```

**Operation**

The operation listed here is declared as `SOAP.`*`operation`*, `REST.`*`HTTP_method`*`.`*`resource_type`*, `WEBSITE.`*`HTTP_method`*`.`*`resource_type`*, or `BATCH.DELETE.OBJECT`, or `S3.action.resource_type` for Lifecycle and logging.

**Example entry**

```
REST.PUT.OBJECT
```

**Key**

The key (object name) part of the request.

**Example entry**

```
/photos/2019/08/puppy.jpg
```

### Request-URI

The `Request-URI` part of the HTTP request message.

**Example Entry**

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

### HTTP status

The numeric HTTP status code of the response.

**Example entry**

```
200
```

### Error Code

The Amazon S3 Error code, or `-` if no error occurred.

**Example entry**

```
NoSuchBucket
```

### Bytes Sent

The number of response bytes sent, excluding HTTP protocol overhead, or `-` if zero.

**Example entry**

```
2662992
```

### Object Size

The total size of the object in question.

**Example entry**

```
3462992
```

**Total Time**

The number of milliseconds that the request was in flight from the server's perspective. This value is measured from the time that your request is received to the time that the last byte of the response is sent. Measurements made from the client's perspective might be longer because of network latency.

**Example entry**

```
70
```

**Turn-Around Time**

The number of milliseconds that Amazon S3 spent processing your request. This value is measured from the time that the last byte of your request was received until the time that the first byte of the response was sent.

**Example entry**

```
10
```

**Referer**

The value of the HTTP `Referer` header, if present. HTTP user-agents (for example, browsers) typically set this header to the URL of the linking or embedding page when making a request.

**Example entry**

```
"http://www.example.com/webservices"
```

**User-Agent**

The value of the HTTP `User-Agent` header.

**Example entry**

```
"curl/7.15.1"
```

**Version Id**

The version ID in the request, or `-` if the operation doesn't take a `versionId` parameter.

**Example entry**

```
3HL4kqtJvjVBH40Nrjfkd
```

### Host Id

The `x-amz-id-2` or Amazon S3 extended request ID.

**Example entry**

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

### Signature Version

The signature version, `SigV2` or `SigV4`, that was used to authenticate the request or a `-` for unauthenticated requests.

**Example entry**

```
SigV2
```

### Cipher Suite

The Secure Sockets Layer (SSL) cipher that was negotiated for an HTTPS request or a `-` for HTTP.

**Example entry**

```
ECDHE-RSA-AES128-GCM-SHA256
```

### Authentication Type

The type of request authentication used: `AuthHeader` for authentication headers, `QueryString` for query string (presigned URL), or a `-` for unauthenticated requests.

**Example entry**

```
AuthHeader
```

### Host Header

The endpoint used to connect to Amazon S3.

**Example entry**

```
s3.us-west-2.amazonaws.com
```

Some earlier Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, see Legacy endpoints. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

**TLS version**

The Transport Layer Security (TLS) version negotiated by the client. The value is one of following: `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, or `-` if TLS wasn't used.

**Example entry**

```
TLSv1.2
```

**Access Point ARN**

The Amazon Resource Name (ARN) of the access point of the request. If the access point ARN is malformed or not used, the field will contain a `-`. For more information about access points, see Using access points. For more information about ARNs, see Amazon Resource Name (ARN) in the *AWS Reference Guide*.

**Example entry**

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

**aclRequired**

A string that indicates whether the request required an access control list (ACL) for authorization. If the request required an ACL for authorization, the string is `Yes`. If no ACLs were required, the string is `-`. For more information about ACLs, see Access control list (ACL) overview. For more information about using the `aclRequired` field to disable ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

**Example entry**

```
Yes
```

## Additional logging for copy operations

A copy operation involves a GET and a PUT. For that reason, we log two records when performing a copy operation. The previous section describes the fields related to the PUT part of the operation. The following list describes the fields in the record that relate to the GET part of the copy operation.

**Bucket Owner**

The canonical user ID of the bucket that stores the object being copied. The canonical user ID is another form of the AWS account ID. For more information about the canonical user ID, see AWS account identifiers in the *AWS General Reference*. For information about how to find the canonical user ID for your account, see Finding the canonical user ID for your AWS account.

**Example entry**

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

**Bucket**

The name of the bucket that stores the object that's being copied.

**Example entry**

```
```

**Time**

The time at which the request was received; these dates and times are in Coordinated Universal Time (UTC). The format, using `strftime()` terminology, is as follows: `[%d/%B/%Y:%H:%M:%S %z]`

**Example entry**

```
[06/Feb/2019:00:00:38 +0000]
```

**Remote IP**

The apparent IP address of the requester. Intermediate proxies and firewalls might obscure the actual IP address of the machine that's making the request.

**Example entry**

```
192.0.2.3
```

**Requester**

The canonical user ID of the requester, or a `-` for unauthenticated requests. If the requester was an IAM user, this field will return the requester's IAM user name along with the AWS account root user that the IAM user belongs to. This identifier is the same one used for access control purposes.

**Example entry**

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

If the requester is using an assumed role, this field returns the assumed IAM role.

**Example entry**

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

**Request ID**

A string generated by Amazon S3 to uniquely identify each request.

**Example entry**

```
3E57427F33A59F07
```

**Operation**

The operation listed here is declared as `SOAP.operation`, `REST.HTTP_method.resource_type`, `WEBSITE.HTTP_method.resource_type`, or `BATCH.DELETE.OBJECT`.

**Example entry**

```
REST.COPY.OBJECT_GET
```

**Key**

The key (object name) of the object being copied, or `-` if the operation doesn't take a key parameter.

**Example entry**

```
/photos/2019/08/puppy.jpg
```

**Request-URI**

The `Request-URI` part of the HTTP request message.

**Example entry**

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar"
```

**HTTP status**

The numeric HTTP status code of the `GET` portion of the copy operation.

**Example entry**

```
200
```

**Error Code**

The Amazon S3 Error code of the `GET` portion of the copy operation, or `-` if no error occurred.

**Example entry**

```
NoSuchBucket
```

**Bytes Sent**

The number of response bytes sent, excluding the HTTP protocol overhead, or `-` if zero.

**Example entry**

```
2662992
```

**Object Size**

The total size of the object in question.

**Example entry**

```
3462992
```

**Total Time**

The number of milliseconds that the request was in flight from the server's perspective. This value is measured from the time that your request is received to the time that the last byte of the response is sent. Measurements made from the client's perspective might be longer because of network latency.

**Example entry**

```
70
```

**Turn-Around Time**

The number of milliseconds that Amazon S3 spent processing your request. This value is measured from the time that the last byte of your request was received until the time that the first byte of the response was sent.

**Example entry**

```
10
```

**Referer**

The value of the HTTP `Referer` header, if present. HTTP user-agents (for example, browsers) typically set this header to the URL of the linking or embedding page when making a request.

**Example entry**

```
"http://www.example.com/webservices"
```

**User-Agent**

The value of the HTTP `User-Agent` header.

**Example entry**

```
"curl/7.15.1"
```

**Version Id**

The version ID of the object being copied, or `-` if the `x-amz-copy-source` header didn't specify a `versionId` parameter as part of the copy source.

**Example Entry**

```
3HL4kqtJvjVBH40Nrjfkd
```

### Host Id

The `x-amz-id-2` or Amazon S3 extended request ID.

**Example entry**

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

### Signature Version

The signature version, `SigV2` or `SigV4`, that was used to authenticate the request, or a `-` for unauthenticated requests.

**Example entry**

```
SigV4
```

### Cipher Suite

The Secure Sockets Layer (SSL) cipher that was negotiated for an HTTPS request, or a `-` for HTTP.

**Example entry**

```
ECDHE-RSA-AES128-GCM-SHA256
```

### Authentication Type

The type of request authentication used: `AuthHeader` for authentication headers, `QueryString` for query strings (presigned URLs), or a `-` for unauthenticated requests.

**Example entry**

```
AuthHeader
```

**Host Header**

The endpoint that was used to connect to Amazon S3.

**Example entry**

```
s3.us-west-2.amazonaws.com
```

Some earlier Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, see Legacy endpoints. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

**TLS version**

The Transport Layer Security (TLS) version negotiated by the client. The value is one of following: `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, or `-` if TLS wasn't used.

**Example entry**

```
TLSv1.2
```

**Access Point ARN**

The Amazon Resource Name (ARN) of the access point of the request. If the access point ARN is malformed or not used, the field will contain a `-`. For more information about access points, see Using access points. For more information about ARNs, see Amazon Resource Name (ARN) in the *AWS Reference Guide*.

**Example entry**

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

**aclRequired**

A string that indicates whether the request required an access control list (ACL) for authorization. If the request required an ACL for authorization, the string is `Yes`. If no ACLs were required, the string is `-`. For more information about ACLs, see Access control list (ACL) overview. For more information about using the `aclRequired` field to disable ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

**Example entry**

```
Yes
```

### Custom access log information

You can include custom information to be stored in the access log record for a request. To do this, add a custom query-string parameter to the URL for the request. Amazon S3 ignores query-string parameters that begin with `x-`, but includes those parameters in the access log record for the request, as part of the `Request-URI` field of the log record.

For example, a `GET` request for `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-user=johndoe"` works the same as the request for `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg"`, except that the `"x-user=johndoe"` string is included in the `Request-URI` field for the associated log record. This functionality is available in the REST interface only.

### Programming considerations for extensible server access log format

Occasionally, we might extend the access log record format by adding new fields to the end of each line. Therefore, make sure that any of your code that parses server access logs can handle trailing fields that it might not understand.

## Deleting Amazon S3 log files

An Amazon S3 bucket with server access logging enabled can accumulate many server log objects over time. Your application might need these access logs for a specific period after they are created, and after that, you might want to delete them. You can use Amazon S3 Lifecycle configuration to set rules so that Amazon S3 automatically queues these objects for deletion at the end of their life.

You can define a lifecycle configuration for a subset of objects in your S3 bucket by using a shared prefix. If you specified a prefix in your server access logging configuration, you can set a lifecycle configuration rule to delete log objects that have that prefix.

For example, suppose that your log objects have the prefix `logs/`. You can set a lifecycle configuration rule to delete all objects in the bucket that have the prefix `logs/` after a specified period of time.

For more information about lifecycle configuration, see Managing your storage lifecycle.

For general information about server access logging, see Logging requests with server access logging.

## Using Amazon S3 server access logs to identify requests

You can identify Amazon S3 requests by using Amazon S3 server access logs.

> **Note**
>
> - To identify Amazon S3 requests, we recommend that you use AWS CloudTrail data events instead of Amazon S3 server access logs. CloudTrail data events are easier to set up and contain more information. For more information, see Identifying Amazon S3 requests using CloudTrail.
> - Depending on how many access requests you get, analyzing your logs might require more resources or time than using CloudTrail data events.

**Topics**

- Querying access logs for requests by using Amazon Athena
- Identifying Signature Version 2 requests by using Amazon S3 access logs
- Identifying object access requests by using Amazon S3 access logs

### Querying access logs for requests by using Amazon Athena

You can identify Amazon S3 requests with Amazon S3 access logs by using Amazon Athena.

Amazon S3 stores server access logs as objects in an S3 bucket. It is often easier to use a tool that can analyze the logs in Amazon S3. Athena supports analysis of S3 objects and can be used to query Amazon S3 access logs.

**Example**

The following example shows how you can query Amazon S3 server access logs in Amazon Athena. Replace the *`user input placeholders`* used in the following examples with your own information.

> **Note**
>
> To specify an Amazon S3 location in an Athena query, you must provide an S3 URI for the bucket where your logs are delivered to. This URI must include the bucket name and prefix in the following format: `s3://amzn-s3-demo-bucket1-logs/prefix/`

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. In the Query Editor, run a command similar to the following. Replace *s3_access_logs_db* with the name that you want to give to your database.

```
CREATE DATABASE s3_access_logs_db
```

> **Note**
>
> It's a best practice to create the database in the same AWS Region as your S3 bucket.

3. In the Query Editor, run a command similar to the following to create a table schema in the database that you created in step 2. Replace *s3_access_logs_db.mybucket_logs* with the name that you want to give to your table. The `STRING` and `BIGINT` data type values are the access log properties. You can query these properties in Athena. For `LOCATION`, enter the S3 bucket and prefix path as noted earlier.

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` STRING,
  `httpstatus` STRING,
  `errorcode` STRING,
  `bytessent` BIGINT,
  `objectsize` BIGINT,
  `totaltime` STRING,
  `turnaroundtime` STRING,
```

```
  `referrer` STRING,
  `useragent` STRING,
  `versionid` STRING,
  `hostid` STRING,
  `sigv` STRING,
  `ciphersuite` STRING,
  `authtype` STRING,
  `endpoint` STRING,
  `tlsversion` STRING,
  `accesspointarn` STRING,
  `aclrequired` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://amzn-s3-demo-bucket1-logs/prefix/'
```

4. In the navigation pane, under **Database**, choose your database.
5. Under **Tables**, choose **Preview table** next to your table name.

   In the **Results** pane, you should see data from the server access logs, such as `bucketowner`, `bucket`, `requestdatetime`, and so on. This means that you successfully created the Athena table. You can now query the Amazon S3 server access logs.

**Example — Show who deleted an object and when (timestamp, IP address, and IAM user)**

```
SELECT requestdatetime, remoteip, requester, key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

### Example — Show all operations that were performed by an IAM user

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

### Example — Show all operations that were performed on an object in a specific time period

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE Key='prefix/images/picture.jpg'
    AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
    BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
    AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

### Example — Show how much data was transferred to a specific IP address in a specific time period

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

**Note**

To reduce the time that you retain your logs, you can create an S3 Lifecycle configuration for your server access logs bucket. Create lifecycle configuration rules to remove log files periodically. Doing so reduces the amount of data that Athena analyzes for each query. For more information, see Setting a lifecycle configuration on a bucket.

## Identifying Signature Version 2 requests by using Amazon S3 access logs

Amazon S3 support for Signature Version 2 will be turned off (deprecated). After that, Amazon S3 will no longer accept requests that use Signature Version 2, and all requests must use Signature Version 4 signing. You can identify Signature Version 2 access requests by using Amazon S3 access logs.

> **Note**
>
> To identify Signature Version 2 requests, we recommend that you use AWS CloudTrail data events instead of Amazon S3 server access logs. CloudTrail data events are easier to set up and contain more information than server access logs. For more information, see Identifying Amazon S3 Signature Version 2 requests by using CloudTrail.

**Example — Show all requesters that are sending Signature Version 2 traffic**

```
SELECT requester, sigv, Count(sigv) as sigcount
FROM s3_access_logs_db.mybucket_logs
GROUP BY requester, sigv;
```

## Identifying object access requests by using Amazon S3 access logs

You can use queries on Amazon S3 server access logs to identify Amazon S3 object access requests, for operations such as GET, PUT, and DELETE, and discover further information about those requests.

The following Amazon Athena query example shows how to get all PUT object requests for Amazon S3 from a server access log.

**Example — Show all requesters that are sending PUT object requests in a certain period**

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.PUT.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
```

```
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

The following Amazon Athena query example shows how to get all GET object requests for Amazon S3 from the server access log.

**Example — Show all requesters that are sending GET object requests in a certain period**

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.GET.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

The following Amazon Athena query example shows how to get all anonymous requests to your S3 buckets from the server access log.

**Example — Show all anonymous requesters that are making requests to a bucket during a certain period**

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db.mybucket_logs
WHERE requester IS NULL AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

The following Amazon Athena query shows how to identify all requests to your S3 buckets that required an access control list (ACL) for authorization. You can use this information to migrate those ACL permissions to the appropriate bucket policies and disable ACLs. After you've created these bucket policies, you can disable ACLs for these buckets. For more information about disabling ACLs, see Prerequisites for disabling ACLs.

**Example — Identify all requests that required an ACL for authorization**

```
SELECT bucket_name, requester, key, operation, aclrequired, requestdatetime
```

```
FROM s3_access_logs_db
WHERE aclrequired = 'Yes' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-05-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2022-08-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
```

**Note**

- You can modify the date range as needed to suit your needs.
- These query examples might also be useful for security monitoring. You can review the results for `PutObject` or `GetObject` calls from unexpected or unauthorized IP addresses or requesters and for identifying any anonymous requests to your buckets.
- This query only retrieves information from the time at which logging was enabled.
- If you are using AWS CloudTrail logs, see Identifying access to S3 objects by using CloudTrail.

# Monitoring metrics with Amazon CloudWatch

Amazon CloudWatch metrics for Amazon S3 can help you understand and improve the performance of applications that use Amazon S3. There are several ways that you can use CloudWatch with Amazon S3.

**Daily storage metrics for buckets**

Monitor bucket storage using CloudWatch, which collects and processes storage data from Amazon S3 into readable, daily metrics. These storage metrics for Amazon S3 are reported once per day and are provided to all customers at no additional cost.

**Request metrics**

Monitor Amazon S3 requests to quickly identify and act on operational issues. The metrics are available at 1-minute intervals after some latency for processing. These CloudWatch metrics are billed at the same rate as the Amazon CloudWatch custom metrics. For information about CloudWatch pricing, see Amazon CloudWatch pricing. To learn how to opt in to getting these metrics, see CloudWatch metrics configurations.

When enabled, request metrics are reported for all object operations. By default, these 1-minute metrics are available at the Amazon S3 bucket level. You can also define a filter for the metrics using a shared prefix, object tag, or access point:

- **Access point** – Access points are named network endpoints that are attached to buckets and simplify managing data access at scale for shared datasets in S3. With the access point filter, you can gain insights into your access point usage. For more information about access points, see Monitoring and logging access points.
- **Prefix** – Although the Amazon S3 data model is a flat structure, you can use prefixes to infer a hierarchy. A prefix is similar to a directory name that enables you to group similar objects together in a bucket. The S3 console supports prefixes with the concept of folders. If you filter by prefix, objects that have the same prefix are included in the metrics configuration. For more information about prefixes, see Organizing objects using prefixes.
- **Tags** – Tags are key-value name pairs that you can add to objects. Tags help you find and organize objects easily. You can also use tags as a filter for metrics configurations so that only objects with those tags are included in the metrics configuration. For more information about object tags, see Categorizing your storage using tags.

To align these metrics to specific business applications, workflows, or internal organizations, you can filter on a shared prefix, object tag, or access point.

**Replication metrics**

**Replication metrics** – Monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, the maximum replication time to the destination AWS Region, and the total number of operations that failed replication. Replication rules that have S3 Replication Time Control (S3 RTC) or S3 Replication metrics enabled will publish replication metrics.

For more information, see Monitoring progress with replication metrics and S3 Event Notifications or Meeting compliance requirements using S3 Replication Time Control (S3 RTC).

**Amazon S3 Storage Lens metrics**

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. S3 Storage Lens metrics are available in the `AWS/S3/Storage-Lens` namespace. The CloudWatch publishing option is available for S3 Storage Lens dashboards upgraded to *advanced metrics and recommendations*. You can enable the CloudWatch publishing option for a new or existing dashboard configuration in S3 Storage Lens.

For more information, see Monitor S3 Storage Lens metrics in CloudWatch.

All CloudWatch statistics are retained for a period of 15 months so that you can access historical information and gain a better perspective on how your web application or service is performing. For more information about CloudWatch, see What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*. You may need some additional configurations to your CloudWatch alarms, depending on your use cases. For example, you can use metric math expression to create an alarm. For more information, see Use CloudWatch metrics, Use metric math, Using Amazon CloudWatch alarms, and Create a CloudWatch alarm based on a metric math expression in the *Amazon CloudWatch User Guide*.

**Best-effort CloudWatch metrics delivery**

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

For more information, see the following topics.

**Topics**

- Metrics and dimensions
- Accessing CloudWatch metrics
- CloudWatch metrics configurations

## Metrics and dimensions

The storage metrics and dimensions that Amazon S3 sends to Amazon CloudWatch are listed in the following tables.

### Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

### Topics



## Amazon S3 daily storage metrics for buckets in CloudWatch

The `AWS/S3` namespace includes the following daily storage metrics for buckets.

| Metric | Description |
| --- | --- |
| `BucketSizeBytes` | The amount of data in bytes that is stored in a bucket in the following storage classes: |

| Metric | Description |
| --- | --- |
| | • S3 Standard (STANDARD)<br>• S3 Intelligent-Tiering (INTELLIGENT_TIERING )<br>• S3 Standard-Infrequent Access (STANDARD_IA )<br>• S3 One Zone-Infrequent Access (ONEZONE_IA )<br>• Reduced Redundancy Storage (RRS) (REDUCED_REDUNDANCY )<br>• S3 Glacier Instant Retrieval (GLACIER_IR )<br>• S3 Glacier Deep Archive (DEEP_ARCHIVE )<br>• S3 Glacier Flexible Retrieval (GLACIER)<br>• S3 Express One Zone (EXPRESS_ONEZONE )<br><br>This value is calculated by summing the size of all objects and metadata (such as bucket names) in the bucket (both current and noncurrent objects), including the size of all parts for all incomplete multipart uploads to the bucket.<br><br>**Note**<br>The S3 Express One Zone storage class is available only for directory buckets.<br><br>Valid storage-type filters (see the StorageType dimension):<br><br>• S3 Standard: StandardStorage<br>• S3 Intelligent-Tiering: IntelligentTieringFAStorage , IntelligentTieringIAStorage , IntelligentTieringAAStorage , IntelligentTieringAIAStorage , IntelligentTieringDAAStorage<br>• S3 Standard-Infrequent Access: StandardIAStorage , StandardIASizeOverhead , StandardIAObjectOverhead<br>• S3 One Zone-Infrequent Access: OneZoneIAStorage , OneZoneIASizeOverhead |

| Metric | Description |
| --- | --- |
| | • Reduced Redundancy Storage (RRS): `ReducedRedundancyStorage`<br>• S3 Glacier Instant Retrieval: `GlacierInstantRetrievalSizeOverhead` , `GlacierInstantRetrievalStorage`<br>• S3 Glacier Flexible Retrieval: `GlacierStorage` , `GlacierStagingStorage` , `GlacierObjectOverhead` , `GlacierS3ObjectOverhead`<br>• S3 Glacier Deep Archive: `DeepArchiveStorage` , `DeepArchiveObjectOverhead` , `DeepArchiveS3ObjectOverhead` , `DeepArchiveStagingStorage`<br>• S3 Express One Zone: `ExpressOneZoneStorage`<br><br>Units: Bytes<br><br>Valid statistics: Average<br><br>For more information about the `StorageType` dimensions, see the section called “Amazon S3 dimensions in CloudWatch”. |
| `NumberOfObjects` | The total number of objects stored in a general purpose bucket for all storage classes. This value is calculated by counting all objects in the bucket, which includes current and noncurrent objects, delete markers, and the total number of parts for all incomplete multipart uploads to the bucket. For directory buckets with objects in the S3 Express One Zone storage class, this value is calculated by counting all objects in the bucket, but it doesn't include incomplete multiple uploads to the bucket.<br><br>Valid storage type filters: `AllStorageTypes` (see the `StorageType` dimension)<br><br>Units: Count<br><br>Valid statistics: Average |

## Amazon S3 request metrics in CloudWatch

The AWS/S3 namespace includes the following request metrics. These metrics include non-billable requests (in the case of GET requests from CopyObject and Replication).

> **Note**
>
> Amazon S3 request metrics in CloudWatch aren't supported for directory buckets.

| Metric | Description |
| --- | --- |
| AllRequests | The total number of HTTP requests made to an Amazon S3 bucket, regardless of type. If you're using a metrics configuration with a filter, then this metric returns only the HTTP requests that meet the filter's requirements.<br><br>Units: Count<br><br>Valid statistics: Sum |
| GetRequests | The number of HTTP GET requests made for objects in an Amazon S3 bucket. This doesn't include list operations. This metric is incremented for the source of each CopyObject request.<br><br>Units: Count<br><br>Valid statistics: Sum<br><br>**Note**<br>Paginated list-oriented requests, such as ListMultipartUploads, ListParts, ListObjectVersions, and others, are not included in this metric. |
| PutRequests | The number of HTTP PUT requests made for objects in an Amazon S3 bucket. This metric is incremented for the destination of each CopyObject request. |

| Metric | Description |
| --- | --- |
| | Units: Count<br><br>Valid statistics: Sum |
| `DeleteRequests` | The number of HTTP `DELETE` requests made for objects in an Amazon S3 bucket. This metric also includes DeleteObjects requests. This metric shows the number of requests made, not the number of objects deleted.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `HeadRequests` | The number of HTTP `HEAD` requests made to an Amazon S3 bucket.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `PostRequests` | The number of HTTP `POST` requests made to an Amazon S3 bucket.<br><br>Units: Count<br><br>Valid statistics: Sum<br><br>**Note**<br>DeleteObjects and SelectObjectContent requests are not included in this metric. |
| `SelectRequests` | The number of Amazon S3 SelectObjectContent requests made for objects in an Amazon S3 bucket.<br><br>Units: Count<br><br>Valid statistics: Sum |

| Metric | Description |
| --- | --- |
| `SelectBytesScanned` | The number of bytes of data scanned with Amazon S3 SelectObjectContent requests in an Amazon S3 bucket.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |
| `SelectBytesReturned` | The number of bytes of data returned with Amazon S3 SelectObjectContent requests in an Amazon S3 bucket.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |
| `ListRequests` | The number of HTTP requests that list the contents of a bucket.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `BytesDownloaded` | The number of bytes downloaded for requests made to an Amazon S3 bucket, where the response includes a body.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |

| Metric | Description |
| --- | --- |
| `BytesUploaded` | The number of bytes uploaded for requests made to an Amazon S3 bucket, where the request includes a body.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |
| `4xxErrors` | The number of HTTP 4*xx* client error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period.<br><br>Units: Count<br><br>Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count |
| `5xxErrors` | The number of HTTP 5*xx* server error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period.<br><br>Units: Count<br><br>Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count |
| `FirstByte Latency` | The per-request time from the complete request being received by an Amazon S3 bucket to when the response starts to be returned.<br><br>Units: Milliseconds<br><br>Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100 |

| Metric | Description |
| --- | --- |
| `TotalRequestLatency` | The elapsed per-request time from the first byte received to the last byte sent to an Amazon S3 bucket. This metric includes the time taken to receive the request body and send the response body, which is not included in `FirstByteLatency` .<br><br>Units: Milliseconds<br><br>Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100 |

## S3 Replication metrics in CloudWatch

You can monitor the progress of replication with S3 Replication metrics by tracking bytes pending, operations pending, and replication latency. For more information, see Monitoring progress with replication metrics.

> **Note**
>
> You can enable alarms for your replication metrics in Amazon CloudWatch. When you set up alarms for your replication metrics, set the **Missing data treatment** field to **Treat missing data as ignore (maintain the alarm state)**.

| Metric | Description |
| --- | --- |
| `ReplicationLatency` | The maximum number of seconds by which the replication destination AWS Region is behind the source AWS Region for a given replication rule.<br><br>Units: Seconds<br><br>Valid statistics: Max |
| `BytesPendingReplication` | The total number of bytes of objects pending replication for a given replication rule. |

| Metric | Description |
| --- | --- |
| | Units: Bytes<br><br>Valid statistics: Max |
| `OperationsPendingReplication` | The number of operations pending replication for a given replication rule.<br><br>Units: Count<br><br>Valid statistics: Max |
| `OperationsFailedReplication` | The number of operations that failed to replicate for a given replication rule.<br><br>Units: Count<br><br>Valid statistics: Sum (total number of failed operations), Average (failure rate), Sample Count (total number of replication operations) |

## S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. S3 Storage Lens metrics are published to the `AWS/S3/Storage-Lens` namespace in CloudWatch. The CloudWatch publishing option is available for S3 Storage Lens dashboards that have been upgraded to advanced metrics and recommendations.

For a list of S3 Storage Lens metrics that are published to CloudWatch, see Amazon S3 Storage Lens metrics glossary. For a complete list of dimensions, see Dimensions.

## S3 Object Lambda request metrics in CloudWatch

S3 Object Lambda includes the following request metrics.

| Metric | Description |
| --- | --- |
| `AllRequests` | The total number of HTTP requests made to an Amazon S3 bucket by using an Object Lambda Access Point. |

| Metric | Description |
| --- | --- |
| | Units: Count<br><br>Valid statistics: Sum |
| GetRequests | The number of HTTP GET requests made for objects by using an Object Lambda Access Point. This metric does not include list operations.<br><br>Units: Count<br><br>Valid statistics: Sum |
| BytesUploaded | The number of bytes uploaded to an Amazon S3 bucket by using an Object Lambda Access Point, where the request includes a body.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |
| PostRequests | The number of HTTP POST requests made to an Amazon S3 bucket by using an Object Lambda Access Point.<br><br>Units: Count<br><br>Valid statistics: Sum |
| PutRequests | The number of HTTP PUT requests made for objects in an Amazon S3 bucket by using an Object Lambda Access Point.<br><br>Units: Count<br><br>Valid statistics: Sum |

| Metric | Description |
| --- | --- |
| `DeleteRequests` | The number of HTTP `DELETE` requests made for objects in an Amazon S3 bucket by using an Object Lambda Access Point. This metric includes DeleteObjects requests. This metric shows the number of requests made, not the number of objects deleted.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `BytesDownloaded` | The number of bytes downloaded for requests made to an Amazon S3 bucket by using an Object Lambda Access Point, where the response includes a body.<br><br>Units: Bytes<br><br>Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9 |
| `FirstByte Latency` | The per-request time from the complete request being received by an Amazon S3 bucket through an Object Lambda Access Point to when the response starts to be returned. This metric is dependent on the AWS Lambda function's running time to transform the object before the function returns the bytes to the Object Lambda Access Point.<br><br>Units: Milliseconds<br><br>Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100 |

| Metric | Description |
| --- | --- |
| `TotalRequestLatency` | The elapsed per-request time from the first byte received to the last byte sent to an Object Lambda Access Point. This metric includes the time taken to receive the request body and send the response body, which is not included in `FirstByteLatency` .<br><br>Units: Milliseconds<br><br>Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100 |
| `HeadRequests` | The number of HTTP `HEAD` requests made to an Amazon S3 bucket by using an Object Lambda Access Point.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `ListRequests` | The number of HTTP `GET` requests that list the contents of an Amazon S3 bucket. This metric includes both `ListObjects` and `ListObjectsV2` operations.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `4xxErrors` | The number of HTTP 4*xx* client error status code requests made to an Amazon S3 bucket by using an Object Lambda Access Point with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period.<br><br>Units: Count<br><br>Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count |

| Metric | Description |
| --- | --- |
| `5xxErrors` | The number of HTTP 5*xx* server error status code requests made to an Amazon S3 bucket by using an Object Lambda Access Point with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period.<br><br>Units: Count<br><br>Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count |
| `ProxiedRequests` | The number of HTTP requests to an Object Lambda Access Point that return the standard Amazon S3 API response. (Such requests do not have a Lambda function configured.)<br><br>Units: Count<br><br>Valid statistics: Sum |
| `InvokedLambda` | The number of HTTP requests to an S3 object where a Lambda function was invoked.<br><br>Units: Count<br><br>Valid statistics: Sum |
| `LambdaResponseRequests` | The number of `WriteGetObjectResponse` requests made by the Lambda function. This metric applies only to `GetObject` requests. |
| `LambdaResponse4xx` | The number of HTTP 4*xx* client errors that occur when calling `WriteGetObjectResponse` from a Lambda function. This metric provides the same information as `4xxErrors`, but only for `WriteGetObjectResponse` calls. |
| `LambdaResponse5xx` | The number of HTTP 5*xx* server errors that occur when calling `WriteGetObjectResponse` from a Lambda function. This metric provides the same information as `5xxErrors`, but only for `WriteGetObjectResponse` calls. |

## Amazon S3 on Outposts metrics in CloudWatch

For a list of metrics in CloudWatch that are used for S3 on Outposts buckets, see CloudWatch metrics.

## Amazon S3 dimensions in CloudWatch

The following dimensions are used to filter Amazon S3 metrics.

| Dimension | Description |
| --- | --- |
| `BucketName` | This dimension filters the data that you request for the identified bucket only. |
| `StorageType` | This dimension filters the data that you have stored in a bucket by the following types of storage:<br>• `StandardStorage` – The number of bytes used for objects in the `STANDARD` storage class.<br>• `IntelligentTieringAAStorage` – The number of bytes used for objects in the Archive Access tier of the `INTELLIGENT_TIERING` storage class.<br>• `IntelligentTieringAIAStorage` – The number of bytes used for objects in the Archive Instant Access tier of the `INTELLIGENT_TIERING` storage class.<br>• `IntelligentTieringDAAStorage` – The number of bytes used for objects in the Deep Archive Access tier of the `INTELLIGENT_TIERING` storage class.<br>• `IntelligentTieringFAStorage` – The number of bytes used for objects in the Frequent Access tier of the `INTELLIGENT_TIERING` storage class.<br>• `IntelligentTieringIAStorage` – The number of bytes used for objects in the Infrequent Access tier of the `INTELLIGENT_TIERING` storage class.<br>• `StandardIAStorage` – The number of bytes used for objects in the S3 Standard-Infrequent Access (`STANDARD_IA`) storage class. |

| Dimension | Description |
| --- | --- |
| | • StandardIASizeOverhead – The number of bytes used for objects smaller than 128 KB in the STANDARD_IA storage class.<br>• IntAAObjectOverhead – For each object in the INTELLIGENT_TIERING storage class in the Archive Access tier, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval rates for this additional storage.<br>• IntAAS3ObjectOverhead – For each object in the INTELLIGENT_TIERING storage class in the Archive Access tier, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage.<br>• IntDAAObjectOverhead – For each object in the INTELLIGENT_TIERING storage class in the Deep Archive Access tier, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Deep Archive storage rates for this additional storage.<br>• IntDAAS3ObjectOverhead – For each object in the INTELLIGENT_TIERING storage class in the Deep Archive Access tier, Amazon S3 adds 8 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Standard rates for this additional storage.<br>• OneZoneIAStorage – The number of bytes used for objects in the S3 One Zone-Infrequent Access (ONEZONE_IA ) storage class.<br>• OneZoneIASizeOverhead – The number of bytes used for objects smaller than 128 KB in the ONEZONE_IA storage class. |

| Dimension | Description |
| --- | --- |
| | • `ReducedRedundancyStorage` – The number of bytes used for objects in the Reduced Redundancy Storage (RRS) class.<br>• `GlacierInstantRetrievalSizeOverhead` – The number of bytes used for objects smaller than 128 KB in the S3 Glacier Instant Retrieval storage class.<br>• `GlacierInstantRetrievalStorage` – The number of bytes used for objects in the S3 Glacier Instant Retrieval storage class.<br>• `GlacierStorage` – The number of bytes used for objects in the S3 Glacier Flexible Retrieval storage class.<br>• `GlacierStagingStorage` – The number of bytes used for parts of multipart objects before the `CompleteMultipartUpload` request is completed on objects in the S3 Glacier Flexible Retrieval storage class.<br>• `GlacierObjectOverhead` – For each archived object, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval rates for this additional storage.<br>• `GlacierS3ObjectOverhead` – For each object archived to S3 Glacier Flexible Retrieval, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage.<br>• `DeepArchiveStorage` – The number of bytes used for objects in the S3 Glacier Deep Archive storage class.<br>• `DeepArchiveObjectOverhead` – For each archived object, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Deep Archive rates for this additional storage.<br>• `DeepArchiveS3ObjectOverhead` – For each object archived to S3 Glacier Deep Archive, Amazon S3 uses 8 KB of |

| Dimension | Description |
| --- | --- |
| | storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage.<br>• `DeepArchiveStagingStorage` – The number of bytes used for parts of multipart objects before the `CompleteMultipartUpload` request is completed on objects in the S3 Glacier Deep Archive storage class.<br>• `ExpressOneZoneStorage` – The number of bytes used for objects in the S3 Express One Zone storage class. |
| `FilterId` | This dimension filters metrics configurations that you specify for the request metrics on a bucket. When you create a metrics configuration, you specify a filter ID (for example, a prefix, a tag, or an access point). For more information, see Creating a metrics configuration. |

## S3 Replication dimensions in CloudWatch

The following dimensions are used to filter S3 Replication metrics.

| Dimension | Description |
| --- | --- |
| `SourceBucket` | The name of the bucket objects are replicated from. |
| `DestinationBucket` | The name of the bucket objects are replicated to. |
| `RuleId` | A unique identifier for the rule that triggered this replication metric to update. |

## S3 Storage Lens dimensions in CloudWatch

For a list of dimensions that are used to filter S3 Storage Lens metrics in CloudWatch, see Dimensions.

### S3 Object Lambda request dimensions in CloudWatch

The following dimensions are used to filter data from an Object Lambda Access Point.

| Dimension | Description |
| --- | --- |
| `AccessPointName` | The name of the access point of which requests are being made. |
| `DataSourceARN` | The source the Object Lambda Access Point is retrieving the data from. If the request invokes a Lambda function this refers to the Lambda Amazon Resource Name (ARN). Otherwise this refers to the access point ARN. |

## Accessing CloudWatch metrics

You can use the following procedures to view the storage metrics for Amazon S3. To get the Amazon S3 metrics involved, you must set a start and end timestamp. For metrics for any given 24-hour period, set the time period to 86400 seconds, the number of seconds in a day. Also, remember to set the `BucketName` and `StorageType` dimensions.

#### Using the AWS CLI

For example, if you want to use the AWS CLI to get the average of a specific bucket's size in bytes, you could use the following command:

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=amzn-s3-demo-bucket
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

This example produces the following output.

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
```

```
    ],
    "Label": "BucketSizeBytes"
}
```

### Using the S3 console

#### To view metrics by using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Metrics**.
3. Choose the **S3** namespace.
4. (Optional) To view a metric, enter the metric name in the search box.
5. (Optional) To filter by the **StorageType** dimension, enter the name of the storage class in the search box.

#### To view a list of valid metrics stored for your AWS account by using the AWS CLI

- At a command prompt, use the following command.

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

For more information about the permissions required to access CloudWatch dashboards, see Amazon CloudWatch dashboard permissions in the *Amazon CloudWatch User Guide*.

## CloudWatch metrics configurations

With Amazon CloudWatch request metrics for Amazon S3, you can receive 1-minute CloudWatch metrics, set CloudWatch alarms, and access CloudWatch dashboards to view near-real-time operations and performance of your Amazon S3 storage. For applications that depend on cloud storage, these metrics let you quickly identify and act on operational issues. When enabled, these 1-minute metrics are available at the Amazon S3 bucket-level, by default.

If you want to get the CloudWatch request metrics for the objects in a bucket, you must create a metrics configuration for the bucket. For more information, see Creating a CloudWatch metrics configuration for all the objects in your bucket.

You can also use a shared prefix, object tags, or an access point to define a filter for the metrics collected. This method of defining a filter allows you to align metrics filters to specific business

applications, workflows, or internal organizations. For more information, see Creating a metrics configuration that filters by prefix, object tag, or access point. For more information about the CloudWatch metrics that are available and the differences between storage and request metrics, see Monitoring metrics with Amazon CloudWatch.

Keep the following in mind when using metrics configurations:

- You can have a maximum of 1,000 metrics configurations per bucket.
- You can choose which objects in a bucket to include in metrics configurations by using filters. You can filter on a shared prefix, object tag, or access point to align metrics filters to specific business applications, workflows, or internal organizations. To request metrics for the entire bucket, create a metrics configuration without a filter.
- Metrics configurations are necessary only to enable request metrics. Bucket-level daily storage metrics are always turned on, and are provided at no additional cost. Currently, it's not possible to get daily storage metrics for a filtered subset of objects.
- Each metrics configuration enables the full set of available request metrics. Operation-specific metrics (such as `PostRequests`) are reported only if there are requests of that type for your bucket or your filter.
- Request metrics are reported for object-level operations. They are also reported for operations that list bucket contents, like GET Bucket (List Objects), GET Bucket Object Versions, and List Multipart Uploads, but they are not reported for other operations on buckets.
- Request metrics support filtering by prefix, object tags, or access point, but storage metrics do not.

**Best-effort CloudWatch metrics delivery**

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

For more information about working with CloudWatch metrics in Amazon S3, see the following topics.

**Topics**

- Creating a CloudWatch metrics configuration for all the objects in your bucket
- Creating a metrics configuration that filters by prefix, object tag, or access point
- Deleting a metrics filter

## Creating a CloudWatch metrics configuration for all the objects in your bucket

When you configure request metrics, you can create a CloudWatch metrics configuration for all the objects in your bucket, or you can filter by prefix, object tag, or access point. The procedures in this topic show you how to create a configuration for all the objects in your bucket. To create a configuration that filters by object tag, prefix, or access point, see Creating a metrics configuration that filters by prefix, object tag, or access point.

There are three types of Amazon CloudWatch metrics for Amazon S3: storage metrics, request metrics, and replication metrics. Storage metrics are reported once per day and are provided to all customers at no additional cost. Request metrics are available at one-minute intervals after some latency for processing. Request metrics are billed at the standard CloudWatch rate. You must opt in to request metrics by configuring them in the console or using the Amazon S3 API. S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

You can add metrics configurations to a bucket using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 REST API.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the **Buckets** list, choose the name of the bucket that contains the objects you want request metrics for.
3. Choose the **Metrics** tab.
4. Under **Bucket metrics**, choose **View additional charts**.
5. Choose the **Request metrics** tab.
6. Choose **Create filter**.
7. In the **Filter name** box, enter your filter name.

   Names can only contain letters, numbers, periods, dashes, and underscores. We recommend using the name `EntireBucket` for a filter that applies to all objects.
8. Under **Filter scope**, choose **This filter applies to all objects in the bucket**.

   You can also define a filter so that the metrics are only collected and reported on a subset of objects in the bucket. For more information, see Creating a metrics configuration that filters by prefix, object tag, or access point.
9. Choose **Save changes**.
10. On the **Request metrics** tab, under **Filters**, choose the filter that you just created.

    After about 15 minutes, CloudWatch begins tracking these request metrics. You can see them on the **Request metrics** tab. You can see graphs for the metrics on the Amazon S3 or CloudWatch console. Request metrics are billed at the standard CloudWatch rate. For more information, see Amazon CloudWatch pricing.

### Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

**Using the AWS CLI**

1. Install and set up the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
2. Open a terminal.
3. Run the following command to add a metrics configuration.

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id"}'
```

## Creating a metrics configuration that filters by prefix, object tag, or access point

There are three types of Amazon CloudWatch metrics for Amazon S3: storage metrics, request metrics, and replication metrics. Storage metrics are reported once per day and are provided to all customers at no additional cost. Request metrics are available at one-minute intervals after some latency for processing. Request metrics are billed at the standard CloudWatch rate. You must opt in to request metrics by configuring them in the console or using the Amazon S3 API. S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

When you configure CloudWatch metrics, you can create a filter for all the objects in your bucket, or you can filter the configuration into groups of related objects within a single bucket. You can filter objects in a bucket for inclusion in a metrics configuration based on one or more of the following filter types:

- **Object key name prefix** – Although the Amazon S3 data model is a flat structure, you can infer a hierarchy by using a prefix. The Amazon S3 console supports these prefixes with the concept of folders. If you filter by prefix, objects that have the same prefix are included in the metrics configuration. For more information about prefixes, see Organizing objects using prefixes.
- **Tag** – You can add tags, which are key-value name pairs, to objects. Tags help you find and organize objects easily. You can also use tags as filters for metrics configurations. For more information about object tags, see Categorizing your storage using tags.

- **Access point** – S3 Access Points are named network endpoints that are attached to buckets and simplify managing data access at scale for shared datasets in S3. When you create an access point filter, Amazon S3 includes requests to the access point that you specify in the metrics configuration. For more information, see Monitoring and logging access points.

  > **Note**
  >
  > When you create a metrics configuration that filters by access point, you must use the access point Amazon Resource Name (ARN), not the access point alias. Make sure that you use the ARN for the access point itself, not the ARN for a specific object. For more information about access point ARNs, see Using access points.

If you specify a filter, only requests that operate on single objects can match the filter and be included in the reported metrics. Requests like DeleteObjectsand `ListObjects` requests don't return any metrics for configurations with filters.

To request more complex filtering, choose two or more elements. Only objects that have all of those elements are included in the metrics configuration. If you don't set filters, all of the objects in the bucket are included in the metrics configuration.

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the objects that you want request metrics for.
3. Choose the **Metrics** tab.
4. Under **Bucket metrics**, choose **View additional charts**.
5. Choose the **Request metrics** tab.
6. Choose **Create filter**.
7. In the **Filter name** box, enter your filter name.

   Names can contain only letters, numbers, periods, dashes, and underscores.
8. Under **Filter scope**, choose **Limit the scope of this filter using a prefix, object tags, and an S3 Access Point, or a combination of all three**.
9. Under **Filter type**, choose at least one filter type: **Prefix**, **Object tags**, or **Access Point**.

10. To define a prefix filter and limit the scope of the filter to a single path, in the **Prefix** box, enter a prefix.
11. To define an object tags filter, under **Object tags**, choose **Add tag**, and then enter a tag **Key** and **Value**.
12. To define an access point filter, in the **S3 Access Point** field, enter the access point ARN, or choose **Browse S3** to navigate to the access point.

    > **Important**
    >
    > You cannot enter an access point alias. You must enter the ARN for the access point itself, not the ARN for a specific object.

13. Choose **Save changes**.

    Amazon S3 creates a filter that uses the prefix, tags, or access point that you specified.
14. On the **Request metrics** tab, under **Filters**, choose the filter that you just created.

    You have now created a filter that limits the request metrics scope by prefix, object tags, or access point. About 15 minutes after CloudWatch begins tracking these request metrics, you can see charts for the metrics on both the Amazon S3 and CloudWatch consoles. Request metrics are billed at the standard CloudWatch rate. For more information, see Amazon CloudWatch pricing.

    You can also configure request metrics at the bucket level. For information, see Creating a CloudWatch metrics configuration for all the objects in your bucket.

### Using the AWS CLI

1. Install and set up the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
2. Open a terminal.
3. To add a metrics configuration, run one of the following commands:

### Example : To filter by prefix

```
aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Prefix":"prefix1"}} '
```

### Example : To filter by tags

```
aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Tag": {"Key": "string", "Value": "string"}} '
```

### Example : To filter by access point

```
aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
```

### Example : To filter by prefix, tags, and access point

```
aws s3api put-bucket-metrics-configuration --endpoint https://
s3.Region.amazonaws.com --bucket DOC-EXAMPLE-BUCKET1 --id metrics-config-id --
metrics-configuration '
{
    "Id": "metrics-config-id",
    "Filter": {
        "And": {
            "Prefix": "string",
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

### Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

## Deleting a metrics filter

You can delete an Amazon CloudWatch request metrics filter if you no longer need it. When you delete a filter, you are no longer charged for request metrics that use that *specific filter*. However, you will continue to be charged for any other filter configurations that exist.

When you delete a filter, you can no longer use the filter for request metrics. Deleting a filter cannot be undone.

For information about creating a request metrics filter, see the following topics:

- Creating a CloudWatch metrics configuration for all the objects in your bucket
- Creating a metrics configuration that filters by prefix, object tag, or access point

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose your bucket name.
3. Choose the **Metrics** tab.
4. Under **Bucket metrics**, choose **View additional charts**.
5. Choose the **Request metrics** tab.
6. Choose **Manage filters**.
7. Choose your filter.

> **Important**
>
> Deleting a filter cannot be undone.

8. Choose **Delete**.

   Amazon S3 deletes your filter.

#### Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

# Amazon S3 Event Notifications

You can use the Amazon S3 Event Notifications feature to receive notifications when certain events happen in your S3 bucket. To enable notifications, add a notification configuration that identifies the events that you want Amazon S3 to publish. Make sure that it also identifies the destinations where you want Amazon S3 to send the notifications. You store this configuration in the *notification* subresource that's associated with a bucket. For more information, see Bucket configuration options. Amazon S3 provides an API for you to manage this subresource.

> **Important**
>
> Amazon S3 event notifications are designed to be delivered at least once. Typically, event notifications are delivered in seconds but can sometimes take a minute or longer.

## Overview of Amazon S3 Event Notifications

Currently, Amazon S3 can publish notifications for the following events:

- New object created events
- Object removal events
- Restore object events
- Reduced Redundancy Storage (RRS) object lost events
- Replication events
- S3 Lifecycle expiration events
- S3 Lifecycle transition events
- S3 Intelligent-Tiering automatic archival events
- Object tagging events
- Object ACL PUT events

For full descriptions of all the supported event types, see Supported event types for SQS, SNS, and Lambda.

Amazon S3 can send event notification messages to the following destinations. You specify the Amazon Resource Name (ARN) value of these destinations in the notification configuration.

- Amazon Simple Notification Service (Amazon SNS) topics
- Amazon Simple Queue Service (Amazon SQS) queues
- AWS Lambda function
- Amazon EventBridge

For more information, see Supported event destinations.

**Note**

Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

**Warning**

If your notification writes to the same bucket that triggers the notification, it could cause an execution loop. For example, if the bucket triggers a Lambda function each time an object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.
For more information and an example of using Amazon S3 notifications with AWS Lambda, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

For more information about the number of event notification configurations that you can create per bucket, see Amazon S3 service quotas in *AWS General Reference*.

For more information about event notifications, see the following sections.

**Topics**

- Event notification types and destinations
- Using Amazon SQS, Amazon SNS, and Lambda
- Using EventBridge

## Event notification types and destinations

Amazon S3 supports several event notification types and destinations where the notifications can be published. You can specify the event type and destination when configuring your event notifications. Only one destination can be specified for each event notification. Amazon S3 event notifications send one event entry for each notification message.

**Topics**

- Supported event destinations
- Supported event types for SQS, SNS, and Lambda
- Supported event types for Amazon EventBridge
- Event ordering and duplicate events

## Supported event destinations

Amazon S3 can send event notification messages to the following destinations.

- Amazon Simple Notification Service (Amazon SNS) topics
- Amazon Simple Queue Service (Amazon SQS) queues
- AWS Lambda
- Amazon EventBridge

However, only one destination type can be specified for each event notification.

> **Note**
>
> You must grant Amazon S3 permissions to post messages to an Amazon SNS topic or an Amazon SQS queue. You must also grant Amazon S3 permission to invoke an AWS Lambda function on your behalf. For instructions on how to grant these permissions, see Granting permissions to publish event notification messages to a destination.

### Amazon SNS topic

Amazon SNS is a flexible, fully managed push messaging service. You can use this service to push messages to mobile devices or distributed services. With SNS, you can publish a message once, and deliver it one or more times. Currently, Standard SNS is only allowed as an S3 event notification destination, whereas SNS FIFO is not allowed.

Amazon SNS both coordinates and manages sending and delivering messages to subscribing endpoints or clients. You can use the Amazon SNS console to create an Amazon SNS topic that your notifications can be sent to.

The topic must be in the same AWS Region as your Amazon S3 bucket. For instructions on how to create an Amazon SNS topic, see Getting started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide* and the Amazon SNS FAQ.

Before you can use the Amazon SNS topic that you created as an event notification destination, you need the following:

- The Amazon Resource Name (ARN) for the Amazon SNS topic

- A valid Amazon SNS topic subscription. With it, topic subscribers are notified when a message is published to your Amazon SNS topic.

### Amazon SQS queue

Amazon SQS offers reliable and scalable hosted queues for storing messages as they travel between computers. You can use Amazon SQS to transmit any volume of data without requiring other services to be always available. You can use the Amazon SQS console to create an Amazon SQS queue that your notifications can be sent to.

The Amazon SQS queue must be in the same AWS Region as your Amazon S3 bucket. For instructions on how to create an Amazon SQS queue, see What is Amazon Simple Queue Service and Getting started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*.

Before you can use the Amazon SQS queue as an event notification destination, you need the following:

- The Amazon Resource Name (ARN) for the Amazon SQS queue

> **Note**
>
> Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

### Lambda function

You can use AWS Lambda to extend other AWS services with custom logic, or create your own backend that operates at AWS scale, performance, and security. With Lambda, you can create discrete, event-driven applications that run only when needed. You can also use it to scale these applications automatically from a few requests a day to thousands a second.

Lambda can run custom code in response to Amazon S3 bucket events. You upload your custom code to Lambda and create what's called a Lambda function. When Amazon S3 detects an event of a specific type, it can publish the event to AWS Lambda and invoke your function in Lambda. In response, Lambda runs your function. One event type it might detect, for example, is an object created event.

You can use the AWS Lambda console to create a Lambda function that uses the AWS infrastructure to run the code on your behalf. The Lambda function must be in the same Region as your S3 bucket. You must also have the name or the ARN of a Lambda function to set up the Lambda function as an event notification destination.

> **Warning**
>
> If your notification writes to the same bucket that triggers the notification, it could cause an execution loop. For example, if the bucket triggers a Lambda function each time an object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.
>
> For more information and an example of using Amazon S3 notifications with AWS Lambda, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

#### Amazon EventBridge

Amazon EventBridge is a serverless event bus, which receives events from AWS services. You can set up rules to match events and deliver them to targets, such as an AWS service or an HTTP endpoint. For more information, see What is EventBridge in the *Amazon EventBridge User Guide*.

Unlike other destinations, you can either enable or disable events to be delivered to EventBridge for a bucket. If you enable delivery, all events are sent to EventBridge. Moreover, you can use EventBridge rules to route events to additional targets.

### Supported event types for SQS, SNS, and Lambda

Amazon S3 can publish events of the following types. You specify these event types in the notification configuration.

| Event types | Description |
| --- | --- |
| *s3:TestEvent* | When a notification is enabled, Amazon S3 publishes a test notification. This is to ensure that the topic exists and that the bucket owner has permission to publish the specified topic. |

| Event types | Description |
| --- | --- |
| | If enabling the notification fails, you don't receive a test notification. |
| *s3:ObjectCreated:**<br><br>*s3:ObjectCreated:Put*<br><br>*s3:ObjectCreated:Post*<br><br>*s3:ObjectCreated:Copy*<br><br>*s3:ObjectCreated:CompleteMultipartUpload* | Amazon S3 API operations such as PUT, POST, and COPY can create an object. With these event types, you can enable notifications when an object is created using a specific API operation. Alternatively, you can use the `s3:ObjectCreated:*` event type to request notification regardless of the API that was used to create an object.<br><br>`s3:ObjectCreated:CompleteMultipartUpload` includes objects that are created using UploadPartCopy for Copy operations. |
| *s3:ObjectRemoved:**<br><br>*s3:ObjectRemoved:Delete*<br><br>*s3:ObjectRemoved:DeleteMarkerCreated* | By using the *ObjectRemoved* event types, you can enable notification when an object or a batch of objects is removed from a bucket.<br><br>You can request notification when an object is deleted or a versioned object is permanently deleted by using the `s3:ObjectRemoved:Delete` event type. Alternatively, you can request notification when a delete marker is created for a versioned object using `s3:ObjectRemoved:DeleteMarkerCreated`. For instructions on how to delete versioned objects, see Deleting object versions from a versioning-enabled bucket. You can also use a wildcard `s3:ObjectRemoved:*` to request notification anytime an object is deleted.<br><br>These event notifications don't alert you for automatic deletes from lifecycle configurations or from failed operations. |

| Event types | Description |
| --- | --- |
| *s3:ObjectRestore:**<br><br>*s3:ObjectRestore:Post*<br><br>*s3:ObjectRestore:Completed*<br><br>*s3:ObjectRestore:Delete* | By using the *ObjectRestore* event types, you can receive notifications for event initiation and completion when restoring objects from the S3 Glacier Flexible Retrieval storage class, S3 Glacier Deep Archive storage class, S3 Intelligent-Tiering Archive Access tier, and S3 Intellige nt-Tiering Deep Archive Access tier. You can also receive notifications for when the restored copy of an object expires.<br><br>The `s3:ObjectRestore:Post` event type notifies you of object restoration initiation. The `s3:Object Restore:Completed` event type notifies you of restoration completion. The `s3:ObjectRestore:D elete` event type notifies you when the temporary copy of a restored object expires. |
| *s3:ReducedRedundancyLostObject* | You receive this notification event when Amazon S3 detects that an object of the RRS storage class is lost. |

| Event types | Description |
| --- | --- |
| *s3:Replication:**<br><br>*s3:Replication:OperationFailedReplication*<br><br>*s3:Replication:OperationMissedThreshold*<br><br>*s3:Replication:OperationReplicatedAfterThreshold*<br><br>*s3:Replication:OperationNotTracked* | By using the *Replication* event types, you can receive notifications for replication configurations that have S3 Replication metrics or S3 Replication Time Control (S3 RTC) enabled. You can monitor the minute-by-minute progress of replication events by tracking bytes pending, operations pending, and replication latency. For informati on about replication metrics, see Monitoring progress with replication metrics and S3 Event Notifications.<br><br>The `s3:Replication:OperationFailedReplic ation` event type notifies you when an object that was eligible for replication failed to replicate. The `s3:Replic ation:OperationMissedThreshold` event type notifies you when an object that was eligible for replicati on exceeds the 15-minute threshold for replication.<br><br>The `s3:Replication:OperationReplicatedAf terThreshold` event type notifies you when an object that was eligible for replication that uses S3 Replication Time Control replicates after the 15-minute threshold . The `s3:Replication:OperationNotTracked` event type notifies you when an object that was eligible for replication that uses S3 Replication Time Control but is no longer tracked by replication metrics. |

| Event types | Description |
| --- | --- |
| *s3:LifecycleExpiration:**<br><br>*s3:LifecycleExpiration:Delete*<br><br>*s3:LifecycleExpiration:DeleteMarkerCreated* | By using the *LifecycleExpiration* event types, you can receive a notification when Amazon S3 deletes an object based on your S3 Lifecycle configuration.<br><br>The `s3:LifecycleExpiration:Delete` event type notifies you when an object in an unversioned bucket is deleted. It also notifies you when an object version is permanently deleted by an S3 Lifecycle configuration. The `s3:LifecycleExpiration:DeleteMarkerCreated` event type notifies you when S3 Lifecycle creates a delete marker when a current version of an object in versioned bucket is deleted. |
| *s3:LifecycleTransition* | You receive this notification event when an object is transitioned to another Amazon S3 storage class by an S3 Lifecycle configuration. |
| *s3:IntelligentTiering* | You receive this notification event when an object within the S3 Intelligent-Tiering storage class moved to the Archive Access tier or Deep Archive Access tier. |
| *s3:ObjectTagging:**<br><br>*s3:ObjectTagging:Put*<br><br>*s3:ObjectTagging:Delete* | By using the *ObjectTagging* event types, you can enable notification when an object tag is added or deleted from an object.<br><br>The `s3:ObjectTagging:Put` event type notifies you when a tag is PUT on an object or an existing tag is updated. The `s3:ObjectTagging:Delete` event type notifies you when a tag is removed from an object. |
| *s3:ObjectAcl:Put* | You receive this notification event when an ACL is PUT on an object or when an existing ACL is changed. An event is not generated when a request results in no change to an object's ACL. |

### Supported event types for Amazon EventBridge

For a list of event types Amazon S3 will send to Amazon EventBridge, see Using EventBridge.

### Event ordering and duplicate events

Amazon S3 Event Notifications is designed to deliver notifications at least once, but they aren't guaranteed to arrive in the same order that the events occurred. On rare occasions, Amazon S3's retry mechanism might cause duplicate S3 Event Notifications for the same object event. For more about handling duplicate or out of order events, see Manage event ordering and duplicate events with Amazon S3 Event Notifications on the *AWS Storage Blog*.

## Using Amazon SQS, Amazon SNS, and Lambda

Enabling notifications is a bucket-level operation. You store notification configuration information in the *notification* subresource that's associated with a bucket. After you create or change the bucket notification configuration, it usually takes about five minutes for the changes to take effect. When the notification is first enabled, an `s3:TestEvent` occurs. You can use any of the following methods to manage notification configuration:

- **Using the Amazon S3 console** — You can use the console UI to set a notification configuration on a bucket without having to write any code. For more information, see Enabling and configuring event notifications using the Amazon S3 console.
- **Programmatically using the AWS SDKs** — Internally, both the console and the SDKs call the Amazon S3 REST API to manage *notification* subresources that are associated with the bucket. For examples of notification configurations that use AWS SDK, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

> **Note**
>
> You can also make the Amazon S3 REST API calls directly from your code. However, this can be cumbersome because to do so you must write code to authenticate your requests.

Regardless of the method that you use, Amazon S3 stores the notification configuration as XML in the *notification* subresource that's associated with a bucket. For information about bucket subresources, see Bucket configuration options.

**Topics**

## Granting permissions to publish event notification messages to a destination

You must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. This is so that Amazon S3 can publish event notification messages to a destination.

To troubleshoot publishing event notification messages to a destination, see Troubleshoot to publish Amazon S3 event notifications to an Amazon Simple Notification Service topic .

**Topics**



### Granting permissions to invoke an AWS Lambda function

Amazon S3 publishes event messages to AWS Lambda by invoking a Lambda function and providing the event message as an argument.

When you use the Amazon S3 console to configure event notifications on an Amazon S3 bucket for a Lambda function, the console sets up the necessary permissions on the Lambda function. This is so that Amazon S3 has permissions to invoke the function from the bucket. For more information, see Enabling and configuring event notifications using the Amazon S3 console.

You can also grant Amazon S3 permissions from AWS Lambda to invoke your Lambda function. For more information, see Tutorial: Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

### Granting permissions to publish messages to an SNS topic or an SQS queue

To grant Amazon S3 permissions to publish messages to the SNS topic or SQS queue, attach an AWS Identity and Access Management (IAM) policy to the destination SNS topic or SQS queue.

For an example of how to attach a policy to an SNS topic or an SQS queue, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue). For more information about permissions, see the following topics:

- Example cases for Amazon SNS access control in the *Amazon Simple Notification Service Developer Guide*
- Identity and access management in Amazon SQS in the *Amazon Simple Queue Service Developer Guide*

### IAM policy for a destination SNS topic

The following is an example of an AWS Identity and Access Management (IAM) policy that you attach to the destination SNS topic. For instructions on how to use this policy to set up a destination Amazon SNS topic for event notifications, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

### IAM policy for a destination SQS queue

The following is an example of an IAM policy that you attach to the destination SQS queue. For instructions on how to use this policy to set up a destination Amazon SQS queue for event notifications, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

To use this policy, you must update the Amazon SQS queue ARN, bucket name, and bucket owner's AWS account ID.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "arn:aws:sqs:Region:account-id:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

For both the Amazon SNS and Amazon SQS IAM policies, you can specify the `StringLike` condition in the policy instead of the `ArnLike` condition.

When `ArnLike` is used, the partition, service, account-id, resource-type, and partial resource-id portions of the ARN must have exact matching to the ARN in the request context. Only the region and resource path allow partial matching.

When `StringLike` is used instead of `ArnLike`, matching ignores the ARN structure and allows partial matching, regardless of the portion that was wildcarded. For more information, see IAM JSON policy elements in the *IAM User Guide*.

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
  }
```

**AWS KMS key policy**

If the SQS queue or SNS topics are encrypted with an AWS Key Management Service (AWS KMS) customer managed key, you must grant the Amazon S3 service principal permission to work with the encrypted topics or queue. To grant the Amazon S3 service principal permission, add the following statement to the key policy for the customer managed key.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about AWS KMS key policies, see Using key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

For more information about using server-side encryption with AWS KMS for Amazon SQS and Amazon SNS, see the following:

- Key management in the *Amazon Simple Notification Service Developer Guide*.

- Key management in the *Amazon Simple Queue Service Developer Guide*.
- Encrypting messages published to Amazon SNS with AWS KMS in the *AWS Compute Blog*.

## Enabling and configuring event notifications using the Amazon S3 console

You can enable certain Amazon S3 bucket events to send a notification message to a destination whenever those events occur. This section explains how to use the Amazon S3 console to enable event notifications. For information about how to use event notifications with the AWS SDKs and the Amazon S3 REST APIs, see Configuring event notifications programmatically.

**Prerequisites**: Before you can enable event notifications for your bucket, you must set up one of the destination types and then configure permissions. For more information, see Supported event destinations and Granting permissions to publish event notification messages to a destination.

> **Note**
>
> Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

### Topics

- Enabling Amazon SNS, Amazon SQS, or Lambda notifications using the Amazon S3 console

### Enabling Amazon SNS, Amazon SQS, or Lambda notifications using the Amazon S3 console

**To enable and configure event notifications for an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable events for.
3. Choose **Properties**.
4. Navigate to the **Event Notifications** section and choose **Create event notification**.
5. In the **General configuration** section, specify descriptive event name for your event notification. Optionally, you can also specify a prefix and a suffix to limit the notifications to objects with keys ending in the specified characters.

a. Enter a description for the **Event name**.

   If you don't enter a name, a globally unique identifier (GUID) is generated and used for the name.

b. (Optional) To filter event notifications by prefix, enter a **Prefix**.

   For example, you can set up a prefix filter so that you receive notifications only when files are added to a specific folder (for example, `images/`).

c. (Optional) To filter event notifications by suffix, enter a **Suffix**.

   For more information, see Configuring event notifications using object key name filtering.

6. In the **Event types** section, select one or more event types that you want to receive notifications for.

   For a list of the different event types, see Supported event types for SQS, SNS, and Lambda.

7. In the **Destination** section, choose the event notification destination.

> **Note**
>
> Before you can publish event notifications, you must grant the Amazon S3 principal the necessary permissions to call the relevant API. This is so that it can publish notifications to a Lambda function, SNS topic, or SQS queue.

   a. Select the destination type: **Lambda Function**, **SNS Topic**, or **SQS Queue**.

   b. After you choose your destination type, choose a function, topic, or queue from the list.

   c. Or, if you prefer to specify an Amazon Resource Name (ARN), select **Enter ARN** and enter the ARN.

   For more information, see Supported event destinations.

8. Choose **Save changes**, and Amazon S3 sends a test message to the event notification destination.

## Configuring event notifications programmatically

By default, notifications aren't enabled for any type of event. Therefore, the *notification* subresource initially stores an empty configuration.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

To enable notifications for events of specific types, you replace the XML with the appropriate configuration that identifies the event types you want Amazon S3 to publish and the destination where you want the events published. For each destination, you add a corresponding XML configuration.

### To publish event messages to an SQS queue

To set an SQS queue as the notification destination for one or more event types, add the `QueueConfiguration`.

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

### To publish event messages to an SNS topic

To set an SNS topic as the notification destination for specific event types, add the `TopicConfiguration`.

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
```

```
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

**To invoke the AWS Lambda function and provide an event message as an argument**

To set a Lambda function as the notification destination for specific event types, add the `CloudFunctionConfiguration`.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
     <CloudFunction>cloud-function-arn</CloudFunction>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </CloudFunctionConfiguration>
   ...
</NotificationConfiguration>
```

**To remove all notifications configured on a bucket**

To remove all notifications configured on a bucket, save an empty `<NotificationConfiguration/>` element in the *notification* subresource.

When Amazon S3 detects an event of the specific type, it publishes a message with the event information. For more information, see Event message structure.

For more information about configuring event notifications, see the following topics:

- Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).
- Configuring event notifications using object key name filtering

## Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue)

You can receive Amazon S3 notifications using Amazon Simple Notification Service (Amazon SNS) or Amazon Simple Queue Service (Amazon SQS). In this walkthrough, you add a notification configuration to your bucket using an Amazon SNS topic and an Amazon SQS queue.

> **Note**
>
> Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

**Topics**



### Walkthrough summary

This walkthrough helps you do the following:

- Publish events of the `s3:ObjectCreated:*` type to an Amazon SQS queue.
- Publish events of the `s3:ReducedRedundancyLostObject` type to an Amazon SNS topic.

For information about notification configuration, see Using Amazon SQS, Amazon SNS, and Lambda.

You can do all these steps using the console, without writing any code. In addition, code examples using AWS SDKs for Java and .NET are also provided to help you add notification configurations programmatically.

The procedure includes the following steps:

1. Create an Amazon SQS queue.

   Using the Amazon SQS console, create an SQS queue. You can access any messages Amazon S3 sends to the queue programmatically. But, for this walkthrough, you verify notification messages in the console.

You attach an access policy to the queue to grant Amazon S3 permission to post messages.

2. Create an Amazon SNS topic.

   Using the Amazon SNS console, create an SNS topic and subscribe to the topic. That way, any events posted to it are delivered to you. You specify email as the communications protocol. After you create a topic, Amazon SNS sends an email. You use the link in the email to confirm the topic subscription.

   You attach an access policy to the topic to grant Amazon S3 permission to post messages.

3. Add notification configuration to a bucket.

### Step 1: Create an Amazon SQS queue

Follow the steps to create and subscribe to an Amazon Simple Queue Service (Amazon SQS) queue.

1. Using the Amazon SQS console, create a queue. For instructions, see Getting Started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*.
2. Replace the access policy that's attached to the queue with the following policy.

   a. In the Amazon SQS console, in the **Queues** list, choose the queue name.

   b. On the **Access policy** tab, choose **Edit**.

   c. Replace the access policy that's attached to the queue. In it, provide your Amazon SQS ARN, source bucket name, and bucket owner account ID.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "SQS-queue-ARN",
            "Condition": {
```

```
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

d. Choose **Save**.

3. (Optional) If the Amazon SQS queue or the Amazon SNS topic is server-side encryption enabled with AWS Key Management Service (AWS KMS), add the following policy to the associated symmetric encryption customer managed key.

   You must add the policy to a customer managed key because you cannot modify the AWS managed key for Amazon SQS or Amazon SNS.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

   For more information about using SSE for Amazon SQS and Amazon SNS with AWS KMS, see the following:

   - Key management in the *Amazon Simple Notification Service Developer Guide*.

- Key management in the *Amazon Simple Queue Service Developer Guide*.

4. Note the queue ARN.

   The SQS queue that you created is another resource in your AWS account. It has a unique Amazon Resource Name (ARN). You need this ARN in the next step. The ARN is of the following format:

   ```
   arn:aws:sqs:aws-region:account-id:queue-name
   ```

### Step 2: Create an Amazon SNS topic

Follow the steps to create and subscribe to an Amazon SNS topic.

1. Using Amazon SNS console, create a topic. For instructions, see Creating an Amazon SNS topic in the *Amazon Simple Notification Service Developer Guide*.
2. Subscribe to the topic. For this exercise, use email as the communications protocol. For instructions, see Subscribing to an Amazon SNS topic in the *Amazon Simple Notification Service Developer Guide*.

   You get an email requesting you to confirm your subscription to the topic. Confirm the subscription.
3. Replace the access policy attached to the topic with the following policy. In it, provide your SNS topic ARN, bucket name, and bucket owner's account ID.

   ```
   {
       "Version": "2012-10-17",
       "Id": "example-ID",
       "Statement": [
           {
               "Sid": "Example SNS topic policy",
               "Effect": "Allow",
               "Principal": {
                   "Service": "s3.amazonaws.com"
               },
               "Action": [
                   "SNS:Publish"
               ],
               "Resource": "SNS-topic-ARN",
               "Condition": {
   ```

```
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

4. Note the topic ARN.

   The SNS topic you created is another resource in your AWS account, and it has a unique ARN. You will need this ARN in the next step. The ARN will be of the following format:

   ```
   arn:aws:sns:aws-region:account-id:topic-name
   ```

**Step 3: Add a notification configuration to your bucket**

You can enable bucket notifications either by using the Amazon S3 console or programmatically by using AWS SDKs. Choose any one of the options to configure notifications on your bucket. This section provides code examples using the AWS SDKs for Java and .NET.

**Option A: Enable notifications on a bucket using the console**

Using the Amazon S3 console, add a notification configuration requesting Amazon S3 to do the following:

- Publish events of the **All object create events** type to your Amazon SQS queue.
- Publish events of the **Object in RRS lost** type to your Amazon SNS topic.

After you save the notification configuration, Amazon S3 posts a test message, which you get via email.

For instructions, see Enabling and configuring event notifications using the Amazon S3 console.

### Option B: Enable notifications on a bucket using the AWS SDKs

.NET

The following C# code example provides a complete code listing that adds a notification configuration to a bucket. You must update the code and provide your bucket name and SNS topic ARN. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }

        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new
 PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };
```

```
                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
                });

                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server.
 Message:'{0}' ", e.Message);
            }
        }
    }
}
```

Java

The following example shows how to add a notification configuration to a bucket. For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.EnumSet;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String snsTopicARN = "*** SNS Topic ARN ***";
        String sqsQueueARN = "*** SQS Queue ARN ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
            BucketNotificationConfiguration notificationConfiguration = new
 BucketNotificationConfiguration();

            // Add an SNS topic notification.
            notificationConfiguration.addConfiguration("snsTopicConfig",
                    new TopicConfiguration(snsTopicARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Add an SQS queue notification.
            notificationConfiguration.addConfiguration("sqsQueueConfig",
                    new QueueConfiguration(sqsQueueARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Create the notification configuration request and set the bucket
 notification
            // configuration.
            SetBucketNotificationConfigurationRequest request = new
 SetBucketNotificationConfigurationRequest(
                    bucketName, notificationConfiguration);
            s3Client.setBucketNotificationConfiguration(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
```

```
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

**Step 4: Test the setup**

Now, you can test the setup by uploading an object to your bucket and verifying the event notification in the Amazon SQS console. For instructions, see Receiving a Message in the *Amazon Simple Queue Service Developer Guide "Getting Started" section*.

## Configuring event notifications using object key name filtering

When configuring an Amazon S3 event notification, you must specify which supported Amazon S3 event types cause Amazon S3 to send the notification. If an event type that you didn't specify occurs in your S3 bucket, Amazon S3 doesn't send the notification.

You can configure notifications to be filtered by the prefix and suffix of the key name of objects. For example, you can set up a configuration where you're sent a notification only when image files with a "`.jpg`" file name extension are added to a bucket. Or, you can have a configuration that delivers a notification to an Amazon SNS topic when an object with the prefix "`images/`" is added to the bucket, while having notifications for objects with a "`logs/`" prefix in the same bucket delivered to an AWS Lambda function.

> **Note**
>
> A wildcard character ("*") can't be used in filters as a prefix or suffix. If your prefix or suffix contains a space, you must replace it with the "+" character. If you use any other special characters in the value of the prefix or suffix, you must enter them in URL-encoded (percent-encoded) format. For a complete list of special characters that must be converted to URL-encoded format when used in a prefix or suffix for event notifications, see Safe characters.

You can set up notification configurations that use object key name filtering in the Amazon S3 console. You can do so by using Amazon S3 APIs through the AWS SDKs or the REST APIs directly. For information about using the console UI to set a notification configuration on a bucket, see Enabling and configuring event notifications using the Amazon S3 console.

Amazon S3 stores the notification configuration as XML in the *notification* subresource associated with a bucket as described in Using Amazon SQS, Amazon SNS, and Lambda. You use the `Filter` XML structure to define the rules for notifications to be filtered by the prefix or suffix of an object key name. For information about the `Filter` XML structure, see PUT Bucket notification in the *Amazon Simple Storage Service API Reference*.

Notification configurations that use `Filter` cannot define filtering rules with overlapping prefixes, overlapping suffixes, or prefix and suffix overlapping. The following sections have examples of valid notification configurations with object key name filtering. They also contain examples of notification configurations that are not valid because of prefix and suffix overlapping.

**Topics**

- Examples of valid notification configurations with object key name filtering
- Examples of notification configurations with invalid prefix and suffix overlapping

### Examples of valid notification configurations with object key name filtering

The following notification configuration contains a queue configuration identifying an Amazon SQS queue for Amazon S3 to publish events to of the `s3:ObjectCreated:Put` type. The events are published whenever an object that has a prefix of `images/` and a `jpg` suffix is PUT to a bucket.

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
              </FilterRule>
          </S3Key>
```

```
    </Filter>
    <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
    <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

The following notification configuration has multiple non-overlapping prefixes. The configuration defines that notifications for PUT requests in the `images/` folder go to queue-A, while notifications for PUT requests in the `logs/` folder go to queue-B.

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>1</Id>
    <Filter>
           <S3Key>
               <FilterRule>
                   <Name>prefix</Name>
                   <Value>images/</Value>
               </FilterRule>
           </S3Key>
    </Filter>
    <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
    <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
    <Id>2</Id>
    <Filter>
           <S3Key>
               <FilterRule>
                   <Name>prefix</Name>
                   <Value>logs/</Value>
               </FilterRule>
           </S3Key>
    </Filter>
    <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
    <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

The following notification configuration has multiple non-overlapping suffixes. The configuration defines that all `.jpg` images newly added to the bucket are processed by Lambda cloud-function-A, and all newly added `.png` images are processed by cloud-function-B. The `.png` and `.jpg`

suffixes aren't overlapping even though they have the same last letter. If a given string can end with both suffixes, the two suffixes are considered overlapping. A string can't end with both `.png` and `.jpg`, so the suffixes in the example configuration aren't overlapping suffixes.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

Your notification configurations that use `Filter` can't define filtering rules with overlapping prefixes for the same event types. They can only do so, if the overlapping prefixes that are used with suffixes that don't overlap. The following example configuration shows how objects created with a common prefix but non-overlapping suffixes can be delivered to different destinations.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
```

```
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

### Examples of notification configurations with invalid prefix and suffix overlapping

For the most part, your notification configurations that use `Filter` can't define filtering rules with overlapping prefixes, overlapping suffixes, or overlapping combinations of prefixes and suffixes for the same event types. You can have overlapping prefixes as long as the suffixes don't overlap. For an example, see Configuring event notifications using object key name filtering.

You can use overlapping object key name filters with different event types. For example, you can create a notification configuration that uses the prefix `image/` for the `ObjectCreated:Put` event type and the prefix `image/` for the `ObjectRemoved:*` event type.

You get an error if you try to save a notification configuration that has invalid overlapping name filters for the same event types when using the Amazon S3 console or API. This section shows examples of notification configurations that aren't valid because of overlapping name filters.

Any existing notification configuration rule is assumed to have a default prefix and suffix that match any other prefix and suffix, respectively. The following notification configuration isn't valid because it has overlapping prefixes. Specifically, the root prefix overlaps with any other prefix. The same thing is true if you use a suffix instead of a prefix in this example. The root suffix overlaps with any other suffix.

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

The following notification configuration isn't valid because it has overlapping suffixes. If a given string can end with both suffixes, the two suffixes are considered overlapping. A string can end with `jpg` and pg. So, the suffixes overlap. The same is true for prefixes. If a given string can begin with both prefixes, the two prefixes are considered overlapping.

```
 <NotificationConfiguration>
    <TopicConfiguration>
        <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
```

```
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

The following notification configuration isn't valid because it has overlapping prefixes and suffixes.

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
```

```
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

## Event message structure

The notification message that Amazon S3 sends to publish an event is in the JSON format.

For a general overview and instructions on configuring event notifications, see Amazon S3 Event Notifications.

This example shows *version 2.2* of the event notification JSON structure. Amazon S3 uses *versions 2.1*, *2.2*, and *2.3* of this event structure. Amazon S3 uses version 2.2 for cross-Region replication event notifications. It uses version 2.3 for S3 Lifecycle, S3 Intelligent-Tiering, object ACL, object tagging, and object restoration delete events. These versions contain extra information specific to these operations. Versions 2.2 and 2.3 are otherwise compatible with version 2.1, which Amazon S3 currently uses for all other event notification types.

```
{
   "Records":[
      {
         "eventVersion":"2.2",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"The time, in ISO-8601 format, for example,
 1970-01-01T00:00:00.000Z, when Amazon S3 finished processing the request",
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
```

```
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
               "name":"bucket-name",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
               "key":"object-key",
               "size":"object-size in bytes",
               "eTag":"object eTag",
               "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
               "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence, only used with PUTs and DELETEs"
            }
         },
         "glacierEventData": {
            "restoreEventData": {
               "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format, for
 example, 1970-01-01T00:00:00.000Z, of Restore Expiry",
               "lifecycleRestoreStorageClass": "Source storage class for restore"
            }
         }
      }
   ]
}
```

Note the following about the event message structure:

- The `eventVersion` key value contains a major and minor version in the form `<major>.<minor>`.

The major version is incremented if Amazon S3 makes a change to the event structure that's not backward compatible. This includes removing a JSON field that's already present or changing how the contents of a field are represented (for example, a date format).

The minor version is incremented if Amazon S3 adds new fields to the event structure. This might occur if new information is provided for some or all existing events. This might also occur if new information is provided on only newly introduced event types. Applications should ignore new fields to stay forward compatible with new minor versions of the event structure.

If new event types are introduced but the structure of the event is otherwise unmodified, the event version doesn't change.

To ensure that your applications can parse the event structure correctly, we recommend that you do an equal-to comparison on the major version number. To ensure that the fields that are expected by your application are present, we also recommend doing a greater-than-or-equal-to comparison on the minor version.

- The `eventName` references the list of event notification types but doesn't contain the `s3:` prefix.
- The `responseElements` key value is useful if you want to trace a request by following up with AWS Support. Both `x-amz-request-id` and `x-amz-id-2` help Amazon S3 trace an individual request. These values are the same as those that Amazon S3 returns in the response to the request that initiates the events. This is so they can be used to match the event to the request.
- The `s3` key provides information about the bucket and object involved in the event. The object key name value is URL encoded. For example, "red flower.jpg" becomes "red+flower.jpg" (Amazon S3 returns "`application/x-www-form-urlencoded`" as the content type in the response).
- The `sequencer` key provides a way to determine the sequence of events. Event notifications aren't guaranteed to arrive in the same order that the events occurred. However, notifications from events that create objects (PUTs) and delete objects contain a `sequencer`. It can be used to determine the order of events for a given object key.

  If you compare the `sequencer` strings from two event notifications on the same object key, the event notification with the greater `sequencer` hexadecimal value is the event that occurred later. If you're using event notifications to maintain a separate database or index of your Amazon S3 objects, we recommend that you compare and store the `sequencer` values as you process each event notification.

  Note the following:

  - You can't use `sequencer` to determine order for events on different object keys.
  - The sequencers can be of different lengths. So, to compare these values, first right pad the shorter value with zeros, and then do a lexicographical comparison.
- The `glacierEventData` key is only visible for `s3:ObjectRestore:Completed` events.
- The `restoreEventData` key contains attributes that are related to your restore request.
- The `replicationEventData` key is only visible for replication events.
- The `intelligentTieringEventData` key is only visible for S3 Intelligent-Tiering events.
- The `lifecycleEventData` key is only visible for S3 Lifecycle transition events.

### Example messages

The following are examples of Amazon S3 event notification messages.

#### Amazon S3 test message

After you configure an event notification on a bucket, Amazon S3 sends the following test message.

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"bucketname",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

#### Example message when an object is created using a PUT request

The following message is an example of a message Amazon S3 sends to publish an `s3:ObjectCreated:Put` event.

```
{
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
```

```
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

For a definition of each IAM identification prefix (for example, AIDA, AROA, AGPA), see IAM identifiers in the *IAM User Guide*.

## Using EventBridge

Amazon S3 can send events to Amazon EventBridge whenever certain events happen in your bucket. Unlike other destinations, you don't need to select which event types you want to deliver. After EventBridge is enabled, all events below are sent to EventBridge. You can use EventBridge

rules to route events to additional targets. The following lists the events Amazon S3 sends to EventBridge.

| Event type | Description |
| --- | --- |
| *Object Created* | An object was created.<br><br>The reason field in the event message structure indicates which S3 API was used to create the object: PutObject, POST Object, CopyObject, or CompleteMultipartUpload. |
| *Object Deleted (DeleteObject)*<br><br>*Object Deleted (Lifecycle expiration)* | An object was deleted.<br><br>When an object is deleted using an S3 API call, the reason field is set to DeleteObject. When an object is deleted by an S3 Lifecycle expiration rule, the reason field is set to Lifecycle Expiration. For more information, see Expiring objects.<br><br>When an unversioned object is deleted, or a versioned object is permanently deleted, the deletion-type field is set to Permanently Deleted. When a delete marker is created for a versioned object, the deletion-type field is set to Delete Marker Created. For more information, see Deleting object versions from a versioning-enabled bucket. |
| *Object Restore Initiated* | An object restore was initiated from S3 Glacier or S3 Glacier Deep Archive storage class or from S3 Intellige nt-Tiering Archive Access or Deep Archive Access tier. For more information, see Working with archived objects. |
| *Object Restore Completed* | An object restore was completed. |
| *Object Restore Expired* | The temporary copy of an object restored from S3 Glacier or S3 Glacier Deep Archive expired and was deleted. |

| Event type | Description |
| --- | --- |
| *Object Storage Class Changed* | An object was transitioned to a different storage class. For more information, see Transitioning objects using Amazon S3 Lifecycle. |
| *Object Access Tier Changed* | An object was transitioned to the S3 Intelligent-Tiering Archive Access tier or Deep Archive Access tier. For more information, see Amazon S3 Intelligent-Tiering. |
| *Object ACL Updated* | An object's access control list (ACL) was set using PutObjectACL. An event is not generated when a request results in no change to an object's ACL. For more informati on, see Access control list (ACL) overview. |
| *Object Tags Added* | A set of tags was added to an object using PutObject Tagging. For more information, see Categorizing your storage using tags. |
| *Object Tags Deleted* | All tags were removed from an object using DeleteObj ectTagging. For more information, see Categorizing your storage using tags. |

**Note**

For more information about how Amazon S3 event types map to EventBridge event types, see Amazon EventBridge mapping and troubleshooting.

You can use Amazon S3 Event Notifications with EventBridge to write rules that take actions when an event occurs in your bucket. For example, you can have it send you a notification. For more information, see What is EventBridge in the *Amazon EventBridge User Guide*.

For more information about the actions and data types you can interact with using the EventBridge API, see the Amazon EventBridge API Reference in the *Amazon EventBridge API Reference*.

For information about pricing, see Amazon EventBridge pricing.

**Topics**



## Amazon EventBridge permissions

Amazon S3 does not require any additional permissions to deliver events to Amazon EventBridge.

## Enabling Amazon EventBridge

You can enable Amazon EventBridge using the S3 console, AWS Command Line Interface (AWS CLI), or Amazon S3 REST API.

### Using the S3 console

**To enable EventBridge event delivery in the S3 console.**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable events for.
3. Choose **Properties**.
4. Navigate to the **Event Notifications** section and find the **Amazon EventBridge** subsection. Choose **Edit**.
5. Under **Send notifications to Amazon EventBridge for all events in this bucket** choose **On**.

> **Note**
>
> After you enable EventBridge, it takes around five minutes for the changes to take effect.

### Using the AWS CLI

The following example creates a bucket notification configuration for bucket `amzn-s3-demo-bucket1` with Amazon EventBridge enabled.

```
aws s3api put-bucket-notification-configuration --bucket amzn-s3-demo-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

### Using the REST API

You can programmatically enable Amazon EventBridge on a bucket by calling the Amazon S3 REST API. For more information see, see PutBucketNotificationConfiguration in the *Amazon Simple Storage Service API Reference*.

The following example shows the XML used to create a bucket notification configuration with Amazon EventBridge enabled.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

### Creating EventBridge rules

Once enabled you can create Amazon EventBridge rules for certain tasks. For example, you can send email notifications when an object is created. For a full tutorial, see Tutorial: Send a notification when an Amazon S3 object is created in the *Amazon EventBridge User Guide*.

## EventBridge event message structure

The notification message that Amazon S3 sends to publish an event is in the JSON format. When Amazon S3 sends an event to Amazon EventBridge, the following fields are present.

- **version** — Currently 0 (zero) for all events.
- **id** — A Version 4 UUID generated for every event.
- **detail-type** — The type of event that's being sent. See Using EventBridge for a list of event types.
- **source** — Identifies the service that generated the event.
- **account** — The 12-digit AWS account ID of the bucket owner.
- **time** — The time the event occurred.
- **region** — Identifies the AWS Region of the bucket.
- **resources** — A JSON array that contains the Amazon Resource Name (ARN) of the bucket.

- **detail** — A JSON object that contains information about the event. For more information about what can be included in this field, see Event message detail field.

### Event message structure examples

The following are examples of some of the Amazon S3 event notification messages that can be sent to Amazon EventBridge.

### Object created

```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "PutObject"
  }
}
```

### Object deleted (using DeleteObject)


```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```


### Object deleted (using lifecycle expiration)


```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
```

```
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```

### Object restore completed

```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
```

```
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```

### Event message detail field

The detail field contains a JSON object with information about the event. The following fields may be present in the detail field.

- **version** — Currently 0 (zero) for all events.
- **bucket** — Information about the Amazon S3 bucket involved in the event.
- **object** — Information about the Amazon S3 object involved in the event.
- **request-id** — Request ID in S3 response.
- **requester** — AWS account ID or AWS service principal of requester.
- **source-ip-address** — Source IP address of S3 request. Only present for events triggered by an S3 request.
- **reason** — For **Object Created** events, the S3 API used to create the object: PutObject, POST Object, CopyObject, or CompleteMultipartUpload. For **Object Deleted** events, this is set to **DeleteObject** when an object is deleted by an S3 API call, or **Lifecycle Expiration** when an object is deleted by an S3 Lifecycle expiration rule. For more information, see Expiring objects.
- **deletion-type** — For **Object Deleted** events, when an unversioned object is deleted, or a versioned object is permanently deleted, this is set to **Permanently Deleted**. When a delete marker is created for a versioned object, this is set to **Delete Marker Created**. For more information, see Deleting object versions from a versioning-enabled bucket.

> **Note**
>
> Some object attributes (such as `etag` and `size`) are present only when a delete marker is created.

- **restore-expiry-time** — For **Object Restore Completed** events, the time when the temporary copy of the object will be deleted from S3. For more information, see Working with archived objects.
- **source-storage-class** — For **Object Restore Initiated** and **Object Restore Completed** events, the storage class of the object being restored. For more information, see Working with archived objects.
- **destination-storage-class** — For **Object Storage Class Changed** events, the new storage class of the object. For more information, see Transitioning objects using Amazon S3 Lifecycle.
- **destination-access-tier** — For **Object Access Tier Changed** events, the new access tier of the object. For more information, see Amazon S3 Intelligent-Tiering.

## Amazon EventBridge mapping and troubleshooting

The following table describes how Amazon S3 event types are mapped to Amazon EventBridge event types.

| S3 event type | Amazon EventBridge detail type |
| --- | --- |
| ObjectCreated:Put<br>ObjectCreated:Post<br>ObjectCreated:Copy<br>ObjectCreated:CompleteMultipartUpload | Object Created |
| ObjectRemoved:Delete<br>ObjectRemoved:DeleteMarkerCreated<br>LifecycleExpiration:Delete<br>LifecycleExpiration:DeleteMarkerCreated | Object Deleted |
| ObjectRestore:Post | Object Restore Initiated |

| S3 event type | Amazon EventBridge detail type |
| --- | --- |
| ObjectRestore:Completed | Object Restore Completed |
| ObjectRestore:Delete | Object Restore Expired |
| LifecycleTransition | Object Storage Class Changed |
| IntelligentTiering | Object Access Tier Changed |
| ObjectTagging:Put | Object Tags Added |
| ObjectTagging:Delete | Object Tags Deleted |
| ObjectAcl:Put | Object ACL Updated |

### Amazon EventBridge troubleshooting

For information about how to troubleshoot EventBridge, see Troubleshooting Amazon EventBridge in the *Amazon EventBridge User Guide*.

# Using analytics and insights

You can use analytics and insights in Amazon S3 to understand, analyze, and optimize your storage usage . For more information, see the topics below.

**Topics**



# Amazon S3 analytics – Storage Class Analysis

By using Amazon S3 analytics *Storage Class Analysis* you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. This new Amazon S3 analytics feature observes data access patterns to help you determine when to transition less frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class. For more information about storage classes, see Using Amazon S3 storage classes.

After storage class analysis observes the infrequent access patterns of a filtered set of data over a period of time, you can use the analysis results to help you improve your lifecycle configurations. You can configure storage class analysis to analyze all the objects in a bucket. Or, you can configure filters to group objects together for analysis by common prefix (that is, objects that have names that begin with a common string), by object tags, or by both prefix and tags. You'll most likely find that filtering by object groups is the best way to benefit from storage class analysis.

> **Important**
>
> Storage class analysis only provides recommendations for Standard to Standard IA classes.

You can have multiple storage class analysis filters per bucket, up to 1,000, and will receive a separate analysis for each filter. Multiple filter configurations allow you analyze specific groups of objects to improve your lifecycle configurations that transition objects to STANDARD_IA.

Storage class analysis provides storage usage visualizations in the Amazon S3 console that are updated daily. You can also export this daily usage data to an S3 bucket and view them in a spreadsheet application, or with business intelligence tools, like Amazon QuickSight.

There are costs associated with the storage class analysis. For pricing information, see *Management and replication* Amazon S3 pricing.

**Topics**



## How do I set up storage class analysis?

You set up storage class analysis by configuring what object data you want to analyze. You can configure storage class analysis to do the following:

- **Analyze the entire contents of a bucket.**

  You'll receive an analysis for all the objects in the bucket.
- **Analyze objects grouped together by prefix and tags.**

  You can configure filters that group objects together for analysis by prefix, or by object tags, or by a combination of prefix and tags. You receive a separate analysis for each filter you configure. You can have multiple filter configurations per bucket, up to 1,000.
- **Export analysis data.**

  When you configure storage class analysis for a bucket or filter, you can choose to have the analysis data exported to a file each day. The analysis for the day is added to the file to form a historic analysis log for the configured filter. The file is updated daily at the destination of your choice. When selecting data to export, you specify a destination bucket and optional destination prefix where the file is written.

You can use the Amazon S3 console, the REST API, or the AWS CLI or AWS SDKs to configure storage class analysis.

- For information about how to configure storage class analysis in the Amazon S3 console, see Configuring storage class analysis.
- To use the Amazon S3 API, use the PutBucketAnalyticsConfiguration REST API, or the equivalent, from the AWS CLI or AWS SDKs.

## How do I use storage class analysis?

You use storage class analysis to observe your data access patterns over time to gather information to help you improve the lifecycle management of your STANDARD_IA storage. After you configure a filter, you'll start seeing data analysis based on the filter in the Amazon S3 console in 24 to 48 hours. However, storage class analysis observes the access patterns of a filtered data set for 30 days or longer to gather information for analysis before giving a result. The analysis continues to run after the initial result and updates the result as the access patterns change

When you first configure a filter, the Amazon S3 console may take a moment to analyze your data.

Storage class analysis observes the access patterns of a filtered object data set for 30 days or longer to gather enough information for the analysis. After storage class analysis has gathered sufficient information, you'll see a message in the Amazon S3 console that analysis is complete.

When performing the analysis for infrequently accessed objects storage class analysis looks at the filtered set of objects grouped together based on age since they were uploaded to Amazon S3. Storage class analysis determines if the age group is infrequently accessed by looking at the following factors for the filtered data set:

- Objects in the STANDARD storage class that are larger than 128 KB.
- How much average total storage you have per age group.
- Average number of bytes transferred out (not frequency) per age group.
- Analytics export data only includes requests with data relevant to storage class analysis. This might cause differences in the number of requests, and the total upload and request bytes compared to what are shown in storage metrics or tracked by your own internal systems.
- Failed GET and PUT requests are not counted for the analysis. However, you will see failed requests in storage metrics.

#### How Much of My Storage did I Retrieve?

The Amazon S3 console graphs how much of the storage in the filtered data set has been retrieved for the observation period.

#### What Percentage of My Storage did I Retrieve?

The Amazon S3 console also graphs what percentage of the storage in the filtered data set has been retrieved for the observation period.

As stated earlier in this topic, when you are performing the analysis for infrequently accessed objects, storage class analysis looks at the filtered set of objects grouped together based on the age since they were uploaded to Amazon S3. The storage class analysis uses the following predefined object age groups:

- Amazon S3 Objects less than 15 days old
- Amazon S3 Objects 15-29 days old
- Amazon S3 Objects 30-44 days old
- Amazon S3 Objects 45-59 days old
- Amazon S3 Objects 60-74 days old
- Amazon S3 Objects 75-89 days old
- Amazon S3 Objects 90-119 days old
- Amazon S3 Objects 120-149 days old
- Amazon S3 Objects 150-179 days old
- Amazon S3 Objects 180-364 days old
- Amazon S3 Objects 365-729 days old
- Amazon S3 Objects 730 days and older

Usually it takes about 30 days of observing access patterns to gather enough information for an analysis result. It might take longer than 30 days, depending on the unique access pattern of your data. However, after you configure a filter you'll start seeing data analysis based on the filter in the Amazon S3 console in 24 to 48 hours. You can see analysis on a daily basis of object access broken down by object age group in the Amazon S3 console.

**How Much of My Storage is Infrequently Accessed?**

The Amazon S3 console shows the access patterns grouped by the predefined object age groups. The **Frequently accessed** or **Infrequently accessed** text shown is meant as a visual aid to help you in the lifecycle creation process.

## How can I export storage class analysis data?

You can choose to have storage class analysis export analysis reports to a comma-separated values (CSV) flat file. Reports are updated daily and are based on the object age group filters you configure. When using the Amazon S3 console you can choose the export report option when you create a filter. When selecting data export you specify a destination bucket and optional

destination prefix where the file is written. You can export the data to a destination bucket in a different account. The destination bucket must be in the same region as the bucket that you configure to be analyzed.

You must create a bucket policy on the destination bucket to grant permissions to Amazon S3 to verify what AWS account owns the bucket and to write objects to the bucket in the defined location. For an example policy, see Grant permissions for S3 Inventory and S3 analytics.

After you configure storage class analysis reports, you start getting the exported report daily after 24 hours. After that, Amazon S3 continues monitoring and providing daily exports.

You can open the CSV file in a spreadsheet application or import the file into other applications like Amazon QuickSight. For information on using Amazon S3 files with Amazon QuickSight, see Create a Data Set Using Amazon S3 Files in the *Amazon QuickSight User Guide*.

Data in the exported file is sorted by date within object age group as shown in following examples. If the storage class is STANDARD the row also contains data for the columns `ObjectAgeForSIATransition` and `RecommendedObjectAgeForSIATransition`.

| Date | ConfigId | Filter | StorageClass | ObjectAge | ObjectCount | DataUploaded_MB | Storage_MB | DataRetrieved_MB | GetRequestCount | CumulativeAccessRatio | ObjectAgeForSIATransition | RecommendedObjectAgeForSIATransition |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8/17/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | 0.4313 | | | 0 | | |
| 9/2/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | | | | 0.04096734 | | |
| 8/22/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | 0.4313 | | | 0 | | |
| 8/25/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | 0.4313 | | | 0 | | |
| 9/6/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | | | | 0.04096734 | | |
| 8/30/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | | | | 0.04096734 | | |
| 8/28/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | | | | 0.04096734 | | |
| 8/21/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | 0.4313 | | | 0 | | |
| 9/5/2021 | SalesMaterial | SalesMaterial | STANDARD | 000-014 | | | | | | 0.04096734 | | |

At the end of the report the object age group is given as ALL. The ALL rows contain cumulative totals, including objects smaller than 128 KB, for all the age groups for that day.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8/24/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |
| 9/3/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0.02426125 | 015-029 |
| 8/28/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0.03545875 | 015-029 |
| 8/17/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |
| 8/25/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |
| 9/6/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0.0209529 | 015-029 |
| 9/4/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0.02304819 | 015-029 |
| 8/22/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |
| 8/21/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |
| 8/30/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0.03073092 | 015-029 |
| 8/20/2021 | SalesMaterial | SalesMaterial | STANDARD | ALL | 3 | | 0.4599 | | | 0 | 000-014 |

The next section describes the columns used in the report.

### Exported file layout

The following table describe the layout of the exported file.

## Configuring storage class analysis

By using the Amazon S3 analytics storage class analysis tool, you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. Storage

class analysis observes data access patterns to help you determine when to transition less frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class. For more information about STANDARD_IA, see the Amazon S3 FAQ and Using Amazon S3 storage classes.

You set up storage class analysis by configuring what object data you want to analyze. You can configure storage class analysis to do the following:

- **Analyze the entire contents of a bucket.**

  You'll receive an analysis for all the objects in the bucket.

- **Analyze objects grouped together by prefix and tags.**

  You can configure filters that group objects together for analysis by prefix, or by object tags, or by a combination of prefix and tags. You receive a separate analysis for each filter you configure. You can have multiple filter configurations per bucket, up to 1,000.

- **Export analysis data.**

  When you configure storage class analysis for a bucket or filter, you can choose to have the analysis data exported to a file each day. The analysis for the day is added to the file to form a historic analysis log for the configured filter. The file is updated daily at the destination of your choice. When selecting data to export, you specify a destination bucket and optional destination prefix where the file is written.

You can use the Amazon S3 console, the REST API, or the AWS CLI or AWS SDKs to configure storage class analysis.

**Important**

Storage class analysis does not give recommendations for transitions to the ONEZONE_IA or S3 Glacier Flexible Retrieval storage classes.
If you want to configure storage class analysis to export your findings as a .csv file and the destination bucket uses default bucket encryption with a AWS KMS key, you must update the AWS KMS key policy to grant Amazon S3 permission to encrypt the .csv file. For instructions, see Granting Amazon S3 permission to use your customer managed key for encryption.

For more information about analytics, see Amazon S3 analytics – Storage Class Analysis.

**Using the S3 console**

**To configure storage class analysis**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket for which you want to configure storage class analysis.
3. Choose the **Metrics** tab.
4. Under **Storage Class Analysis**, choose **Create analytics configuration**.
5. Type a name for the filter. If you want to analyze the whole bucket, leave the **Prefix** field empty.
6. In the **Prefix** field, type text for the prefix for the objects that you want to analyze.
7. To add a tag, choose **Add tag**. Enter a key and value for the tag. You can enter one prefix and multiple tags.
8. Optionally, you can choose **Enable** under **Export CSV** to export analysis reports to a comma-separated values (.csv) flat file. Choose a destination bucket where the file can be stored. You can type a prefix for the destination bucket. The destination bucket must be in the same AWS Region as the bucket for which you are setting up the analysis. The destination bucket can be in a different AWS account.

   If the destination bucket for the .csv file uses default bucket encryption with a KMS key, you must update the AWS KMS key policy to grant Amazon S3 permission to encrypt the .csv file. For instructions, see Granting Amazon S3 permission to use your customer managed key for encryption.
9. Choose **Create Configuration**.

Amazon S3 creates a bucket policy on the destination bucket that grants Amazon S3 write permission. This will allow it to write the export data to the bucket.

If an error occurs when you try to create the bucket policy, you'll be given instructions on how to fix it. For example, if you chose a destination bucket in another AWS account and do not have permissions to read and write to the bucket policy, you'll see the following message. You must have the destination bucket owner add the displayed bucket policy to the destination bucket. If

the policy is not added to the destination bucket you won't get the export data because Amazon S3 doesn't have permission to write to the destination bucket. If the source bucket is owned by a different account than that of the current user, then the correct account ID of the source bucket must be substituted in the policy.

For information about the exported data and how the filter works, see Amazon S3 analytics – Storage Class Analysis.

**Using the REST API**

To configure Storage Class Analysis using the REST API, use the PutBucketAnalyticsConfiguration. You can also use the equivalent operation with the AWS CLI or AWS SDKs.

You can use the following REST APIs to work with Storage Class Analysis:

- DELETE Bucket Analytics configuration
- GET Bucket Analytics configuration
- List Bucket Analytics Configuration

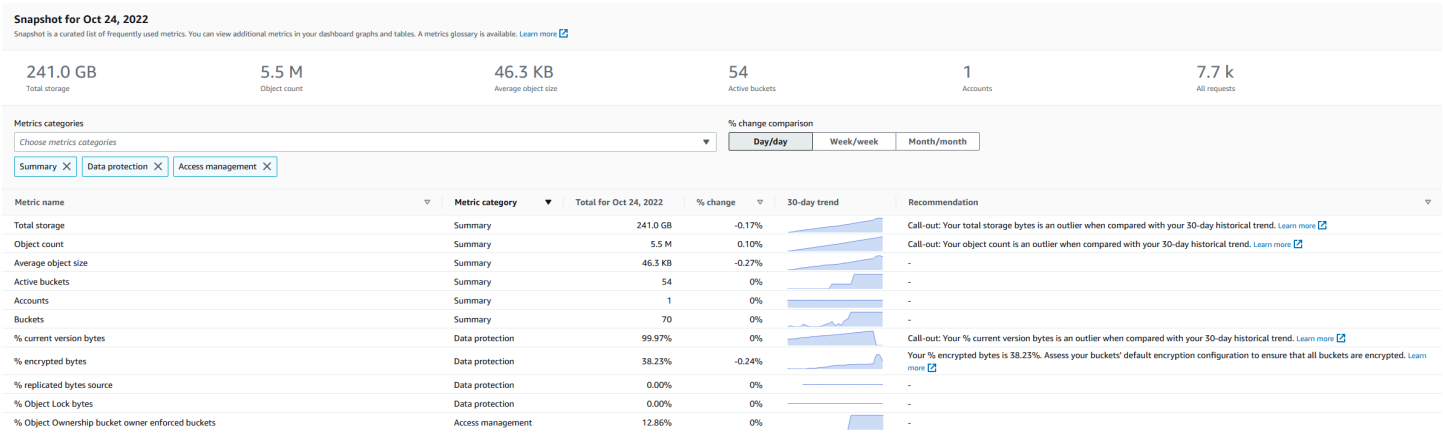
# Assessing your storage activity and usage with Amazon S3 Storage Lens

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object storage and activity. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

You can use S3 Storage Lens metrics to generate summary insights. For example, you can find out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and access-management best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to abort incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an

interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.




## S3 Storage Lens metrics and features

S3 Storage Lens provides an interactive *default dashboard* that is updated daily. S3 Storage Lens preconfigures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. Metrics from this dashboard are also summarized in your account snapshot on the **Buckets** page. For more information, see Default dashboard.

To create other dashboards and scope them by AWS Regions, S3 buckets, or accounts (for AWS Organizations), you create an S3 Storage Lens dashboard configuration. You can create and manage S3 Storage Lens dashboard configurations by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. When you create or edit an S3 Storage Lens dashboard, you define your dashboard scope and metrics selection.

S3 Storage Lens offers free metrics and advanced metrics and recommendations, which you can upgrade to for an additional charge. With advanced metrics and recommendations, you can access additional metrics and features for gaining insight into your storage. These features include advanced metric categories, prefix aggregation, contextual recommendations, and Amazon CloudWatch publishing. Prefix aggregation and contextual recommendations are available only in the Amazon S3 console. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

### Metrics categories

Within the free and advanced tiers, metrics are organized into categories that align with key use cases, such as cost optimization and data protection. Free metrics include summary, cost optimization, data protection, access management, performance, and event metrics. When you upgrade to advanced metrics and recommendations, you can enable advanced cost-optimization and data-protection metrics. You can use these advanced metrics to further reduce your S3 storage costs and improve your data-protection stance. You can also enable activity metrics and detailed status-code metrics to improve the performance of application workloads that are accessing your S3 buckets. For more information about the free and advanced metrics categories, see Metrics selection.

You can assess your storage based on S3 best practices, such as analyzing the percentage of your buckets that have encryption or S3 Object Lock or S3 Versioning enabled. You can also identify potential cost-savings opportunities. For example, you can use S3 Lifecycle rule count metrics to identify buckets that are missing lifecycle expiration or transition rules. You can also analyze your request activity per bucket to find buckets where objects could be transitioned to a lower-cost storage class. For more information, see Amazon S3 Storage Lens metrics use cases.

**Metrics export**

In addition to viewing the dashboard on the S3 console, you can export metrics in CSV or Parquet format to an S3 bucket for further analysis with the analytics tool of your choice. For more information, see Viewing Amazon S3 Storage Lens metrics using a data export.

**Amazon CloudWatch publishing**

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. The CloudWatch publishing option is available for dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations. For more information about support for S3 Storage Lens metrics in CloudWatch, see Monitor S3 Storage Lens metrics in CloudWatch.

For more information about using S3 Storage Lens, see the following topics.

**Topics**

- Understanding Amazon S3 Storage Lens
- Using Amazon S3 Storage Lens with AWS Organizations

- Amazon S3 Storage Lens permissions
- Viewing metrics with Amazon S3 Storage Lens
- Amazon S3 Storage Lens metrics use cases
- Amazon S3 Storage Lens metrics glossary
- Working with Amazon S3 Storage Lens by using the console and API
- Working with S3 Storage Lens groups

## Understanding Amazon S3 Storage Lens

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface and AWS SDKs. For more information, see Default encryption FAQ.

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive

recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket. You can create and manage S3 Storage Lens dashboards by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

## S3 Storage Lens concepts and terminology

This section contains the terminology and concepts that are essential for successfully understanding and using Amazon S3 Storage Lens.

**Topics**



### Dashboard configuration

S3 Storage Lens requires a dashboard configuration that contains the properties required to aggregate metrics on your behalf for a single dashboard or export. When you create a configuration, you choose the dashboard name and the home Region, which you can't change after you create the dashboard. You can optionally add tags and configure a metrics export in CSV or Parquet format.

In the dashboard configuration, you also define the dashboard scope and the metrics selection. The scope can include all the storage for your organization account or sections that are filtered by Region, bucket, and account. When you configure the metrics selection, you choose between free

metrics and advanced metrics and recommendations, which you can upgrade to for an additional charge. With advanced metrics and recommendations, you can access additional metrics and features. These features include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

### Default dashboard

The S3 Storage Lens default dashboard on the console is named **default-account-dashboard**. S3 preconfigures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. You can't modify the configuration scope of the default dashboard, but you can upgrade the metrics selection from free metrics to advanced metrics and recommendations. You can configure the optional metrics export or even disable the dashboard. However, you can't delete the default dashboard.

> **Note**
>
> If you disable your default dashboard, it is no longer updated. You'll no longer receive any new daily metrics in your S3 Storage Lens dashboard, your metrics export, or the account snapshot on the S3 **Buckets** page. If your dashboard uses advanced metrics and recommendations, you'll no longer be charged. You can still see historic data in the dashboard until the 14-day period for data queries expires. This period is 15 months if you've enabled advanced metrics and recommendations. To access historic data, you can re-enable the dashboard within the expiration period.

### Dashboards

You can create additional S3 Storage Lens dashboards and scope them by AWS Regions, S3 buckets, or accounts (for AWS Organizations). When you create or edit a S3 Storage Lens dashboard, you define your dashboard scope and metrics selection. S3 Storage Lens offers free metrics and advanced metrics and recommendations, which you can upgrade to for an additional charge. With advanced metrics and recommendations, you can access additional metrics and features for gaining insight into your storage. These include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

You can also disable or delete dashboards. If you disable a dashboard, it is no longer updated, and you will no longer receive any new daily metrics. You can still see historic data until the 14-day

expiration period. If you enabled advanced metrics and recommendations for that dashboard, this period is 15 months. To access historic data, you can re-enable the dashboard within the expiration period.

If you delete your dashboard, you lose all your dashboard configuration settings. You will no longer receive any new daily metrics, and you also lose access to the historical data associated with that dashboard. If you want to access the historic data for a deleted dashboard, you must create another dashboard with the same name in the same home Region.

**Note**

- You can use S3 Storage Lens to create up to 50 dashboards per home Region.
- Organization-level dashboards can be limited only to a Regional scope.

### Account snapshot

The S3 Storage Lens **Account snapshot** summarizes metrics from your default dashboard and displays your total storage, object count, and average object size on the S3 console **Buckets** page. This account snapshot gives you quick access to insights about your storage without having to leave the **Buckets** page. The account snapshot also provides one-click access to your interactive S3 Storage Lens dashboard.

You can use your dashboard to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate insights at the organization, account, bucket, object, or prefix level. You can also send a once-daily metrics export to an S3 bucket in CSV or Parquet format.

You can't modify the dashboard scope of the **default-account dashboard** because it's linked to the **Account snapshot**. However, you can upgrade the metrics selection in your **default-account-dashboard** from free metrics to paid advanced metrics and recommendations. After upgrading, you can then display all requests, bytes uploaded, and bytes downloaded in the S3 Storage Lens **Account snapshot**.

**Note**

If you disable your default dashboard, your **Account snapshot** is no longer updated. To continue displaying metrics in the **Account snapshot**, you can re-enable the **default-account-dashboard**.

### Metrics export

An S3 Storage Lens metrics export is a file that contains all the metrics identified in your S3 Storage Lens configuration. This information is generated daily in CSV or Parquet format and is sent to an S3 bucket. You can use the metrics export for further analysis by using the metrics tool of your choice. The S3 bucket for your metrics export must be in the same Region as your S3 Storage Lens configuration. You can generate an S3 Storage Lens metrics export from the S3 console by editing your dashboard configuration. You can also configure a metrics export by using the AWS CLI and AWS SDKs.

### Home Region

The home Region is the AWS Region where all S3 Storage Lens metrics for a given dashboard configuration are stored. You must choose a home Region when you create your S3 Storage Lens dashboard configuration. After you choose a home Region, you can't change it. Also, if you're creating a Storage Lens group, we recommend that you choose the same home Region as your Storage Lens dashboard.

**Note**

You can choose one of the following Regions as your home Region:

- US East (N. Virginia) – `us-east-1`
- US East (Ohio) – `us-east-2`
- US West (N. California) – `us-west-1`
- US West (Oregon) – `us-west-2`
- Asia Pacific (Mumbai) – `ap-south-1`
- Asia Pacific (Seoul) – `ap-northeast-2`
- Asia Pacific (Singapore) – `ap-southeast-1`
- Asia Pacific (Sydney) – `ap-southeast-2`

- Asia Pacific (Tokyo) – `ap-northeast-1`
- Canada (Central) – `ca-central-1`
- China (Beijing) – `cn-north-1`
- China (Ningxia) – `cn-northwest-1`
- Europe (Frankfurt) – `eu-central-1`
- Europe (Ireland) – `eu-west-1`
- Europe (London) – `eu-west-2`
- Europe (Paris) – `eu-west-3`
- Europe (Stockholm) – `eu-north-1`
- South America (São Paulo) – `sa-east-1`

**Retention period**

S3 Storage Lens metrics are retained so that you can see historical trends and compare differences in your storage and activity over time. You can use Amazon S3 Storage Lens metrics for queries so that you can see historical trends and compare differences in your storage usage and activity over time.

All S3 Storage Lens metrics are retained for a period of 15 months. However, metrics are only available for queries for a specific duration, which depends on your metrics selection. This duration can't be modified. Free metrics are available for queries for a 14-day period, and advanced metrics are available for queries for a 15-month period.

**Metrics categories**

Within the free and advanced tiers, S3 Storage Lens metrics are organized into categories that align with key use cases, such as cost optimization and data protection. Free metrics include summary, cost optimization, data protection, access management, performance, and event metrics. When you upgrade to advanced metrics and recommendations, you can enable additional cost-optimization and data-protection metrics that you can use to further reduce your S3 storage costs and ensure your data is protected. You can also enable activity metrics and detailed status-code metrics that you can use to improve the performance of application workflows.

The following list shows all of the free and advanced metric categories. For a complete list of the individual metrics included in each category, see Metrics glossary.

**Summary metrics**

Summary metrics provide general insights about your S3 storage, including your total storage bytes and object count.

**Cost-optimization metrics**

Cost-optimization metrics provide insights that you can use to manage and optimize your storage costs. For example, you can identify buckets that have incomplete multipart uploads that are more than 7-days old.

With advanced metrics and recommendations, you can enable advanced cost-optimization metrics. These metrics include S3 Lifecycle rule count metrics that you can use to get per-bucket expiration and transition S3 Lifecycle rule counts.

**Data-protection metrics**

Data-protection metrics provide insights for data-protection features, such as encryption and S3 Versioning. You can use these metrics to identify buckets that are not following data-protection best practices. For example, you can identify buckets that are not using default encryption with AWS Key Management Service keys (SSE-KMS) or S3 Versioning.

With advanced metrics and recommendations, you can enable advanced data-protection metrics. These metrics include per-bucket replication rule count metrics.

**Access-management metrics**

Access-management metrics provide insights for S3 Object Ownership. You can use these metrics to see which Object Ownership settings your buckets use.

**Event metrics**

Event metrics provide insights for S3 Event Notifications. With event metrics, you can see which buckets have S3 Event Notifications configured.

**Performance metrics**

Performance metrics provide insights for S3 Transfer Acceleration. With performance metrics, you can see which buckets have Transfer Acceleration enabled.

**Activity metrics (advanced)**

If you upgrade your dashboard to **Advanced metrics and recommendations**, you can enable activity metrics. Activity metrics provide details about how your storage is requested (for example, All requests, Get requests, Put requests), Bytes uploaded or downloaded, and errors.

Prefix-level activity metrics can be used to help you determine which prefixes are being used infrequently, so that you can transition to a more optimal storage class using S3 Lifecycle.

**Detailed status code metrics (advanced)**

If you upgrade your dashboard to **Advanced metrics and recommendations**, you can enable detailed status code metrics. Detailed status code metrics provide insights for HTTP status codes, such as 403 Forbidden and 503 Service Unavailable, that you can use to troubleshoot access or performance issues. For example, you can look at the **403 Forbidden error count** metric to identify workloads that are accessing buckets without the correct permissions applied.

Prefix-level detailed status code metrics can be used to gain a better understanding of the HTTP status code occurrences by prefix. For example, 503 error count metrics enable you to identify prefixes receiving throttling requests during data ingestion.

**Recommendations**

S3 Storage Lens provides automated recommendations to help you optimize your storage. Recommendations are placed contextually alongside relevant metrics in the S3 Storage Lens dashboard. Historical data is not eligible for recommendations because recommendations are relevant to what is happening in the most recent period. Recommendations appear only when they are relevant.

S3 Storage Lens recommendations come in the following forms:

- **Suggestions**

  Suggestions alert you to trends within your storage and activity that might indicate a storage-cost optimization opportunity or a data-protection best practice. You can use the suggested topics in the *Amazon S3 User Guide* and the S3 Storage Lens dashboard to drill down for more details about the specific Regions, buckets, or prefixes.

- **Call-outs**

  Call-outs are recommendations that alert you to interesting anomalies within your storage and activity over a period that might need further attention or monitoring.

  - **Outlier call-outs**

S3 Storage Lens provides call-outs for metrics that are outliers, based on your recent 30-day trend. The outlier is calculated by using a standard score, also known as a *z-score*. In this score, the current day's metric is subtracted from the average of the last 30 days for that metric. The current day's metric is then divided by the standard deviation for that metric over the last 30 days. The resulting score is usually between -3 and +3. This number represents the number of standard deviations that the current day's metric is from the mean.

S3 Storage Lens considers metrics with a score >2 or <-2 to be outliers because they are higher or lower than 95 percent of normally distributed data.

- **Significant change call-outs**

  The significant change call-out applies to metrics that are expected to change less frequently. Therefore, it is set to a higher sensitivity than the outlier calculation, which is typically in the range of +/- 20 percent versus the prior day, week, or month.

  **Addressing call-outs in your storage and activity** – If you receive a significant change call-out, it's not necessarily a problem. The call-out could be the result of an anticipated change in your storage. For example, you might have recently added a large number of new objects, deleted a large number of objects, or made similar planned changes.

  If you see a significant change call-out on your dashboard, take note of it and determine whether it can be explained by recent circumstances. If not, use the S3 Storage Lens dashboard to drill down for more details to understand the specific Regions, buckets, or prefixes that are driving the fluctuation.

- **Reminders**

  Reminders provide insights into how Amazon S3 works. They can help you learn more about ways to use S3 features to reduce storage costs or apply data-protection best practices.

### Metrics selection

S3 Storage Lens offers two metrics selections that you can choose for your dashboard and export: *free metrics* and *advanced metrics and recommendations*.

- **Free metrics**

  S3 Storage Lens offers free metrics for all dashboards and configurations. Free metrics contain metrics that are relevant to your storage, such as the number of buckets and the objects in your

account. Free metrics also include use-case based metrics (for example, cost-optimization and data-protection metrics) that you can use to investigate whether your storage is configured according to S3 best practices. All free metrics are collected daily. Data is available for queries for 14 days. For more information about which metrics are available with free metrics, see the Amazon S3 Storage Lens metrics glossary.

- **Advanced metrics and recommendations**

  S3 Storage Lens offers free metrics for all dashboards and configurations with the option to upgrade to advanced metrics and recommendations. Additional charges apply. For more information, see Amazon S3 pricing.

  Advanced metrics and recommendations include all the metrics in free metrics along with additional metrics, such as advanced data-protection and cost-optimization metrics, activity metrics, and detailed status-code metrics. Advanced metrics and recommendations also provide recommendations to help you optimize your storage. Recommendations are placed contextually alongside relevant metrics in the dashboard.

  Advanced metrics and recommendations include the following features:

  - **Advanced metrics** – Generate additional metrics. For a complete list of advanced metric categories, see Metrics categories. For a complete list of metrics, see the Amazon S3 Storage Lens metrics glossary.
  - **Amazon CloudWatch publishing** – Publishes S3 Storage Lens metrics to CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch API operations and features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. For more information, see Monitor S3 Storage Lens metrics in CloudWatch.
  - **Prefix aggregation** – Collects metrics at the prefix level. Enabling prefix aggregation extends all metrics that are included in your dashboard configuration at the prefix level. Metrics are only generated for prefixes that meet the configured threshold. Note that metrics that are applicable at the prefix level are available with **Prefix aggregation**, except for bucket-level settings and rule count metrics. Prefix-level metrics are not published to CloudWatch.
  - **Storage Lens group aggregation** – Collects metrics at the Storage Lens group level. After you enable **Advanced metrics and recommendations** and **Storage Lens group aggregation**, you can specify which Storage Lens groups to include or exclude from your Storage Lens dashboard. At least one Storage Lens group must be specified. Storage Lens groups that are

specified must also reside within the designated home Region in the dashboard account. Storage Lens group-level metrics are not published to CloudWatch.

All advanced metrics are collected daily. Data is available for querying for up to 15 months. For more information about the storage metrics that are aggregated by S3 Storage Lens, see Amazon S3 Storage Lens metrics glossary.

> **Note**
>
> Recommendations are available only when you use the S3 Storage Lens dashboard on the Amazon S3 console.

### S3 Storage Lens and AWS Organizations

AWS Organizations is an AWS service that helps you aggregate all of your AWS accounts under one organization hierarchy. Amazon S3 Storage Lens works with AWS Organizations to provide a single view of object storage and activity across your Amazon S3 storage.

For more information, see Using Amazon S3 Storage Lens with AWS Organizations.

- **Trusted access**

  Using your organization's management account, you must enable *trusted access* for S3 Storage Lens to aggregate storage metrics and usage data for all member accounts in your organization. You can then create dashboards or exports for your organization by using your management account or by giving delegated administrator access to other accounts in your organization.

  You can disable trusted access for S3 Storage Lens at any time, which stops S3 Storage Lens from aggregating metrics for your organization.

- **Delegated administrator**

  You can create dashboards and metrics for S3 Storage Lens for your organization by using your AWS Organizations management account, or by giving *delegated administrator* access to other accounts in your organization. You can deregister delegated administrators at any time. Deregistering a delegated administrator also automatically stops all organization-level dashboards created by that delegated administrator from aggregating new storage metrics.

For more information, see Amazon S3 Storage Lens and AWS Organizations in the *AWS Organizations User Guide*.

**Amazon S3 Storage Lens service-linked roles**

Along with AWS Organizations trusted access, Amazon S3 Storage Lens uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to S3 Storage Lens. Service-linked roles are predefined by S3 Storage Lens and include all the permissions that it requires to collect daily storage and activity metrics from member accounts in your organization.

For more information, see Using service-linked roles for Amazon S3 Storage Lens.

## Using Amazon S3 Storage Lens with AWS Organizations

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

You can use Amazon S3 Storage Lens to collect storage metrics and usage data for all AWS accounts that are part of your AWS Organizations hierarchy. To do this, you must be using AWS Organizations, and you must enable S3 Storage Lens trusted access by using your AWS Organizations management account.

After enabling trusted access, you can add delegated administrator access to accounts in your organization. These accounts can then create S3 Storage Lens configurations and dashboards that collect organization-wide storage metrics and user data.

For more information about enabling trusted access, see Amazon S3 Storage Lens and AWS Organizations in the *AWS Organizations User Guide*.

**Topics**

- Enabling trusted access for S3 Storage Lens
- Disabling trusted access for S3 Storage Lens
- Registering a delegated administrator for S3 Storage Lens
- Deregistering a delegated administrator for S3 Storage Lens

## Enabling trusted access for S3 Storage Lens

By enabling trusted access, you allow Amazon S3 Storage Lens to have access to your AWS Organizations hierarchy, membership, and structure through AWS Organizations API operations. S3 Storage Lens then becomes a trusted service for your entire organization's structure.

Whenever a dashboard configuration is created, S3 Storage Lens creates service-linked roles in your organization's management or delegated administrator accounts. The service-linked role grants S3 Storage Lens permission to do the following:

- Describe organizations
- List accounts
- Verify a list of AWS service access for the organizations
- Get delegated administrators for the organizations

S3 Storage Lens can then ensure that it has access to collect the cross-account metrics for the accounts in your organizations. For more information, see Using service-linked roles for Amazon S3 Storage Lens.

After enabling trusted access, you can assign delegated administrator access to accounts in your organization. When an account is marked as a delegated administrator for a service, the account receives authorization to access all read-only organization API operations. This access provides the delegated administrator visibility to the members and structures of your organization so that they too can create S3 Storage Lens dashboards.

> **Note**
>
> Only the management account can enable trusted access for Amazon S3 Storage Lens.

## Disabling trusted access for S3 Storage Lens

By disabling trusted access, you limit S3 Storage Lens to working only on an account level. In addition, each account holder can see only the S3 Storage Lens information for the scope of their account, and not their entire organization. Any dashboards that require trusted access are no longer updated, but will retain their historic data for the period that data is available for queries.

**Note**

- Disabling trusted access for S3 Storage Lens also automatically stops all organization-level dashboards from collecting and aggregating storage metrics.
- Your management and delegated administrator accounts will still be able to see the historic data for your existing organization-level dashboards during the period that data is available for queries.

## Registering a delegated administrator for S3 Storage Lens

You can create organization-level dashboards by using your organization's management account or delegated administrator accounts. Delegated administrator accounts allow other accounts besides your management account to create organization-level dashboards. Only the management account of an organization can register and deregister other accounts as delegated administrators for the organization.

To register a delegated administrator by using the Amazon S3 console, see Registering delegated administrators for S3 Storage Lens.

You can also register a delegated administrator by using the AWS Organizations REST API, AWS CLI, or SDKs from the management account. For more information, see RegisterDelegatedAdministrator in the *AWS Organizations API Reference*.

**Note**

Before you can designate a delegated administrator by using the AWS Organizations REST API, AWS CLI, or SDKs, you must call the EnableAWSOrganizationsAccess operation.

### Deregistering a delegated administrator for S3 Storage Lens

You can also deregister a delegated administrator account. Delegated administrator accounts allow other accounts besides your management account to create organization-level dashboards. Only the management account of an organization can deregister accounts as delegated administrators for the organization.

To deregister a delegated administrator by using the S3 console, see Deregistering delegated administrators for S3 Storage Lens.

You can also deregister a delegated administrator by using the AWS Organizations REST API, AWS CLI, or SDKs from the management account. For more information, see DeregisterDelegatedAdministrator in the *AWS Organizations API Reference*.

> **Note**
>
> - Deregistering a delegated administrator also automatically stops all organization-level dashboards created by that delegated administrator from aggregating new storage metrics.
> - The deregistered delegated administrator will still be able to see the historic data for the dashboards that they created while data is available for queries.

## Amazon S3 Storage Lens permissions

Amazon S3 Storage Lens requires new permissions in AWS Identity and Access Management (IAM) to authorize access to S3 Storage Lens actions. To grant these permissions, you can use an identity-based IAM policy. You can attach this policy to IAM users, groups, or roles to grant them permissions. Such permissions can include the ability to enable or disable S3 Storage Lens, or to access any S3 Storage Lens dashboard or configuration.

The IAM user or role must belong to the account that created or owns the dashboard or configuration, unless both of the following conditions are true:

- Your account is a member of AWS Organizations.
- You were given access to create organization-level dashboards by your management account as a delegated administrator.

**Note**

- You can't use your account's root user credentials to view Amazon S3 Storage Lens dashboards. To access S3 Storage Lens dashboards, you must grant the required IAM permissions to a new or existing IAM user. Then, sign in with those user credentials to access S3 Storage Lens dashboards. For more information, see Security best practices in IAM in the *IAM User Guide*.
- Using S3 Storage Lens on the Amazon S3 console can require multiple permissions. For example, to edit a dashboard on the console, you need the following permissions:
  - `s3:ListStorageLensConfigurations`
  - `s3:GetStorageLensConfiguration`
  - `s3:PutStorageLensConfiguration`

**Topics**



## Setting account permissions to use S3 Storage Lens

To create and manage S3 Storage Lens dashboards and Storage Lens dashboard configurations, you must have the following permissions, depending on which actions you want to perform:

**Amazon S3 Storage Lens related IAM permissions**

| Action | IAM permissions |
| --- | --- |
| Create or update an S3 Storage Lens dashboard in the Amazon S3 console. | `s3:ListStorageLensConfigurations`<br>`s3:GetStorageLensConfiguration`<br>`s3:GetStorageLensConfigurationTagging`<br>`s3:PutStorageLensConfiguration` |

| Action | IAM permissions |
| --- | --- |
| | `s3:PutStorageLensConfigurationTagging` |
| Get the tags of an S3 Storage Lens dashboard on the Amazon S3 console. | `s3:ListStorageLensConfigurations`<br>`s3:GetStorageLensConfigurationTagging` |
| View an S3 Storage Lens dashboard on the Amazon S3 console. | `s3:ListStorageLensConfigurations`<br>`s3:GetStorageLensConfiguration`<br>`s3:GetStorageLensDashboard` |
| Delete an S3 Storage Lens dashboard on Amazon S3 console. | `s3:ListStorageLensConfigurations`<br>`s3:GetStorageLensConfiguration`<br>`s3:DeleteStorageLensConfiguration` |
| Create or update an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK. | `s3:PutStorageLensConfiguration`<br>`s3:PutStorageLensConfigurationTagging` |
| Get the tags of an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK. | `s3:GetStorageLensConfigurationTagging` |
| View an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK. | `s3:GetStorageLensConfiguration` |
| Delete an S3 Storage Lens configuration by using the AWS CLI or AWS SDK. | `s3:DeleteStorageLensConfiguration` |

**Note**

- S3 Storage Lens dashboard views are logged in CloudTrail with the event name `GetStorageLensDashboardDataInternal`.
- You can use resource tags in an IAM policy to manage permissions.
- An IAM user or role with these permissions can see metrics from buckets and prefixes that they might not have direct permission to read or list objects from.
- For S3 Storage Lens dashboards with prefix-level metrics enabled, if a selected prefix path matches with an object key, the dashboard might display the object key as another prefix.
- For metrics exports, which are stored in a bucket in your account, permissions are granted by using the existing `s3:GetObject` permission in the IAM policy. Similarly, for an AWS Organizations entity, the organization's management account or delegated administrator accounts can use IAM policies to manage access permissions for organization-level dashboard and configurations.

## Setting account permissions to use S3 Storage Lens groups

You can use S3 Storage Lens groups to understand the distribution of your storage within buckets based on prefix, suffix, object tag, object size, or object age. You can attach Storage Lens groups to your dashboards to view their aggregated metrics.

To work with Storage Lens groups, you need certain permissions. For more information, see the section called “Storage Lens groups permissions”.

## Setting permissions to use S3 Storage Lens with AWS Organizations

You can use Amazon S3 Storage Lens to collect storage metrics and usage data for all accounts that are part of your AWS Organizations hierarchy. The following are the actions and permissions related to using S3 Storage Lens with Organizations.

**AWS Organizations related IAM permissions for using S3 Storage Lens**

| Action | IAM Permissions |
| --- | --- |
| Enable trusted access for S3 Storage Lens for your organization. | `organizations:EnableAWSServiceAccess` |
| Disable trusted access for S3 Storage Lens for your organization. | `organizations:DisableAWSServiceAccess` |
| Register a delegated administrator to create S3 Storage Lens dashboards or configurations for your organization. | `organizations:RegisterDelegatedAdministrator` |
| Deregister a delegated administrator so that they can no longer create S3 Storage Lens dashboards or configurations for your organization. | `organizations:DeregisterDelegatedAdministrator` |
| Additional permissions to create S3 Storage Lens organization-wide configurations. | `organizations:DescribeOrganization`<br>`organizations:ListAccounts`<br>`organizations:ListAWSServiceAccessForOrganization`<br>`organizations:ListDelegatedAdministrators`<br>`iam:CreateServiceLinkedRole` |

## Viewing metrics with Amazon S3 Storage Lens

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account,

AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

By default, all dashboards are configured with free metrics, which include metrics that you can use to understand usage and activity across your S3 storage, optimize your storage costs, and implement data-protection and access-management best practices. Free metrics are aggregated down to the bucket level. With free metrics, data is available for queries for up to 14 days.

Advanced metrics and recommendations include the following additional features that you can use to gain further insight into usage and activity across your storage and best practices for optimizing your storage:

- Contextual recommendations (available only in the dashboard)
- Advanced metrics (including activity metrics aggregated by bucket)
- Prefix aggregation
- Storage Lens group aggregation
- Storage Lens group aggregation
- Amazon CloudWatch publishing

Advanced metrics data is available for queries for 15 months. There are additional charges for using S3 Storage Lens with advanced metrics. For more information, see Amazon S3 pricing. For more information about free and advanced metrics, see Metrics selection.

**Topics**



## Viewing S3 Storage Lens metrics on the dashboards

In the Amazon S3 console, S3 Storage Lens provides an interactive default dashboard that you can use to visualize insights and trends in your data. You can also use this dashboard to flag outliers and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate insights at the account, bucket, AWS Region, prefix, or Storage Lens group level. If you've enabled S3 Storage Lens to work with AWS Organizations, you can also generate insights at the organization level (such as data for all

accounts that are part of your AWS Organizations hierarchy). The dashboard always loads for the latest date that has metrics available.

The S3 Storage Lens default dashboard on the console is named **default-account-dashboard**. Amazon S3 pre-configures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. You can't modify the configuration scope of the default dashboard, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations. With advanced metrics and recommendations, you can access additional metrics and features. These features include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing.

You can disable the default dashboard, but you can't delete it. If you disable your default dashboard, it is no longer updated. You also will no longer receive any new daily metrics in S3 Storage Lens or in the **Account snapshot** section on the **Buckets** page. You can still see historic data in the default dashboard until the 14-day period for data queries expires. This period is 15 months if you've enabled advanced metrics and recommendations. To access this data, you can re-enable the default dashboard within the expiration period.

You can create additional S3 Storage Lens dashboards and scope them by AWS Regions, S3 buckets, or accounts. You can also scope your dashboards by organization if you've enabled Storage Lens to work with AWS Organizations. When you create or edit an S3 Storage Lens dashboard, you define your dashboard scope and metrics selection.

You can disable or delete any additional dashboards that you create.

- If you disable a dashboard, it is no longer updated, and you will no longer receive any new daily metrics. You can still see historic data for free metrics until the 14-day expiration period. If you enabled advanced metrics and recommendations for that dashboard, this period is 15 months. To access this data, you can re-enable the dashboard within the expiration period.
- If you delete your dashboard, you lose all your dashboard configuration settings. You will no longer receive any new daily metrics, and you also lose access to the historical data associated with that dashboard. If you want to access the historic data for a deleted dashboard, you must create another dashboard with the same name in the same home Region.

**Topics**

### Viewing an Amazon S3 Storage Lens dashboard

The following procedure shows how to view an S3 Storage Lens dashboard in the S3 console. For use-case based walkthroughs that show how to use your dashboard to optimize costs, implement best practices, and improve the performance of applications that access your S3 buckets, see Amazon S3 Storage Lens metrics use cases.

> **Note**
>
> You can't use your account's root user credentials to view Amazon S3 Storage Lens dashboards. To access S3 Storage Lens dashboards, you must grant the required AWS Identity and Access Management (IAM) permissions to a new or existing IAM user. Then, sign in with those user credentials to access S3 Storage Lens dashboards. For more information, see Amazon S3 Storage Lens permissions and Security best practices in IAM in the *IAM User Guide*.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

   Your dashboard opens in S3 Storage Lens. The **Snapshot for *date*** section shows the latest date that S3 Storage Lens has collected metrics for. Your dashboard always loads the latest date that has metrics available.
4. (Optional) To change the date for your S3 Storage Lens dashboard, in the top-right date selector, choose a new date.
5. (Optional) To apply temporary filters to further limit the scope of your dashboard data, do the following:

   a. Expand the **Filters** section.

   b. To filter by specific accounts, AWS Regions, storage classes, buckets, prefixes, or Storage Lens groups, choose the options to filter by.

> **Note**
>
> The **Prefixes** filter and the **Storage Lens groups** filter can't be applied at the same time.

c. To update a filter, choose **Apply**.

d. To remove a filter, click on the **X** next to the filter.

6. In any section in your S3 Storage Lens dashboard, to see data for a specific metric, for **Metric**, choose the metric name.
7. In any chart or visualization in your S3 Storage Lens dashboard, you can drill down into deeper levels of aggregation by using the **Accounts**, **AWS Regions**, **Storage classes**, **Buckets**, **Prefixes**, or **Storage Lens groups** tabs. For an example, see Uncover cold Amazon S3 buckets.

### Understanding your S3 Storage Lens dashboard

Your S3 Storage Lens dashboard has a primary **Overview** tab, and up to five additional tabs that represent each aggregation level:

- **Accounts**
- **AWS Regions**
- **Storage classes**
- **Buckets**
- **Prefixes**
- **Storage Lens groups**

On the **Overview** tab, your dashboard data is aggregated into three different sections: **Snapshot for *date***, **Trends and distributions**, and **Top N overview**.

For more information about your S3 Storage Lens dashboard, see the following sections.

### Snapshot

The **Snapshot for *date*** section shows summary metrics that S3 Storage Lens has aggregated for the date selected. These summary metrics include the following metrics:

- **Total storage** – The total amount of storage used in bytes.

- **Object count** – The total number of objects in your AWS account.
- **Average object size** – The average object size.
- **Active buckets** – The total number of active buckets in active usage with storage > 0 bytes in your account.
- **Accounts** – The number of accounts whose storage is in scope. This value is **1** unless you are using AWS Organizations and your S3 Storage Lens has trusted access with a valid service-linked role. For more information, see Using service-linked roles for Amazon S3 Storage Lens.
- **Buckets** – The total number of buckets in your account.

### Metric data

For each metric that appears in the snapshot, you can see the following data:

- **Metric name** – The name of the metric.
- **Metric category** – The category that the metric is organized into.
- **Total for *date*** – The total count for the date selected.
- **% change** – The percentage change from the last snapshot date.
- **30-day trend** – A trend-line showing the changes for the metric over a 30-day period.
- **Recommendation** – A contextual recommendation based on the data that's provided in the snapshot. Recommendations are available with advanced metrics and recommendations. For more information, see Recommendations.

### Metrics categories

You can optionally update your dashboard **Snapshot for *date*** section to display metrics for other categories. If you want to see snapshot data for additional metrics, you can choose from the following **Metrics categories**:

- **Cost optimization**
- **Data protection**
- **Activity** (available with advanced metrics)
- **Access management**
- **Performance**
- **Events**

The **Snapshot for *date*** section displays only a selection of metrics for each category. To see all metrics for a specific category, choose the metric in the **Trends and distributions** or **Top N overview** sections. For more information about metric categories, see Metrics categories. For a complete list of S3 Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

**Trends and distributions**

The second section of the **Overview** tab is **Trends and distributions**. In the **Trends and distributions** section, you can choose two metrics to compare over a date range that you define. The **Trends and distributions** section shows the relationship between two metrics over time. This section displays charts that you can use to see the **Storage class** and **Region** distribution between the two trends that you are tracking. You can optionally drill down into a data point in one of the charts for deeper analysis.

For a walkthrough that uses the **Trends and distributions** section, see Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS).

**Top N overview**

The third section of the S3 Storage Lens dashboard is **Top N overview** (sorted in ascending or descending order). This section displays your selected metrics across the top number of accounts, AWS Regions, buckets, prefixes, or Storage Lens groups. If you enabled S3 Storage Lens to work with AWS Organizations, you can also see your selected metrics across your organization.

For a walkthrough that uses the **Top N overview** section, see Identify your largest S3 buckets.

**Drill down and analyze by options**

To provide a fluid experience for analysis, the S3 Storage Lens dashboard provides an action menu, which appears when you choose any chart value. To use this menu, choose any chart value to see the associated metrics values, and then choose from two options in the box that appears:

- The **Drill down** action applies the selected value as a filter across all tabs of your dashboard. You can then drill down into that value for deeper analysis.
- The **Analyze by** action takes you to the **Dimension** tab that you select and applies that tab value as a filter. These tabs include **Accounts**, **AWS Regions**, **Storage classes**, **Buckets**, **Prefixes** (for dashboards that have **Advanced metrics** and **Prefix aggregation** enabled), and **Storage Lens groups** (for dashboards that have **Advanced metrics** and **Storage Lens group aggregation** enabled). With **Analyze by**, you can view the data in the context of the new dimension for deeper analysis.

The **Drill down** and **Analyze by** actions might be disabled if the outcome would yield illogical results or would not have any value. Both the **Drill down** and **Analyze by** actions apply filters on top of any existing filters across all tabs of the dashboard. You can also remove the filters as needed.

#### Tabs

The dimension-level tabs provide a detailed view of all values within a particular dimension. For example, the **AWS Regions** tab shows metrics for all AWS Regions, and the **Buckets** tab shows metrics for all buckets. Each dimension tab contains an identical layout consisting of four sections:

- A trend chart that displays your top *N* items within the dimension over the last 30 days for the selected metric. By default, this chart displays the top 10 items, but you can decrease it to at least 3 items or increase it up to 50 items.
- A histogram chart that shows a vertical bar chart for the selected date and metric. If you have a large number of items to display in this chart, you might need to scroll horizontally.
- A bubble analysis chart that plots all items within the dimension. This chart represents the first metric on the x axis and the second metric on the y axis. The third metric is represented by the size of the bubble.
- A metric grid view that contains each item in the dimension listed in rows. The columns represent each available metric, arranged in metrics category tabs for easier navigation.

## Viewing Amazon S3 Storage Lens metrics using a data export

Amazon S3 Storage Lens metrics are generated daily in CSV or Apache Parquet-formatted metrics export files and placed in an S3 bucket in your account. From there, you can ingest the metrics export into the analytics tools of your choice, such as Amazon QuickSight and Amazon Athena, where you can analyze storage usage and activity trends.

#### Topics

### Using an AWS KMS key to encrypt your metrics exports

To grant Amazon S3 Storage Lens permission to encrypt your metrics exports by using a customer managed key, you must use a key policy. To update your key policy so that you can use a KMS key to encrypt your S3 Storage Lens metrics exports, follow these steps.

#### To grant S3 Storage Lens permissions to encrypt data by using your KMS key

1. Sign into the AWS Management Console by using the AWS account that owns the customer managed key.
2. Open the AWS KMS console at https://console.aws.amazon.com/kms.
3. To change the AWS Region, use the **Region selector** in the upper-right corner of the page.
4. In the left navigation pane, choose **Customer managed keys**.
5. Under **Customer managed keys**, choose the key that you want to use to encrypt the metrics exports. AWS KMS keys are Region-specific and must be in the same Region as the metrics export destination S3 bucket.
6. Under **Key policy**, choose **Switch to policy view**.
7. To update the key policy, choose **Edit**.
8. Under **Edit key policy**, add the following key policy to the existing key policy. To use this policy, replace the *`user input placeholders`* with your information.

```
{
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
       "StringEquals": {
           "aws:SourceArn": "arn:aws:s3:us-east-1:source-account-id:storage-
lens/your-dashboard-name",
           "aws:SourceAccount": "source-account-id"
        }
     }
}
```

9. Choose **Save changes**.

For more information about creating customer managed keys and using key policies, see the following topics in the *AWS Key Management Service Developer Guide*:

- Getting started
- Using key policies in AWS KMS

You can also use the AWS KMS PUT key policy API operation (PutKeyPolicy) to copy the key policy to the customer managed keys that you want to use to encrypt the metrics exports by using the REST API, AWS CLI, and SDKs.

**What is an S3 Storage Lens export manifest?**

Given the large amount of data aggregated, an S3 Storage Lens daily metrics export can be split into multiple files. The manifest file `manifest.json` describes where the metrics export files for that day are located. Whenever a new export is delivered, it is accompanied by a new manifest. Each manifest contained in the `manifest.json` file provides metadata and other basic information about the export.

The manifest information includes the following properties:

- `sourceAccountId` – The account ID of the configuration owner.
- `configId` – A unique identifier for the dashboard.
- `destinationBucket` – The destination bucket Amazon Resource Name (ARN) that the metrics export is placed in.
- `reportVersion` – The version of the export.
- `reportDate` – The date of the report.
- `reportFormat` – The format of the report.
- `reportSchema` – The schema of the report.
- `reportFiles` – The actual list of the export report files that are in the destination bucket.

The following is an example of a manifest in a `manifest.json` file for a CSV-formatted export.

```
{
   "sourceAccountId":"123456789012",
```

```
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"CSV",

 "reportSchema":"version_number,configuration_id,report_date,aws_account_number,aws_region,stor
   "reportFiles":[
        {
            "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-
configuration-id/V_1/reports/dt=2020-11-03/a38f6bc4-2e3d-4355-ac8a-e2fdcf3de158.csv",
            "size":1603959,
            "md5Checksum":"2177e775870def72b8d84febe1ad3574"
        }
      ]
}
```

The following is an example of a manifest in a `manifest.json` file for a Parquet-formatted export.

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"Parquet",
   "reportSchema":"message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles":[
      {
         "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-configuration-
id/V_1/reports/dt=2020-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
         "size":14714,
         "md5Checksum":"b5c741ee0251cd99b90b3e8eff50b944"
      }
}
```

You can configure your metrics export to be generated as part of your dashboard configuration in the Amazon S3 console or by using the Amazon S3 REST API, AWS CLI, and SDKs.

### Understanding the Amazon S3 Storage Lens export schema

The following table contains the schema of your S3 Storage Lens metrics export.

| Attribute name | Data type | Column name | Description |
|---|---|---|---|
| VersionNumber | String | version_number | The version of the S3 Storage Lens metrics being used. |
| ConfigurationId | String | configuration_id | The configuration_id of your S3 Storage Lens configuration. |
| ReportDate | String | report_date | The date that the metrics were tracked. |
| AwsAccountNumber | String | aws_account_number | Your AWS account number. |
| AwsRegion | String | aws_region | The AWS Region for which the metrics are being tracked. |
| StorageClass | String | storage_class | The storage class of the bucket in question. |
| RecordType | ENUM | record_type | The type of artifact that is being reported (ACCOUNT, BUCKET, or PREFIX). |
| RecordValue | String | record_value | The value of the RecordType artifact. |

| Attribute name | Data type | Column name | Description |
| --- | --- | --- | --- |
| | | | **Note** The `record_value` is URL-encoded. |
| `BucketName` | String | `bucket_name` | The name of the bucket that is being reported. |
| `MetricName` | String | `metric_name` | The name of the metric that is being reported. |
| `MetricValue` | Long | `metric_value` | The value of the metric that is being reported. |

### Example of an S3 Storage Lens metrics export

The following is an example of an S3 Storage Lens metrics export based on this schema.

**Note**

You can identify metrics for Storage Lens groups by looking for the `STORAGE_LENS_GROUP_BUCKET` or `STORAGE_LENS_GROUP_ACCOUNT` values in the `record_type` column. The `record_value` column will display the Amazon Resource Name (ARN) for the Storage Lens group, for example, `arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1`.

| version_r | configuration_id | report_date | aws_account_number | aws_region | storage_class | record_type | record_value | bucket_name | metric_name | metric_value |
|---|---|---|---|---|---|---|---|---|---|---|
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | StorageBytes | 2478830621 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | ObjectCount | 1598962 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | ReplicatedStorageBytes | 20000 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | ReplicatedObjectCount | 20 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | EncryptedStorageBytes | 2478828742 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | EncryptedObjectCount | 1598961 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | DeleteMarkerObjectCount | 1500 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | ObjectLockEnabledStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | ObjectLockEnabledObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | CurrentVersionStorageBytes | 2478830621 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | CurrentVersionObjectCount | 1598962 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | NonCurrentVersionStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | NonCurrentVersionObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | IncompleteMultipartUploadStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | eu-west-1 | STANDARD | ACCOUNT | | | IncompleteMultipartUploadObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | StorageBytes | 29996800 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | ObjectCount | 12264 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | ReplicatedStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | ReplicatedObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | EncryptedStorageBytes | 29996800 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | EncryptedObjectCount | 12264 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | DeleteMarkerObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | ObjectLockEnabledStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | ObjectLockEnabledObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | CurrentVersionStorageBytes | 29996800 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | CurrentVersionObjectCount | 12264 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | NonCurrentVersionStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | NonCurrentVersionObjectCount | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | IncompleteMultipartUploadStorageBytes | 0 |
| V_1 | sample-cmh-exclude | 11/3/2020 | 546264889236 | us-west-1 | STANDARD | PREFIX | AWSLogs%2F546264889236%2FCloudTrail%2Fus | cloudtrail-log-sf | IncompleteMultipartUploadObjectCount | 0 |

The following is an example of an S3 Storage Lens metrics export with Storage Lens groups data.

| version_numbe | configuration_id | report_date | aws_account_num | aws_region | storage_class | record_type | record_value | bucket_name | metric_name | metric_value |
|---|---|---|---|---|---|---|---|---|---|---|
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | StorageBytes | 3128548856 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ObjectCount | 4440 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ReplicatedStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ReplicatedObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | EncryptedStorageBytes | 3128548856 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | EncryptedObjectCount | 4440 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | DeleteMarkerObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ObjectLockEnabledStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ObjectLockEnabledObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | CurrentVersionStorageBytes | 3128548856 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | CurrentVersionObjectCount | 4440 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | NonCurrentVersionStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | NonCurrentVersionObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | IncompleteMultipartUploadStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | IncompleteMultipartUploadObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | DeleteMarkerStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ReplicatedStorageBytesSource | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | ReplicatedObjectCountSource | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | IncompleteMPUStorageBytesOlderThan7Days | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_ACCOUNT | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | | IncompleteMPUObjectCountOlderThan7Days | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | StorageBytes | 676863200 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ObjectCount | 3000 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ReplicatedStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ReplicatedObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | EncryptedStorageBytes | 676863200 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | EncryptedObjectCount | 3000 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | DeleteMarkerObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ObjectLockEnabledStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ObjectLockEnabledObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | CurrentVersionStorageBytes | 676863200 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | CurrentVersionObjectCount | 3000 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | NonCurrentVersionStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | NonCurrentVersionObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | IncompleteMultipartUploadStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | IncompleteMultipartUploadObjectCount | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | DeleteMarkerStorageBytes | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ReplicatedStorageBytesSource | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | ReplicatedObjectCountSource | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | IncompleteMPUStorageBytesOlderThan7Days | 0 |
| V_1 | sample-sfo-exclude | 11/15/23 | 546264889236 | us-west-1 | STANDARD | STORAGE_LENS_GROUP_BUCKET | arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180 | cloud-trail-log-sfo | IncompleteMPUObjectCountOlderThan7Days | 0 |

## Monitor S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

You can enable the CloudWatch publishing option for new or existing dashboard configurations by using the Amazon S3 console, Amazon S3 REST API, AWS CLI, and AWS SDKs. Dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations can use the CloudWatch publishing option. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply. For more information, see Amazon CloudWatch pricing.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics and recommendations, you can access organization, account, and bucket-level metrics in CloudWatch. Prefix-level metrics are not available in CloudWatch.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.

After you enable the CloudWatch publishing option, you can use the following CloudWatch features to monitor and analyze your S3 Storage LensStorage Lens data:

- Dashboards – Use CloudWatch dashboards to create customized S3 Storage Lens dashboards. Share your CloudWatch dashboard with people who don't have direct access to your AWS account, across teams, with stakeholders, and with people external to your organizations.
- Alarms and triggered actions – Configure alarms that watch metrics and take action when a threshold is breached. For example, you can configure an alarm that sends an Amazon SNS notification when the **Incomplete Multipart Upload Bytes** metric exceeds 1 GB for three consecutive days.
- Anomaly detection – Enable anomaly detection to continuously analyze metrics, determine normal baselines, and surface anomalies. You can create an anomaly detection alarm based on

the expected value of a metric. For example, you can monitor anomalies for the **Object Lock Enabled Bytes** metric to detect unauthorized removal of Object Lock settings.

- Metric math – You can also use metric math to query multiple S3 Storage Lens metrics and use math expressions to create new time series based on these metrics. For example, you can create a new metric to get the average object size by dividing `StorageBytes` by `ObjectCount`.

For more information about the CloudWatch publishing option for S3 Storage Lens metrics, see the following topics.

**Topics**

- S3 Storage Lens metrics and dimensions
- Enabling CloudWatch publishing for S3 Storage Lens
- Working with S3 Storage Lens metrics in CloudWatch

### S3 Storage Lens metrics and dimensions

To send S3 Storage Lens metrics to CloudWatch, you must enable the CloudWatch publishing option within S3 Storage Lens advanced metrics and recommendations. After advanced metrics are enabled, you can use CloudWatch dashboards to monitor S3 Storage Lens metrics alongside other application metrics and create a unified view of your operational health. You can use dimensions to filter your S3 Storage Lens metrics in CloudWatch by organization, account, bucket, storage class, Region, and metrics configuration ID.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics and recommendations, you can access organization, account, and bucket-level metrics in CloudWatch. Prefix-level metrics are not available in CloudWatch.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for

S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.

For more information about S3 Storage Lens metrics and dimensions in CloudWatch, see the following topics.

**Topics**

- Metrics
- Dimensions

### Metrics

S3 Storage Lens metrics are available as metrics within CloudWatch. S3 Storage Lens metrics are published to the `AWS/S3/Storage-Lens` namespace. This namespace is only for S3 Storage Lens metrics. Amazon S3 bucket, request, and replication metrics are published to the `AWS/S3` namespace.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics and recommendations, you can access organization, account, and bucket-level metrics in CloudWatch. Prefix-level metrics are not available in CloudWatch.

In S3 Storage Lens, metrics are aggregated and stored only in the designated home Region. S3 Storage Lens metrics are also published to CloudWatch in the home Region that you specify in the S3 Storage Lens configuration.

For a complete list of S3 Storage Lens metrics, including a list of those metrics available in CloudWatch, see Amazon S3 Storage Lens metrics glossary.

**Note**

The valid statistic for S3 Storage Lens metrics in CloudWatch is Average. For more information about statistics in CloudWatch, see CloudWatch statistics definitions in the *Amazon CloudWatch User Guide*.

### Granularity of S3 Storage Lens metrics in CloudWatch

S3 Storage Lens offers metrics at organization, account, bucket, and prefix granularity. S3 Storage Lens publishes organization, account, and bucket-level S3 Storage Lens metrics to CloudWatch. Prefix-level S3 Storage Lens metrics are not available in CloudWatch.

For more information about the granularity of S3 Storage Lens metrics available in CloudWatch, see the following list:

- **Organization** – Metrics aggregated across the member accounts in your organization. S3 Storage Lens publishes metrics for member accounts to CloudWatch in the management account.
  - **Organization and account** – Metrics for the member accounts in your organization.
  - **Organization and bucket** – Metrics for Amazon S3 buckets in the member accounts of your organization.
- **Account** (Non-organization level) – Metrics aggregated across the buckets in your account.
- **Bucket** (Non-organization level) – Metrics for a specific bucket. In CloudWatch, S3 Storage Lens publishes these metrics to the AWS account that created the S3 Storage Lens configuration. S3 Storage Lens publishes these metrics only for non-organization configurations.

### Dimensions

When S3 Storage Lens sends data to CloudWatch, dimensions are attached to each metric. Dimensions are categories that describe the characteristics of metrics. You can use dimensions to filter the results that CloudWatch returns.

For example, all S3 Storage Lens metrics in CloudWatch have the `configuration_id` dimension. You can use this dimension to differentiate between metrics associated with a specific S3 Storage Lens configuration. The `organization_id` identifies organization-level metrics. For more information about dimensions in CloudWatch, see Dimensions in the *CloudWatch User Guide*.

Different dimensions are available for S3 Storage Lens metrics depending on the granularity of the metrics. For example, you can use the `organization_id` dimension to filter organization-level metrics by the AWS Organizations ID. However, you can't use this dimension for bucket and account-level metrics. For more information, see Filtering metrics using dimensions.

To see which dimensions are available for your S3 Storage Lens configuration, see the following table.

| Dimension | Description | Bucket | Account | Organization | Organization and account | Organization and bucket |
|---|---|---|---|---|---|---|
| `configuration_id` | The dashboard name for the S3 Storage Lens configuration reported in the metrics | | | | | |
| `metrics_version` | The version of the S3 Storage Lens metrics. The metrics version has a fixed value of `1.0`. | | | | | |
| `organization_id` | The AWS Organizations ID for the metrics | | | | | |
| `aws_account_number` | The AWS account that's associated with the metrics | | | | | |
| `aws_region` | The AWS Region for the metrics | | | | | |
| `bucket_name` | The name of the S3 bucket that's reported in the metrics | | | | | |
| `storage_class` | The storage class for the bucket that's reported in the metrics | | | | | |
| `record_type` | The granularity of the metrics: ORGANIZATION, ACCOUNT, BUCKET | BUCKET | ACCOUNT | ORGANIZATION | | |

### Enabling CloudWatch publishing for S3 Storage Lens

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications,

including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics and recommendations, you can access organization, account, and bucket-level metrics in CloudWatch. Prefix-level metrics are not available in CloudWatch.

You can enable CloudWatch support for new or existing dashboard configurations by using the S3 console, Amazon S3 REST APIs, AWS CLI, and AWS SDKs. The CloudWatch publishing option is available for dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply.

To enable the CloudWatch publishing option for S3 Storage Lens metrics, see the following topics.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.
> Currently, S3 Storage Lens metrics cannot be consumed through CloudWatch streams.

### Using the S3 console

When you update an S3 Storage Lens dashboard, you can't change the dashboard name or home Region. You also can't change the scope of the default dashboard, which is scoped to your entire account's storage.

**To update an S3 Storage Lens dashboard to enable CloudWatch publishing**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **S3 Storage Lens**, **Dashboards**.
3. Choose the dashboard that you want to edit, and then choose **Edit.**
4. Under **Metrics selection**, choose **Advanced metrics and recommendations**.

   Advanced metrics and recommendations are available for an additional charge. Advanced metrics and recommendations include a 15-month period for data queries, usage metrics aggregated at the prefix level, activity metrics aggregated by bucket, the CloudWatch publishing option, and contextual recommendations that help you optimize storage costs and apply data-protection best practices. For more information, see Amazon S3 pricing.
5. Under **Select Advanced metrics and recommendations features**, select **CloudWatch publishing**.

   > **Important**
   >
   > If your configuration enables prefix aggregation for usage metrics, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.
6. Choose **Save changes**.

**To create a new S3 Storage Lens dashboard that enables CloudWatch support**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. Choose **Create dashboard**.
4. Under **General**, define the following configuration options:

   a. For **Dashboard name**, enter your dashboard name.

      Dashboard names must be fewer than 65 characters and must not contain special characters or spaces. You can't change the dashboard name after you create your dashboard.

   b. Choose the **Home Region** for your dashboard.

Metrics for all Regions included in this dashboard scope are stored centrally in the designated home Region. In CloudWatch, S3 Storage Lens metrics are also available in the home Region. You can't change the home Region after you create your dashboard.

5. (Optional) To add tags, choose **Add tag** and enter the tag **Key** and **Value**.

> **Note**
>
> You can add up to 50 tags to your dashboard configuration.

6. Define the scope for your configuration:

   a. If you're creating an organization-level configuration, choose the accounts to include in the configuration: **Include all accounts in your configuration** or **Limit the scope to your signed-in account**.

   > **Note**
   >
   > When you create an organization-level configuration that includes all accounts, you can include or exclude only Regions, not buckets.

   b. Choose the Regions and buckets that you want S3 Storage Lens to include in the dashboard configuration by doing the following:

   - To include all Regions, choose **Include Regions and buckets**.
   - To include specific Regions, clear **Include all Regions**. Under **Choose Regions to include**, choose the Regions that you want S3 Storage Lens to include in the dashboard.
   - To include specific buckets, clear **Include all buckets**. Under **Choose buckets to include**, choose the buckets that you want S3 Storage Lens to include in the dashboard.

   > **Note**
   >
   > You can choose up to 50 buckets.

7. For **Metrics selection**, choose **Advanced metrics and recommendations**.

   For more information about advanced metrics and recommendations pricing, see Amazon S3 pricing.

8. Under **Advanced metrics and recommendations features**, select the options that you want to enable:

   - **Advanced metrics**
   - **CloudWatch publishing**

     > **Important**
     >
     > If you enable prefix aggregation for your S3 Storage Lens configuration, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.

   - **Prefix aggregation**

   > **Note**
   >
   > For more information about advanced metrics and recommendations features, see Metrics selection.

9. If you enabled **Advanced metrics**, select the **Advanced metrics categories** that you want to display in your S3 Storage Lens dashboard:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Advanced cost optimization metrics**
   - **Advanced data protection metrics**

   For more information about metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

10. (Optional) Configure your metrics export.

    For more information about how to configure a metrics export, see step Creating an Amazon S3 Storage Lens dashboard.

11. Choose **Create dashboard**.

### Using the AWS CLI

The following AWS CLI example enables the CloudWatch publishing option by using a S3 Storage Lens organization-level advanced metrics and recommendations configuration. To use this example, replace the *user input placeholders* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
```

```
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true, //Mark this as false if you want only free metrics.
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
 configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
 metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true //Mark this as false if you want to export only free metrics.
    }
  }
}
```

### Using the AWS SDK for Java

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
```

```
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
```

```
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Using the REST API

To enable the CloudWatch publishing option by using the Amazon S3 REST API, you can use PutStorageLensConfiguration.

### Next steps

After you enable the CloudWatch publishing option, you can access your S3 Storage Lens metrics in CloudWatch. You also can leverage CloudWatch features to monitor and analyze your S3 Storage Lens data in CloudWatch. For more information, see the following topics:

- S3 Storage Lens metrics and dimensions
- Working with S3 Storage Lens metrics in CloudWatch

## Working with S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

You can enable the CloudWatch publishing option for new or existing dashboard configurations by using the Amazon S3 console, Amazon S3 REST APIs, AWS CLI, and AWS SDKs. The CloudWatch publishing option is available for dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply. For more information, see Amazon CloudWatch pricing.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics and recommendations, you can access organization, account, and bucket-level metrics in CloudWatch. Prefix-level metrics are not available in CloudWatch.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must

be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.
Currently, S3 Storage Lens metrics cannot be consumed through CloudWatch streams.

For more information about working with S3 Storage Lens metrics in CloudWatch, see the following topics.

**Topics**

- Working with CloudWatch dashboards
- Setting alarms, triggering actions, and using anomaly detection
- Filtering metrics using dimensions
- Calculating new metrics with metric math
- Using search expressions in graphs

### Working with CloudWatch dashboards

You can use CloudWatch dashboards to monitor S3 Storage Lens metrics alongside other application metrics and create a unified view of your operational health. Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view.

CloudWatch has broad permissions control that doesn't support limiting access to a specific set of metrics or dimensions. Users in your account or organization who have access to CloudWatch will have access to metrics for all S3 Storage Lens configurations where the CloudWatch support option is enabled. You can't manage permissions for specific dashboards as you can in S3 Storage Lens. For more information about CloudWatch permissions, see Managing access permissions to your CloudWatch resources in the *Amazon CloudWatch User Guide*.

For more information about using CloudWatch dashboards and configuring permissions, see Using Amazon CloudWatch dashboards and Sharing CloudWatch dashboards in the *Amazon CloudWatch User Guide*.

### Setting alarms, triggering actions, and using anomaly detection

You can configure CloudWatch alarms that watch S3 Storage Lens metrics in CloudWatch and take action when a threshold is breached. For example, you can configure an alarm that sends an Amazon SNS notification when the **Incomplete Multipart Upload Bytes** metric exceeds 1 GB for three consecutive days.

You can also enable anomaly detection to continuously analyze your S3 Storage Lens metrics, determine normal baselines, and surface anomalies. You can create an anomaly detection alarm based on a metric's expected value. For example, you can monitor anomalies for the **Object Lock Enabled Bytes** metric to detect unauthorized removal of Object Lock settings.

For more information and examples, see Using Amazon CloudWatch alarms and Creating an alarm from a metric on a graph in the *Amazon CloudWatch User Guide*.

### Filtering metrics using dimensions

You can use dimensions to filter S3 Storage Lens metrics in the CloudWatch console. For example, you can filter by `configuration_id`, `aws_account_number`, `aws_region`, `bucket_name`, and more.

S3 Storage Lens supports multiple dashboard configurations per account. This means that different configurations can include the same bucket. When these metrics are published to CloudWatch, the bucket will have duplicate metrics within CloudWatch. To view metrics only for a specific S3 Storage Lens configuration in CloudWatch, you can use the `configuration_id` dimension. When you filter by `configuration_id`, you see only the metrics that are associated with the configuration that you identify.

For more information about filtering by configuration ID, see Searching for available metrics in the *Amazon CloudWatch User Guide*.

### Calculating new metrics with metric math

You can use metric math to query multiple S3 Storage Lens metrics and use math expressions to create new time series based on these metrics. For example, you can create a new metric for unencrypted objects by subtracting Encrypted Objects from Object Count. You can also create a metric to get the average object size by dividing `StorageBytes` by `ObjectCount` or the number bytes accessed on one day by dividing `BytesDownloaded` by `StorageBytes`.

For more information, see Using metric math in the *Amazon CloudWatch User Guide*.

### Using search expressions in graphs

With S3 Storage Lens metrics, you can create a search expression. For example, you can create a search expression for all metrics that are named **IncompleteMultipartUploadStorageBytes** and add SUM to the expression. With this search expression, you can see your total incomplete multipart upload bytes across all dimensions of your storage in a single metric.

This example shows the syntax that you would use to create a search expression for all metrics named **IncompleteMultipartUploadStorageBytes**.

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

For more information about this syntax, see CloudWatch search expression syntax in the *Amazon CloudWatch User Guide*. To create a CloudWatch graph with a search expression, see Creating a CloudWatch graph with a search expressionin the *Amazon CloudWatch User Guide*.

## Amazon S3 Storage Lens metrics use cases

You can use your Amazon S3 Storage Lens dashboard to visualize insights and trends, flag outliers, and receive recommendations. S3 Storage Lens metrics are organized into categories that align with key use cases. You can use these metrics to do the following:

- Identify cost-optimization opportunities
- Apply data-protection best practices
- Apply access-management best practices
- Improve the performance of application workloads

For example, with cost-optimization metrics, you can identify opportunities to reduce your Amazon S3 storage costs. You can identify buckets with multipart uploads that are more than 7-days old or buckets that are accumulating noncurrent versions.

Similarly, you can use data-protection metrics to identify buckets that aren't following data-protection best practices within your organization. For example, you can identify buckets that don’t use AWS Key Management Service keys (SSE-KMS) for default encryption or don't have S3 Versioning enabled.

With S3 Storage Lens access-management metrics, you can identify bucket settings for S3 Object Ownership so that you can migrate access control list (ACL) permissions to bucket policies and disable ACLs.

If you have S3 Storage Lens advanced metrics enabled, you can use detailed status-code metrics to get counts for successful or failed requests that you can use to troubleshoot access or performance issues.

With advanced metrics, you can also access additional cost-optimization and data-protection metrics that you can use to identify opportunities to further reduce your overall S3 storage costs and better align with best practices for protecting your data. For example, advanced cost-optimization metrics include lifecycle rule counts that you can use to identify buckets that don't have lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. Advanced data-protection metrics include replication rule counts.

For more information about metrics categories, see Metrics categories. For a complete list of S3 Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

**Topics**

- Using Amazon S3 Storage Lens to optimize your storage costs
- Using S3 Storage Lens to protect your data
- Using S3 Storage Lens to audit Object Ownership settings
- Using S3 Storage Lens metrics to improve performance

## Using Amazon S3 Storage Lens to optimize your storage costs

You can use S3 Storage Lens cost-optimization metrics to reduce the overall cost of your S3 storage. Cost-optimization metrics can help you confirm that you've configured Amazon S3 cost effectively and according to best practices. For example, you can identify the following cost-optimization opportunities:

- Buckets with incomplete multipart uploads older than 7 days
- Buckets that are accumulating numerous noncurrent versions
- Buckets that don't have lifecycle rules to abort incomplete multipart uploads
- Buckets that don't have lifecycle rules to expire noncurrent versions objects
- Buckets that don't have lifecycle rules to transition objects to a different storage class

You can then use this data to add additional lifecycle rules to your buckets.

The following examples show how you can use cost- optimization metrics in your S3 Storage Lens dashboard to optimize your storage costs.

**Topics**



### Identify your largest S3 buckets

You pay for storing objects in S3 buckets. The rate that you're charged depends on your objects' sizes, how long you store the objects, and their storage classes. With S3 Storage Lens, you get a centralized view of all the buckets in your account. To see all the buckets in all of your organization's accounts, you can configure an AWS Organizations-level S3 Storage Lens dashboard. From this dashboard view, you can identify your largest buckets.

#### Step 1: Identify your largest buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

   When the dashboard opens, you can see the latest date that S3 Storage Lens has collected metrics for. Your dashboard always loads to the latest date that has metrics available.
4. To see a ranking of your largest buckets by the **Total storage** metric for a selected date range, scroll down to the **Top N overview for *date*** section.

   You can toggle the sort order to show the smallest buckets. You can also adjust the **Metric** selection to rank your buckets by any of the available metrics. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

5. For more detailed insights about your buckets, scroll up to the top of the page, and then choose the **Bucket** tab.

   On the **Bucket** tab, you can see details such as the recent growth rate, the average object size, the largest prefixes, and the number of objects.

### Step 2: Navigate to your buckets and investigate

After you've identified your largest S3 buckets, you can navigate to each bucket within the S3 console to view the objects in the bucket, understand its associated workload, and identify its internal owners. You can contact the bucket owners to find out whether the growth is expected or whether the growth needs further monitoring and control.

### Uncover cold Amazon S3 buckets

If you have S3 Storage Lens advanced metrics enabled, you can use activity metrics to understand how cold your S3 buckets are. A "cold" bucket is one whose storage is no longer accessed (or very rarely accessed). This lack of activity typically indicates that the bucket's objects aren't frequently accessed.

Activity metrics, such as **GET Requests** and **Download Bytes**, indicate how often your buckets are accessed each day. To understand the consistency of the access pattern and to spot buckets that are no longer being accessed at all, you can trend this data over several months. The **Retrieval rate** metric, which is computed as **Download bytes / Total storage**, indicates the proportion of storage in a bucket that is accessed daily.

> **Note**
>
> Download bytes are duplicated in cases where the same object is downloaded multiple times during the day.

### Prerequisite

To see activity metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Activity metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

**Step 1: Identify active buckets**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. Choose the **Bucket** tab, and then scroll down to the **Bubble analysis by buckets for *date*** section.

   In the **Bubble analysis by buckets for *date*** section, you can plot your buckets on multiple dimensions by using any three metrics to represent the **X-axis**, **Y-axis**, and **Size** of the bubble.
5. To find buckets that have gone cold, for **X-axis**, **Y-axis**, and **Size**, choose the **Total storage**, **% retrieval rate**, and **Average object size** metrics.
6. In the **Bubble analysis by buckets for *date*** section, locate any buckets with retrieval rates of zero (or near zero) and a larger relative storage size, and choose the bubble that represents the bucket.

   A box will appear with choices for more granular insights. Do one of the following:

   a. To update the **Bucket** tab to display metrics only for the selected bucket, choose **Drill down**, and then choose **Apply**.

   b. To aggregate your bucket-level data to by account, AWS Region, storage class, or bucket, choose **Analyze by** and then make a choice for **Dimension**. For example, to aggregate by storage class, choose **Storage class** for **Dimension**.

   To find buckets that have gone cold, do a bubble analysis using the **Total storage**, **% retrieval rate**, and **Average object size** metrics. Look for any buckets with retrieval rates of zero (or near zero) and a larger relative storage size.

   The **Bucket** tab of your dashboard updates to display data for your selected aggregation or filter. If you aggregated by storage class or another dimension, that new tab opens in your dashboard (for example, the **Storage class** tab).

**Step 2: Investigate cold buckets**

From here, you can identify the owners of cold buckets in your account or organization and find out if that storage is still needed. You can then optimize costs by configuring lifecycle expiration configurations for these buckets or archiving the data in one of the Amazon S3 Glacier storage classes.

To avoid the problem of cold buckets going forward, you can automatically transition your data by using S3 Lifecycle configurations for your buckets, or you can enable auto-archiving with S3 Intelligent-Tiering.

You can also use step 1 to identify hot buckets. Then, you can ensure that these buckets use the correct S3 storage class to ensure that they serve their requests most effectively in terms of performance and cost.

**Locate incomplete multipart uploads**

You can use multipart uploads to upload very large objects (up to 5 TB) as a set of parts for improved throughput and quicker recovery from network issues. In cases where the multipart upload process doesn't finish, the incomplete parts remain in the bucket (in an unusable state). These incomplete parts incur storage costs until the upload process is finished, or until the incomplete parts are removed. For more information, see Uploading and copying objects using multipart upload.

With S3 Storage Lens, you can identify the number of incomplete multipart upload bytes in your account or across your entire organization, including incomplete multipart uploads that are more than 7 days old. For a complete list of incomplete multipart upload metrics, see Amazon S3 Storage Lens metrics glossary.

As a best practice, we recommend configuring lifecycle rules to expire incomplete multipart uploads that are older than a specific number of days. When you create your lifecycle rule to expire incomplete multipart uploads, we recommend 7 days as a good starting point.

**Step 1: Review overall trends for incomplete multipart uploads**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

4. In the **Snapshot for *date*** section, under **Metrics categories**, choose **Cost optimization**.

   The **Snapshot for *date*** section updates to display **Cost optimization** metrics, which include **Incomplete multipart upload bytes greater than 7 days old**.

   In any chart in your S3 Storage Lens dashboard, you can see metrics for incomplete multipart uploads. You can use these metrics to further assess the impact of incomplete multipart upload bytes on your storage, including their contribution to overall growth trends. You can also drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Bucket**, or **Storage class** tabs for a deeper analysis of your data. For an example, see Uncover cold Amazon S3 buckets.

**Step 2: Identify buckets that have the most incomplete multipart upload bytes but don't have lifecycle rules to abort incomplete multipart uploads**

**Prerequisite**

To see the **Abort incomplete multipart upload lifecycle rule count** metric in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. To identify specific buckets that are accumulating incomplete multipart uploads greater than 7 days old, go to the **Top N overview for *date*** section.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. You can increase or decrease the number of buckets in the **Top N** field. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. (This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.)

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

5. For **Metric**, choose **Incomplete multipart upload bytes greater than 7 days old** in the **Cost optimization** category.

   Under **Top *number* buckets**, you can see the buckets with the most incomplete multipart upload storage bytes that are greater than 7 days old.

6. To view more detailed bucket-level metrics for incomplete multipart uploads, scroll to the top of the page, and then choose the **Bucket** tab.

7. Scroll down to the **Buckets** section. For **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.

8. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon
   (  ).

9. Clear the toggles for all cost-optimization metrics until only **Incomplete multipart upload bytes greater than 7 days old** and **Abort incomplete multipart upload lifecycle rule count** remain selected.

10. (Optional) Under **Page size**, choose the number of buckets to display in the list.

11. Choose **Confirm**.

    The **Buckets** list updates to display bucket-level metrics for incomplete multipart uploads and lifecycle rule counts. You can use this data to identify buckets that have the most incomplete multipart upload bytes that are greater than 7 days old and are missing lifecycle rules to abort incomplete multipart uploads. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to delete abandoned incomplete multipart uploads.

### Step 3: Add a lifecycle rule to delete incomplete multipart uploads after 7 days

To automatically manage incomplete multipart uploads, you can use the S3 console to create a lifecycle configuration to expire incomplete multipart upload bytes from a bucket after a specified number of days. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

### Reduce the number of noncurrent versions retained

When enabled, S3 Versioning retains multiple distinct copies of the same object that you can use to quickly recover data if an object is accidentally deleted or overwritten. If you've enabled S3 Versioning without configuring lifecycle rules to transition or expire noncurrent versions, a large number of previous noncurrent versions can accumulate, which can have storage-cost implications. For more information, see Using versioning in S3 buckets.

### Step 1: Identify buckets with the most noncurrent object versions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In the **Snapshot for *date*** section, under **Metric categories**, choose **Cost optimization**.

   The **Snapshot for *date*** section updates to display **Cost optimization** metrics, which include the metric for **% noncurrent version bytes**. The **% noncurrent version bytes** metric represents the proportion of your total storage bytes that is attributed to noncurrent versions, within the dashboard scope and for the selected date.

   > **Note**
   >
   > If your **% noncurrent version bytes** is greater than 10 percent of your storage at the account level, you might be storing too many object versions.

5. To identify specific buckets that are accumulating a large number of noncurrent versions:

   a. Scroll down to the **Top N overview for *date*** section. For **Top N**, enter the number of buckets that you would like to see data for.

   b. For **Metric**, choose **% noncurrent version bytes**.

Under **Top *number* buckets**, you can see the buckets (for the number that you specified) with the highest **% noncurrent version bytes**. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

c. To view more detailed bucket-level metrics for noncurrent object versions, scroll to the top of the page, and then choose the **Bucket** tab.

   In any chart or visualization in your S3 Storage Lens dashboard, you can drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Storage class**, or **Bucket** tabs. For an example, see Uncover cold Amazon S3 buckets.

d. In the **Buckets** section, for **Metric categories**, select **Cost optimization**. Then, clear **Summary**.

   You can now see the **% noncurrent version bytes** metric, along with other metrics related to noncurrent versions.

**Step 2: Identify buckets that are missing transition and expiration lifecycle rules for managing noncurrent versions**

**Prerequisite**

To see the **Noncurrent version transition lifecycle rule count** and **Noncurrent version expiration lifecycle rule count** metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. For **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon (  ).
7. Clear the toggles for all cost-optimization metrics until only the following remain selected:

   - **% noncurrent version bytes**
   - **Noncurrent version transition lifecycle rule count**
   - **Noncurrent version expiration lifecycle rule count**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list updates to display metrics for noncurrent version bytes and noncurrent version lifecycle rule counts. You can use this data to identify buckets that have a high percentage of noncurrent version bytes but are missing transition and expiration lifecycle rules. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to these buckets.

#### Step 3: Add lifecycle rules to transition or expire noncurrent object versions

After you've determined which buckets require further investigation, you can navigate to the buckets within the S3 console and add a lifecycle rule to expire noncurrent versions after a specified number of days. Alternatively, to reduce costs while still retaining noncurrent versions, you can configure a lifecycle rule to transition noncurrent versions to one of the Amazon S3 Glacier storage classes. For more information, see Example 6: Specifying a lifecycle rule for a versioning-enabled bucket.

#### Identify buckets that don't have lifecycle rules and review lifecycle rule counts

S3 Storage Lens provides S3 Lifecycle rule count metrics that you can use to identify buckets that are missing lifecycle rules. To find buckets that don't have lifecycle rules, you can use the **Total**

**buckets without lifecycle rules** metric. A bucket with no S3 Lifecycle configuration might have storage that you no longer need or can migrate to a lower-cost storage class. You can also use lifecycle rule count metrics to identify buckets that are missing specific types of lifecycle rules, such as expiration or transition rules.

**Prerequisite**

To see lifecycle rule count metrics and the **Total buckets without lifecycle rules** metric in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

**Step 1: Identify buckets without lifecycle rules**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. To identify specific buckets without lifecycle rules, scroll down to the **Top N overview for *date*** section.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. In the **Top N** field, you can increase the number of buckets. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

   > **Note**
   >
   > With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

5. For **Metric**, choose **Total buckets without lifecycle rules** from the **Cost optimization** category.
6. Review the following data for **Total buckets without lifecycle rules**:

   - **Top *number* accounts** - See which accounts that have the most buckets without lifecycle rules.
   - **Top *number* Regions** - View a breakdown of buckets without lifecycle rules by Region.

- **Top *number* buckets** - See which buckets don't have lifecycle rules.

In any chart or visualization in your S3 Storage Lens dashboard, you can drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Storage class**, or **Bucket** tabs. For an example, see Uncover cold Amazon S3 buckets.

After you identify which buckets don't have lifecycle rules, you can also review specific lifecycle rule counts for your buckets.

**Step 2: Review lifecycle rule counts for your buckets**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In your S3 Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon (  ).
7. Clear the toggles for all cost-optimization metrics until only the following remain selected:
   - **Transition lifecycle rule count**
   - **Expiration lifecycle rule count**
   - **Noncurrent version transition lifecycle rule count**
   - **Noncurrent version expiration lifecycle rule count**
   - **Abort incomplete multipart upload lifecycle rule count**
   - **Total lifecycle rule count**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

The **Buckets** list updates to display lifecycle rule count metrics for your buckets. You can use this data to identify buckets without lifecycle rules or buckets that are missing specific kinds of lifecycle rules, for example, expiration or transition rules. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to these buckets.

#### Step 3: Add lifecycle rules

After you've identified buckets with no lifecycle rules, you can add lifecycle rules. For more information, see Setting a lifecycle configuration on a bucket and Examples of S3 Lifecycle configuration.

## Using S3 Storage Lens to protect your data

You can use Amazon S3 Storage Lens data-protection metrics to identify buckets where data-protection best practices haven't been applied. You can use these metrics to take action and apply standard settings that align with best practices for protecting your data across the buckets in your account or organization. For example, you can use data-protection metrics to identify buckets that don't use AWS Key Management Service (AWS KMS) keys (SSE-KMS) for default encryption or requests that use AWS Signature Version 2 (SigV2).

The following use cases provide strategies for using your S3 Storage Lens dashboard to identify outliers and apply data-protection best practices across your S3 buckets.

#### Topics

- Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS)
- Identify buckets that have S3 Versioning enabled
- Identify requests that use AWS Signature Version 2 (SigV2)
- Count the total number of replication rules for each bucket
- Identify percentage of Object Lock bytes

#### Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS)

With Amazon S3 default encryption, you can set the default encryption behavior for an S3 bucket. For more information, see the section called “Setting default bucket encryption”.

You can use the **SSE-KMS enabled bucket count** and **% SSE-KMS enabled buckets** metrics to identify buckets that use server-side encryption with AWS KMS keys (SSE-KMS) for default encryption. S3 Storage Lens also provides metrics for unencrypted bytes, unencrypted objects, encrypted bytes, and encrypted objects. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

You can analyze SSE-KMS encryption metrics in the context of general encryption metrics to identify buckets that don't use SSE-KMS. If you want to use SSE-KMS for all the buckets in your account or organization, you can then update the default encryption settings for these buckets to use SSE-KMS. In addition to SSE-KMS, you can use server-side encryption with Amazon S3 managed keys (SSE-S3) or customer-provided keys (SSE-C). For more information, see Protecting data with encryption.

**Step 1: Identify which buckets are using SSE-KMS for default encryption**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, choose **% SSE-KMS enabled bucket count** for the primary metric and **% encrypted bytes** for the secondary metric.

   The **Trend for *date*** chart updates to display trends for SSE-KMS and encrypted bytes.
5. To view more granular, bucket-level insights for SSE-KMS:

   a. Choose a point on the chart. A box will appear with choices for more granular insights.

   b. Choose the **Buckets** dimension. Then choose **Apply**.
6. In the **Distribution by buckets for *date*** chart, choose the **SSE-KMS enabled bucket count** metric.
7. You can now see which buckets have SSE-KMS enabled and which do not.

**Step 2: Update bucket default encryption settings**

Now that you've determined which buckets use SSE-KMS in the context of your **% encrypted bytes**, you can identify buckets that don't use SSE-KMS. You can then optionally navigate to these buckets within the S3 console and update their default encryption settings to use SSE-KMS or SSE-S3. For more information, see Configuring default encryption.

### Identify buckets that have S3 Versioning enabled

When enabled, the S3 Versioning feature retains multiple versions of the same object that can be used to quickly recover data if an object is accidentally deleted or overwritten. You can use the **Versioning-enabled bucket count** metric to see which buckets use S3 Versioning. Then, you can take action in the S3 console to enable S3 Versioning for other buckets.

#### Step 1: Identify buckets that have S3 Versioning enabled

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, choose **Versioning-enabled bucket count** for the primary metric and **Buckets** for the secondary metric.

   The **Trend for *date*** chart updates to display trends for S3 Versioning enabled buckets. Right below the trends line, you can see the **Storage class distribution** and **Region distribution** subsections.
5. To view more granular insights for any of the buckets that you see in the **Trend for *date*** chart so that you can perform a deeper analysis, do the following:

   a. Choose a point on the chart. A box will appear with choices for more granular insights.

   b. Choose a dimension to apply to your data for deeper analysis: **Account**, **AWS Region**, **Storage class**, or **Bucket**. Then choose **Apply**.
6. In the **Bubble analysis by buckets for *date*** section, choose the **Versioning-enabled bucket count**, **Buckets**, and **Active buckets** metrics.

   The **Bubble analysis by buckets for *date*** section updates to display data for the metrics that you selected. You can use this data to see which buckets have S3 Versioning enabled in the context of your total bucket count. In the **Bubble analysis by buckets for *date*** section, you can plot your buckets on multiple dimensions by using any three metrics to represent the **X-axis**, **Y-axis**, and **Size** of the bubble.

#### Step 2: Enable S3 Versioning

After you've identified buckets that have S3 Versioning enabled, you can identify buckets that have never had S3 Versioning enabled or are versioning suspended. Then, you can optionally enable

versioning for these buckets in the S3 console. For more information, see Enabling versioning on buckets.

**Identify requests that use AWS Signature Version 2 (SigV2)**

You can use the **All unsupported signature requests** metric to identify requests that use AWS Signature Version 2 (SigV2). This data can help you identify specific applications that are using SigV2. You can then migrate these applications to AWS Signature Version 4 (SigV4).

SigV4 is the recommended signing method for all new S3 applications. SigV4 provides improved security and is supported in all AWS Regions. For more information, see Amazon S3 update - SigV2 deprecation period extended & modified.

**Prerequisite**

To see **All unsupported signature requests** in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Advanced data protection metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

**Step 1: Examine SigV2 signing trends by AWS account, Region, and bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. To identify specific buckets, accounts, and Regions with requests that use SigV2:

    a. Under **Top N overview for *date***, in **Top N**, enter the number of buckets that you would like to see data for.

    b. For **Metric**, choose **All unsupported signature requests** from the **Data protection** category.

    The **Top N overview for *date*** updates to display data for SigV2 requests by account, AWS Region, and bucket. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

**Step 2: Identify buckets that are accessed by applications through SigV2 requests**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, choose **Data protection**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Data protection** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific data-protection metrics, choose the preferences icon (  ).
7. Clear the toggles for all data-protection metrics until only the following metrics remain selected:

   - **All unsupported signature requests**
   - **% all unsupported signature requests**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list updates to display bucket-level metrics for SigV2 requests. You can use this data to identify specific buckets that have SigV2 requests. Then, you can use this information to migrate your applications to SigV4. For more information, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

### Count the total number of replication rules for each bucket

S3 Replication enables automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. For more information, see Replicating objects overview.

You can use S3 Storage Lens replication rule count metrics to get detailed per-bucket information about your buckets that are configured for replication. This information includes replication rules within and across buckets and Regions.

#### Prerequisite

To see replication rule count metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Advanced data protection metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

#### Step 1: Count the total number of replication rules for each bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, choose **Data protection**. Then clear **Summary**.
6. To filter the **Buckets** list to display only replication rule count metrics, choose the preferences icon
   ( ).
7. Clear the toggles for all data-protection metrics until only the replication rule count metrics remain selected:
   - **Same-Region Replication rule count**
   - **Cross-Region Replication rule count**
   - **Same-account replication rule count**
   - **Cross-account replication rule count**
   - **Total replication rule count**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.

9. Choose **Confirm**.

### Step 2: Add replication rules

After you have a per-bucket replication rule count, you can optionally create additional replication rules. For more information, see Examples for configuring live replication.

#### Identify percentage of Object Lock bytes

With S3 Object Lock, you can store objects by using a *write-once-read-many (WORM)* model. You can use Object Lock to help prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can enable Object Lock only when you create a bucket and also enable S3 Versioning. However, you can edit the retention period for individual object versions or apply legal holds for buckets that have Object Lock enabled. For more information, see Using S3 Object Lock.

You can use Object Lock metrics in S3 Storage Lens to see the **% Object Lock bytes** metric for your account or organization. You can use this information to identify buckets in your account or organization that aren't following your data-protection best practices.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Snapshot** section, under **Metrics categories**, choose **Data protection**.

   The **Snapshot** section updates to display data-protection metrics, including the **% Object Lock bytes** metric. You can see the overall percentage of Object Lock bytes for your account or organization.
5. To see the **% Object Lock bytes** per bucket, scroll down to the **Top N overview** section.

   To get object-level data for Object Lock, you can also use the **Object Lock object count** and **% Object Lock objects** metrics.
6. For **Metric**, choose **% Object Lock bytes** from the **Data protection** category.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. In the **Top N** field, you can increase the number of buckets. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the

trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

7. Review the following data for **% Object Lock bytes**:

- **Top *number* accounts** - See which accounts have the highest and lowest **% Object Lock bytes**.
- **Top *number* Regions** - View a breakdown of **% Object Lock bytes** by Region.
- **Top *number* buckets** - See which buckets have the highest and lowest **% Object Lock bytes**.

## Using S3 Storage Lens to audit Object Ownership settings

Amazon S3 Object Ownership is an S3 bucket-level setting that you can use to disable access control lists (ACLs) and control ownership of the objects in your bucket. If you set Object Ownership to bucket owner enforced, you can disable access control lists (ACLs) and take ownership of every object in your bucket. This approach simplifies access management for data stored in Amazon S3.

By default, when another AWS account uploads an object to your S3 bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through ACLs. You can use Object Ownership to change this default behavior.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. Therefore, we recommend that you disable ACLs, except in unusual circumstances where you must control access for each object individually. By setting Object Ownership to bucket owner enforced, you can disable ACLs and rely on policies for access control. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

With S3 Storage Lens access-management metrics, you can identify buckets that don't have disabled ACLs. After identifying these buckets, you can migrate ACL permissions to policies and disable ACLs for these buckets.

### Topics

- Step 1: Identify general trends for Object Ownership settings
- Step 2: Identify bucket-level trends for Object Ownership settings
- Step 3: Update your Object Ownership setting to bucket owner enforced to disable ACLs

### Step 1: Identify general trends for Object Ownership settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Snapshot for *date*** section, under **Metrics categories**, choose **Access management**.

   The **Snapshot for *date*** section updates to display the **% Object Ownership bucket owner enforced** metric. You can see the overall percentage of buckets in your account or organization that use the bucket owner enforced setting for Object Ownership to disable ACLs.

### Step 2: Identify bucket-level trends for Object Ownership settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. To view more detailed bucket-level metrics, choose the **Bucket** tab.
5. In the **Distribution by buckets for *date*** section, choose the **% Object Ownership bucket owner enforced** metric.

   The chart updates to show a per-bucket breakdown for **% Object Ownership bucket owner enforced**. You can see which buckets use the bucket owner enforced setting for Object Ownership to disable ACLs.
6. To view the bucket owner enforced settings in context, scroll down to the **Buckets** section. For **Metrics categories**, select **Access management**. Then clear **Summary**.

   The **Buckets** list displays data for all three Object Ownership settings: bucket owner enforced, bucket owner preferred, and object writer.

7. To filter the **Buckets** list to display metrics only for a specific Object Ownership setting, choose the preferences icon (  ).
8. Clear the metrics that you don't want to see.
9. (Optional) Under **Page size**, choose the number of buckets to display in the list.
10. Choose **Confirm**.

#### Step 3: Update your Object Ownership setting to bucket owner enforced to disable ACLs

After you've identified buckets that use the object writer and bucket owner preferred setting for Object Ownership, you can migrate your ACL permissions to bucket policies. When you've finished migrating your ACL permissions, you can then update your Object Ownership settings to bucket owner enforced in order to disable ACLs. For more information, see Prerequisites for disabling ACLs.

## Using S3 Storage Lens metrics to improve performance

If you have S3 Storage Lens advanced metrics enabled, you can use detailed status-code metrics to get counts for successful or failed requests. You can use this information to troubleshoot access or performance issues. Detailed status-code metrics show counts for HTTP status codes, such as 403 Forbidden and 503 Service Unavailable. You can examine overall trends for detailed status-code metrics across S3 buckets, accounts, and organizations. Then, you can drill down into bucket-level metrics to identify workloads that are currently accessing these buckets and causing errors.

For example, you can look at the **403 Forbidden error count** metric to identify workloads that are accessing buckets without the correct permissions applied. After you've identified these workloads, you can do a deep dive outside of S3 Storage Lens to troubleshoot your 403 Forbidden errors.

This example shows you how to do a trend analysis for the 403 Forbidden error by using the **403 Forbidden error count** and the **% 403 Forbidden errors** metrics. You can use these metrics to identify workloads that are accessing buckets without the correct permissions applied. You can do a similar trend analysis for any of the other **Detailed status code metrics**. For more information, see Amazon S3 Storage Lens metrics glossary.

#### Prerequisite

To see **Detailed status code metrics** in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Detailed status code metrics**. For more information, see Creating and updating Amazon S3 Storage Lens dashboards.

**Topics**



### Step 1: Do a trend analysis for an individual HTTP status code

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, for **Primary metric**, choose **403 Forbidden error count** from the **Detailed status codes** category. For **Secondary metric**, choose **% 403 Forbidden errors**.
5. Scroll down to the **Top N overview for *date*** section. For **Metrics**, choose **403 Forbidden error count** or **% 403 Forbidden errors** from the **Detailed status codes** category.

   The **Top N overview for *date*** section updates to display the top 403 Forbidden error counts by account, AWS Region, and bucket.

### Step 2: Analyze error counts by bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. For **Metrics categories**, select **Detailed status code metrics**. Then clear **Summary**.

The **Buckets** list updates to display all the available detailed status code metrics. You can use this information to see which buckets have a large proportion of certain HTTP status codes and which status codes are common across buckets.

6. To filter the **Buckets** list to display only specific detailed status-code metrics, choose the preferences icon (  ).
7. Clear the toggles for any detailed status-code metrics that you don't want to view in the **Buckets** list.
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list displays error count metrics for the number of buckets that you specified. You can use this information to identify specific buckets that are experiencing many errors and troubleshoot errors by bucket.

#### Step 3: Troubleshoot errors

After you identify buckets with a high proportion of specific HTTP status codes, you can troubleshoot these errors. For more information, see the following:

- Why am I getting a 403 Forbidden error when I try to upload files in Amazon S3?
- Why am I getting a 403 Forbidden error when I try to modify a bucket policy in Amazon S3?
- How do I troubleshoot 403 Forbidden errors from my Amazon S3 bucket where all the resources are from the same AWS account?
- How do I troubleshoot an HTTP 500 or 503 error from Amazon S3?

## Amazon S3 Storage Lens metrics glossary

The Amazon S3 Storage Lens metrics glossary provides a complete list of free and advanced metrics for S3 Storage Lens.

S3 Storage Lens offers free metrics for all dashboards and configurations, with the option to upgrade to advanced metrics.

- **Free metrics** contain metrics that are relevant to your storage usage, such as the number of buckets and the objects in your account. Free metrics also include use-case based metrics, such as cost-optimization and data-protection metrics. All free metrics are collected daily, and data is available for queries for up to 14 days.
- **Advanced metrics and recommendations** include all the metrics in free metrics along with additional metrics, such as advanced data-protection and cost-optimization metrics. Advanced metrics also include additional metric categories, such as activity metrics and detailed status-code metrics. Advanced metrics data is available for queries for 15 months.

  There are additional charges when you use S3 Storage Lens with advanced metrics and recommendations. For more information, see Amazon S3 pricing. For more information about advanced metrics and recommendations features, see Metrics selection.

> **Note**
>
> For Storage Lens groups, only free tier storage metrics are available. Advanced tier metrics are not available at the Storage Lens group level.

### Metric names

The **Metric name** column in the following table provides the name of each S3 Storage Lens in the S3 console. The **CloudWatch and export** column provides the name of each metric in Amazon CloudWatch and the metrics export file that you can configure in your S3 Storage Lens dashboard.

### Derived metric formulas

Derived metrics are not available for the metrics export and the CloudWatch publishing option. However, you can use the metrics formulas shown in the **Derived metrics formula** column to compute them.

### Interpreting the Amazon S3 Storage Lens prefix symbols for metrics unit multiples (K, M, G, and so on)

S3 Storage Lens metrics unit multiples are written with prefix symbols. These prefix symbols match the International System of Units (SI) symbols that are standardized by the International Bureau of Weights and Measures (BIPM). These symbols are also used in the Unified Code for Units of Measure (UCUM). For more information, see List of SI prefix symbols.

**Note**

- The unit of measurement for S3 storage bytes is in binary gigabytes (GB), where 1 GB is $2^{30}$ bytes, 1 TB is $2^{40}$ bytes, and 1 PB is $2^{50}$ bytes. This unit of measurement is also known as a gibibyte (GiB), as defined by the International Electrotechnical Commission (IEC).
- When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 queues the object for removal and removes it asynchronously. Therefore, there might be a delay between the expiration date and the date when Amazon S3 removes an object. S3 Storage Lens doesn't include metrics for objects that have expired but haven't been removed. For more information about expiration actions in S3 Lifecycle, see Expiring objects.

### S3 Storage Lens metrics glossary

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri meti form |
|---|---|---|---|---|---|---|
| Total storage | StorageBytes | The total storage, inclusive of incomplet e multipart uploads, object metadata, and delete markers | Free | Sum | N | - |
| Object count | ObjectCount | The total object count | Free | Sum | N | - |
| Average object size | - | The average object size | Free | Sum | Y | sum(Stora geBytes)/ sum(Objec tCount) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Active buckets | - | The total number of buckets with storage > 0 bytes | Free | Sum | Y | - |
| Buckets | - | The total number of buckets | Free | Sum | Y | - |
| Accounts | - | The number of accounts whose storage is in scope | Free | Sum | Y | - |
| Current version bytes | CurrentVersionStorageBytes | The number of bytes that are a current version of an object | Free | Cost optimization | N | - |
| % current version bytes | - | The percentage of bytes in scope that are current versions of objects | Free | Cost optimization | Y | sum(CurrentVersionStorageBytes)/sum(StorageBytes) |
| Current version object count | CurrentVersionObjectCount | The count of current version objects | Free | Cost optimization | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % current version objects | - | The percentage of objects in scope that are a current version | Free | Cost optii ion | Y | sum(Curre ntVersion ObjectCou nt)/ sum(O bjectCoun t) |
| Noncurrent version bytes | NonCurren tVersionS torageBytes | The number of noncurrent version bytes | Free | Cost optii ion | N | - |
| % noncurrent version bytes | - | The percentag e of bytes in scope that are noncurrent versions | Free | Cost optii ion | Y | sum(NonCu rrentVers ionStorag eBytes)/ s um(Storag eBytes) |
| Noncurrent version object count | NonCurren tVersionO bjectCount | The count of the noncurrent object versions | Free | Cost optii ion | N | - |
| % noncurrent version objects | - | The percentag e of objects in scope that are a noncurrent version | Free | Cost optii ion | Y | sum(NonCu rrentVers ionObject Count)/ su m(ObjectC ount) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Delete marker bytes | DeleteMarkerStorageBytes | The number of bytes in scope that are delete markers | Free | Cost optimization | N | - |
| % delete marker bytes | - | The percentage of bytes in scope that are delete markers | Free | Cost optimization | Y | sum(DeleteMarkerStorageBytes)/ sum(StorageBytes) |
| Delete marker object count | DeleteMarkerObjectCount | The total number of objects with a delete marker | Free | Cost optimization | N | - |
| % delete marker objects | - | The percentage of objects in scope with a delete marker | Free | Cost optimization | Y | sum(DeleteMarkerObjectCount)/ sum(ObjectCount) |
| Incomplete multipart upload bytes | IncompleteMultipartUploadStorageBytes | The total bytes in scope for incomplete multipart uploads | Free | Cost optimization | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % incomplete multipart upload bytes | - | The percentage of bytes in scope that are the result of incomplete multipart uploads | Free | Cost optimization | Y | sum(IncompleteMultipartUploadStorageBytes)/ sum(StorageBytes) |
| Incomplete multipart upload object count | IncompleteMultipartUploadObjectCount | The number of objects in scope that are incomplete multipart uploads | Free | Cost optimization | N | - |
| % incomplete multipart upload objects | - | The percentage of objects in scope that are incomplete multipart uploads | Free | Cost optimization | Y | sum(IncompleteMultipartUploadObjectCount)/ sum(ObjectCount) |
| Incomplete multipart upload storage bytes greater than 7 days old | IncompleteMPUStorageBytesOlderThan7Days | The total bytes in scope for incomplete multipart uploads that are more than 7 days old | Free | Cost optimization | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % incomplete multipart upload storage bytes greater than 7 days old | - | The percentage of bytes for incomplete multipart uploads that are more than 7 days old | Free | Cost optimization | Y | sum(IncompleteMPUStorageBytesOlderThan7Days)/sum(StorageBytes) |
| Incomplete multipart upload object count greater than 7 days old | IncompleteMPUObjectCountOlderThan7Days | The number of objects that are incomplete multipart uploads more than 7 days old | Free | Cost optimization | N | - |
| % incomplete multipart upload object count greater than 7 days old | - | The percentage of objects that are incomplete multipart uploads more than 7 days old | Free | Cost optimization | Y | sum(IncompleteMPUObjectCountOlderThan7Days)/sum(ObjectCount) |
| Transition lifecycle rule count | TransitionLifecycleRuleCount | The count of lifecycle rules to transition objects to another storage class | Adva | Cost optimization | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Average transition lifecycle rules per bucket | - | The average number of lifecycle rules to transition objects to another storage class | Adva | Cost optii ion | Y | sum(Trans itionLife cycleRule Count)/ su m(Distinc tNumberOf Buckets) |
| Expiration lifecycle rule count | Expiratio nLifecycl eRuleCount | The count of lifecycle rules to expire objects | Adva | Cost optii ion | N | - |
| Average expiration lifecycle rules per bucket | - | The average number of lifecycle rules to expire objects | Adva | Cost optii ion | Y | sum(Expir ationLife cycleRule Count)/ su m(Distinc tNumberOf Buckets) |
| Noncurren t version transition lifecycle rule count | Noncurren tVersionT ransition Lifecycle RuleCount | The count of lifecycle rules to transitio n noncurrent object versions to another storage class | Adva | Cost optii ion | N | |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Average noncurrent t version transition lifecycle rules per bucket | - | The average number of lifecycle rules to transitio n noncurrent object versions to another storage class | Adva | Cost optii ion | Y | sum(Noncu rrentVers ionTransi tionLifec ycleRuleC ount)/ sum (Distinct NumberOfB uckets) |
| Noncurren t version expiration lifecycle rule count | Noncurren tVersionE xpiration Lifecycle RuleCount | The count of lifecycle rules to expire noncurrent object versions | Adva | Cost optii ion | N | - |
| Average noncurren t version expiration lifecycle rules per bucket | - | The average number of lifecycle rules to expire noncurrent object versions | Adva | Cost optii ion | Y | sum(Noncu rrentVers ionExpira tionLifec ycleRuleC ount)/ sum (Distinct NumberOfB uckets) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Abort incomplete multipart upload lifecycle rule count | AbortIncompleteMPULifecycleRuleCount | The count of lifecycle rules to delete incomplete multipart uploads | Adva | Cost optii ion | N | - |
| Average abort incomplete multipart upload lifecycle rules per bucket | - | The average number of lifecycle rules to delete incomplete multipart uploads | Adva | Cost optii ion | Y | sum(AbortIncompleteMPULifecycleRuleCount)/ sum (DistinctNumberOfBuckets) |
| Expired object delete marker lifecycle rule count | ExpiredObjectDeleteMarkerLifecycleRuleCount | The count of lifecycle rules to remove expired object delete markers | Adva | Cost optii ion | N | - |
| Average expired object delete marker lifecycle rules per bucket | - | The average number of lifecycle rules to remove expired object delete markers | Adva | Cost optii ion | Y | sum(ExpiredObjectDeleteMarkerLifecycleRuleCount)/ sum(DistinctNumberOfBuckets) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Total lifecycle rule count | TotalLife cycleRuleCount | The total count of lifecycle rules | Adva | Cost opti ion | N | - |
| Average lifecycle rule count per bucket | - | The average number of lifecycle rules | Adva | Cost opti ion | Y | sum(Total LifecycleRuleCount)/ sum(DistinctNumberOfBuckets) |
| Encrypted bytes | Encrypted StorageBytes | The total number of encrypted bytes | Free | Data prot n | N | - |
| % encrypted bytes | - | The percentage of total bytes that are encrypted | Free | Data prot n | Y | sum(EncryptedObjectCount)/ sum(StorageBytes) |
| Encrypted object count | Encrypted ObjectCount | The total count of objects that are encrypted | Free | Data prot n | N | - |
| % encrypted objects | - | The percentage of objects that are encrypted | Free | Data prot n | Y | sum(EncryptedStorageBytes)/ sum(ObjectCount) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Unencrypted bytes | UnencryptedStorageBytes | The number of bytes that are unencrypted | Free | Data prot n | Y | sum(StorageBytes) - sum(EncryptedStorageBytes) |
| % unencrypted bytes | - | The percentage of bytes that are unencrypted | Free | Data prot n | Y | sum(UnencryptedStorageBytes)/ sum(StorageBytes) |
| Unencrypted object count | UnencryptedObjectCount | The total count of objects that are unencrypted | Free | Data prot n | Y | sum(ObjectCount) - sum(EncryptedObjectCount) |
| % unencrypted objects | - | The percentage of unencrypted objects | Free | Data prot n | Y | sum(UnencryptedStorageBytes)/ sum(ObjectCount) |
| Replicated storage bytes source | ReplicatedStorageBytesSource | The total number of bytes that are replicated from the source bucket | Free | Data prot n | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % replicated bytes source | - | The percentage of total bytes that are replicated from the source bucket | Free | Data prot n | Y | sum(ReplicatedStorageBytesSource)/ sum(StorageBytes) |
| Replicated object count source | ReplicatedObjectCountSource | The count of replicated objects from the source bucket | Free | Data prot n | N | - |
| % replicated objects source | - | The percentage of total objects that are replicated from the source bucket | Free | Data prot n | Y | sum(ReplicatedStorageObjectCount)/ sum(ObjectCount) |
| Replication storage bytes destination | ReplicatedStorageBytes | The total number of bytes that are replicated to the destination bucket | Free | Data prot n | Y | - |
| % replicated bytes destination | - | The percentage of total bytes that are replicated to the destination bucket | Free | Data prot n | Y | sum(ReplicatedStorageBytes) / sum(StorageBytes) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Replicated object count destination | ReplicatedObjectCount | The count of objects that are replicated to the destination bucket | Free | Data prot n | Y | - |
| % replicated objects destination | - | The percentage of total objects that are replicated to the destination bucket | Free | Data prot n | Y | sum(ReplicatedObjectCount)/ sum(ObjectCount) |
| Object Lock bytes | ObjectLockEnabledStorageBytes | The total count of Object Lock enabled storage bytes | Free | Data prot n | Y | sum(UnencryptedStorageBytes)/ sum(ObjectLockEnabledStorageCount) - sum(ObjectLockEnabledStorageBytes) |
| % Object Lock bytes | - | The percentage of Object Lock enabled storage bytes | Free | Data prot n | Y | sum(ObjectLockEnabledStorageBytes)/ sum(StorageBytes) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Object Lock object count | ObjectLockEnabledObjectCount | The total count of Object Lock objects | Free | Data prot n | Y | - |
| % Object Lock objects | - | The percentage of total objects that have Object Lock enabled | Free | Data prot n | Y | sum(ObjectLockEnabledObjectCount)/ sum(ObjectCount) |
| Versioning-enabled bucket count | VersioningEnabledBucketCount | The count of buckets that have S3 Versioning enabled | Free | Data prot n | N | - |
| % versioning-enabled buckets | - | The percentage of buckets that have S3 Versioning enabled | Free | Data prot n | Y | sum(VersioningEnabledBucketCount)/ sum(DistinctNumberOfBuckets) |
| MFA delete-enabled bucket count | MFADeleteEnabledBucketCount | The count of buckets that have MFA (multi-factor authentication) delete enabled | Free | Data prot n | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % MFA delete-enabled buckets | - | The percentage of buckets that have MFA (multi-factor authentication) delete enabled | Free | Data prot n | Y | sum(MFADeleteEnabledBucketCount)/ sum (Distinct NumberOfBuckets) |
| SSE-KMS enabled bucket count | SSEKMSEnabledBucketCount | The count of buckets that use server-side encryption with AWS Key Management Service keys (SSE-KMS) for default bucket encryption | Free | Data prot n | N | - |
| % SSE-KMS enabled buckets | - | The percentage of buckets that SSE-KMS for default bucket encryption | Free | Data prot n | Y | sum(SSEKMSEnabledBucketCount)/ sum(DistinctNumberOfBuckets) |
| All unsupported signature requests | AllUnsupportedSignatureRequests | The total number of requests that use unsupported AWS signature versions | Adva | Data prot n | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % all unsupported signature requests | - | The percentage of requests that use unsupported AWS signature versions | Adva | Data prot n | Y | sum(AllUnsupported SignatureRequests) / sum(AllRequests) |
| All unsupported TLS requests | AllUnsupportedTLSRequests | The number of requests that use unsupported Transport Layer Security (TLS) versions | Adva | Data prot n | N | - |
| % all unsupported TLS requests | - | The percentage of requests that use unsupported TLS versions | Adva | Data prot n | Y | sum(AllUnsupportedTLSRequests)/ sum(AllRequests) |
| All SSE-KMS requests | AllSSEKMSRequests | The total number of requests that specify SSE-KMS | Adva | Data prot n | N | - |
| % all SSE-KMS requests | - | The percentage of requests that specify SSE-KMS | Adva | Data prot n | Y | sum(AllSSEKMSRequests)/ sum(AllRequests) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Same-Region Replication rule count | SameRegionReplicationRuleCount | The count of replication rules for Same-Region Replication (SRR) | Adva | Data prot n | N | - |
| Average Same-Region Replication rules per bucket | - | The average number of replication rules for SRR | Adva | Data prot n | Y | sum(SameRegionReplicationRuleCount)/ sum(DistinctNumberOfBuckets) |
| Cross-Region Replication rule count | CrossRegionReplicationRuleCount | The count of replication rules for Cross-Region Replication (CRR) | Adva | Data prot n | N | - |
| Average Cross-Region Replication rules per bucket | - | The average number of replication rules for CRR | Adva | Data prot n | Y | sum(CrossRegionReplicationRuleCount) / sum(DistinctNumberOfBuckets) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Same-account replication rule count | SameAccountReplicationRuleCount | The count of replication rules for replication within the same account | Adva | Data prot n | N | - |
| Average same-account replication rules per bucket | - | The average number of replication rules for replication within the same account | Adva | Data prot n | Y | sum(SameAccountReplicationRuleCount) / sum(DistinctNumberOfBuckets) |
| Cross-account replication rule count | CrossAccountReplicationRuleCount | The count of replication rules for cross-account replication | Adva | Data prot n | N | - |
| Average cross-account replication rules per bucket | - | The average number of replication rules for cross-account replication | Adva | Data prot n | Y | sum(CrossAccountReplicationRuleCount)/ sum(DistinctNumberOfBuckets) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Invalid destination replication rule count | InvalidDestinationReplicationRuleCount | The count of replication rules with a replication destination that's not valid | Adva | Data prot n | N | - |
| Average invalid destination replication rules per bucket | - | The average number of replication rules with a replication destination that's not valid | Adva | Data prot n | Y | sum(InvalidReplicationRuleCount)/ sum(DistinctNumberOfBuckets) |
| Total replication rule count | - | The total replication rule count | Adva | Data prot n | Y | - |
| Average replication rule count per bucket | - | The average total replication rule count | Adva | Data prot n | Y | sum(all replication rule count metrics)/ sum(DistinctNumberOfBuckets) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Object Ownership bucket owner enforced bucket count | ObjectOwnershipBucketOwnerEnforcedBucketCount | The total count of buckets that have access control lists (ACLs) disabled by using the bucket owner enforced setting for Object Ownership | Free | Acce man t | N | - |
| % Object Ownership bucket owner enforced buckets | - | The percentage of buckets that have ACLs disabled by using the bucket owner enforced setting for Object Ownership | Free | Acce man t | Y | sum(ObjectOwnershipBucketOwnerEnforcedBucketCount)/ sum (Distinct NumberOfBuckets) |
| Object Ownership bucket owner preferred bucket count | ObjectOwnershipBucketOwnerPreferredBucketCount | The total count of buckets that use the bucket owner preferred setting for Object Ownership | Free | Acce man t | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % Object Ownership bucket owner preferred buckets | - | The percentage of buckets that use the bucket owner preferred setting for Object Ownership | Free | Acce man t | Y | sum(ObjectOwnershipBucketOwnerPreferredBucketCount)/ sum(DistinctNumberOfBuckets) |
| Object Ownership object writer bucket count | ObjectOwnershipObjectWriterBucketCount | The total count of buckets that use the object writer setting for Object Ownership | Free | Acce man t | N | - |
| % Object Ownership object writer buckets | - | The percentage of buckets that use the object writer setting for Object Ownership | Free | Acce man t | Y | sum(ObjectOwnershipObjectWriterBucketCount)/ sum(DistinctNumberOfBuckets) |
| Transfer Acceleration enabled bucket count | TransferAccelerationEnabledBucketCount | The total count of buckets that have Transfer Acceleration enabled | Free | Perf ce | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % Transfer Accelerat ion enabled buckets | - | The percentage of buckets that have Transfer Accelerat ion enabled | Free | Perf ce | Y | sum(Trans ferAccele rationEna bledBucke tCount)/ s um(Distin ctNumberO fBuckets) |
| Event Notificat ion enabled bucket count | EventNoti ficationE nabledBuc ketCount | The total count of buckets that have Event Notificat ions enabled | Free | Ever | N | |
| % Event Notificat ion enabled buckets | - | The percentage of buckets that have Event Notificat ions enabled | Free | Ever | Y | sum(Event Notificat ionEnable dBucketCo unt)/ sum( DistinctN umberOfBu ckets) |
| All requests | AllRequests | The total number of requests made | Adva | Activ | N | - |
| Get requests | GetRequests | The total number of GET requests made | Adva | Activ | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| Put requests | PutRequests | The total number of PUT requests made | Adva | Activ | N | - |
| Head requests | HeadRequests | The total number of HEAD requests made | Adva | Activ | N | - |
| Delete requests | DeleteRequests | The total number of DELETE requests made | Adva | Activ | N | - |
| List requests | ListRequests | The total number of LIST requests made | Adva | Activ | N | - |
| Post requests | PostRequests | The total number of POST requests made | Adva | Activ | N | - |
| Select requests | SelectRequests | The total number of S3 Select requests | Adva | Activ | N | - |
| Select scanned bytes | SelectScannedBytes | The number of S3 Select bytes scanned | Adva | Activ | N | - |
| Select returned bytes | SelectReturnedBytes | The number of S3 Select bytes returned | Adva | Activ | N | - |
| Bytes downloaded | BytesDownloaded | The number of bytes downloaded | Adva | Activ | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % retrieval rate | - | The percentage of bytes downloaded | Adva | Activ | Y | sum(BytesDownloaded)/sum(StorageBytes) |
| Bytes uploaded | BytesUploaded | The number of bytes uploaded | Adva | Activ | N | - |
| % ingest ratio | - | The percentage of bytes uploaded | Adva | Activ | Y | sum(BytesUploaded)/sum(StorageBytes) |
| 4xx errors | 4xxErrors | The total number of HTTP 4xx status codes | Adva | Activ | N | - |
| 5xx errors | 5xxErrors | The total number of HTTP 5xx status codes | Adva | Activ | N | - |
| Total errors | - | The sum of all 4xx and 5xx errors | Adva | Activ | Y | sum(4xxErrors) + sum(5xxErrors) |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metı form |
|---|---|---|---|---|---|---|
| % error rate | - | The total number of 4xx and 5xx errors as a percentage of total requests | Adva | Activ | Y | sum(Total Errors)/ s um(TotalR equests) |
| 200 OK status count | 200OKStat usCount | The total count of 200 OK status codes | Adva | Deta statı code | N | - |
| % 200 OK status | - | The total number of 200 OK status codes as a percentage of total requests | Adva | Deta statı code | Y | sum(200OK StatusCou nt)/ sum(A llRequest s) |
| 206 Partial Content status count | 206Partia lContentS tatusCount | The total count of 206 Partial Content status codes | Adva | Deta statı code | N | - |
| % 206 Partial Content status | - | The total number of 206 Partial Content status codes as a percentage of total requests | Adva | Deta statı code | Y | sum(206Pa rtialCont entStatus Count)/ su m(AllRequ ests) |
| 400 Bad Request error count | 400BadReq uestErrorCount | The total count of 400 Bad Request status codes | Adva | Deta statı code | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % 400 Bad Request errors | - | The total number of 400 Bad Request status codes as a percentage of total requests | Adva | Deta statu code | Y | sum(400BadRequestErrorCount)/ sum(All Requests) |
| 403 Forbidden error count | 403ForbiddenErrorCount | The total count of 403 Forbidden status codes | Adva | Deta statu code | N | - |
| % 403 Forbidden errors | - | The total number of 403 Forbidden status codes as a percentage of total requests | Adva | Deta statu code | Y | sum(403ForbiddenErrorCount) / sum(AllRequests) |
| 404 Not Found error count | 404NotFoundErrorCount | The total count of 404 Not Found status codes | Adva | Deta statu code | N | - |
| % 404 Not Found errors | - | The total number of 404 Not Found status codes as a percentage of total requests | Adva | Deta statu code | Y | sum(404NotFoundErrorCount)/ sum(AllRequests) |
| 500 Internal Server Error count | 500InternalServerErrorCount | The total count of 500 Internal Server Error status codes | Adva | Deta statu code | N | - |

| Metric name | CloudWatch and export | Description | Tier[1] | Cate | D | Deri metr form |
|---|---|---|---|---|---|---|
| % 500 Internal Server Errors | - | The total number of 500 Internal Server Error status codes as a percentage of total requests | Adva | Deta statu code | Y | sum(500InternalServerErrorCount)/ sum(AllRequests) |
| 503 Service Unavailable error count | 503ServiceUnavailableErrorCount | The total count of 503 Service Unavailable status codes | Adva | Deta statu code | N | - |
| % 503 Service Unavailable errors | - | The total number of 503 Service Unavailable status codes as a percentage of total requests | Adva | Deta statu code | Y | sum(503ServiceUnavailableErrorCount) / sum(AllRequests) |

[1] All free tier storage metrics are available at the Storage Lens group level. Advanced tier metrics are not available at the Storage Lens group level.

[2] Rule count metrics and bucket settings metrics aren't available at the prefix level.

## Working with Amazon S3 Storage Lens by using the console and API

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security

best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

The following sections contain examples of creating, updating, and viewing S3 Storage Lens configurations and performing operations related to the feature. If you are using S3 Storage Lens with AWS Organizations, these examples also cover those use cases. In the examples, replace any variable values with those that are specific to you.

**Topics**



## Using Amazon S3 Storage Lens on the console

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to

deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

> **Note**
>
> Any updates to your dashboard configuration can take up to 48 hours to accurately display or visualize.

**Topics**



### Creating and updating Amazon S3 Storage Lens dashboards

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

The Amazon S3 Storage Lens default dashboard is **default-account-dashboard**. This dashboard is preconfigured by Amazon S3 to help you visualize summarized insights and trends for your entire account's aggregated free and advanced metrics on the console. You can't modify the default dashboard's configuration scope, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations, configure the optional metrics export, or even disable the default dashboard. The default dashboard cannot be deleted.

You can also create additional S3 Storage Lens custom dashboards that can be scoped to your organization in AWS Organizations or to specific Regions or buckets within an account.

### Creating an Amazon S3 Storage Lens dashboard

Use the following steps to create an Amazon S3 Storage Lens dashboard on the Amazon S3 console.

#### Step 1: Define the dashboard scope

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, under **S3 Storage Lens**, choose **Dashboards**.
4. Choose **Create dashboard**.
5. On the **Dashboard** page, in the **General** section, do the following:

    a. View the **Home Region** for your dashboard. The home Region is the AWS Region where the configuration and metrics for this Storage Lens dashboard are stored.

    b. Enter a dashboard name.

    Dashboard names must be fewer than 65 characters and must not contain special characters or spaces.

    > **Note**
    >
    > You can't change this dashboard name after the dashboard is created.

    c. You can optionally choose to add **Tags** to your dashboard. You can use tags to manage permissions for your dashboard and track costs for S3 Storage Lens.

    For more information, see Controlling access using resource tags in the *IAM User Guide* and AWS-Generated Cost Allocation Tags in the *AWS Billing User Guide*.

    > **Note**
    >
    > You can add up to 50 tags to your dashboard configuration.

6. In the **Dashboard scope** section, do the following:

   a. Choose the Regions and buckets that you want S3 Storage Lens to include or exclude in the dashboard.

   b. Choose the buckets in your selected Regions that you want S3 Storage Lens to include or exclude. You can either include or exclude buckets, but not both. This option is not available when you create organization-level dashboards.

> **Note**
>
> - You can either include or exclude Regions and buckets. This option is limited to Regions only when creating organization-level dashboards across member accounts in your organization.
> - You can choose up to 50 buckets to include or exclude.

**Step 2: Configure the metrics selection**

1. In the **Metrics selection** section, choose the type of metrics that you want to aggregate for this dashboard.

   - To include free metrics aggregated at the bucket level and available for queries for 14 days, choose **Free metrics**.
   - To enable advanced metrics and other advanced options, choose **Advanced metrics and recommendations**. These options include advanced prefix aggregation, Amazon CloudWatch publishing, and contextual recommendations. Data is available for queries for 15 months. Advanced metrics and recommendations have an additional cost. For more information, see Amazon S3 pricing.

   For more information about advanced metrics and free metrics, see Metrics selection.

2. Under **Advanced metrics and recommendations features**, select the options that you want to enable:

   - **Advanced metrics**
   - **CloudWatch publishing**
   - **Prefix aggregation**

> **Important**
>
> If you enable prefix aggregation for your S3 Storage Lens configuration, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.

3. If you enabled **Advanced metrics**, select the **Advanced metrics categories** that you want to display in your S3 Storage Lens dashboard:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Advanced cost optimization metrics**
   - **Advanced data protection metrics**

   For more information about metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

4. If you chose to enable prefix aggregation, configure the following:

   a. Choose the minimum prefix threshold size for this dashboard.

      For example, a prefix threshold of 5 percent indicates that prefixes that make up 5 percent or more of the bucket's total storage size will be aggregated.

   b. Choose the prefix depth.

      This setting indicates the maximum number of levels up to which the prefixes are evaluated. The prefix depth must be less than 10.

   c. Enter a prefix delimiter character.

      This value is used to identify each prefix level. The default value in Amazon S3 is the / character, but your storage structure might use other delimiter characters.

### (Optional) Step 3: Export metrics for the dashboard

1. In the **Metrics export** section, to create a metrics export that will be placed daily in a destination bucket of your choice, choose **Enable**.

The metrics export is in CSV or Apache Parquet format. It represents the same scope of data as your S3 Storage Lens dashboard data without the recommendations.

2. If you enabled the metrics export, choose the output format of your daily metrics export: **CSV** or **Apache Parquet**.

   Parquet is an open source file format for Hadoop that stores nested data in a flat columnar format.

3. Choose the destination S3 bucket for your metrics export.

   You can choose a bucket in the current account of the S3 Storage Lens dashboard. Or you can choose another AWS account if you have the destination bucket permissions and the destination bucket owner's account ID.

4. Choose the destination S3 bucket (format: `s3://bucket-name/prefix`).

   The bucket must be in the home Region of your S3 Storage Lens dashboard. The S3 console shows you the **Destination bucket permission** that will be added by Amazon S3 to the destination bucket policy. Amazon S3 updates the bucket policy on the destination bucket to allow S3 to place data in that bucket.

5. (Optional) To enable server-side encryption for your metrics export, choose **Specify an encryption key**. Then, choose the **Encryption type**: **Amazon S3 managed keys (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**.

   You can choose between an Amazon S3 managed key (SSE-S3) and an AWS Key Management Service (AWS KMS) key (SSE-KMS).

6. (Optional) To specify an AWS KMS key, you must choose a KMS key or enter a key Amazon Resource Name (ARN).

   If you choose a customer managed key, you must grant S3 Storage Lens permission to encrypt in the AWS KMS key policy. For more information, see Using an AWS KMS key to encrypt your metrics exports.

7. Choose **Create dashboard**.

To gain further visibility into your storage, you can create one or more S3 Storage Lens groups and attach them to your dashboard. An S3 Storage Lens group is a custom defined filter for objects based on prefixes, suffixes, object tags, object size, object age, or a combination of these filters.

You can use S3 Storage Lens groups to gain granular visibility into large shared buckets, such as data lakes, to make better-informed business decisions. For example, you can streamline storage allocation and optimize cost reporting by breaking down storage usage to specific groups of objects for individual projects and cost centers within a bucket or across multiple buckets.

To use S3 Storage Lens groups, you must upgrade your dashboard to use advanced metrics and recommendations. For more information about S3 Storage Lens groups, see the section called "Working with S3 Storage Lens groups".

**Updating an Amazon S3 Storage Lens dashboard**

Use the following steps to update an Amazon S3 Storage Lens dashboard on the Amazon S3 console.

**Step 1: Update the dashboard scope**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens, Dashboards**.
3. Choose the dashboard that you want to edit, and then choose **Edit**.

   The **Edit dashboard** page opens.

   > **Note**
   >
   > You can't change the following:
   >
   > - The dashboard name
   > - The home Region
   > - The dashboard scope of the default dashboard, which is scoped to your entire account's storage

4. (Optional) On the dashboard configuration page, in the **General** section, update and add tags to your dashboard.

   You can use tags to manage permissions for your dashboard and to track costs for S3 Storage Lens. For more information, see Controlling access using resource tags in the *IAM User Guide* and AWS-Generated Cost Allocation Tags in the *AWS Billing User Guide*.

> **Note**
>
> You can add up to 50 tags to your dashboard configuration.

5. In the **Dashboard scope** section, do the following:

   a. Update the Regions and buckets that you want S3 Storage Lens to include or exclude in the dashboard.

   > **Note**
   >
   > - You can either include or exclude Regions and buckets. This option is limited to Regions only when creating organization-level dashboards across member accounts in your organization.
   > - You can choose up to 50 buckets to include or exclude.

   b. Update the buckets in your selected Regions that you want S3 Storage Lens to include or exclude. You can either include or exclude buckets, but not both. This option is not present when creating organization-level dashboards.

**Step 2: Update the metrics selection**

1. In the **Metrics selection** section, choose the type of metrics that you want to aggregate for this dashboard.

   - To include free metrics aggregated at the bucket level and available for queries for 14 days, choose **Free metrics**.
   - To enable advanced metrics and other advanced options, choose **Advanced metrics and recommendations**. These options include advanced prefix aggregation, Amazon CloudWatch publishing, and contextual recommendations. Data is available for queries for 15 months. Advanced metrics and recommendations have an additional cost. For more information, see Amazon S3 pricing.

   For more information about advanced metrics and free metrics, see Metrics selection.

2. Under **Advanced metrics and recommendations features**, select the options that you want to enable:

- **Advanced metrics**
- **CloudWatch publishing**
- **Prefix aggregation**

> **Important**
>
> If you enable prefix aggregation for your S3 Storage Lens configuration, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.

3. If you enabled **Advanced metrics**, select the **Advanced metrics categories** that you want to display in your S3 Storage Lens dashboard:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Advanced cost optimization metrics**
   - **Advanced data protection metrics**

   For more information metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

4. If you chose to enable prefix aggregation, configure the following:

   a. Choose the minimum prefix threshold size for this dashboard.

      For example, a prefix threshold of 5 percent indicates that prefixes that make up 5 percent or more of the bucket's total storage size will be aggregated.

   b. Choose the prefix depth.

      This setting indicates the maximum number of levels up to which the prefixes are evaluated. The prefix depth must be less than 10.

   c. Enter a prefix delimiter character.

      This is the value used to identify each prefix level. The default value in Amazon S3 is the / character, but your storage structure might use other delimiter characters.

### (Optional) Step 3: Export metrics for the dashboard

1. In the **Metrics export** section, to create a metrics export that will be placed daily in a destination bucket of your choice, choose **Enable**. To disable the metrics export, choose **Disable**.

   The metrics export is in CSV or Apache Parquet format. It represents the same scope of data as your S3 Storage Lens dashboard data without the recommendations.

2. If enabled, choose the output format of your daily metrics export: **CSV** or **Apache Parquet**.

   Parquet is an open source file format for Hadoop that stores nested data in a flat columnar format.

3. Choose the destination S3 bucket for your metrics export.

   You can choose a bucket in the current account of the S3 Storage Lens dashboard. Or you can choose another AWS account if you have the destination bucket permissions and the destination bucket owner's account ID.

4. Choose the destination S3 bucket (format: `s3://bucket-name/prefix`).

   The bucket must be in the home Region of your S3 Storage Lens dashboard. The S3 console shows you the **Destination bucket permission** that will be added by Amazon S3 to the destination bucket policy. Amazon S3 updates the bucket policy on the destination bucket to allow S3 to place data in that bucket.

5. (Optional) To enable server-side encryption for your metrics export, choose **Specify an encryption key**. Then, choose the **Encryption type**: **Amazon S3 managed keys (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**.

   You can choose between an Amazon S3 managed key (SSE-S3) and an AWS Key Management Service (AWS KMS) key (SSE-KMS).

6. (Optional) To specify an AWS KMS key, you must choose a KMS key or enter a key Amazon Resource Name (ARN). Under **AWS KMS key**, specify your KMS key in one of the following ways:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> The AWS managed key (`aws/S3`) is not supported for SSE-KMS encryption with S3 Storage Lens.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  If you choose a customer managed key, you must grant S3 Storage Lens permission to encrypt in the AWS KMS key policy. For more information, see Using an AWS KMS key to encrypt your metrics exports.

  For more information about creating an AWS KMS key, see Creating Keys in the *AWS Key Management Service Developer Guide*.

7. Choose **Save changes**.

To gain further visibility into your storage, you can create one or more S3 Storage Lens groups and attach them to your dashboard. An S3 Storage Lens group is a custom defined filter for objects based on prefixes, suffixes, object tags, object size, object age, or a combination of these filters.

You can use S3 Storage Lens groups to gain granular visibility into large shared buckets, such as data lakes, to make better-informed business decisions. For example, you can streamline storage allocation and optimize cost reporting by breaking down storage usage to specific groups of objects for individual projects and cost centers within a bucket or across multiple buckets.

To use S3 Storage Lens groups, you must upgrade your dashboard to use advanced metrics and recommendations. For more information about S3 Storage Lens groups, see the section called "Working with S3 Storage Lens groups".

**Disabling or deleting Amazon S3 Storage Lens dashboards**

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an

interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

The Amazon S3 Storage Lens default dashboard is **default-account-dashboard**. This dashboard is preconfigured by Amazon S3 to help you visualize summarized insights and trends for your entire account's aggregated free and advanced metrics on the console. You can't modify the default dashboard's configuration scope, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations, configure the optional metrics export, or even disable the default dashboard. The default dashboard cannot be deleted.

You can delete or disable an Amazon S3 Storage Lens dashboard from the Amazon S3 console. Disabling or deleting a dashboard prevents it from generating metrics in the future. A disabled dashboard still retains its configuration information, so that it can be easily resumed when re-enabled. A disabled dashboard retains its historical data until it's no longer available for queries.

Data for free metrics selections is available for queries for 14 days, and data for advanced metrics and recommendations selections is available for queries for 15 months.

**Disabling an Amazon S3 Storage Lens dashboard**

**To disable an S3 Storage Lens dashboard**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to disable, and then choose **Disable** at the top of the list.
4. On the confirmation page, confirm that you want to disable the dashboard by entering the name of dashboard in the text field, and then choose **Confirm**.

### Deleting an Amazon S3 Storage Lens dashboard

> **Note**
>
> You can't delete the default dashboard. However, you can disable it. Before deleting a dashboard that you've created, consider the following:
>
> - As an alternative to deleting a dashboard, you can *disable* the dashboard so that it is available to be re-enabled in the future. For more information, see Disabling an Amazon S3 Storage Lens dashboard.
> - Deleting the dashboard deletes all the configuration settings that are associated with it.
> - Deleting a dashboard makes all the historic metrics data unavailable. This historical data is still retained for 15 months. If you want to access this data again, create a dashboard with the same name in the same home Region as the one that was deleted.

#### To delete an S3 Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to delete, and then choose **Delete** at the top of the list.
4. On the **Delete dashboards** page, confirm that you want to delete the dashboard by entering the name of dashboard in the text field. Then choose **Confirm**.

### Working with AWS Organizations to create organization-level dashboards

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket.

The Amazon S3 Storage Lens default dashboard is **default-account-dashboard**. This dashboard is preconfigured by Amazon S3 to help you visualize summarized insights and trends for your entire account's aggregated free and advanced metrics on the console. You can't modify the default dashboard's configuration scope, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations, configure the optional metrics export, or even disable the default dashboard. The default dashboard cannot be deleted.

You can also create additional S3 Storage Lens dashboards that are focused on specific AWS Regions, S3 buckets, or other AWS accounts in your organization.

An S3 Storage Lens dashboard provides a rich resource of information about its storage scope. A dashboard visualizes more than 30 metrics that represent trends and information, including storage summary, cost efficiency, data protection, and activity.

Amazon S3 Storage Lens can be used to collect storage metrics and usage data for all accounts that are part of your AWS Organizations hierarchy. To do this, you must be using AWS Organizations, and you must enable S3 Storage Lens trusted access by using your AWS Organizations management account.

When trusted access is enabled, you can add delegate administrator access to accounts in your organization. These accounts can then create organization-wide dashboards and configurations for S3 Storage Lens. For more information about enabling trusted access, see Amazon S3 Lens and AWS Organizations in the *AWS Organizations User Guide*.

The following console controls are available only to the AWS Organizations management accounts.

**Enabling trusted access for S3 Storage Lens in your organization**

Enabling trusted access allows Amazon S3 Storage Lens to access your AWS Organizations hierarchy, membership, and structure through AWS Organizations API operations. S3 Storage Lens becomes a trusted service for your entire organization's structure. It can create service-linked roles in your organization's management or delegated administrator accounts whenever a dashboard configuration is created.

The service-linked role grants S3 Storage Lens permissions to describe organizations, list accounts, verify a list of service access for the organizations, and get delegated administrators for the organization. This allows S3 Storage Lens to collect cross-account storage usage and activity metrics for dashboards within accounts in your organization.

For more information, see Using service-linked roles for Amazon S3 Storage Lens.

**Note**

- Trusted access can be enabled only by the *management account*.
- Only the management account and delegated administrators can create S3 Storage Lens dashboards or configurations for your organization.

**To enable S3 Storage Lens to have trusted access**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Organization settings**.
3. In **Organizations access**, choose **Edit**.

   The **Organization access** page opens. Here you can **Enable trusted access** for S3 Storage Lens. This allows you and any other account holders that you add as delegated administrators to create dashboards for all accounts and storage in your organization.

**Disabling S3 Storage Lens trusted access in your organization**

Disabling trusted access will limit S3 Storage Lens to work only on an account level. Each account holder will only be able to see the benefits of S3 Storage Lens limited to the scope of their account, and not their organization. Any dashboards requiring trusted access will no longer be updated, but those dashboards will be able to query their historic data per the respective period that data is available for queries.

Removing an account as a delegated administrator limits the account owner's S3 Storage Lens dashboard metrics access to work only on an account level. Any organizational dashboards that they created will no longer be updated, but they will be able to query their historic data per the period that it is available for queries.

**Note**

- Disabling trusted access also automatically disables all organization-level dashboards because S3 Storage Lens will no longer have trusted access to the organization accounts to collect and aggregate storage metrics.

- The management and delegate administrator accounts can still see the historic data for these disabled dashboards and can query this data while it is available.

**To disable trusted access for S3 Storage Lens**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Organization settings**.
3. In **Organizations access**, choose **Edit**.

   The **Organization access** page opens. Here you can **Disable trusted access** for S3 Storage Lens.

### Registering delegated administrators for S3 Storage Lens

After enabling trusted access, you can register delegate administrator access to accounts in your organization. When an account is registered as a delegate administrator, the account receives authorization to access all read-only AWS Organizations API operations. This provides visibility to the members and structures of your organization so that they can create S3 Storage Lens dashboards on your behalf.

**To register delegated administrators for S3 Storage Lens**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Organization settings**.
3. In the **delegated access** section, for **Accounts**, choose **Add account**.

   The **Delegated admin access** page opens. Here you can add an AWS account ID as a delegated administrator to create organization-level dashboards for all accounts and storage in your organization.

### Deregistering delegated administrators for S3 Storage Lens

You can deregister delegate administrator access to accounts in your organization. When an account is deregistered as a delegated administrator, the account loses authorization to access all

read-only AWS Organizations API operations that provide visibility to the members and structures of your organization.

> **Note**
>
> - Deregistering a delegated administrator also automatically disables all organization-level dashboards created by the delegated administrator.
> - The delegate administrator accounts can still see the historic data for these disabled dashboards according to the respective period that data is available for queries.

**To deregister accounts for delegated administrator access**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Organization settings**.
3. In the **Accounts with delegated access** section, choose the account ID you want to deregister, and then choose **Remove**.

## Amazon S3 Storage Lens examples using the AWS CLI

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket. For more information, see Assessing storage activity and usage with Amazon S3 Storage Lens.

The following examples show how you can use S3 Storage Lens with the AWS Command Line Interface.

**Topics**

- Helper files for using Amazon S3 Storage Lens
- Using Amazon S3 Storage Lens configurations with the AWS CLI
- Using Amazon S3 Storage Lens with AWS Organizations examples using the AWS CLI

### Helper files for using Amazon S3 Storage Lens

Use the following JSON files and its key inputs for your examples.

### S3 Storage Lens example configuration in JSON

### Example `config.json`

The `config.json` file contains the details of a S3 Storage Lens Organizations-level *advanced metrics and recommendations* configuration. To use the following example, replace the *`user input placeholders`* with your own information.

> **Note**
>
> Additional charges apply for advanced metrics and recommendations. For more information, see advanced metrics and recommendations.

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
```

```
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
```

```
  }
}
```

### S3 Storage Lens example configuration with Storage Lens groups in JSON

### Example `config.json`

The `config.json` file contains the details that you want to apply to your Storage Lens configuration when using Storage Lens groups. To use the example, replace the *`user input placeholders`* with your own information.

To attach all Storage Lens groups to your dashboard, update your Storage Lens configuration with the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

To include only two Storage Lens groups in your Storage Lens dashboard configuration (*`slg-1`* and *`slg-2`*), use the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
```

```
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

To exclude only certain Storage Lens groups from being attached to your dashboard configuration, use the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Exclude": [
```

```
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

### S3 Storage Lens example tags configuration in JSON

### Example `tags.json`

The `tags.json` file contains the tags that you want to apply to your S3 Storage Lens configuration. To use this example, replace the *`user input placeholders`* with your own information.

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
]
```

### S3 Storage Lens example configuration IAM permissions

### Example `permissions.json` – Specific dashboard name

This example policy shows an S3 Storage Lens IAM `permissions.json` file with a specific dashboard name specified. Replace *`value1`*, *`us-east-1`*, *`your-dashboard-name`*, and *`example-account-id`* with your own values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
```

```
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/your-
dashboard-name"
        }
    ]
}
```

### Example `permissions.json` – No specific dashboard name

This example policy shows an S3 Storage Lens IAM `permissions.json` file without a specific dashboard name specified. Replace *value1*, *us-east-1*, and *example-account-id* with your own values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/*"
        }
    ]
}
```

### Using Amazon S3 Storage Lens configurations with the AWS CLI

You can use the AWS CLI to list, create, delete, get, tag, and update your S3 Storage Lens configurations. The following examples use the helper JSON files for key inputs. To use these examples, replace the *`user input placeholders`* with your own information.

#### Create an S3 Storage Lens configuration

#### Example Create an S3 Storage Lens configuration

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

#### Create an S3 Storage Lens configuration without tags

#### Example Create an S3 Storage Lens configuration without tags

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

#### Get an S3 Storage Lens configuration

#### Example Get an S3 Storage Lens configuration

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

#### List S3 Storage Lens configurations without a next token

#### Example List S3 Storage Lens configurations without a next token

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1
```

#### List S3 Storage Lens configurations

#### Example List S3 Storage Lens configurations

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1 --next-token=abcdefghij1234
```

### Delete an S3 Storage Lens configuration

### Example Delete an S3 Storage Lens configuration

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

### Add tags to an S3 Storage Lens configuration

### Example Add tags to an S3 Storage Lens configuration

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

### Get tags for an S3 Storage Lens configuration

### Example Get tags for an S3 Storage Lens configuration

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

### Delete tags for an S3 Storage Lens configuration

### Example Delete tags for an S3 Storage Lens configuration

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

### Using Amazon S3 Storage Lens with AWS Organizations examples using the AWS CLI

Use Amazon S3 Storage Lens to collect storage metrics and usage data for all accounts that are part of your AWS Organizations hierarchy. For more information, see Using Amazon S3 Storage Lens with AWS Organizations.

### Enable Organizations trusted access for S3 Storage Lens

### Example Enable Organizations trusted access for S3 Storage Lens

```
aws organizations enable-aws-service-access --service-principal storage-lens.s3.amazonaws.com
```

### Disable Organizations trusted access for S3 Storage Lens

### Example Disable Organizations trusted access for S3 Storage Lens

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

### Register Organizations delegated administrators for S3 Storage Lens

### Example Register Organizations delegated administrators for S3 Storage Lens

To use this example, replace *111122223333* with the appropriate AWS account ID.

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

### Deregister Organizations delegated administrators for S3 Storage Lens

### Example Deregister Organizations delegated administrators for S3 Storage Lens

To use this example, replace *111122223333* with the appropriate AWS account ID.

```
aws organizations deregister-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

## Amazon S3 Storage Lens examples using the SDK for Java

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket. For more information, see Assessing storage activity and usage with Amazon S3 Storage Lens.

The following examples show how you can use S3 Storage Lens with the AWS SDK for Java.

### Topics

- Using Amazon S3 Storage Lens configurations using the SDK for Java

### Using Amazon S3 Storage Lens configurations using the SDK for Java

You can use the SDK for Java to list, create, get and update your S3 Storage Lens configurations. The following examples use the helper JSON files for key inputs.

#### Topics



### Create and update an S3 Storage Lens configuration

#### Example Create and update an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
```

```
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
```

```
                    .withPrefixLevel(new
PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
```

```
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Delete an S3 Storage Lens configuration

## Example Delete an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;
```

```
public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Get an S3 Storage Lens configuration

### Example Get an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;

import static com.amazonaws.regions.Regions.US_WEST_2;
```

```
public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### List S3 Storage Lens configurations

### Example List S3 Storage Lens configurations

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
```

```
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "Source Account ID";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();

            System.out.println(configurations.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Add tags to an S3 Storage Lens configuration

### Example Add tags to an S3 Storage Lens configuration

```
package aws.example.s3control;
```

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Get tags for an S3 Storage Lens configuration

### Example Get tags for an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();
```

```
            System.out.println(s3Tags.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Delete tags for an S3 Storage Lens configuration

### Example Delete tags for an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationTaggingRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
```

```
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Update the default S3 Storage Lens configuration with advanced metrics and recommendations

### Example Update the default S3 Storage Lens configuration with advanced metrics and recommendations

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;
```

```
import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "Source Account ID";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
```

```
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

**Note**

Additional charges apply for advanced metrics and recommendations. For more information, see advanced metrics and recommendations.

### Attach a Storage Lens group to an S3 Storage Lens dashboard

#### Example Attach all Storage Lens groups to a dashboard

The following SDK for Java example attaches all Storage Lens groups in the account *111122223333* to the *DashBoardConfigurationId* dashboard:

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;
```

```
public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Example Attach two Storage Lens groups to a dashboard

The following AWS SDK for Java example attaches two Storage Lens groups (*StorageLensGroupName1* and *StorageLensGroupName2*) to the *ExampleDashboardConfigurationId* dashboard.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
```

```
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Example Attach all Storage Lens groups with exclusions

The following SDK for Java example attaches all Storage Lens groups to the *ExampleDashboardConfigurationId* dashboard, excluding the two specified (*StorageLensGroupName1* and *StorageLensGroupName2*):

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Using Amazon S3 Storage Lens with AWS Organizations examples using the SDK for Java

Use Amazon S3 Storage Lens to collect storage metrics and usage data for all accounts that are part of your AWS Organizations hierarchy. For more information, see Using Amazon S3 Storage Lens with AWS Organizations.

#### Topics

- Enable Organizations trusted access for S3 Storage Lens
- Disable Organizations trusted access for S3 Storage Lens
- Register Organizations delegated administrators for S3 Storage Lens
- Deregister Organizations delegated administrators for S3 Storage Lens

### Enable Organizations trusted access for S3 Storage Lens

#### Example Enable Organizations trusted access for S3 Storage Lens

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
```

```
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;

public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

### Disable Organizations trusted access for S3 Storage Lens

### Example Disable Organizations trusted access for S3 Storage Lens

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
```

```
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

### Register Organizations delegated administrators for S3 Storage Lens

### Example Register Organizations delegated administrators for S3 Storage Lens

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
```

```
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
 RegisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

### Deregister Organizations delegated administrators for S3 Storage Lens

### Example Deregister Organizations delegated administrators for S3 Storage Lens

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
```

```
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

## Working with S3 Storage Lens groups

An Amazon S3 Storage Lens group aggregates metrics using custom filters based on object metadata. Storage Lens groups help you drill down into characteristics of your data, such as distribution of objects by age, your most common file types, and more. For example, you can filter metrics by object tag to identify your fastest-growing datasets, or visualize your storage based on object size and age to inform your storage archive strategy. As a result, Amazon S3 Storage Lens groups helps you to better understand and optimize your S3 storage.

When you use Storage Lens groups, you can analyze and filter S3 Storage Lens metrics using object metadata such as prefixes, suffixes, object tags, object size, or object age. You can also apply a combination of these filters. After you attach your Storage Lens group to your S3 Storage Lens

dashboard, you can view S3 Storage Lens metrics aggregated by Amazon S3 Storage Lens groups directly in your dashboard.

For example, you can also filter your metrics by object size or age bands to determine which portion of your storage consists of small objects. You can then use this information with S3 Intelligent-Tiering or S3 Lifecycle to transition small objects to different storage classes for cost and storage optimization.

**Topics**



## How S3 Storage Lens groups work

You can use Storage Lens groups to aggregate metrics using custom filters based on object metadata. When you define a custom filter, you can use prefixes, suffixes, object tags, object sizes, object age, or a combination of these custom filters. During Storage Lens group creation, you can also include a single filter or multiple filter conditions. To specify multiple filter conditions, you use `And` or `Or` logical operators.

When you create and configure a Storage Lens group, the Storage Lens group itself acts as a custom filter in the dashboard that you attach the group to. In your dashboard, you can then use the Storage Lens group filter to obtain storage metrics based on the custom filter that you defined in the group.

To view the data for your Storage Lens group in your S3 Storage Lens dashboard, you must attach the group to the dashboard after you've created the group. After your Storage Lens group is attached to your Storage Lens dashboard, your dashboard will collect storage usage metrics within 48 hours. You can then visualize this data in the Storage Lens dashboard or export it through a metrics export. If you forget to attach a Storage Lens group to a dashboard, your Storage Lens group data won't be captured or displayed anywhere.

**Note**

- When you create a S3 Storage Lens group, you're creating an AWS resource. Therefore, each Storage Lens group has its own Amazon Resource Name (ARN), which you can specify when attaching it to or excluding it from a S3 Storage Lens dashboard.

- If your Storage Lens group isn't attached to a dashboard, you won't incur any additional charges for creating a Storage Lens group.
- S3 Storage Lens aggregates usage metrics for an object under all matching Storage Lens groups. Therefore, if an object matches the filter conditions for two or more Storage Lens groups, you will see repeated counts for the same object across your storage usage.

You can create a Storage Lens group at the account level in a specified home Region (from the list of supported AWS Regions). Then, you can attach your Storage Lens group to multiple Storage Lens dashboards, as long as the dashboards are in the same AWS account and home Region. You can create up to 50 Storage Lens groups per home Region in each AWS account.

You can create and manage S3 Storage Lens groups by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 REST API.

**Topics**



### Viewing Storage Lens group aggregated metrics

You can view the aggregated metrics for your Storage Lens groups by attaching the groups to a dashboard. The Storage Lens groups that you want to attach must reside within the designated home Region in the dashboard account.

To attach a Storage Lens group to a dashboard, you must specify the group in the **Storage Lens group aggregation** section of your dashboard configuration. If you have several Storage Lens groups, you can filter the **Storage Lens group aggregation** results to include or exclude only the groups that you want. For more information about attaching groups to your dashboards, see the section called “Attach or remove a Storage Lens group”.

After you've attached your groups, you will see the additional Storage Lens group aggregation data in your dashboard within 48 hours.

> **Note**
>
> To view aggregated metrics for your Storage Lens group, you must attach the group to an S3 Storage Lens dashboard.

### Storage Lens groups permissions

Storage Lens groups require certain permissions in AWS Identity and Access Management (IAM) to authorize access to S3 Storage Lens group actions. To grant these permissions, you can use an identity-based IAM policy. You can attach this policy to IAM users, groups, or roles to grant them permissions. Such permissions can include the ability to create or delete Storage Lens groups, view their configurations, or manage their tags.

The IAM user or role that you grant permissions to must belong to the account that created or owns the Storage Lens group.

To use Storage Lens groups and to view your Storage Lens groups metrics, you must first have the appropriate permissions to use S3 Storage Lens. For more information, see the section called “S3 Storage Lens permissions”.

To create and manage S3 Storage Lens groups, you must have the following IAM permissions, depending on which actions you want to perform:

| Action | IAM permissions |
| --- | --- |
| Create a new Storage Lens group | `s3:CreateStorageLensGroup` |
| Create a new Storage Lens group with tags | `s3:CreateStorageLensGroup` , `s3:TagResource` |
| Update an existing Storage Lens group | `s3:UpdateStorageLensGroup` |
| Return the details of a Storage Lens group configuration | `s3:GetStorageLensGroup` |
| List all Storage Lens groups in your home Region | `s3:ListStorageLensGroups` |
| Delete a Storage Lens group | `s3:DeleteStorageLensGroup` |

| Action | IAM permissions |
| --- | --- |
| List the tags that were added to your Storage Lens group | `s3:ListTagsForResource` |
| Add or update a Storage Lens group tag for an existing Storage Lens group | `s3:TagResource` |
| Delete a tag from a Storage Lens group | `s3:UntagResource` |

Here's an example of how to configure your IAM policy in the account that creates the Storage Lens group. To use this policy, replace *us-east-1* with the home Region that your Storage Lens group is located in. Replace *111122223333* with your AWS account ID, and replace *example-storage-lens-group* with the name of your Storage Lens group. To apply these permissions to all Storage Lens groups, replace *example-storage-lens-group* with an *.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup,
                "s3:TagResource",
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-storage-lens-group"
        }
    ]
}
```

For more information about S3 Storage Lens permissions, see Amazon S3 Storage Lens permissions. For more information about IAM policy language, see Policies and permissions in Amazon S3.

**Storage Lens groups configuration**

**S3 Storage Lens group name**

We recommend giving your Storage Lens groups names that indicate their purpose so that you can easily determine which groups you want to attach to your dashboards. To attach a Storage Lens group to a dashboard, you must specify the group in the **Storage Lens group aggregation** section of the dashboard configuration.

Storage Lens group names must be unique within the account. They must not exceed 64 characters, and can contain only letters (a-z, A-Z), numbers (0-9), hyphens (-), and underscores (_).

**Home Region**

The home Region is the AWS Region where your Storage Lens group is created and maintained. Your Storage Lens group is created in the same home Region as your Amazon S3 Storage Lens dashboard. The Storage Lens group configuration and metrics are also stored in this Region. You can create up to 50 Storage Lens groups in a home Region.

After you create your Storage Lens group, you can't edit the home Region.

**Scope**

To include objects in your Storage Lens group, they must be in scope for your Amazon S3 Storage Lens dashboard. The scope of your Storage Lens dashboard is determined by the buckets that you included in the **Dashboard scope** of your S3 Storage Lens dashboard configuration.

You can use different filters for your objects to define the scope of your Storage Lens group. To view these Storage Lens group metrics in your S3 Storage Lens dashboard, objects must match the filters that you include in your Storage Lens groups. For example, suppose that your Storage Lens group includes objects with the prefix `marketing` and the suffix `.png`, but no objects match those criteria. In this case, metrics for this Storage Lens group won't be generated in your daily metrics export, and no metrics for this group will be visible in your dashboard.

**Filters**

You can use the following filters in an S3 Storage Lens group:

- **Prefixes** – Specifies the prefix of included objects, which is a string of characters at the beginning of the object key name. For example, a value of `images` for the **Prefixes** filter includes objects with any of the following prefixes: `images/`, `images-marketing`, and `images/production`. The maximum length of a prefix is 1,024 bytes.
- **Suffixes** – Specifies the suffix of included objects (for example, `.png`, `.jpeg`, or `.csv`). The maximum length of a suffix is 1,024 bytes.
- **Object tags** – Specifies the list of object tags that you want to filter on. A tag key can't exceed 128 Unicode characters, and a tag value can't exceed 256 Unicode characters. Note that if the object tag value field is left empty, S3 Storage Lens groups only matches the object to other objects that also have empty tag values.
- **Age** – Specifies the object age range of included objects in days. Only integers are supported.
- **Size** – Specifies the object size range of included objects in bytes. Only integers are supported. The maximum allowable value is 5 TB.

### Storage Lens group object tags

You can create a Storage Lens group that includes up to 10 object tag filters. The following example includes two object tag key-value pairs as filters for a Storage Lens group that's named *`Marketing-Department`*. To use this example, replace *`Marketing-Department`* with the name of your group, and replace *`object-tag-key-1`*, *`object-tag-value-1`*, and so on with the object tag key-value pairs that you want to filter on.

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
                }
            ]
    }
}
```

### Logical operators (And or Or)

To include multiple filter conditions in your Storage Lens group, you can use logical operators (either And or Or). In the following example, the Storage Lens group that's named *Marketing-Department* has an And operator that contains Prefix, ObjectAge, and ObjectSize filters. Because an And operator is used, only objects that match **all** of these filter conditions will be included the Storage Lens group's scope.

To use this example, replace the *user input placeholders* with the values that you want to filter on.

```
{
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> If you want to include objects that match **any** of the conditions in the filters, replace the And logical operator with the Or logical operator in this example.

### AWS resource tags

Each S3 Storage Lens group is counted as an AWS resource with its own Amazon Resource Name (ARN). Therefore, when you configure your Storage Lens group, you can optionally add AWS resource tags to the group. You can add up to 50 tags for each Storage Lens group. To create a Storage Lens group with tags, you must have the `s3:CreateStorageLensGroup` and `s3:TagResource` permissions.

You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it. You can also use tags to track and allocate costs.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

You can edit tag keys and values, and you can remove tags from a resource at any time. Also, be aware of the following limitations:

- Tag keys and tag values are case sensitive.
- If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.
- If you delete a resource, any tags for the resource are also deleted.
- Don't include private or sensitive data in your AWS resource tags.
- System tags (or tags with tag keys that begin with `aws:`) aren't supported.
- The length of each tag key can't exceed 128 characters. The length of each tag value can't exceed 256 characters.

### Storage Lens groups metrics export

S3 Storage Lens group metrics are included in the Amazon S3 Storage Lens metrics export for the dashboard that the Storage Lens group is attached to. For general information about the Storage Lens metrics export feature, see Viewing Amazon S3 Storage Lens metrics using a data export.

Your metrics export for Storage Lens groups includes any S3 Storage Lens metrics that are in scope for the dashboard that you attached the Storage Lens group to. The export also includes additional metrics data for Storage Lens groups.

After you create your Storage Lens group, your metrics export is sent daily to the bucket that you selected when you configured the metrics export for the dashboard that your group is attached to. It can take up to 48 hours for you to receive the first metrics export.

To generate metrics in the daily export, objects must match the filters that you include in your Storage Lens groups. If no objects match the filters that you included in your Storage Lens group, then no metrics will be generated. However, if an object matches two or more Storage Lens groups, the object is listed separately for each group when it appears in the metrics export.

You can identify metrics for Storage Lens groups by looking for one of the following values in the `record_type` column of the metrics export for your dashboard:

- `STORAGE_LENS_GROUP_BUCKET`
- `STORAGE_LENS_GROUP_ACCOUNT`

The `record_value` column displays the resource ARN for the Storage Lens group (for example, `arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-Department`).

## Using Storage Lens groups

Amazon S3 Storage Lens groups aggregates metrics using custom filters based on object metadata. You can analyze and filter S3 Storage Lens metrics using prefixes, suffixes, object tags, object size, or object age. With Amazon S3 Storage Lens groups, you can also categorize your usage within and across Amazon S3 buckets. As a result, you'll be able to better understand and optimize your S3 storage.

To start visualizing the data for a Storage Lens group, you must first attach your Storage Lens group to an S3 Storage Lens dashboard. If you need to manage Storage Lens groups in the dashboard, you can edit the dashboard configuration. To check which Storage Lens groups are under your account, you can list them. To check which Storage Lens groups are attached to your dashboard, you can always check the **Storage Lens groups** tab in the dashboard. To review or update the scope of an existing Storage Lens group, you can view its details. You can also permanently delete a Storage Lens group.

To manage permissions, you can create and add user-defined AWS resource tags to your Storage Lens groups. You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

**Topics**



### Creating a Storage Lens group

The following examples demonstrate how to create an Amazon S3 Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To create a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Storage Lens groups**.
4. Choose **Create Storage Lens group**.

5. Under **General**, view your **Home Region** and enter your **Storage Lens group name**.
6. Under **Scope**, choose the filter that you want to apply to your Storage Lens group. To apply multiple filters, choose your filters, and then choose the **AND** or **OR** logical operator.

   - For the **Prefixes** filter, choose **Prefixes**, and enter a prefix string. To add multiple prefixes, choose **Add prefix**. To remove a prefix, choose **Remove** next to the prefix that you want to remove.
   - For the **Object tags** filter, choose **Object tags**, and enter the key-value pair for your object. Then, choose **Add tag**. To remove a tag, choose **Remove** next to the tag that you want to remove.
   - For the **Suffixes** filter, choose **Suffixes**, and enter a suffix string. To add multiple suffixes, choose **Add suffix**. To remove a suffix, choose **Remove** next to the suffix that you want to remove.
   - For the **Age** filter, specify the object age range in days. Choose **Specify minimum object age**, and enter the minimum object age. Then, choose **Specify maximum object age**, and enter the maximum object age.
   - For the **Size** filter, specify the object size range and unit of measurement. Choose **Specify minimum object size**, and enter the minimum object size. Choose **Specify maximum object size**, and enter the maximum object size.
7. (Optional) For AWS resource tags, add the key-value pair, and then choose **Add tag**.
8. Choose **Create Storage Lens group**.

### Using the AWS CLI

The following example AWS CLI command creates a Storage Lens group. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

The following example AWS CLI command creates a Storage Lens group with two AWS resource tags. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
```

```
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

For example JSON configurations, see Storage Lens groups configuration.

### Using the AWS SDK for Java

The following AWS SDK for Java example creates a Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

### Example – Create a Storage Lens group with a single filter

The following example creates a Storage Lens group named *Marketing-Department*. This group has an object age filter that specifies the age range as *30* to *90* days. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;

public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90)
                            .build())
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(objectAgeFilter)
```

```
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Example – Create a Storage Lens group with an AND operator that includes multiple filters

The following example creates a Storage Lens group named *Marketing-Department*. This group uses the AND operator to indicate that objects must match **all** of the filter conditions. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
```

```
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
 StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
```

```
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

**Example – Create a Storage Lens group with an OR operator that includes multiple filters**

The following example creates a Storage Lens group named *Marketing-Department*. This group uses an OR operator to apply a prefix filter (*prefix-1*, *prefix-2*, *prefix3/sub-prefix-1*) or an object size filter with a size range between *1000* bytes and *6000* bytes. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";
```

```
        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Example – Create a Storage Lens group with a single filter and two AWS resource tags

The following example creates a Storage Lens group named *Marketing-Department* that has a suffix filter. This example also adds two AWS resource tags to the Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();
```

```
            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For example JSON configurations, see Storage Lens groups configuration.

**Attaching or removing S3 Storage Lens groups to or from your dashboard**

After you've upgraded to the advanced tier in Amazon S3 Storage Lens, you can attach a Storage Lens group to your dashboard. If you have several Storage Lens groups, you can include or exclude the groups that you want.

Your Storage Lens groups must reside within the designated home Region in the dashboard account. After you attach a Storage Lens group to your dashboard, you'll receive the additional Storage Lens group aggregation data in your metrics export within 48 hours.

> **Note**
>
> If you want to view aggregated metrics for your Storage Lens group, you must attach it to your Storage Lens dashboard. For examples of Storage Lens group JSON configuration files, see S3 Storage Lens example configuration with Storage Lens groups in JSON.

### Attaching a Storage Lens group to an S3 Storage Lens dashboard

#### To attach a Storage Lens group to a Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Storage Lens**, choose **Dashboards**.
3. Choose the option button for the Storage Lens dashboard that you want to attach a Storage Lens group to.
4. Choose **Edit**.
5. Under **Metrics selection**, choose **Advanced metrics and recommendations**.
6. Select **Storage Lens group aggregation**.

> **Note**
>
> By default, **Advanced metrics** is also selected. However, you can also deselect this setting as it's not required to aggregate Storage Lens groups data.

7. Scroll down to **Storage Lens group aggregation** and specify the Storage Lens group or groups that you either want to include or exclude in the data aggregation. You can use the following filtering options:

   - If you want to include certain Storage Lens groups, choose **Include Storage Lens groups**. Under **Storage Lens groups to include**, select your Storage Lens groups.
   - If you want to include all Storage Lens groups, select **Include all Storage Lens groups in home Region in this account**.
   - If you want to exclude certain Storage Lens groups, choose **Exclude Storage Lens groups**. Under **Storage Lens groups to exclude**, select the Storage Lens groups that you want to exclude.

8. Choose **Save changes**. If you've configured your Storage Lens groups correctly, you will see the additional Storage Lens group aggregation data in your dashboard within 48 hours.

### Removing a Storage Lens group from an S3 Storage Lens dashboard

#### To remove a Storage Lens group from an S3 Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Storage Lens**, choose **Dashboards**.
3. Choose the option button for the Storage Lens dashboard that you want to remove a Storage Lens group from.
4. Choose **View dashboard configuration**.
5. Choose **Edit**.
6. Scroll down to the **Metrics selection** section.
7. Under **Storage Lens group aggregation**, choose the **X** next to the Storage Lens group that you want to remove. This removes your Storage Lens group.

   If you included all of your Storage Lens groups in your dashboard, clear the check box next to **Include all Storage Lens groups in home Region in this account**.
8. Choose **Save changes**.

> **Note**
>
> It will take up to 48 hours for your dashboard to reflect the configuration updates.

### Visualizing your Storage Lens groups data

You can visualize your Storage Lens groups data by attaching the group to your Amazon S3 Storage Lens dashboard. After you've included the Storage Lens group in the Storage Lens group aggregation in your dashboard configuration, it can take up to 48 hours for the Storage Lens group data to display in your dashboard.

After the dashboard configuration has been updated, any newly attached Storage Lens groups appear in the list of available resources under the **Storage Lens groups** tab. You can also further analyze storage usage in your **Overview** tab by slicing the data by another dimension. For example, you can choose one of the items listed under the **Top 3** categories and choose **Analyze by** to slice the data by another dimension. You can't apply the same dimension as the filter itself.

> **Note**
>
> You can't apply a Storage Lens group filter along with a prefix filter, or the reverse. You also can't further analyze a Storage Lens group by using a prefix filter.

You can use the **Storage Lens groups** tab in the Amazon S3 Storage Lens dashboard to customize the data visualization for the Storage Lens groups that are attached to your dashboard. You can either visualize the data for some Storage Lens groups that are attached to your dashboard, or all of them.

When visualizing Storage Lens group data in your S3 Storage Lens dashboard, be aware of the following:

- S3 Storage Lens aggregates usage metrics for an object under all matching Storage Lens groups. Therefore, if an object matches the filter conditions for two or more Storage Lens groups, you will see repeated counts for the same object across your storage usage.
- Objects must match the filters that you include in your Storage Lens groups. If no objects match the filters that you include in your Storage Lens group, then no metrics are generated. To determine if there are any unassigned objects, check your total object count in the dashboard at the account level and bucket level.

### Updating a Storage Lens group

The following examples demonstrate how to update an Amazon S3 Storage Lens group. You can update a Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To update a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **Scope**, choose **Edit**.

5. On the **Scope** page, select the filter that you want to apply to your Storage Lens group. To apply multiple filters, select your filters, and choose the **AND** or **OR** logical operator.

   - For the **Prefixes** filter, select **Prefixes**, and enter a prefix string. To add multiple prefixes, choose **Add prefix**. To remove a prefix, choose **Remove** next to the prefix that you want to remove.
   - For the **Object tags** filter, enter the key-value pair for your object. Then, choose **Add tag**. To remove a tag, choose **Remove** next to the tag that you want to remove.
   - For the **Suffixes** filter, select **Suffixes**, and enter a suffix string. To add multiple suffixes, choose **Add suffix**. To remove a suffix, choose **Remove** next to the suffix that you want to remove.
   - For the **Age** filter, specify the object age range in days. Choose **Specify minimum object age**, and enter the minimum object age. For **Specify maximum object age**, enter the maximum object age.
   - For the **Size** filter, specify the object size range and unit of measurement. Choose **Specify minimum object size**, and enter the minimum object size. For **Specify maximum object size**, enter the maximum object size.

6. Choose **Save changes**. The details page for the Storage Lens group appears.
7. (Optional) If you want to add a new AWS resource tag, scroll to the **AWS resource tags** section, then choose **Add tags**. The **Add tags** page appears.

   Add the new key-value pair, then choose **Save changes**. The details page for the Storage Lens group appears.
8. (Optional) If you want to remove an existing AWS resource tag, scroll to the **AWS resource tags** section, and select the resource tag. Then, choose **Delete**. The **Delete AWS tags** dialog box appears.

   Choose **Delete** again to permanently delete the AWS resource tag.

   > **Note**
   >
   > After you permanently delete an AWS resource tag, it can't be restored.

### Using the AWS CLI

The following AWS CLI example command returns the configuration details for a Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

The following AWS CLI example updates a Storage Lens group. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control update-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

For example JSON configurations, see Storage Lens groups configuration.

### Using the AWS SDK for Java

The following AWS SDK for Java example returns the configuration details for the *Marketing-Department* Storage Lens group in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
```

```
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

The following example updates the Storage Lens group named *Marketing-Department* in account *111122223333*. This example updates the dashboard scope to include objects that match any of the following suffixes: *.png*, *.gif*, *.jpg*, or *.jpeg*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
```

```
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For example JSON configurations, see Storage Lens groups configuration.

### Managing AWS resource tags with Storage Lens groups

Each Amazon S3 Storage Lens group is counted as an AWS resource with its own Amazon Resource Name (ARN). Therefore, when you configure your Storage Lens group, you can optionally add AWS resource tags to the group. You can add up to 50 tags for each Storage Lens group. To create a Storage Lens group with tags, you must have the `s3:CreateStorageLensGroup` and `s3:TagResource` permissions.

You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it. You can also use tags to track and allocate costs.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

You can edit tag keys and values, and you can remove tags from a resource at any time. Also, be aware of the following limitations:

- Tag keys and tag values are case sensitive.
- If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.
- If you delete a resource, any tags for the resource are also deleted.
- Don't include private or sensitive data in your AWS resource tags.
- System tags (with tag keys that begin with `aws:`) aren't supported.
- The length of each tag key can't exceed 128 characters. The length of each tag value can't exceed 256 characters.

The following examples demonstrate how to use AWS resource tags with Storage Lens groups.

**Topics**



### Adding an AWS resource tag to a Storage Lens group

The following examples demonstrate how to add AWS resource tags to an Amazon S3 Storage Lens group. You can add resource tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

#### To add an AWS resource tag to a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, choose **Add tags**.
5. On the **Add tags** page, add the new key-value pair.

> **Note**
>
> Adding a new tag with the same key as an existing tag overwrites the previous tag value.

6. (Optional) To add more than one new tag, choose **Add tag** again to continue adding new entries. You can add up to 50 AWS resource tags to your Storage Lens group.
7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove.
8. Choose **Save changes**.

### Using the AWS CLI

The following example AWS CLI command adds two resource tags to an existing Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

### Using the AWS SDK for Java

The following AWS SDK for Java example adds two AWS resource tags to an existing Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

### Updating Storage Lens group tag values

The following examples demonstrate how to update Storage Lens group tag values by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To update an AWS resource tag for a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, select the tag that you want to update.
5. Add the new tag value, using the same key of the key-value pair that you want to update. Choose the checkmark icon to update the tag value.

   > **Note**
   >
   > Adding a new tag with the same key as an existing tag overwrites the previous tag value.

6. (Optional) If you want to add new tags, choose **Add tag** to add new entries. The **Add tags** page appears.

   You can add up to 50 AWS resource tags for your Storage Lens group. When you're finished adding new tags, choose **Save changes**.
7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove. When you're finished removing tags, choose **Save changes**.

#### Using the AWS CLI

The following example AWS CLI command updates two tag values for the Storage Lens group named *`marketing-department`*. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

### Using the AWS SDK for Java

The following AWS SDK for Java example updates two Storage Lens group tag values. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Deleting an AWS resource tag from a Storage Lens group

The following examples demonstrate how to delete an AWS resource tag from a Storage Lens group. You can delete tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To delete an AWS resource tag from a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, select the key-value pair that you want to delete.
5. Choose **Delete**. The **Delete AWS resource tags** dialog box appears.

   > **Note**
   >
   > If tags are used to control access, proceeding with this action can affect related resources. After you permanently delete a tag, it can't be restored.

6. Choose **Delete** to permanently delete the key-value pair.

### Using the AWS CLI

The following AWS CLI command deletes two AWS resource tags from an existing Storage Lens group: To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control untag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
--region us-east-1 --tag-keys k1 k2
```

### Using the AWS SDK for Java

The following AWS SDK for Java example deletes two AWS resource tags from the Storage Lens group Amazon Resource Name (ARN) that you specify in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
```

```
                    .build();
            s3ControlClient.untagResource(untagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Listing Storage Lens group tags

The following examples demonstrate how to list the AWS resource tags associated with a Storage Lens group. You can list tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

**To review the list of tags and tag values for a Storage Lens group**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you're interested in.
4. Scroll down to the **AWS resource tags** section. All of the user-defined AWS resource tags that are added to your Storage Lens group are listed along with their tag values.

#### Using the AWS CLI

The following AWS CLI example command lists all the Storage Lens group tag values for the Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
```

```
--region us-east-1
```

### Using the AWS SDK for Java

The following AWS SDK for Java example lists the Storage Lens group tag values for the Storage Lens group Amazon Resource Name (ARN) that you specify. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

### Listing all Storage Lens groups

The following examples demonstrate how to list all Amazon S3 Storage Lens groups in an AWS account and home Region. These examples show how list all Storage Lens groups by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To list all Storage Lens groups in an account and home Region

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, the list of Storage Lens groups in your account is displayed.

#### Using the AWS CLI

The following AWS CLI example lists all of the Storage Lens groups for your account. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

#### Using the AWS SDK for Java

The following AWS SDK for Java example lists the Storage Lens groups for account *`111122223333`*. To use this example, replace the *`user input placeholders`* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
```

```
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;

public class ListStorageLensGroups {
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Viewing Storage Lens group details

The following examples demonstrate how to view Amazon S3 Storage Lens group configuration details. You can view these details by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

#### To view Storage Lens group configuration details

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the option button next to the Storage Lens group that you're interested in.
4. Choose **View details**. You can now review the details of your Storage Lens group.

### Using the AWS CLI

The following AWS CLI example returns the configuration details for a Storage Lens group. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

### Using the AWS SDK for Java

The following AWS SDK for Java example returns the configuration details for the Storage Lens group named *Marketing-Department* in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
```

```
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### Deleting a Storage Lens group

The following examples demonstrate how to delete an Amazon S3 Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

#### To delete a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the option button next to the Storage Lens group that you want to delete.
4. Choose **Delete**. A **Delete Storage Lens group** dialog box displays.
5. Choose **Delete** again to permanently delete your Storage Lens group.

> **Note**
>
> After you delete a Storage Lens group, it can't be restored.

### Using the AWS CLI

The following AWS CLI example deletes the Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

### Using the AWS SDK for Java

The following AWS SDK for Java example deletes the Storage Lens group named *Marketing-Department* in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

# Tracing Amazon S3 requests using AWS X-Ray

AWS X-Ray collects data about requests that your application serves. You can then view and filter the data to identify and troubleshoot performance issues and errors in your distributed applications and micro-services architecture. For any traced request to your application, it shows you detailed information about the request, the response, and the calls that your application makes to downstream AWS resources, micro-services, databases, and HTTP web APIs.

For more information, see What is AWS X-Ray? in the *AWS X-Ray Developer Guide*.

**Topics**

- How X-Ray works with Amazon S3
- Available Regions

## How X-Ray works with Amazon S3

AWS X-Ray supports trace context propagation for Amazon S3, so you can view end-to-end requests as they travel through your entire application. X-Ray aggregates the data that is generated by the individual services such as Amazon S3, AWS Lambda, and Amazon EC2, and the many resources that make up your application. It provides you with an overall view of how your application is performing.

Amazon S3 integrates with X-Ray to propagate trace context and give you one request chain with upstream and downstream nodes. If an upstream service includes a valid-formatted trace header with its S3 request, Amazon S3 passes the trace header when delivering event notifications to downstream services such as Lambda, Amazon SQS, and Amazon SNS. If you have all these services actively integrated with X-Ray, they are linked in one request chain to give you the complete details of your Amazon S3 requests.

To send X-Ray trace headers through Amazon S3, you must include a formatted X-Amzn-Trace-Id in your requests. You can also instrument the Amazon S3 client using the AWS X-Ray SDKs. For a list of the supported SDKs, see the AWS X-Ray documentation.

### Service maps

X-Ray *service maps* show you the relationships between Amazon S3 and other AWS services and resources in your application in near-real time. To see the end-to-end requests using the X-Ray service maps, you can use the X-Ray console to view a map of the connections between Amazon S3 and other services that your application uses. You can easily detect where high latency is occurring, visualize node distribution for these services, and then drill down into the specific services and paths impacting application performance.

### X-Ray Analytics

You can also use the X-Ray Analytics console to analyze traces, view metrics such as latency and failure rates, and generate insights to help you identify and troubleshoot issues. This console also shows you metrics such as average latency and failure rates. For more information, see AWS X-Ray console in the *AWS X-Ray Developer Guide*.

## Available Regions

AWS X-Ray support for Amazon S3 is available in all AWS X-Ray Regions. For more information, see Amazon S3 and AWS X-Ray in the *AWS X-Ray Developer Guide*.

# Hosting a static website using Amazon S3

You can use Amazon S3 to host a static website. On a *static* website, individual webpages include static content. They might also contain client-side scripts.

By contrast, a *dynamic* website relies on server-side processing, including server-side scripts, such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting, but AWS has other resources for hosting dynamic websites. To learn more about website hosting on AWS, see Web Hosting.

**Note**

You can use the AWS Amplify Console to host a single-page web app. The AWS Amplify Console supports single-page apps built with single-page app frameworks (for example, React JS, Vue JS, Angular JS, and Nuxt) and static site generators (for example, Gatsby JS, React-static, Jekyll, and Hugo). For more information, see Getting Started in the *AWS Amplify Console User Guide*.

Amazon S3 website endpoints do not support HTTPS. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3. For more information, see How do I use CloudFront to serve HTTPS requests for my Amazon S3 bucket? To use HTTPS with a custom domain, see Configuring a static website using a custom domain registered with Route 53.

For more information about hosting a static website on Amazon S3, including instructions and step-by-step walkthroughs, see the following topics.

**Topics**

# Website endpoints

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Website endpoints are different from the endpoints where you send REST API requests. For more information about the differences between the endpoints, see Key differences between a website endpoint and a REST API endpoint.

Depending on your Region, your Amazon S3 website endpoint follows one of these two formats.

- **s3-website dash (-) Region** - `http://bucket-name.s3-website-Region.amazonaws.com`
- **s3-website dot (.) Region** - `http://bucket-name.s3-website.Region.amazonaws.com`

These URLs return the default index document that you configure for the website. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website Endpoints.

**Note**

To augment the security of your Amazon S3 static websites, the Amazon S3 website endpoint domains (for example, *s3-website-us-east-1.amazonaws.com* or *s3-website.ap-south-1.amazonaws.com*) are registered in the Public Suffix List (PSL). For further security, we recommend that you use cookies with a `__Host-` prefix if you ever need to set sensitive cookies in the domain name for your Amazon S3 static websites. This practice will help to defend your domain against cross-site request forgery attempts (CSRF). For more information see the Set-Cookie page in the Mozilla Developer Network.

If you want your website to be public, you must make all your content publicly readable for your customers to be able to access it at the website endpoint. For more information, see Setting permissions for website access.

**Important**

Amazon S3 website endpoints do not support HTTPS or access points. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3. For more information, see How do I use CloudFront to serve HTTPS requests for my Amazon S3 bucket? To use HTTPS with a custom domain, see Configuring a static website using a custom domain registered with Route 53.

Requester Pays buckets do not allow access through a website endpoint. Any request to such a bucket receives a 403 Access Denied response. For more information, see Using Requester Pays buckets for storage transfers and usage.

**Topics**



## Website endpoint examples

The following examples show how you can access an Amazon S3 bucket that is configured as a static website.

**Example — Requesting an object at the root level**

To request a specific object that is stored at the root level in the bucket, use the following URL structure.

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

For example, the following URL requests the `photo.jpg` object that is stored at the root level in the bucket.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

**Example — Requesting an object in a prefix**

To request an object that is stored in a folder in your bucket, use this URL structure.

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

The following URL requests the `docs/doc1.html` object in your bucket.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

## Adding a DNS CNAME

If you have a registered domain, you can add a DNS CNAME entry to point to the Amazon S3 website endpoint. For example, if you registered the domain `www.example-bucket.com`, you could create a bucket `www.example-bucket.com`, and add a DNS CNAME record that points to `www.example-bucket.com.s3-website.`*`Region`*`.amazonaws.com`. All requests to `http://www.example-bucket.com` are routed to `www.example-bucket.com.s3-website.`*`Region`*`.amazonaws.com`.

For more information, see Customizing Amazon S3 URLs with CNAME records.

## Using a custom domain with Route 53

Instead of accessing the website using an Amazon S3 website endpoint, you can use your own domain registered with Amazon Route 53 to serve your content—for example, `example.com`. You can use Amazon S3 with Route 53 to host a website at the root domain. For example, if you have the root domain `example.com` and you host your website on Amazon S3, your website visitors can access the site from their browser by entering either `http://www.example.com` or `http://example.com`.

For an example walkthrough, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

## Key differences between a website endpoint and a REST API endpoint

An Amazon S3 website endpoint is optimized for access from a web browser. The following table summarizes the key differences between a REST API endpoint and a website endpoint.

| Key difference | REST API endpoint | Website endpoint |
|---|---|---|
| Access control | Supports both public and private content | Supports only publicly readable content |
| Error message handling | Returns an XML-formatted error response | Returns an HTML document |
| Redirection support | Not applicable | Supports both object-level and bucket-level redirects |

| Key difference | REST API endpoint | Website endpoint |
|---|---|---|
| Requests supported | Supports all bucket and object operations | Supports only GET and HEAD requests on objects |
| Responses to GET and HEAD requests at the root of a bucket | Returns a list of the object keys in the bucket | Returns the index document that is specified in the website configuration |
| Secure Sockets Layer (SSL) support | Supports SSL connections | Does not support SSL connections |

For a complete list of Amazon S3 endpoints, see Amazon S3 endpoints and quotas in the *AWS General Reference*.

# Enabling website hosting

When you configure a bucket as a static website, you must enable static website hosting, configure an index document, and set permissions.

You can enable static website hosting using the Amazon S3 console, REST API, the AWS SDKs, the AWS CLI, or AWS CloudFormation.

To configure your website with a custom domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

## Using the S3 console

### To enable static website hosting

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to enable static website hosting for.
3. Choose **Properties**.

4. Under **Static website hosting**, choose **Edit**.
5. Choose **Use this bucket to host a website**.
6. Under **Static website hosting**, choose **Enable**.
7. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.
8. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.
9. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

   For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.
10. Choose **Save changes**.

    Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.
11. Under **Static website hosting**, note the **Endpoint**.

    The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

## Using the REST API

For more information about sending REST requests directly to enable static website hosting, see the following sections in the Amazon Simple Storage Service API Reference:

- PUT Bucket website
- GET Bucket website
- DELETE Bucket website

## Using the AWS SDKs

To host a static website on Amazon S3, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket. You can also use the AWS SDKs to create, update, and delete the website configuration programmatically. The SDKs provide wrapper classes around the Amazon S3 REST API. If your application requires it, you can send REST API requests directly from your application.

.NET

The following example shows how to use the AWS SDK for .NET to manage website configuration for a bucket. To add a website configuration to a bucket, you provide a bucket name and a website configuration. The website configuration must include an index document and can contain an optional error document. These documents must be stored in the bucket. For more information, see PUT Bucket website. For more information about the Amazon S3 website feature, see Hosting a static website using Amazon S3.

The following C# code example adds a website configuration to the specified bucket. The configuration specifies both the index document and the error document names. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; //
 For example, index.html.
```

```
        private const string errorDocument = "*** error object key ***"; // For
example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                               string indexDocumentSuffix,
                                               string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
getResponse.WebsiteConfiguration.ErrorDocument);
            }
```

```
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

The following PHP example adds a website configuration to the specified bucket. The `create_website_config` method explicitly provides the index document and error document names. The example also retrieves the website configuration and prints the response. For more information about the Amazon S3 website feature, see Hosting a static website using Amazon S3.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'                => $bucket,
    'WebsiteConfiguration'  => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
```

```
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

### Using the AWS CLI

For more information about using the AWS CLI to configure an S3 bucket as a static website, see website in the *AWS CLI Command Reference*.

Next, you must configure your index document and set permissions. For information, see Configuring an index document and Setting permissions for website access.

You can also optionally configure an error document, web traffic logging, or a redirect.

## Configuring an index document

When you enable website hosting, you must also configure and upload an index document. An *index document* is a webpage that Amazon S3 returns when a request is made to the root of a website or any subfolder. For example, if a user enters `http://www.example.com` in the browser, the user is not requesting any specific page. In that case, Amazon S3 serves up the index document, which is sometimes referred to as the *default page*.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, `index.html`). After you enable static website hosting for your bucket, you upload an HTML file with the index document name to your bucket.

The trailing slash at the root-level URL is optional. For example, if you configure your website with `index.html` as the index document, either of the following URLs returns `index.html`.

```
http://example-bucket.s3-website.Region.amazonaws.com/
```

```
http://example-bucket.s3-website.Region.amazonaws.com
```

For more information about Amazon S3 website endpoints, see Website endpoints.

## Index document and folders

In Amazon S3, a bucket is a flat container of objects. It does not provide any hierarchical organization as the file system on your computer does. However, you can create a logical hierarchy by using object key names that imply a folder structure.

For example, consider a bucket with three objects that have the following key names. Although these are stored with no physical hierarchical organization, you can infer the following logical folder structure from the key names:

- `sample1.jpg` — Object is at the root of the bucket.
- `photos/2006/Jan/sample2.jpg` — Object is in the `photos/2006/Jan` subfolder.
- `photos/2006/Feb/sample3.jpg` — Object is in the `photos/2006/Feb` subfolder.

In the Amazon S3 console, you can also create a folder in a bucket. For example, you can create a folder named `photos`. You can upload objects to the bucket or to the `photos` folder within the bucket. If you add the object `sample.jpg` to the bucket, the key name is `sample.jpg`. If you upload the object to the `photos` folder, the object key name is `photos/sample.jpg`.

If you create a folder structure in your bucket, you must have an index document at each level. In each folder, the index document must have the same name, for example, `index.html`. When a user specifies a URL that resembles a folder lookup, the presence or absence of a trailing slash determines the behavior of the website. For example, the following URL, with a trailing slash, returns the `photos/index.html` index document.

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

However, if you exclude the trailing slash from the preceding URL, Amazon S3 first looks for an object `photos` in the bucket. If the `photos` object is not found, it searches for an index document, `photos/index.html`. If that document is found, Amazon S3 returns a `302 Found` message and points to the `photos/` key. For subsequent requests to `photos/`, Amazon S3 returns `photos/index.html`. If the index document is not found, Amazon S3 returns an error.

# Configure an index document

To configure an index document using the S3 console, use the following procedure. You can also configure an index document using the REST API, the AWS SDKs, the AWS CLI, or AWS CloudFormation.

> **Note**
>
> In a versioning-enabled bucket, you may upload multiple copies of the `index.html` but only the newest version will be resolved to. For more information about using S3 Versioning see, Using versioning in S3 buckets.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

**To configure the index document**

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.
5. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

   After enabling static website hosting, proceed to step 6.
6. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.
7. (Optional) Upload other website content to your bucket.

Next, you must set permissions for website access. For information, see Setting permissions for website access.

You can also optionally configure an error document, web traffic logging, or a redirect.

# Configuring a custom error document

After you configure your bucket as a static website, when an error occurs, Amazon S3 returns an HTML error document. You can optionally configure your bucket with a custom error document so that Amazon S3 returns that document when an error occurs.

> **Note**
>
> Some browsers display their own error message when an error occurs, ignoring the error document that Amazon S3 returns. For example, when an HTTP 404 Not Found error occurs, Google Chrome might ignore the error document that Amazon S3 returns and display its own error.

**Topics**

## Amazon S3 HTTP response codes

The following table lists the subset of HTTP response codes that Amazon S3 returns when an error occurs.

| HTTP error code | Description |
|---|---|
| **301 Moved Permanently** | When a user sends a request directly to the Amazon S3 website endpoint (`http://s3-website.` *`Region`*`.amazonaws.com/`), Amazon S3 returns a **301 Moved Permanently** response and redirects those requests to `https://aws.amazon.com/s3/`. |
| **302 Found** | When Amazon S3 receives a request for a key `x`, `http://`*`bucket-name`*`.s3-website.` *`Region`*`.amazonaws.com/x`, without a trailing slash, it first looks for the object with the key name `x`. If the object is not found, Amazon S3 determines that the request is for subfolder `x` and redirects the request by adding a slash at the end, and returns **302 Found**. |
| **304 Not Modified** | Amazon S3 uses request headers `If-Modified-Since`, `If-Unmodified-Since`, `If-Match` and/or `If-None-Match` to determine whether the requested object is same as the cached copy held by the client. If the object is the same, the website endpoint returns a **304 Not Modified** response. |
| **400 Malformed Request** | The website endpoint responds with a **400 Malformed Request** when a user attempts to access a bucket through the incorrect regional endpoint. |
| **403 Forbidden** | The website endpoint responds with a **403 Forbidden** when a user request translates to an object that is not publicly readable. The object owner must make the object publicly readable using a bucket policy or an ACL. |
| **404 Not Found** | |

| HTTP error code | Description |
| --- | --- |
| | The website endpoint responds with **404 Not Found** for the following reasons:<br>• Amazon S3 determines that the URL of the website refers to an object key that does not exist.<br>• Amazon S3 infers that the request is for an index document that does not exist.<br>• A bucket specified in the URL does not exist.<br>• A bucket specified in the URL exists, but isn't configured as a website.<br><br>You can create a custom document that is returned for **404 Not Found**. Make sure that the document is uploaded to the bucket configured as a website, and that the website hosting configuration is set to use the document.<br><br>For information on how Amazon S3 interprets the URL as a request for an object or an index document, see Configuring an index document. |
| **500 Service Error** | The website endpoint responds with a **500 Service Error** when an internal server error occurs. |
| **503 Service Unavailable** | The website endpoint responds with a **503 Service Unavailable** when Amazon S3 determines that you need to reduce your request rate. |

For each of these errors, Amazon S3 returns a predefined HTML message. The following is an example HTML message that is returned for a **403 Forbidden** response.


## Configuring a custom error document

When you configure your bucket as a static website, you can provide a custom error document that contains a user-friendly error message and additional help. Amazon S3 returns your custom error document for only the HTTP 4XX class of error codes.

To configure a custom error document using the S3 console, follow the steps below. You can also configure an error document using the REST API, the AWS SDKs, the AWS CLI, or AWS CloudFormation. For more information, see the following:

- PutBucketWebsite in the *Amazon Simple Storage Service API Reference*
- AWS::S3::Bucket WebsiteConfiguration in the *AWS CloudFormation User Guide*
- put-bucket-website in the *AWS CLI Command Reference*

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

**To configure an error document**

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error**

**document** name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the **Buckets** list, choose the name of the bucket that you want to use to host a static website.
5. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

   After enabling static website hosting, proceed to step 6.
6. To upload the error document to your bucket, do one of the following:

   - Drag and drop the error document file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

## Setting permissions for website access

When you configure a bucket as a static website, if you want your website to be public, you can grant public read access. To make your bucket publicly readable, you must disable block public access settings for the bucket and write a bucket policy that grants public read access. If your bucket contains objects that are not owned by the bucket owner, you might also need to add an object access control list (ACL) that grants everyone read access.

If you don't want to disable block public access settings for your bucket but you still want your website to be public, you can create a Amazon CloudFront distribution to serve your static website. For more information, see Speeding up your website with Amazon CloudFront or Use an Amazon CloudFront distribution to serve a static website in the *Amazon Route 53 Developer Guide*.

> **Note**
>
> On the website endpoint, if a user requests an object that doesn't exist, Amazon S3 returns HTTP response code `404 (Not Found)`. If the object exists but you haven't granted read permission on it, the website endpoint returns HTTP response code `403 (Access`

`Denied`). The user can use the response code to infer whether a specific object exists. If you don't want this behavior, you should not enable website support for your bucket.

**Topics**



## Step 1: Edit S3 Block Public Access settings

If you want to configure an existing bucket as a static website that has public access, you must edit Block Public Access settings for that bucket. You might also have to edit your account-level Block Public Access settings. Amazon S3 applies the most restrictive combination of the bucket-level and account-level block public access settings.

For example, if you allow public access for a bucket but block all public access at the account level, Amazon S3 will continue to block public access to the bucket. In this scenario, you would have to edit your bucket-level and account-level Block Public Access settings. For more information, see Blocking public access to your Amazon S3 storage.

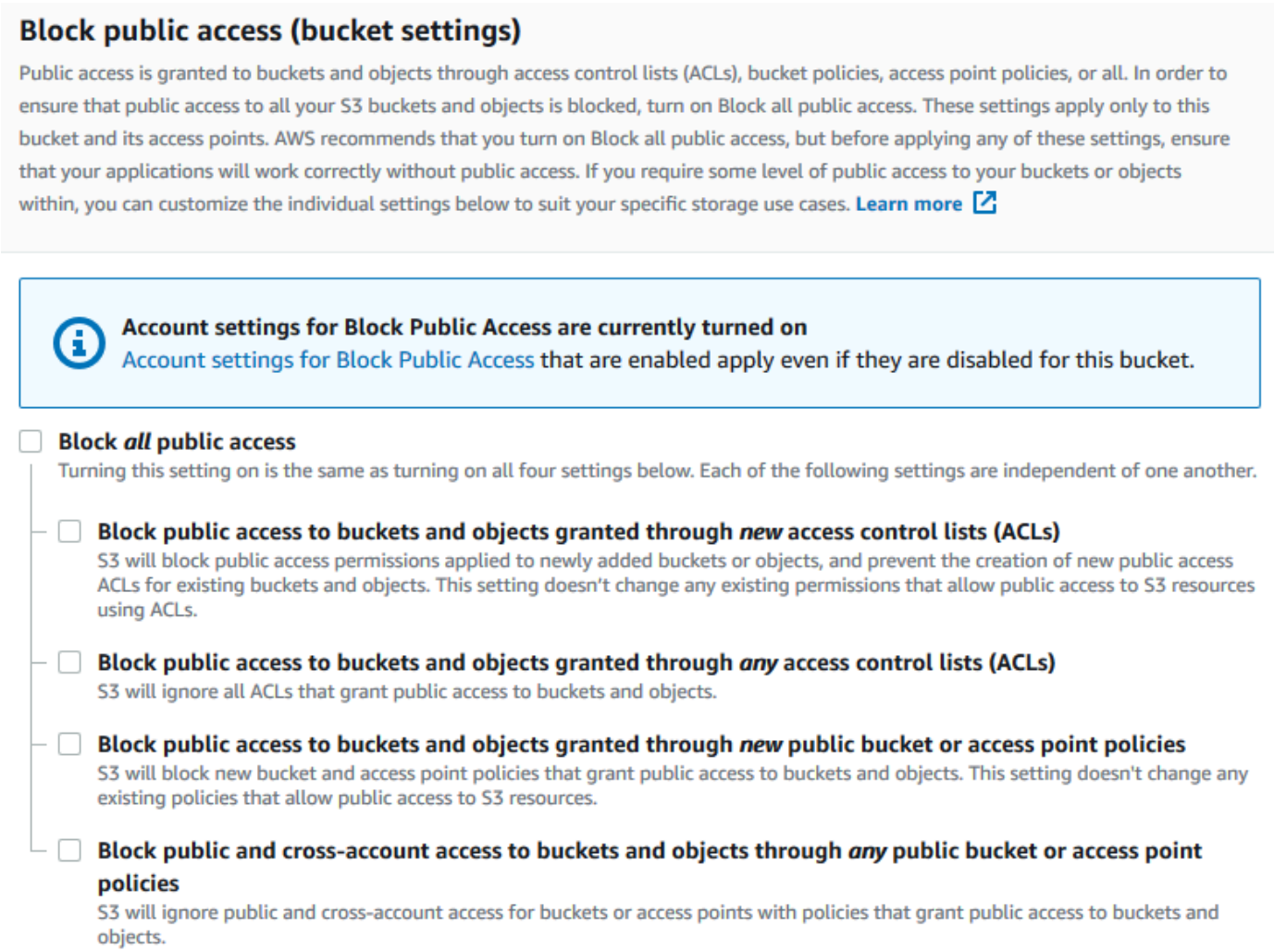
By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

**Warning**

Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.

5. Clear **Block *all* public access**, and choose **Save changes**.




Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

## Step 2: Add a bucket policy

To make the objects in your bucket publicly readable, you must write a bucket policy that grants everyone `s3:GetObject` permission.

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

**Important**

The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *`Bucket-Name`* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

## Object access control lists

You can use a bucket policy to grant public read permission to your objects. However, the bucket policy applies only to objects that are owned by the bucket owner. If your bucket contains objects that aren't owned by the bucket owner, the bucket owner should use the object access control list (ACL) to grant public READ permission on those objects.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

**Important**

If your bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

To make an object publicly readable using an ACL, grant READ permission to the `AllUsers` group, as shown in the following grant element. Add this grant element to the object ACL. For information about managing ACLs, see Access control list (ACL) overview.

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

# (Optional) Logging web traffic

You can optionally enable Amazon S3 server access logging for a bucket that is configured as a static website. Server access logging provides detailed records for the requests that are made to your bucket. For more information, see Logging requests with server access logging. If you plan to use Amazon CloudFront to speed up your website, you can also use CloudFront logging. For more information, see Configuring and Using Access Logs in the *Amazon CloudFront Developer Guide*.

**To enable server access logging for your static website bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the same Region where you created the bucket that is configured as a static website, create a bucket for logging, for example `logs.example.com`.
3. Create a folder for the server access logging log files (for example, `logs`).
4. (Optional) If you want to use CloudFront to improve your website performance, create a folder for the CloudFront log files (for example, `cdn`).

   For more information, see Speeding up your website with Amazon CloudFront.
5. In the **Buckets** list, choose your bucket.
6. Choose **Properties**.
7. Under **Server access logging**, choose **Edit**.
8. Choose **Enable**.
9. Under the **Target bucket**, choose the bucket and folder destination for the server access logs:

   - Browse to the folder and bucket location:

1. Choose **Browse S3**.
2. Choose the bucket name, and then choose the logs folder.
3. Choose **Choose path**.

- Enter the S3 bucket path, for example, **`s3://logs.example.com/logs/`**.

10. Choose **Save changes**.

    In your log bucket, you can now access your logs. Amazon S3 writes website access logs to your log bucket every 2 hours.

# (Optional) Configuring a webpage redirect

If your Amazon S3 bucket is configured for static website hosting, you can configure redirects for your bucket or the objects in it. You have the following options for configuring redirects.

**Topics**

- Redirect requests for your bucket's website endpoint to another bucket or domain
- Configure redirection rules to use advanced conditional redirects
- Redirect requests for an object

## Redirect requests for your bucket's website endpoint to another bucket or domain

You can redirect all requests to a website endpoint for a bucket to another bucket or domain. If you redirect all requests, any request made to the website endpoint is redirected to the specified bucket or domain.

For example, if your root domain is `example.com`, and you want to serve requests for both `http://example.com` and `http://www.example.com`, you must create two buckets named `example.com` and `www.example.com`. Then, maintain the content in the `example.com` bucket, and configure the other `www.example.com` bucket to redirect all requests to the `example.com` bucket. For more information, see Configuring a Static Website Using a Custom Domain Name.

**To redirect requests for a bucket website endpoint**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. Under **Buckets**, choose the name of the bucket that you want to redirect requests from (for example, `www.example.com`).
3. Choose **Properties**.
4. Under **Static website hosting**, choose **Edit**.
5. Choose **Redirect requests for an object**.
6. In the **Host name** box, enter the website endpoint for your bucket or your custom domain.

   For example, if you are redirecting to a root domain address, you would enter **`example.com`**.
7. For **Protocol**, choose the protocol for the redirected requests (**none**,**http**, or **https**).

   If you do not specify a protocol, the default option is **none**.
8. Choose **Save changes**.

## Configure redirection rules to use advanced conditional redirects

Using advanced redirection rules, you can route requests conditionally according to specific object key names, prefixes in the request, or response codes. For example, suppose that you delete or rename an object in your bucket. You can add a routing rule that redirects the request to another object. If you want to make a folder unavailable, you can add a routing rule to redirect the request to another webpage. You can also add a routing rule to handle error conditions by routing requests that return the error to another domain when the error is processed.

When enabling static website hosting for your bucket, you can optionally specify advanced redirection rules. Amazon S3 has a limitation of 50 routing rules per website configuration. If you require more than 50 routing rules, you can use object redirect. For more information, see Using the S3 console.

For more information about configuring routing rules using the REST API, see PutBucketWebsite in the *Amazon Simple Storage Service API Reference*.

> **Important**
>
> To create redirection rules in the new Amazon S3 console, you must use JSON. For JSON examples, see Redirection rules examples.

**To configure redirection rules for a static website**

To add redirection rules for a bucket that already has static website hosting enabled, follow these steps.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of a bucket that you have configured as a static website.
3. Choose **Properties**.
4. Under **Static website hosting**, choose **Edit**.
5. In **Redirection rules** box, enter your redirection rules in JSON.

   In the S3 console you describe the rules using JSON. For JSON examples, see Redirection rules examples. Amazon S3 has a limitation of 50 routing rules per website configuration.
6. Choose **Save changes**.

## Routing rule elements

The following is general syntax for defining the routing rules in a website configuration in JSON and XML To configure redirection rules in the new S3 console, you must use JSON. For JSON examples, see Redirection rules examples.

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]
```

```
Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<RoutingRules> =
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>

Note: <Redirect> must have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

The following table describes the elements in the routing rule.

| Name | Description |
| --- | --- |
| RoutingRules | Container for a collection of RoutingRule elements. |
| RoutingRule | A rule that identifies a condition and the redirect that is applied when the condition is met.<br><br>Condition:<br><br>• A RoutingRules container must contain at least one routing rule. |
| Condition | Container for describing a condition that must be met for the specified redirect to be applied. If the routing rule does not include a condition, the rule is applied to all requests. |
| KeyPrefixEquals | The prefix of the object key name from which requests are redirected.<br><br>KeyPrefixEquals is required if HttpErrorCodeReturnedEquals is not specified. If both KeyPrefixEquals and HttpErrorCodeReturnedEquals are specified, both must be true for the condition to be met. |
| HttpErrorCodeReturnedEquals | The HTTP error code that must match for the redirect to apply. If an error occurs, and if the error code meets this value, then the specified redirect applies.<br><br>HttpErrorCodeReturnedEquals is required if KeyPrefixEquals is not specified. If both KeyPrefixEquals and HttpErrorCodeReturnedEquals are specified, both must be true for the condition to be met. |
| Redirect | |

| Name | Description |
| --- | --- |
| | Container element that provides instructions for redirecting the request. You can redirect requests to another host or another page, or you can specify another protocol to use. A `RoutingRule` must have a `Redirect` element. A `Redirect` element must contain at least one of the following sibling elements: `Protocol`, `HostName`, `ReplaceKeyPrefixWith`, `ReplaceKeyWith`, or `HttpRedirectCode`. |
| `Protocol` | The protocol, `http` or `https`, to be used in the `Location` header that is returned in the response.<br><br>If one of its siblings is supplied, `Protocol` is not required. |
| `HostName` | The hostname to be used in the `Location` header that is returned in the response.<br><br>If one of its siblings is supplied, `HostName` is not required. |
| `ReplaceKeyPrefixWith` | The prefix of the object key name that replaces the value of `KeyPrefixEquals` in the redirect request.<br><br>If one of its siblings is supplied, `ReplaceKeyPrefixWith` is not required. It can be supplied only if `ReplaceKeyWith` is not supplied. |
| `ReplaceKeyWith` | The object key to be used in the `Location` header that is returned in the response.<br><br>If one of its siblings is supplied, `ReplaceKeyWith` is not required. It can be supplied only if `ReplaceKeyPrefixWith` is not supplied. |

| Name | Description |
| --- | --- |
| `HttpRedirectCode` | The HTTP redirect code to be used in the `Location` header that is returned in the response.<br><br>If one of its siblings is supplied, `HttpRedirectCode` is not required. |

### Redirection rules examples

The following examples explain common redirection tasks:

> **Important**
>
> To create redirection rules in the new Amazon S3 console, you must use JSON.

#### Example 1: Redirect after renaming a key prefix

Suppose that your bucket contains the following objects:

- index.html
- docs/article1.html
- docs/article2.html

You decide to rename the folder from `docs/` to `documents/`. After you make this change, you need to redirect requests for prefix `docs/` to `documents/`. For example, request for `docs/article1.html` will be redirected to `documents/article1.html`.

In this case, you add the following routing rule to the website configuration.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
```

```
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

**Example 2: Redirect requests for a deleted folder to a page**

Suppose that you delete the `images/` folder (that is, you delete all objects with the key prefix `images/`). You can add a routing rule that redirects requests for any object with the key prefix `images/` to a page named `folderdeleted.html`.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <RoutingRules>
```

```
    <RoutingRule>
    <Condition>
       <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

**Example 3: Redirect for an HTTP error**

Suppose that when a requested object is not found, you want to redirect requests to an Amazon Elastic Compute Cloud (Amazon EC2) instance. Add a redirection rule so that when an HTTP status code 404 (Not Found) is returned, the site visitor is redirected to an Amazon EC2 instance that handles the request.

The following example also inserts the object key prefix `report-404/` in the redirect. For example, if you request a page `ExamplePage.html` and it results in an HTTP 404 error, the request is redirected to a page `report-404/ExamplePage.html` on the specified Amazon EC2 instance. If there is no routing rule and the HTTP error 404 occurs, the error document that is specified in the configuration is returned.

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
```

```
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

## Redirect requests for an object

You can redirect requests for an object to another object or URL by setting the website redirect location in the metadata of the object. You set the redirect by adding the `x-amz-website-redirect-location` property to the object metadata. On the Amazon S3 console, you set the **Website Redirect Location** in the metadata of the object. If you use the Amazon S3 API, you set `x-amz-website-redirect-location`. The website then interprets the object as a 301 redirect.

To redirect a request to another object, you set the redirect location to the key of the target object. To redirect a request to an external URL, you set the redirect location to the URL that you want. For more information about object metadata, see System-defined object metadata.

When you set a page redirect, you can either keep or delete the source object content. For example, if you have a `page1.html` object in your bucket, you can redirect any requests for this page to another object, `page2.html`. You have two options:

- Keep the content of the `page1.html` object and redirect page requests.
- Delete the content of `page1.html` and upload a zero-byte object named `page1.html` to replace the existing object and redirect page requests.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you have configured as a static website (for example, `example.com`).
3. Under **Objects**, select your object.
4. Choose **Actions**, and choose **Edit metadata**.

5. Choose **Metadata**.
6. Choose **Add Metadata**.
7. Under **Type**, choose **System Defined**.
8. In **Key**, choose **x-amz-website-redirect-location**.
9. In **Value**, enter the key name of the object that you want to redirect to, for example, `/page2.html`.

   For another object in the same bucket, the `/` prefix in the value is required. You can also set the value to an external URL, for example, `http://www.example.com`.
10. Choose **Edit metadata**.

#### Using the REST API

The following Amazon S3 API actions support the `x-amz-website-redirect-location` header in the request. Amazon S3 stores the header value in the object metadata as `x-amz-website-redirect-location`.

- PUT Object
- Initiate Multipart Upload
- POST Object
- PUT Object - Copy

A bucket configured for website hosting has both the website endpoint and the REST endpoint. A request for a page that is configured as a 301 redirect has the following possible outcomes, depending on the endpoint of the request:

- **Region-specific website endpoint –** Amazon S3 redirects the page request according to the value of the `x-amz-website-redirect-location` property.
- **REST endpoint –** Amazon S3 doesn't redirect the page request. It returns the requested object.

For more information about the endpoints, see Key differences between a website endpoint and a REST API endpoint.

When setting a page redirect, you can either keep or delete the object content. For example, suppose that you have a `page1.html` object in your bucket.

- To keep the content of `page1.html` and only redirect page requests, you can submit a PUT Object - Copy request to create a new `page1.html` object that uses the existing `page1.html` object as the source. In your request, you set the `x-amz-website-redirect-location` header. When the request is complete, you have the original page with its content unchanged, but Amazon S3 redirects any requests for the page to the redirect location that you specify.
- To delete the content of the `page1.html` object and redirect requests for the page, you can send a PUT Object request to upload a zero-byte object that has the same object key: `page1.html`. In the PUT request, you set `x-amz-website-redirect-location` for `page1.html` to the new object. When the request is complete, `page1.html` has no content, and requests are redirected to the location that is specified by `x-amz-website-redirect-location`.

When you retrieve the object using the GET Object action, along with other object metadata, Amazon S3 returns the `x-amz-website-redirect-location` header in the response.

# Using cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

This section provides an overview of CORS. The subtopics describe how you can enable CORS using the Amazon S3 console, or programmatically by using the Amazon S3 REST API and the AWS SDKs.

## Cross-origin resource sharing: Use-case scenarios

The following are example scenarios for using CORS.

**Scenario 1**

Suppose that you are hosting a website in an Amazon S3 bucket named `website` as described in Hosting a static website using Amazon S3. Your users load the website endpoint:

```
http://website.s3-website.us-east-1.amazonaws.com
```

Now you want to use JavaScript on the webpages that are stored in this bucket to be able to make authenticated GET and PUT requests against the same bucket by using the Amazon S3

API endpoint for the bucket, `website.s3.us-east-1.amazonaws.com`. A browser would normally block JavaScript from allowing those requests, but with CORS you can configure your bucket to explicitly enable cross-origin requests from `website.s3-website.us-east-1.amazonaws.com`.

**Scenario 2**

Suppose that you want to host a web font from your S3 bucket. Again, browsers require a CORS check (also called a preflight check) for loading web fonts. You would configure the bucket that is hosting the web font to allow any origin to make these requests.

## How does Amazon S3 evaluate the CORS configuration on a bucket?

When Amazon S3 receives a preflight request from a browser, it evaluates the CORS configuration for the bucket and uses the first `CORSRule` rule that matches the incoming browser request to enable a cross-origin request. For a rule to match, the following conditions must be met:

- The `Origin` header in a CORS request to your bucket must match the origins in the `AllowedOrigins` element in your CORS configuration.
- The HTTP methods that are specified in the `Access-Control-Request-Method` in a CORS request to your bucket must match the method or methods listed in the `AllowedMethods` element in your CORS configuration.
- The headers listed in the `Access-Control-Request-Headers` header in a pre-flight request must match the headers in the `AllowedHeaders` element in your CORS configuration.

> **Note**
>
> The ACLs and policies continue to apply when you enable CORS on your bucket.

## How Object Lambda Access Point supports CORS

When S3 Object Lambda receives a request from a browser or the request includes an `Origin` header, S3 Object Lambda always adds an `“AllowedOrigins":"*"` header field.

For more information about using CORS, see the following topics.

**Topics**

# Elements of a CORS configuration

To configure your bucket to allow cross-origin requests, you create a CORS configuration. The CORS configuration is a document with elements that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. You can add up to 100 rules to the configuration. You can add the CORS configuration as the `cors` subresource to the bucket.

If you are configuring CORS in the S3 console, you must use JSON to create a CORS configuration. The new S3 console only supports JSON CORS configurations.

For more information about the CORS configuration and the elements in it, see the topics below. For instructions on how to add a CORS configuration, see Configuring cross-origin resource sharing (CORS).

> **Important**
>
> In the S3 console, the CORS configuration must be JSON.

**Topics**



## `AllowedMethods` element

In the CORS configuration, you can specify the following values for the `AllowedMethods` element.

- GET
- PUT
- POST
- DELETE
- HEAD

### `AllowedOrigins` element

In the `AllowedOrigins` element, you specify the origins that you want to allow cross-domain requests from, for example, `http://www.example.com`. The origin string can contain only one `*` wildcard character, such as `http://*.example.com`. You can optionally specify `*` as the origin to enable all the origins to send cross-origin requests. You can also specify `https` to enable only secure origins.

### `AllowedHeaders` element

The `AllowedHeaders` element specifies which headers are allowed in a preflight request through the `Access-Control-Request-Headers` header. Each header name in the `Access-Control-Request-Headers` header must match a corresponding entry in the element. Amazon S3 will send only the allowed headers in a response that were requested. For a sample list of headers that can be used in requests to Amazon S3, go to Common Request Headers in the *Amazon Simple Storage Service API Reference* guide.

Each AllowedHeaders string in your configuration can contain at most one * wildcard character. For example, `<AllowedHeader>x-amz-*</AllowedHeader>` will enable all Amazon-specific headers.

### `ExposeHeaders` element

Each `ExposeHeader` element identifies a header in the response that you want customers to be able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object). For a list of common Amazon S3 response headers, go to Common Response Headers in the *Amazon Simple Storage Service API Reference* guide.

### `MaxAgeSeconds` element

The `MaxAgeSeconds` element specifies the time in seconds that your browser can cache the response for a preflight request as identified by the resource, the HTTP method, and the origin.

## Examples of CORS configurations

Instead of accessing a website by using an Amazon S3 website endpoint, you can use your own domain, such as `example1.com` to serve your content. For information about using your own domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

The following example CORS configuration has three rules, which are specified as `CORSRule` elements:

- The first rule allows cross-origin PUT, POST, and DELETE requests from the `http://www.example1.com` origin. The rule also allows all headers in a preflight OPTIONS request through the `Access-Control-Request-Headers` header. In response to preflight OPTIONS requests, Amazon S3 returns requested headers.
- The second rule allows the same cross-origin requests as the first rule, but the rule applies to another origin, `http://www.example2.com`.
- The third rule allows cross-origin GET requests from all origins. The `*` wildcard character refers to all origins.

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
```

```
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
```

```
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

The CORS configuration also allows optional configuration parameters, as shown in the following CORS configuration. In this example, the CORS configuration allows cross-origin PUT, POST, and DELETE requests from the `http://www.example.com` origin.

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
```

```
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

The `CORSRule` element in the preceding configuration includes the following optional elements:

- `MaxAgeSeconds`—Specifies the amount of time in seconds (in this example, 3000) that the browser caches an Amazon S3 response to a preflight OPTIONS request for the specified resource. By caching the response, the browser does not have to send preflight requests to Amazon S3 if the original request will be repeated.
- `ExposeHeader`—Identifies the response headers (in this example, `x-amz-server-side-encryption`, `x-amz-request-id`, and `x-amz-id-2`) that customers are able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object).

## Configuring cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

This section shows you how to enable CORS using the Amazon S3 console, the Amazon S3 REST API, and the AWS SDKs. To configure your bucket to allow cross-origin requests, you add a CORS configuration to the bucket. A CORS configuration is a document that defines rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) supported for each origin, and other operation-specific information. In the S3 console, the CORS configuration must be a JSON document.

For example CORS configurations in JSON and XML, see Elements of a CORS configuration.

### Using the S3 console

This section explains how to use the Amazon S3 console to add a cross-origin resource sharing (CORS) configuration to an S3 bucket.

When you enable CORS on the bucket, the access control lists (ACLs) and other access permission policies continue to apply.

> **Important**
>
> In the S3 console, the CORS configuration must be JSON. For examples CORS configurations in JSON and XML, see Elements of a CORS configuration.

**To add a CORS configuration to an S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to create a bucket policy for.
3. Choose **Permissions**.
4. In the **Cross-origin resource sharing (CORS)** section, choose **Edit**.
5. In the **CORS configuration editor** text box, type or copy and paste a new CORS configuration, or edit an existing configuration.

   The CORS configuration is a JSON file. The text that you type in the editor must be valid JSON. For more information, see Elements of a CORS configuration.
6. Choose **Save changes**.

> **Note**
>
> Amazon S3 displays the Amazon Resource Name (ARN) for the bucket next to the **CORS configuration editor** title. For more information about ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces in the *Amazon Web Services General Reference*.

**Using the AWS SDKs**

You can use the AWS SDK to manage cross-origin resource sharing (CORS) for a bucket. For more information about CORS, see Using cross-origin resource sharing (CORS).

The following examples:

- Creates a CORS configuration and sets the configuration on a bucket
- Retrieves the configuration and modifies it by adding a rule
- Adds the modified configuration to the bucket
- Deletes the configuration

Java

**Example**

**Example**

For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new
 ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
```

```
        CORSRule rule1 = new
CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new
CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
            CORSRule rule3 = new
CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
```

```
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in
the
            // configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
configuration) {
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " +
configuration.getRules().size() + " rules\n");

            for (CORSRule rule : configuration.getRules()) {
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
```

```
                System.out.println();
            }
        }
    }
}
```

.NET

**Example**

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
        private static async Task CORSConfigTestAsync()
        {
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
```

```
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST",
 "DELETE"},
                            AllowedOrigins = new List<string> {"http://
*.example.com"}
                          },
                          new CORSRule
                          {
                            Id = "CORSRule2",
                            AllowedMethods = new List<string> {"GET"},
                            AllowedOrigins = new List<string> {"*"},
                            MaxAgeSeconds = 3000,
                            ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                          }
                        }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
 configuration.Rules.Count);
                Console.WriteLine();
```

```
                Console.WriteLine("Pause before configuration delete. To continue,
click Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();

                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
```

```
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
            DeleteCORSConfigurationRequest request = new
 DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:",
 configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
 rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
 rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
 rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
 rule.ExposeHeaders.ToArray()));
            }
        }
    }
}
```

### Using the REST API

To set a CORS configuration on your bucket, you can use the AWS Management Console. If your application requires it, you can also send REST requests directly. The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API actions related to the CORS configuration:

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS object

## Troubleshooting CORS

The following topics can help you troubleshoot some common CORS issues related to S3.

### Topics



### 403 Forbidden error: CORS is not enabled for this bucket

The following `403 Forbidden` error occurs when a cross-origin request is sent to Amazon S3 but CORS is not configured on your S3 bucket.

Error: HTTP/1.1 403 Forbidden CORS Response: CORS is not enabled for this bucket.

The CORS configuration is a document or policy with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. See how to configure CORS on S3 by using the Amazon S3 console, AWS SDKs, and REST API. For more information on CORS and examples of a CORS configuration, see Elements of CORS.

## 403 Forbidden error: This CORS request is not allowed

The following `403 Forbidden` error is received when a CORS rule in your CORS configuration doesn't match the data in your request.

Error: HTTP/1.1 403 Forbidden CORS Response: This CORS request is not allowed.

As a result, this `403 Forbidden` error can occur for multiple reasons:

- Origin is not allowed.
- Methods are not allowed.
- Requested headers are not allowed.

For each request that Amazon S3 receives, you must have a CORS rule in your CORS configuration that matches the data in your request.

### Origin is not allowed

The `Origin` header in a CORS request to your bucket must match the origins in the `AllowedOrigins` element in your CORS configuration. A wildcard character ("`*`") in the `AllowedOrigins` element would match all HTTP methods. For more information on how to update the `AllowedOrigins` element, see Configuring cross-origin resource sharing (CORS).

For example, if only the `http://www.example1.com` domain is included in the `AllowedOrigins` element, then a CORS request sent from the `http://www.example2.com` domain would receive the `403 Forbidden` error.

The following example shows part of a CORS configuration that includes the `http://www.example1.com` domain in the `AllowedOrigins` element.

```
"AllowedOrigins":[
   "http://www.example1.com"
]
```

For a CORS request sent from the `http://www.example2.com` domain to be successful, the `http://www.example2.com` domain should be included in the `AllowedOrigins` element of CORS configuration.

```
"AllowedOrigins":[
```

```
   "http://www.example1.com"
   "http://www.example2.com"
]
```

### Methods are not allowed

The HTTP methods that are specified in the `Access-Control-Request-Method` in a CORS request to your bucket must match the method or methods listed in the `AllowedMethods` element in your CORS configuration. A wildcard character (`"*"`) in `AllowedMethods` would match all HTTP methods. For more information on how to update the `AllowedOrigins` element, see Configuring cross-origin resource sharing (CORS).

In a CORS configuration, you can specify the following methods in the `AllowedMethods` element:

- `GET`
- `PUT`
- `POST`
- `DELETE`
- `HEAD`

The following example shows part of a CORS configuration that includes the `GET` method in the `AllowedMethods` element. Only requests including the `GET` method would succeed.

```
"AllowedMethods":[
   "GET"
]
```

If an HTTP method (for example, `PUT`) was used in a CORS request or included in a pre-flight CORS request to your bucket but the method isn't present in your CORS configuration, the request would result in a `403 Forbidden` error. To allow this CORS request or CORS pre-flight request, the `PUT` method must be added to your CORS configuration.

```
"AllowedMethods":[
   "GET"
   "PUT"
]
```

### Requested headers are not allowed

The headers listed in the `Access-Control-Request-Headers` header in a pre-flight request must match the headers in the `AllowedHeaders` element in your CORS configuration. For a list of common headers that can be used in requests to Amazon S3, see Common Request Headers. For more information on how to update the `AllowedHeaders` element, see Configuring cross-origin resource sharing (CORS).

The following example shows part of a CORS configuration that includes the `Authorization` header in the `AllowedHeaders` element. Only requests for the `Authorization` header would succeed.

```
"AllowedHeaders":  [
    "Authorization"
]
```

If a header (for example `Content-MD5` was included in a CORS request but the header isn't present in your CORS configuration, the request would result in a `403 Forbidden` error. To allow this CORS request , the `Content-MD5` header must be added to your CORS configuration. If you want to pass both `Authorization` and `Content-MD5` headers in a CORS request to your bucket, confirm that both headers are included in the `AllowedHeaders` element in your CORS configuration.

```
"AllowedHeaders":  [
    "Authorization"
    "Content-MD5"
]
```

## Headers not found in CORS response

The `ExposeHeaders` element in your CORS configuration identifies which response headers that you would like to make accessible to scripts and applications running in browsers, in response to a CORS request.

If your objects stored in your S3 bucket have user-defined metadata (for example, `x-amz-meta-custom-header`) along with the response data, this custom header could contain additional metadata or information that you want to access from your client-side JavaScript code. However, by default, browsers block access to custom headers for security reasons. To allow your client-side JavaScript to access custom headers, you need to include the header in your CORS configuration.

In the example below, the `x-amz-meta-custom-header1` header is included in the `ExposeHeaders` element. The `x-amz-meta-custom-header2` isn't included in the `ExposeHeaders` element and is missing from the CORS configuration. In the response, only the values included in the `ExposeHeaders` element would be returned. If the request included the `x-amz-meta-custom-header2` header in the `Access-Control-Expose-Headers` header, the response would still return a `200 OK`. However, only the permitted header, For example `x-amz-meta-custom-header` would be returned and show in the response.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1"
]
```

To ensure all headers appear in the response, add all permitted headers to the `ExposeHeaders` element in your CORS configuration as shown below.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1",
    "x-amz-meta-custom-header2"
]
```

## Considerations of CORS on S3 proxy integrations

If you are experiencing errors and have already checked the CORS configuration on your S3 bucket, and the cross-origin request is sent to proxies such as AWS CloudFront, try the following:

- Configure the settings to allow the `OPTIONS` method for HTTP requests.
- Configure the proxy to forward the following headers: `Origin`, `Access-Control-Request-Headers`, and `Access-Control-Request-Method`.

Some proxies provide pre-defined features for CORS requests. For example, in CloudFront, you can configure a policy that includes the headers that enable CORS requests when the origin is an S3 bucket. For more information, see Control origin requests with a policy and Use managed origin request policies in the *CloudFront Developer Guide*.

# Developing with Amazon S3

This section covers developer-related topics for using Amazon S3. For more information, review the topics below.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



## Making requests

Amazon S3 is a REST service. You can send requests to Amazon S3 using the REST API or the AWS SDK (see Sample Code and Libraries) wrapper libraries that wrap the underlying Amazon S3 REST API, simplifying your programming tasks.

Every interaction with Amazon S3 is either authenticated or anonymous. Authentication is a process of verifying the identity of the requester trying to access an Amazon Web Services (AWS) product. Authenticated requests must include a signature value that authenticates the request sender. The signature value is, in part, generated from the requester's AWS access keys (access key ID and secret access key). For more information about getting access keys, see How Do I Get Security Credentials? in the *AWS General Reference*.

If you are using the AWS SDK, the libraries compute the signature from the keys you provide. However, if you make direct REST API calls in your application, you must write the code to compute the signature and add it to the request.

**Topics**



# About access keys

The following sections review the types of access keys that you can use to make authenticated requests.

## AWS account access keys

The account access keys provide full access to the AWS resources owned by the account. The following are examples of access keys:

- Access key ID (a 20-character, alphanumeric string). For example: AKIAIOSFODNN7EXAMPLE
- Secret access key (a 40-character string). For example: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

The access key ID uniquely identifies an AWS account. You can use these access keys to send authenticated requests to Amazon S3.

## IAM user access keys

You can create one AWS account for your company; however, there may be several employees in the organization who need access to your organization's AWS resources. Sharing your AWS account access keys reduces security, and creating individual AWS accounts for each employee might not be practical. Also, you cannot easily share resources such as buckets and objects because they are owned by different accounts. To share resources, you must grant permissions, which is additional work.

In such scenarios, you can use AWS Identity and Access Management (IAM) to create users under your AWS account with their own access keys and attach IAM user policies that grant appropriate resource access permissions to these users. To better manage these users, IAM enables you to create groups of users and grant group-level permissions that apply to all users in that group.

These users are referred to as IAM users that you create and manage within AWS. The parent account controls a user's ability to access AWS. Any resources an IAM user creates are under the control of and paid for by the parent AWS account. These IAM users can send authenticated requests to Amazon S3 using their own security credentials. For more information about creating and managing users under your AWS account, go to the AWS Identity and Access Management product details page.

## Temporary security credentials

In addition to creating IAM users with their own access keys, IAM also enables you to grant temporary security credentials (temporary access keys and a security token) to any IAM user to enable them to access your AWS services and resources. You can also manage users in your system outside AWS. These are referred to as federated users. Additionally, users can be applications that you create to access your AWS resources.

IAM provides the AWS Security Token Service API for you to request temporary security credentials. You can use either the AWS STS API or the AWS SDK to request these credentials. The API returns temporary security credentials (access key ID and secret access key), and a security token. These credentials are valid only for the duration you specify when you request them. You use the access key ID and secret key the same way you use them when sending requests using your AWS account or IAM user access keys. In addition, you must include the token in each request you send to Amazon S3.

An IAM user can request these temporary security credentials for their own use or hand them out to federated users or applications. When requesting temporary security credentials for federated users, you must provide a user name and an IAM policy defining the permissions you want to associate with these temporary security credentials. The federated user cannot get more permissions than the parent IAM user who requested the temporary credentials.

You can use these temporary security credentials in making requests to Amazon S3. The API libraries compute the necessary signature value using those credentials to authenticate your request. If you send requests using expired credentials, Amazon S3 denies the request.

For information on signing requests using temporary security credentials in your REST API requests, see Signing and authenticating REST requests. For information about sending requests using AWS SDKs, see Making requests using the AWS SDKs.

For more information about IAM support for temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*.

For added security, you can require multifactor authentication (MFA) when accessing your Amazon S3 resources by configuring a bucket policy. For information, see Requiring MFA. After you require MFA to access your Amazon S3 resources, the only way you can access these resources is by providing temporary credentials that are created with an MFA key. For more information, see the AWS Multi-Factor Authentication detail page and Configuring MFA-Protected API Access in the *IAM User Guide*.

## Request endpoints

You send REST requests to the service's predefined endpoint. For a list of all AWS services and their corresponding endpoints, go to Regions and Endpoints in the *AWS General Reference*.

## Making requests to Amazon S3 over IPv6

Amazon Simple Storage Service (Amazon S3) supports the ability to access S3 buckets using the Internet Protocol version 6 (IPv6), in addition to the IPv4 protocol. Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. There are no additional charges for accessing Amazon S3 over IPv6. For more information about pricing, see Amazon S3 Pricing.

**Topics**

- Getting started making requests over IPv6
- Using IPv6 addresses in IAM policies
- Testing IP address compatibility
- Using Amazon S3 dual-stack endpoints

### Getting started making requests over IPv6

To make a request to an S3 bucket over IPv6, you need to use a dual-stack endpoint. The next section describes how to make requests over IPv6 by using dual-stack endpoints.

The following are some things you should know before trying to access a bucket over IPv6:

- The client and the network accessing the bucket must be enabled to use IPv6.
- Both virtual hosted-style and path style requests are supported for IPv6 access. For more information, see Amazon S3 dual-stack endpoints.

- If you use source IP address filtering in your AWS Identity and Access Management (IAM) user or bucket policies, you need to update the policies to include IPv6 address ranges. For more information, see Using IPv6 addresses in IAM policies.
- When using IPv6, server access log files output IP addresses in an IPv6 format. You need to update existing tools, scripts, and software that you use to parse Amazon S3 log files so that they can parse the IPv6 formatted `Remote IP` addresses. For more information, see Amazon S3 server access log format and Logging requests with server access logging.

> **Note**
>
> If you experience issues related to the presence of IPv6 addresses in log files, contact AWS Support.

### Making requests over IPv6 by using dual-stack endpoints

You make requests with Amazon S3 API calls over IPv6 by using dual-stack endpoints. The Amazon S3 API operations work the same way whether you're accessing Amazon S3 over IPv6 or over IPv4. Performance should be the same too.

When using the REST API, you access a dual-stack endpoint directly. For more information, see Dual-stack endpoints.

When using the AWS Command Line Interface (AWS CLI) and AWS SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the Amazon S3 endpoint in the config file.

You can use a dual-stack endpoint to access a bucket over IPv6 from any of the following:

- The AWS CLI, see Using dual-stack endpoints from the AWS CLI.
- The AWS SDKs, see Using dual-stack endpoints from the AWS SDKs.
- The REST API, see Making requests to dual-stack endpoints by using the REST API.

### Features not available over IPv6

The following feature is currently not supported when accessing an S3 bucket over IPv6: Static website hosting from an S3 bucket.

## Using IPv6 addresses in IAM policies

Before trying to access a bucket using IPv6, you must ensure that any IAM user or S3 bucket polices that are used for IP address filtering are updated to include IPv6 address ranges. IP address filtering policies that are not updated to handle IPv6 addresses may result in clients incorrectly losing or gaining access to the bucket when they start using IPv6. For more information about managing access permissions with IAM, see Identity and Access Management for Amazon S3.

IAM policies that filter IP addresses use IP Address Condition Operators. The following bucket policy identifies the 54.240.143.* range of allowed IPv4 addresses by using IP address condition operators. Any IP addresses outside of this range will be denied access to the bucket (`examplebucket`). Since all IPv6 addresses are outside of the allowed range, this policy prevents IPv6 addresses from being able to access `examplebucket`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

You can modify the bucket policy's `Condition` element to allow both IPv4 (`54.240.143.0/24`) and IPv6 (`2001:DB8:1234:5678::/64`) address ranges as shown in the following example. You can use the same type of `Condition` block shown in the example to update both your IAM user and bucket policies.

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
```

```
        }
      }
```

Before using IPv6 you must update all relevant IAM user and bucket policies that use IP address filtering. We do not recommend using IP address filterig in bucket policies.

You can review your IAM user policies using the IAM console at https://console.aws.amazon.com/iam/. For more information about IAM, see the IAM User Guide. For information about editing S3 bucket policies, see Adding a bucket policy by using the Amazon S3 console.

## Testing IP address compatibility

If you are using use Linux/Unix or Mac OS X, you can test whether you can access a dual-stack endpoint over IPv6 by using the `curl` command as shown in the following example:

**Example**

```
curl -v  http://s3.dualstack.us-west-2.amazonaws.com/
```

You get back information similar to the following example. If you are connected over IPv6 the connected IP address will be an IPv6 address.

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
*   Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
 zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

If you are using Microsoft Windows 7 or Windows 10, you can test whether you can access a dual-stack endpoint over IPv6 or IPv4 by using the `ping` command as shown in the following example.

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

## Using Amazon S3 dual-stack endpoints

Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. This section describes how to use dual-stack endpoints.

**Topics**



### Amazon S3 dual-stack endpoints

When you make a request to a dual-stack endpoint, the bucket URL resolves to an IPv6 or an IPv4 address. For more information about accessing a bucket over IPv6, see Making requests to Amazon S3 over IPv6.

When using the REST API, you directly access an Amazon S3 endpoint by using the endpoint name (URI). You can access an S3 bucket through a dual-stack endpoint by using a virtual hosted-style or a path-style endpoint name. Amazon S3 supports only regional dual-stack endpoint names, which means that you must specify the region as part of the name.

Use the following naming conventions for the dual-stack virtual hosted-style and path-style endpoint names:

- Virtual hosted-style dual-stack endpoint:

  *bucketname*.s3.dualstack.*aws-region*.amazonaws.com

- Path-style dual-stack endpoint:

  s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

For more information, about endpoint name style, see Accessing and listing an Amazon S3 bucket. For a list of Amazon S3 endpoints, see Regions and Endpoints in the *AWS General Reference*.

> **Important**
>
> You can use transfer acceleration with dual-stack endpoints. For more information, see Getting started with Amazon S3 Transfer Acceleration.

> **Note**
>
> The two types of VPC endpoints to access Amazon S3 (*Interface VPC endpoints* and *Gateway VPC endpoints*) don't have dual-stack support. For more information about VPC endpoints for Amazon S3, see AWS PrivateLink for Amazon S3.

When using the AWS Command Line Interface (AWS CLI) and AWS SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the Amazon S3 endpoint in the config file. The following sections describe how to use dual-stack endpoints from the AWS CLI and the AWS SDKs.

**Using dual-stack endpoints from the AWS CLI**

This section provides examples of AWS CLI commands used to make requests to a dual-stack endpoint. For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI.

You set the configuration value `use_dualstack_endpoint` to `true` in a profile in your AWS Config file to direct all Amazon S3 requests made by the `s3` and `s3api` AWS CLI commands to the dual-stack endpoint for the specified region. You specify the region in the config file or in a command using the `--region` option.

When using dual-stack endpoints with the AWS CLI, both `path` and `virtual` addressing styles are supported. The addressing style, set in the config file, controls if the bucket name is in the hostname or part of the URL. By default, the CLI will attempt to use virtual style where possible, but will fall back to path style if necessary. For more information, see AWS CLI Amazon S3 Configuration.

You can also make configuration changes by using a command, as shown in the following example, which sets `use_dualstack_endpoint` to `true` and `addressing_style` to `virtual` in the default profile.

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

If you want to use a dual-stack endpoint for specified AWS CLI commands only (not all commands), you can use either of the following methods:

- You can use the dual-stack endpoint per command by setting the `--endpoint-url` parameter to `https://s3.dualstack.aws-region.amazonaws.com` or `http://s3.dualstack.aws-region.amazonaws.com` for any `s3` or `s3api` command.

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- You can set up separate profiles in your AWS Config file. For example, create one profile that sets `use_dualstack_endpoint` to `true` and a profile that does not set `use_dualstack_endpoint`. When you run a command, specify which profile you want to use, depending upon whether or not you want to use the dual-stack endpoint.

> **Note**
>
> When using the AWS CLI you currently cannot use transfer acceleration with dual-stack endpoints. However, support for the AWS CLI is coming soon. For more information, see Using the AWS CLI.

### Using dual-stack endpoints from the AWS SDKs

This section provides examples of how to access a dual-stack endpoint by using the AWS SDKs.

### AWS SDK for Java dual-stack endpoint example

The following example shows how to enable dual-stack endpoints when creating an Amazon S3 client using the AWS SDK for Java.

For instructions on creating and testing a working Java sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {
```

```
    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .withDualstackEnabled(true)
                    .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

If you are using the AWS SDK for Java on Windows, you might have to set the following Java virtual machine (JVM) property:

```
java.net.preferIPv6Addresses=true
```

### AWS .NET SDK dual-stack endpoint example

When using the AWS SDK for .NET you use the `AmazonS3Config` class to enable the use of a dual-stack endpoint as shown in the following example.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;
```

```
namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }
                    Console.WriteLine("Next Continuation Token: {0}",
 response.NextContinuationToken);
                    request.ContinuationToken = response.NextContinuationToken;
```

```
                } while (response.IsTruncated == true);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

For a full .NET sample for listing objects, see Listing object keys programmatically.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

**Using dual-stack endpoints from the REST API**

For information about making requests to dual-stack endpoints by using the REST API, see Making requests to dual-stack endpoints by using the REST API.

# Making requests using the AWS SDKs

**Topics**



You can send authenticated requests to Amazon S3 using either the AWS SDK or by making the REST API calls directly in your application. The AWS SDK API uses the credentials that you provide to compute the signature for authentication. If you use the REST API directly in your applications, you must write the necessary code to compute the signature for authenticating your request. For a list of available AWS SDKs go to, Sample Code and Libraries.

## Making requests using AWS account or IAM user credentials

You can use your AWS account or IAM user security credentials to send authenticated requests to Amazon S3. This section provides examples of how you can send authenticated requests using the AWS SDK for Java, AWS SDK for .NET, and AWS SDK for PHP. For a list of available AWS SDKs, go to Sample Code and Libraries.

Each of these AWS SDKs uses an SDK-specific credentials provider chain to find and use credentials and perform actions on behalf of the credentials owner. What all these credentials provider chains have in common is that they all look for your local AWS credentials file.

For more information, see the topics below:

**Topics**



### To create a local AWS credentials file

The easiest way to configure credentials for your AWS SDKs is to use an AWS credentials file. If you use the AWS Command Line Interface (AWS CLI), you may already have a local AWS credentials file configured. Otherwise, use the following procedure to set up a credentials file:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Create a new user with permissions limited to the services and actions that you want your code to have access to. For more information about creating a new user, see Creating IAM users (Console), and follow the instructions through step 8.
3. Choose **Download .csv** to save a local copy of your AWS credentials.
4. On your computer, navigate to your home directory, and create an `.aws` directory. On Unix-based systems, such as Linux or OS X, this is in the following location:

   ```
   ~/.aws
   ```

   On Windows, this is in the following location:

   ```
   %HOMEPATH%\.aws
   ```
5. In the `.aws` directory, create a new file named `credentials`.
6. Open the credentials `.csv` file that you downloaded from the IAM console, and copy its contents into the `credentials` file using the following format:

   ```
   [default]
   aws_access_key_id = your_access_key_id
   aws_secret_access_key = your_secret_access_key
   ```
7. Save the `credentials` file, and delete the `.csv` file that you downloaded in step 3.

Your shared credentials file is now configured on your local computer, and it's ready to be used with the AWS SDKs.

**Sending authenticated requests using the AWS SDKs**

Use the AWS SDKs to send authenticated requests. For more information about sending authenticated requests, see AWS security credentials or IAM Identity Center Authentication.

Java

To send authenticated requests to Amazon S3 using your AWS account or IAM user credentials, do the following:

- Use the `AmazonS3ClientBuilder` class to create an `AmazonS3Client` instance.

- Run one of the `AmazonS3Client` methods to send requests to Amazon S3. The client generates the necessary signature from the credentials that you provide and includes it in the request.

The following example performs the preceding tasks. For information on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

**Example**

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;
import java.util.List;

public class MakingRequests {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
 ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while (true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
```

```
                for (S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
 summary.getKey(), summary.getSize());
                }
                if (objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
                } else {
                    break;
                }
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

To send authenticated requests using your AWS account or IAM user credentials:

- Create an instance of the `AmazonS3Client` class.
- Run one of the `AmazonS3Client` methods to send requests to Amazon S3. The client generates the necessary signature from the credentials that you provide and includes it in the request it sends to Amazon S3.

For more information, see Making requests using AWS account or IAM user credentials.

> **Note**
>
> - You can create the `AmazonS3Client` client without providing your security credentials. Requests sent using this client are anonymous requests, without a signature. Amazon S3 returns an error if you send anonymous requests for a resource that is not publicly available.

- You can create an AWS account and create the required users. You can also manage credentials for those users. You need these credentials to perform the task in the following example. For more information, see Configure AWS credentials in the *AWS SDK for .NET Developer Guide*.

  You can then also configure your application to actively retrieve profiles and credentials, and then explicitly use those credentials when creating an AWS service client. For more information, see Accessing credentials and profiles in an application in the *AWS SDK for .NET Developer Guide*.

The following C# example shows how to perform the preceding tasks. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

**Example**

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }
```

```
        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {
                    ListObjectsResponse response = await
client.ListObjectsAsync(request);
                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If the response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.Marker = response.NextMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
```

```
    }
}
```

For working examples, see Amazon S3 objects overview and Buckets overview. You can test these examples using your AWS account or an IAM user credentials.

For example, to list all the object keys in your bucket, see Listing object keys programmatically.

PHP

This section explains how to use a class from version 3 of the AWS SDK for PHP to send authenticated requests using your AWS account or IAM user credentials. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example shows how the client makes a request using your security credentials to list all of the buckets for your account.

**Example**

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
```

```
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

> **Note**
>
> You can create the `S3Client` client without providing your security credentials. Requests sent using this client are anonymous requests, without a signature. Amazon S3 returns an error if you send anonymous requests for a resource that is not publicly available. For more information, see Creating Anonymous Clients in the AWS SDK for PHP Documentation.

For working examples, see Amazon S3 objects overview. You can test these examples using your AWS account or IAM user credentials.

For an example of listing object keys in a bucket, see Listing object keys programmatically.

Ruby

Before you can use version 3 of the AWS SDK for Ruby to make calls to Amazon S3, you must set the AWS access credentials that the SDK uses to verify your access to your buckets and objects. If you have shared credentials set up in the AWS credentials profile on your local system, version 3 of the SDK for Ruby can use those credentials without your having to declare them in your code. For more information about setting up shared credentials, see Making requests using AWS account or IAM user credentials.

The following Ruby code snippet uses the credentials in a shared AWS credentials file on a local computer to authenticate a request to get all of the object key names in a specific bucket. It does the following:

1. Creates an instance of the `Aws::S3::Client` class.
2. Makes a request to Amazon S3 by enumerating objects in a bucket using the `list_objects_v2` method of `Aws::S3::Client`. The client generates the necessary signature value from the credentials in the AWS credentials file on your computer, and includes it in the request it sends to Amazon S3.

3. Prints the array of object key names to the terminal.

### Example

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)
```

```
  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

If you don't have a local AWS credentials file, you can still create the `Aws::S3::Client` resource and run code against Amazon S3 buckets and objects. Requests that are sent using version 3 of the SDK for Ruby are anonymous, with no signature by default. Amazon S3 returns an error if you send anonymous requests for a resource that's not publicly available.

You can use and expand the previous code snippet for SDK for Ruby applications, as in the following more robust example.

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end
```

```
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Go

**Example**

The following example uses AWS credentials automatically loaded by the SDK for Go from the shared credentials file.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
```

```
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}
s3Client := s3.NewFromConfig(sdkConfig)
count := 10
fmt.Printf("Let's list up to %v buckets for your account.\n", count)
result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
if err != nil {
 fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 return
}
if len(result.Buckets) == 0 {
 fmt.Println("You don't have any buckets!")
} else {
 if count > len(result.Buckets) {
  count = len(result.Buckets)
 }
 for _, bucket := range result.Buckets[:count] {
  fmt.Printf("\t%v\n", *bucket.Name)
 }
}
}
```

### Related resources

- Developing with Amazon S3 using the AWS SDKs
- AWS SDK for PHP for Amazon S3 Aws\S3\S3Client Class
- AWS SDK for PHP Documentation

## Making requests using IAM user temporary credentials

An AWS account or an IAM user can request temporary security credentials and use them to send authenticated requests to Amazon S3. This section provides examples of how to use the AWS SDK for Java, .NET, and PHP to obtain temporary security credentials and use them to authenticate your requests to Amazon S3.

Java

An IAM user or an AWS account can request temporary security credentials (see Making requests) using the AWS SDK for Java and use them to access Amazon S3. These credentials expire after the specified session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. For more information about making requests, see Making requests.

**To get temporary security credentials and access Amazon S3**

1. Create an instance of the `AWSSecurityTokenService` class. For information about providing credentials, see Developing with Amazon S3 using the AWS SDKs.
2. Retrieve the temporary security credentials for the desired role by calling the `assumeRole()` method of the Security Token Service (STS) client.
3. Package the temporary security credentials into a `BasicSessionCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
4. Create an instance of the `AmazonS3Client` class using the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 will return an error.

> **Note**
>
> If you obtain temporary security credentials using your AWS account security credentials, the temporary credentials are valid for only one hour. You can specify the session duration only if you use IAM user credentials to request a session.

The following example lists a set of object keys in the specified bucket. The example obtains temporary security credentials for a session and uses them to send an authenticated request to Amazon S3.

If you want to test the sample by using IAM user credentials, you must create an IAM user under your AWS account. For more information about how to create an IAM user, see Creating Your First IAM user and Administrators Group in the *IAM User Guide*.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String roleARN = "*** ARN for role to be assumed ***";
        String roleSessionName = "*** Role session name ***";
        String bucketName = "*** Bucket name ***";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
 credentials.
            AWSSecurityTokenService stsClient =
 AWSSecurityTokenServiceClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```
            // Obtain credentials for the IAM role. Note that you cannot assume the
role of
            // an AWS root account;
            // Amazon S3 will deny access. You must use credentials for an IAM user
or an
            // IAM role.
            AssumeRoleRequest roleRequest = new AssumeRoleRequest()
                    .withRoleArn(roleARN)
                    .withRoleSessionName(roleSessionName);
            AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
            Credentials sessionCredentials = roleResponse.getCredentials();

            // Create a BasicSessionCredentials object that contains the credentials
you
            // just retrieved.
            BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());

            // Provide temporary security credentials so that the Amazon S3 client
            // can send authenticated requests to Amazon S3. You create the client
            // using the sessionCredentials object.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
                    .withRegion(clientRegion)
                    .build();

            // Verify that assuming the role worked and the permissions are set
correctly
            // by getting a set of object keys from the bucket.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

.NET

An IAM user or an AWS account can request temporary security credentials using the AWS SDK for .NET and use them to access Amazon S3. These credentials expire after the session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. For more information about making requests, see Making requests.

**To get temporary security credentials and access Amazon S3**

1. Create an instance of the AWS Security Token Service client, `AmazonSecurityTokenServiceClient`. For information about providing credentials, see Developing with Amazon S3 using the AWS SDKs.
2. Start a session by calling the `GetSessionToken` method of the STS client you created in the preceding step. You provide session information to this method using a `GetSessionTokenRequest` object.

   The method returns your temporary security credentials.
3. Package the temporary security credentials in an instance of the `SessionAWSCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
4. Create an instance of the `AmazonS3Client` class by passing in the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 returns an error.

> **Note**
>
> If you obtain temporary security credentials using your AWS account security credentials, those credentials are valid for only one hour. You can specify a session duration only if you use IAM user credentials to request a session.

The following C# example lists object keys in the specified bucket. For illustration, the example obtains temporary security credentials for a default one-hour session and uses them to send authenticated request to Amazon S3.

If you want to test the sample by using IAM user credentials, you must create an IAM user under your AWS account. For more information about how to create an IAM user, see Creating Your First IAM user and Administrators Group in the *IAM User Guide*. For more information about making requests, see Making requests.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
 chain.
                // On local development machines, this is your default profile.
```

```
                Console.WriteLine("Listing objects stored in a bucket");
                SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
                {
                    var listObjectRequest = new ListObjectsRequest
                    {
                        BucketName = bucketName
                    };
                    // Send request to Amazon S3.
                    ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);
                }
            }
            catch (AmazonS3Exception s3Exception)
            {
                Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
            }
            catch (AmazonSecurityTokenServiceException stsException)
            {
                Console.WriteLine(stsException.Message,
stsException.InnerException);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
        {
            using (var stsClient = new AmazonSecurityTokenServiceClient())
            {
                var getSessionTokenRequest = new GetSessionTokenRequest
                {
                    DurationSeconds = 7200 // seconds
                };

                GetSessionTokenResponse sessionTokenResponse =
                            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

                Credentials credentials = sessionTokenResponse.Credentials;
```

```
                var sessionCredentials =
                    new SessionAWSCredentials(credentials.AccessKeyId,
                                              credentials.SecretAccessKey,
                                              credentials.SessionToken);
                return sessionCredentials;
            }
        }
    }
}
```

PHP

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

An IAM user or an AWS account can request temporary security credentials using version 3 of the AWS SDK for PHP. It can then use the temporary credentials to access Amazon S3. The credentials expire when the session duration expires.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. For more information about making requests, see Making requests.

> **Note**
>
> If you obtain temporary security credentials using your AWS account security credentials, the temporary security credentials are valid for only one hour. You can specify the session duration only if you use IAM user credentials to request a session.

**Example**

The following PHP example lists object keys in the specified bucket using temporary security credentials. The example obtains temporary security credentials for a default one-hour session, and uses them to send authenticated request to Amazon S3. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

If you want to test the example by using IAM user credentials, you must create an IAM user under your AWS account. For information about how to create an IAM user, see Creating Your

First IAM user and Administrators Group in the *IAM User Guide*. For examples of setting the session duration when using IAM user credentials to request a session, see Making requests using IAM user temporary credentials .

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key'    => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token'  => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
```

```
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

An IAM user or an AWS account can request temporary security credentials using AWS SDK for Ruby and use them to access Amazon S3. These credentials expire after the session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. For more information about making requests, see Making requests.

> **Note**
>
> If you obtain temporary security credentials using your AWS account security credentials, the temporary security credentials are valid for only one hour. You can specify session duration only if you use IAM user credentials to request a session.

The following Ruby example creates a temporary user to list the items in a specified bucket for one hour. To use this example, you must have AWS credentials that have the necessary permissions to create new AWS Security Token Service (AWS STS) clients, and list Amazon S3 buckets.

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
 user.
# - An existing S3 bucket.

require "aws-sdk-core"
require "aws-sdk-s3"
require "aws-sdk-iam"
```

```
# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts "Error while determining whether the user " \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
```

```
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts "Error while determining whether the role " \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
```

```
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'doc-example-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
```

```
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

### Related resources

- Developing with Amazon S3 using the AWS SDKs
- AWS SDK for PHP for Amazon S3 Aws\S3\S3Client Class
- AWS SDK for PHP Documentation

## Making requests using federated user temporary credentials

You can request temporary security credentials and provide them to your federated users or applications who need to access your AWS resources. This section provides examples of how you can use the AWS SDK to obtain temporary security credentials for your federated users or applications and send authenticated requests to Amazon S3 using those credentials. For a list of available AWS SDKs, see Sample Code and Libraries.

> **Note**
>
> Both the AWS account and an IAM user can request temporary security credentials for federated users. However, for added security, only an IAM user with the necessary permissions should request these temporary credentials to ensure that the federated user gets at most the permissions of the requesting IAM user. In some applications, you might find it suitable to create an IAM user with specific permissions for the sole purpose of granting temporary security credentials to your federated users and applications.

Java

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your AWS resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the session duration is one hour. You can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications.

> **Note**
>
> For added security when requesting temporary security credentials for federated users and applications, we recommend that you use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see AWS Identity and Access Management FAQs .

To provide security credentials and send authenticated request to access resources, do the following:

- Create an instance of the `AWSSecurityTokenServiceClient` class.
- Start a session by calling the `getFederationToken()` method of the Security Token Service (STS) client. Provide session information, including the user name and an IAM policy, that you want to attach to the temporary credentials. You can provide an optional session duration. This method returns your temporary security credentials.
- Package the temporary security credentials in an instance of the `BasicSessionCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
- Create an instance of the `AmazonS3Client` class using the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 returns an error.

**Example**

The example lists keys in the specified S3 bucket. In the example, you obtain temporary security credentials for a two-hour session for your federated user and use the credentials to send authenticated requests to Amazon S3. To run the example, you need to create an IAM user with an attached policy that allows the user to request temporary security credentials and list your AWS resources. The following policy accomplishes this:

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

For more information about how to create an IAM user, see Creating Your First IAM user and Administrators Group in the *IAM User Guide*.

After creating an IAM user and attaching the preceding policy, you can run the following example. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Specify bucket name ***";
        String federatedUser = "*** Federated user name ***";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                    .standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            GetFederationTokenRequest getFederationTokenRequest = new
 GetFederationTokenRequest();
            getFederationTokenRequest.setDurationSeconds(7200);
            getFederationTokenRequest.setName(federatedUser);
```

```
            // Define the policy and add it to the request.
            Policy policy = new Policy();
            policy.withStatements(new Statement(Effect.Allow)
                    .withActions(S3Actions.ListObjects)
                    .withResources(new Resource(resourceARN)));
            getFederationTokenRequest.setPolicy(policy.toJson());

            // Get the temporary security credentials.
            GetFederationTokenResult federationTokenResult =
 stsClient.getFederationToken(getFederationTokenRequest);
            Credentials sessionCredentials = federationTokenResult.getCredentials();

            // Package the session credentials as a BasicSessionCredentials
            // object for an Amazon S3 client object to use.
            BasicSessionCredentials basicSessionCredentials = new
 BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
 AWSStaticCredentialsProvider(basicSessionCredentials))
                    .withRegion(clientRegion)
                    .build();

            // To verify that the client works, send a listObjects request using
            // the temporary security credentials.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects = " +
 objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your AWS resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the duration of a session is one hour. You can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications. For information about sending authenticated requests, see Making requests.

> **Note**
>
> When requesting temporary security credentials for federated users and applications, for added security, we suggest that you use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see AWS Identity and Access Management FAQs .

You do the following:

- Create an instance of the AWS Security Token Service client, `AmazonSecurityTokenServiceClient` class.
- Start a session by calling the `GetFederationToken` method of the STS client. You need to provide session information, including the user name and an IAM policy that you want to attach to the temporary credentials. Optionally, you can provide a session duration. This method returns your temporary security credentials.
- Package the temporary security credentials in an instance of the `SessionAWSCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
- Create an instance of the `AmazonS3Client` class by passing the temporary security credentials. You use this client to send requests to Amazon S3. If you send requests using expired credentials, Amazon S3 returns an error.

### Example

The following C# example lists the keys in the specified bucket. In the example, you obtain temporary security credentials for a two-hour session for your federated user (User1), and use the credentials to send authenticated requests to Amazon S3.

- For this exercise, you create an IAM user with minimal permissions. Using the credentials of this IAM user, you request temporary credentials for others. This example lists only the objects in a specific bucket. Create an IAM user with the following policy attached:

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

  The policy allows the IAM user to request temporary security credentials and access permission only to list your AWS resources. For more information about how to create an IAM user, see Creating Your IAM user User and Administrators Group in the *IAM User Guide*.

- Use the IAM user security credentials to test the following example. The example sends authenticated request to Amazon S3 using temporary security credentials. The example specifies the following policy when requesting temporary security credentials for the federated user (User1), which restricts access to listing objects in a specific bucket (`YourBucketName`). You must update the policy and provide your own existing bucket name.

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

- **Example**

  Update the following sample and provide the bucket name that you specified in the preceding federated user access policy. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
 chain.
                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();
```

```
                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
        {
            AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
            AmazonSecurityTokenServiceClient stsClient =
                new AmazonSecurityTokenServiceClient(
                                            config);

            GetFederationTokenRequest federationTokenRequest =
                                    new GetFederationTokenRequest();
            federationTokenRequest.DurationSeconds = 7200;
            federationTokenRequest.Name = "User1";
            federationTokenRequest.Policy = @"{
               ""Statement"":
               [
```

```
                {
                  ""Sid"":""Stmt1311212314284"",
                  ""Action"":[""s3:ListBucket""],
                  ""Effect"":""Allow"",
                  ""Resource"":""arn:aws:s3:::" + bucketName + @"""
                 }
              ]
             }
            ";

            GetFederationTokenResponse federationTokenResponse =
                        await
 stsClient.GetFederationTokenAsync(federationTokenRequest);
            Credentials credentials = federationTokenResponse.Credentials;

            SessionAWSCredentials sessionCredentials =
                new SessionAWSCredentials(credentials.AccessKeyId,
                                          credentials.SecretAccessKey,
                                          credentials.SessionToken);
            return sessionCredentials;
        }
    }
}
```

PHP

This topic explains how to use classes from version 3 of the AWS SDK for PHP to request temporary security credentials for federated users and applications and use them to access resources stored in Amazon S3. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

You can provide temporary security credentials to your federated users and applications so they can send authenticated requests to access your AWS resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. These credentials expire when the session duration expires. By default, the session duration is one hour. You can explicitly set a different value for the duration when requesting the temporary security credentials for federated users and applications. For more information about temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. For information about providing temporary security credentials to your federated users and applications, see Making requests.

For added security when requesting temporary security credentials for federated users and applications, we recommend using a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For information about identity federation, see AWS Identity and Access Management FAQs.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

**Example**

The following PHP example lists keys in the specified bucket. In the example, you obtain temporary security credentials for an hour session for your federated user (User1). Then you use the temporary security credentials to send authenticated requests to Amazon S3.

For added security when requesting temporary credentials for others, you use the security credentials of an IAM user who has permissions to request temporary security credentials. To ensure that the IAM user grants only the minimum application-specific permissions to the federated user, you can also limit the access permissions of this IAM user. This example lists only objects in a specific bucket. Create an IAM user with the following policy attached:

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

The policy allows the IAM user to request temporary security credentials and access permission only to list your AWS resources. For more information about how to create an IAM user, see Creating Your First IAM user and Administrators Group in the *IAM User Guide*.

You can now use the IAM user security credentials to test the following example. The example sends an authenticated request to Amazon S3 using temporary security credentials. When requesting temporary security credentials for the federated user (User1), the example specifies the following policy, which restricts access to list objects in a specific bucket. Update the policy with your bucket name.

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

In the following example, when specifying the policy resource, replace `YourBucketName` with the name of your bucket.:

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name'              => 'User1',
    'DurationSeconds'    => '3600',
    'Policy'            => json_encode([
        'Statement' => [
            'Sid'               => 'randomstatementid' . time(),
            'Action'            => ['s3:ListBucket'],
            'Effect'            => 'Allow',
            'Resource'          => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);
```

```
// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your AWS resources. When requesting temporary credentials from the IAM service, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the session duration is one hour. However, if you are requesting temporary credentials using IAM user credentials, you can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications. For information about temporary security credentials for your federated users and applications, see Making requests.

**Note**

For added security when you request temporary security credentials for federated users and applications, you might want to use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see AWS Identity and Access Management FAQs .

### Example

The following Ruby code example allows a federated user with a limited set of permissions to lists keys in the specified bucket.

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"
require "aws-sdk-iam"
require "json"

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  return response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  return response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
 #{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
```

```
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::doc-example-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  return response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )
```

```
  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "doc-example-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      "Version" => "2012-10-17",
      "Statement" => [
        "Sid" => "Stmt1",
        "Effect" => "Allow",
        "Action" => "s3:ListBucket",
        "Resource" => "arn:aws:s3:::#{bucket_name}"
      ]
    }
  )

  exit 1 unless credentials.access_key_id

  puts "Access key ID: #{credentials.access_key_id}"
  s3_client = Aws::S3::Client.new(region: region, credentials: credentials)
```

```
  exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

**Related resources**

- Developing with Amazon S3 using the AWS SDKs
- AWS SDK for PHP for Amazon S3 Aws\S3\S3Client Class
- AWS SDK for PHP Documentation

## Making requests using the REST API

This section contains information on how to make requests to Amazon S3 endpoints by using the REST API. For a list of Amazon S3 endpoints, see Regions and Endpoints in the *AWS General Reference*.

### Constructing S3 hostnames for REST API requests

Amazon S3 endpoints follow the structure shown below:

```
s3.Region.amazonaws.com
```

Amazon S3 access points endpoints and dual-stack endpoints also follow the standard structure:

- **Amazon S3 access points** `-s3-accesspoint.Region.amazonaws.com`
- **Dual-stack** - `s3.dualstack.Region.amazonaws.com`

For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

### Virtual hosted-style and path-style requests

When making requests by using the REST API, you can use virtual hosted–style or path-style URIs for the Amazon S3 endpoints. For more information, see Virtual hosting of buckets.

### Example Virtual hosted–Style request

Following is an example of a virtual hosted–style request to delete the puppy.jpg file from the bucket named examplebucket in the US West (Oregon) Region. For more information about virtual hosted-style requests, see Virtual-hosted–style requests.

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

### Example Path-style request

Following is an example of a path-style version of the same request.

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Currently, Amazon S3 supports both virtual-hosted–style and path-style URL access in all AWS Regions. However, path-style URLs will be discontinued in the future. For more information, see the following **Important** note.

For more information about path-style requests, see Path-style requests.

> **Important**
>
> Update (September 23, 2020) – To make sure that customers have the time that they need to transition to virtual-hosted–style URLs, we have decided to delay the deprecation of path-style URLs. For more information, see Amazon S3 Path Deprecation Plan – The Rest of the Story in the *AWS News Blog*.

## Making requests to dual-stack endpoints by using the REST API

When using the REST API, you can directly access a dual-stack endpoint by using a virtual hosted–style or a path style endpoint name (URI). All Amazon S3 dual-stack endpoint names include the

region in the name. Unlike the standard IPv4-only endpoints, both virtual hosted–style and a path-style endpoints use region-specific endpoint names.

**Example Virtual hosted–Style dual-stack endpoint request**

You can use a virtual hosted–style endpoint in your REST request as shown in the following example that retrieves the `puppy.jpg` object from the bucket named `examplebucket` in the US West (Oregon) Region.

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

**Example Path-style dual-stack endpoint request**

Or you can use a path-style endpoint in your request as shown in the following example.

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

For more information about dual-stack endpoints, see Using Amazon S3 dual-stack endpoints.

For more information about making requests using the REST API, see the topics beldow.

**Topics**

- Virtual hosting of buckets
- Request redirection and the REST API

## Virtual hosting of buckets

Virtual hosting is the practice of serving multiple websites from a single web server. One way to differentiate sites in your Amazon S3 REST API requests is by using the apparent hostname of the Request-URI instead of just the path name part of the URI. An ordinary Amazon S3 REST request specifies a bucket by using the first slash-delimited component of the Request-URI path.

Instead, you can use Amazon S3 virtual hosting to address a bucket in a REST API call by using the HTTP `Host` header. In practice, Amazon S3 interprets `Host` as meaning that most buckets are automatically accessible for limited types of requests at `https://bucket-name.s3.region-code.amazonaws.com`. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

Virtual hosting also has other benefits. By naming your bucket after your registered domain name and by making that name a DNS alias for Amazon S3, you can completely customize the URL of your Amazon S3 resources, for example, `http://my.bucket-name.com/`. You can also publish to the "root directory" of your bucket's virtual server. This ability can be important because many existing applications search for files in this standard location. For example, `favicon.ico`, `robots.txt`, and `crossdomain.xml` are all expected to be found at the root.

> **Important**
>
> When you're using virtual-hosted–style buckets with SSL, the SSL wildcard certificate matches only buckets that do not contain dots (`.`). To work around this limitation, use HTTP or write your own certificate-verification logic. For more information, see Amazon S3 Path Deprecation Plan on the *AWS News Blog*.

**Topics**



### Path-style requests

Currently, Amazon S3 supports both virtual-hosted–style and path-style URL access in all AWS Regions. However, path-style URLs will be discontinued in the future. For more information, see the following **Important** note.

In Amazon S3, path-style URLs use the following format:

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

For example, if you create a bucket named `amzn-s3-demo-bucket1` in the US West (Oregon) Region, and you want to access the puppy.jpg object in that bucket, you can use the following path-style URL:

```
https://s3.us-west-2.amazonaws.com/amzn-s3-demo-bucket1/puppy.jpg
```

**Important**

Update (September 23, 2020) – To make sure that customers have the time that they need to transition to virtual-hosted–style URLs, we have decided to delay the deprecation of path-style URLs. For more information, see Amazon S3 Path Deprecation Plan – The Rest of the Story in the *AWS News Blog*.

**Warning**

When hosting website content that will be accessed from a web browser, avoid using path-style URLs, which might interfere with the browser same origin security model. To host website content, we recommend that you use either S3 website endpoints or a CloudFront distribution. For more information, see Website endpoints and Deploy a React-based single-page application to Amazon S3 and CloudFront in the *AWS Perspective Guidance Patterns*.

### Virtual-hosted–style requests

In a virtual-hosted–style URI, the bucket name is part of the domain name in the URL.

Amazon S3 virtual-hosted–style URLs use the following format:

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

In this example, `amzn-s3-demo-bucket1` is the bucket name, US West (Oregon) is the Region, and puppy.png is the key name:

```
https://amzn-s3-demo-bucket1.s3.us-west-2.amazonaws.com/puppy.png
```

### HTTP Host header bucket specification

As long as your GET request does not use the SSL endpoint, you can specify the bucket for the request by using the HTTP Host header. The Host header in a REST request is interpreted as follows:

- If the Host header is omitted or its value is s3.*region-code*.amazonaws.com, the bucket for the request will be the first slash-delimited component of the Request-URI, and the key for the request will be the rest of the Request-URI. This is the ordinary method, as illustrated by the first and second examples in this section. Omitting the Host header is valid only for HTTP 1.0 requests.
- Otherwise, if the value of the Host header ends in .s3.*region-code*.amazonaws.com, the bucket name is the leading component of the Host header's value up to .s3.*region-code*.amazonaws.com. The key for the request is the Request-URI. This interpretation exposes buckets as subdomains of .s3.*region-code*.amazonaws.com, as illustrated by the third and fourth examples in this section.
- Otherwise, the bucket for the request is the lowercase value of the Host header, and the key for the request is the Request-URI. This interpretation is useful when you have registered the same DNS name as your bucket name and have configured that name to be a canonical name (CNAME) alias for Amazon S3. The procedure for registering domain names and configuring CNAME DNS records is beyond the scope of this guide, but the result is illustrated by the final example in this section.

### Examples

This section provides example URLs and requests.

### Example – Path-style URLs and requests

This example uses the following:

- Bucket Name - example.com
- Region - US East (N. Virginia)
- Key Name - homepage.html

The URL is as follows:

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

The request is as follows:

```
GET /example.com/homepage.html HTTP/1.1
Host: s3.us-east-1.amazonaws.com
```

The request with HTTP 1.0 and omitting the `Host` header is as follows:

```
GET /example.com/homepage.html HTTP/1.0
```

For information about DNS-compatible names, see Limitations. For more information about keys, see Keys.

**Example – Virtual-hosted–style URLs and requests**

This example uses the following:

- **Bucket name** - `amzn-s3-demo-bucket1`
- **Region** - Europe (Ireland)
- **Key name** - `homepage.html`

The URL is as follows:

```
http://amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com/homepage.html
```

The request is as follows:

```
GET /homepage.html HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com
```

**Example – CNAME alias method**

To use this method, you must configure your DNS name as a CNAME alias for *`bucket-name`*`.s3.us-east-1.amazonaws.com`. For more information, see Customizing Amazon S3 URLs with CNAME records.

This example uses the following:

- Bucket Name - `example.com`
- **Key name** - `homepage.html`

The URL is as follows:

```
http://www.example.com/homepage.html
```

The example is as follows:

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

**Customizing Amazon S3 URLs with CNAME records**

Depending on your needs, you might not want `s3.`*`region-code`*`.amazonaws.com` to appear on your website or service. For example, if you're hosting website images on Amazon S3, you might prefer `http://images.example.com/` instead of `http://images.example.com.s3.us-east-1.amazonaws.com/`. Any bucket with a DNS-compatible name can be referenced as follows: `http://`*`BucketName`*`.s3.`*`Region`*`.amazonaws.com/[`*`Filename`*`]`, for example, `http://images.example.com.s3.us-east-1.amazonaws.com/mydog.jpg`. By using CNAME, you can map `images.example.com` to an Amazon S3 hostname so that the previous URL could become `http://images.example.com/mydog.jpg`.

Your bucket name must be the same as the CNAME. For example, if you create a CNAME to map `images.example.com` to `images.example.com.s3.us-east-1.amazonaws.com`, both `http://images.example.com/filename` and `http://images.example.com.s3.us-east-1.amazonaws.com/filename` will be the same.

The CNAME DNS record should alias your domain name to the appropriate virtual hosted–style hostname. For example, if your bucket name and domain name are `images.example.com` and your bucket is in the US East (N. Virginia) Region, the CNAME record should alias to `images.example.com.s3.us-east-1.amazonaws.com`.

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3 uses the hostname to determine the bucket name. So the CNAME and the bucket name must be the same. For example, suppose that you have configured `www.example.com` as

a CNAME for `www.example.com.s3.us-east-1.amazonaws.com`. When you access `http://www.example.com`, Amazon S3 receives a request similar to the following:

**Example**

```
GET / HTTP/1.1
Host: www.example.com
Date: date
Authorization: signatureValue
```

Amazon S3 sees only the original hostname `www.example.com` and is unaware of the CNAME mapping used to resolve the request.

You can use any Amazon S3 endpoint in a CNAME alias. For example, `s3.ap-southeast-1.amazonaws.com` can be used in CNAME aliases. For more information about endpoints, see Request Endpoints. To create a static website by using a custom domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

> **Important**
>
> When using custom URLs with CNAMEs, you will need to ensure a matching bucket exists for any CNAME or alias record you configure. For example, if you create DNS entries for `www.example.com` and `login.example.com` to publish web content using S3, you will need to create both buckets `www.example.com` and `login.example.com`.
> When a CNAME or alias records is configured pointing to an S3 endpoint without a matching bucket, any AWS user can create that bucket and publish content under the configured alias, even if ownership is not the same.
> For the same reason, we recommend that you change or remove the corresponding CNAME or alias when deleting a bucket.

**How to associate a hostname with an Amazon S3 bucket**

**To associate a hostname with an Amazon S3 bucket by using a CNAME alias**

1. Select a hostname that belongs to a domain that you control.

   This example uses the `images` subdomain of the `example.com` domain.
2. Create a bucket that matches the hostname.

In this example, the host and bucket names are `images.example.com`. The bucket name must *exactly* match the hostname.

3. Create a CNAME DNS record that defines the hostname as an alias for the Amazon S3 bucket.

   For example:

   ```
   images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
   ```

   > **Important**
   >
   > For request-routing reasons, the CNAME DNS record must be defined exactly as shown in the preceding example. Otherwise, it might appear to operate correctly, but it will eventually result in unpredictable behavior.

   The procedure for configuring CNAME DNS records depends on your DNS server or DNS provider. For specific information, see your server documentation or contact your provider.

### Limitations

SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

### Backward compatibility

The following sections cover various aspects of Amazon S3 backward compatibility that relate to path-style and virtual-hosted–style URL requests.

### Legacy endpoints

Some Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, review the following information. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

**Important**

Although you might see legacy endpoints in your logs, we recommend that you always use the standard endpoint syntax to access your buckets.
Amazon S3 virtual-hosted–style URLs use the following format:

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

In Amazon S3, path-style URLs use the following format:

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

### s3-Region

Some older Amazon S3 Regions support endpoints that contain a dash (-) between s3 and the Region code (for example, s3-us-west-2), instead of a dot (for example, s3.us-west-2). If your bucket is in one of these Regions, you might see the following endpoint format in your server access logs or CloudTrail logs:

```
https://bucket-name.s3-region-code.amazonaws.com
```

In this example, the bucket name is amzn-s3-demo-bucket1 and the Region is US West (Oregon):

```
https://amzn-s3-demo-bucket1.s3-us-west-2.amazonaws.com
```

### Legacy global endpoint

For some Regions, you can use the legacy global endpoint to construct requests that do not specify a Region-specific endpoint. The legacy global endpoint point is as follows:

```
bucket-name.s3.amazonaws.com
```

In your server access logs or CloudTrail logs, you might see requests that use the legacy global endpoint. In this example, the bucket name is amzn-s3-demo-bucket1 and the legacy global endpoint is:

```
https://amzn-s3-demo-bucket1.s3.amazonaws.com
```

#### Virtual-hosted–style requests for US East (N. Virginia)

Requests made with the legacy global endpoint go to the US East (N. Virginia) Region by default. Therefore, the legacy global endpoint is sometimes used in place of the Regional endpoint for US East (N. Virginia). If you create a bucket in US East (N. Virginia) and use the global endpoint, Amazon S3 routes your request to this Region by default.

#### Virtual-hosted–style requests for other Regions

The legacy global endpoint is also used for virtual-hosted–style requests in other supported Regions. If you create a bucket in a Region that was launched before March 20, 2019, and use the legacy global endpoint, Amazon S3 updates the DNS record to reroute the request to the correct location, which might take time. In the meantime, the default rule applies, and your virtual-hosted–style request goes to the US East (N. Virginia) Region. Amazon S3 then redirects it with an HTTP 307 Temporary Redirect to the correct Region.

For S3 buckets in Regions launched after March 20, 2019, the DNS server doesn't route your request directly to the AWS Region where your bucket resides. It returns an HTTP 400 Bad Request error instead. For more information, see Making requests.

#### Path-style requests

For the US East (N. Virginia) Region, you can use the legacy global endpoint for path-style requests.

For all other Regions, the path-style syntax requires that you use the Region-specific endpoint when attempting to access a bucket. If you try to access a bucket with the legacy global endpoint or another endpoint that is different than the one for the Region where the bucket resides, you receive an HTTP response code 307 Temporary Redirect error and a message that indicates the correct URI for your resource. For example, if you use `https://s3.amazonaws.com/`*`bucket-name`* for a bucket that was created in the US West (Oregon) Region, you will receive an HTTP 307 Temporary Redirect error.

## Request redirection and the REST API

### Topics

- Redirects and HTTP user-agents
- Redirects and 100-Continue
- Redirect example

This section describes how to handle HTTP redirects by using the Amazon S3 REST API. For general information about Amazon S3 redirects, see Making requests in the Amazon Simple Storage Service API Reference.

**Redirects and HTTP user-agents**

Programs that use the Amazon S3 REST API should handle redirects either at the application layer or the HTTP layer. Many HTTP client libraries and user agents can be configured to correctly handle redirects automatically; however, many others have incorrect or incomplete redirect implementations.

Before you rely on a library to fulfill the redirect requirement, test the following cases:

- Verify all HTTP request headers are correctly included in the redirected request (the second request after receiving a redirect) including HTTP standards such as Authorization and Date.
- Verify non-GET redirects, such as PUT and DELETE, work correctly.
- Verify large PUT requests follow redirects correctly.
- Verify PUT requests follow redirects correctly if the 100-continue response takes a long time to arrive.

HTTP user-agents that strictly conform to RFC 2616 might require explicit confirmation before following a redirect when the HTTP request method is not GET or HEAD. It is generally safe to follow redirects generated by Amazon S3 automatically, as the system will issue redirects only to hosts within the amazonaws.com domain and the effect of the redirected request will be the same as that of the original request.

**Redirects and 100-Continue**

To simplify redirect handling, improve efficiencies, and avoid the costs associated with sending a redirected request body twice, configure your application to use 100-continues for PUT operations. When your application uses 100-continue, it does not send the request body until it receives an acknowledgement. If the message is rejected based on the headers, the body of the message is not sent. For more information about 100-continue, go to RFC 2616 Section 8.2.3

> **Note**
>
> According to RFC 2616, when using `Expect: Continue` with an unknown HTTP server, you should not wait an indefinite period before sending the request body. This is because

some HTTP servers do not recognize 100-continue. However, Amazon S3 does recognize if your request contains an `Expect: Continue` and will respond with a provisional 100-continue status or a final status code. Additionally, no redirect error will occur after receiving the provisional 100 continue go-ahead. This will help you avoid receiving a redirect response while you are still writing the request body.

### Redirect example

This section provides an example of client-server interaction using HTTP redirects and 100-continue.

Following is a sample PUT to the `quotes.s3.amazonaws.com` bucket.

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 returns the following:

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
  specified temporary endpoint. Continue to use the
  original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
```

```
</Error>
```

The client follows the redirect response and issues a new request to the `quotes.s3-4c25d83b.amazonaws.com` temporary endpoint.

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 returns a 100-continue indicating the client should proceed with sending the request body.

```
HTTP/1.1 100 Continue
```

The client sends the request body.

```
ha ha\n
```

Amazon S3 returns the final response.

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

# Developing with Amazon S3 using the AWS CLI

Follow these steps to download and configure AWS Command Line Interface (AWS CLI).

For a list of Amazon S3 AWS CLI commands, see the following pages in the *AWS CLI Command Reference*:

- s3

- s3api
- s3control

> **Note**
>
> Services in AWS, such as Amazon S3, require that you provide credentials when you access them. The service can then determine whether you have permissions to access the resources that it owns. The console requires your password. You can create access keys for your AWS account to access the AWS CLI or API. However, we don't recommend that you access AWS using the credentials for your AWS account. Instead, we recommend that you use AWS Identity and Access Management (IAM). Create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user that you created. You can then access AWS using a special URL and the credentials of that IAM user. For instructions, go to Creating Your First IAM user and Administrators Group in the *IAM User Guide*.

**To set up the AWS CLI**

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   - Getting Set Up with the AWS Command Line Interface
   - Configuring the AWS Command Line Interface

2. Add a named profile for the administrator user in the AWS CLI config file. You use this profile when executing the AWS CLI commands. For more information, see Named profiles for the AWS CLI in the *AWS Command Line Interface User Guide*.

   ```
   [adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   For a list of available AWS Regions, see Regions and Endpoints in the *AWS General Reference*.

3. Verify the setup by typing the following commands at the command prompt.

   - Try the `help` command to verify that the AWS CLI is installed on your computer:

```
aws help
```

- Run an S3 command using the `adminuser` credentials that you just created. To do this, add the `--profile` parameter to your command to specify the profile name. In this example, the `ls` command lists buckets in your account. The AWS CLI uses the `adminuser` credentials to authenticate the request.

```
aws s3 ls --profile adminuser
```

# Developing with Amazon S3 using the AWS SDKs

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

> **Note**
>
> You can use AWS Amplify for end-to-end fullstack development of web and mobile apps. Amplify Storage seamlessly integrates file storage and management capabilities into frontend web and mobile apps, built on top of Amazon S3. For more information, see Storage in the Amplify user guide.

## Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

| SDK documentation | Code examples |
| --- | --- |
| AWS SDK for C++ | AWS SDK for C++ code examples |
| AWS CLI | AWS CLI code examples |
| AWS SDK for Go | AWS SDK for Go code examples |

| SDK documentation | Code examples |
| --- | --- |
| AWS SDK for Java | AWS SDK for Java code examples |
| AWS SDK for JavaScript | AWS SDK for JavaScript code examples |
| AWS SDK for Kotlin | AWS SDK for Kotlin code examples |
| AWS SDK for .NET | AWS SDK for .NET code examples |
| AWS SDK for PHP | AWS SDK for PHP code examples |
| AWS Tools for PowerShell | Tools for PowerShell code examples |
| AWS SDK for Python (Boto3) | AWS SDK for Python (Boto3) code examples |
| AWS SDK for Ruby | AWS SDK for Ruby code examples |
| AWS SDK for Rust | AWS SDK for Rust code examples |
| AWS SDK for SAP ABAP | AWS SDK for SAP ABAP code examples |
| AWS SDK for Swift | AWS SDK for Swift code examples |

For examples specific to this service, see Code examples for Amazon S3 using AWS SDKs.

**Example availability**

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

## SDK Programming interfaces

Each AWS SDK provides one or more programmatic interfaces for working with Amazon S3. Each SDK provides a low-level interface for Amazon S3, with methods that closely resemble API operations. Some SDKs provide high-level interfaces for Amazon S3, that are abstractions intended to simplify common use cases.

For example, when you perform a multipart upload by using the low-level API operations, you need to use an operation to initiate the upload, another operation to upload parts, and a final operation to complete the upload. A high-level multipart upload API operation allows you to do all of the operations required for upload in a single API call. For examples, see Uploading an object using multipart upload.

Low-level API operations allow greater control over the upload. We recommend that you use the low-level API operations if you need to pause and resume uploads, vary part sizes during the upload, or begin uploads when you don't know the size of the data in advance.

## Specifying the Signature Version in Request Authentication

Amazon S3 supports only AWS Signature Version 4 in most AWS Regions. In some of the older AWS Regions, Amazon S3 supports both Signature Version 4 and Signature Version 2. However, Signature Version 2 is being turned off (deprecated). For more information about the end of support for Signature Version 2, see AWS Signature Version 2 Turned Off (Deprecated) for Amazon S3.

For a list of all the Amazon S3 Regions and the signature versions they support, see Regions and Endpoints in the *AWS General Reference*.

For all AWS Regions, AWS SDKs use Signature Version 4 by default to authenticate requests. When using AWS SDKs that were released before May 2016, you might be required to request Signature Version 4, as shown in the following table.

| SDK | Requesting Signature Version 4 for Request Authentication |
| --- | --- |
| AWS CLI | For the default profile, run the following command:<br>`$ aws configure set default.s3.signature_version s3v4`<br>For a custom profile, run the following command:<br>`$ aws configure set profile.your_profile_name.s3.signature_version s3v4` |
| Java SDK | Add the following in your code: |

| SDK | Requesting Signature Version 4 for Request Authentication |
| --- | --- |
| | `System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");`<br><br>Or, on the command line, specify the following:<br><br>`-Dcom.amazonaws.services.s3.enableV4` |
| JavaScript SDK | Set the `signatureVersion` parameter to `v4` when constructing the client:<br><br>`var s3 = new AWS.S3({signatureVersion: 'v4'});` |
| PHP SDK | Set the `signature` parameter to `v4` when constructing the Amazon S3 service client for PHP SDK v2:<br><br>`<?php`<br>`$client = S3Client::factory([`<br>`'region' => 'YOUR-REGION',`<br>`'version' => 'latest',`<br>`'signature' => 'v4'`<br>`]);`<br><br>When using the PHP SDK v3, set the `signature_version` parameter to `v4` during construction of the Amazon S3 service client:<br><br>`<?php`<br>`$s3 = new Aws\S3\S3Client([`<br>`'version' => '2006-03-01',`<br>`'region' => 'YOUR-REGION',`<br>`'signature_version' => 'v4'`<br>`]);` |
| Python-Boto SDK | Specify the following in the boto default config file:<br><br>`[s3] use-sigv4 = True` |

| SDK | Requesting Signature Version 4 for Request Authentication |
| --- | --- |
| Ruby SDK | Ruby SDK - Version 1: Set the `:s3_signature_version` parameter to `:v4` when constructing the client:<br>`s3 = AWS::S3::Client.new(:s3_signature_version => :v4)`<br>Ruby SDK - Version 3: Set the `signature_version` parameter to `v4` when constructing the client:<br>`s3 = Aws::S3::Client.new(signature_version: 'v4')` |
| .NET SDK | Add the following to the code before creating the Amazon S3 client:<br>`AWSConfigsS3.UseSignatureVersion4 = true;`<br>Or, add the following to the config file:<br>`<appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings>` |

## AWS Signature Version 2 Turned Off (Deprecated) for Amazon S3

Signature Version 2 is being turned off (deprecated) in Amazon S3. Amazon S3 will then only accept API requests that are signed using Signature Version 4.

This section provides answers to common questions regarding the end of support for Signature Version 2.

**What is Signature Version 2/4, and What Does It Mean to Sign Requests?**

The Signature Version 2 or Signature Version 4 signing process is used to authenticate your Amazon S3 API requests. Signing requests enables Amazon S3 to identify who is sending the request and protects your requests from bad actors.

For more information about signing AWS requests, see Signing AWS API Requests in the *AWS General Reference*.

**What Update Are You Making?**

We currently support Amazon S3 API requests that are signed using Signature Version 2 and Signature Version 4 processes. After that, Amazon S3 will only accept requests that are signed using Signature Version 4.

For more information about signing AWS requests, see Changes in Signature Version 4 in the *AWS General Reference*.

**Why Are You Making the Update?**

Signature Version 4 provides improved security by using a signing key instead of your secret access key. Signature Version 4 is currently supported in all AWS Regions, whereas Signature Version 2 is only supported in Regions that were launched before January 2014. This update allows us to provide a more consistent experience across all Regions.

**How Do I Ensure That I'm Using Signature Version 4, and What Updates Do I Need?**

The signature version that is used to sign your requests is usually set by the tool or the SDK on the client side. By default, the latest versions of our AWS SDKs use Signature Version 4. For third-party software, contact the appropriate support team for your software to confirm what version you need. If you are sending direct REST calls to Amazon S3, you must modify your application to use the Signature Version 4 signing process.

For information about which version of the AWS SDKs to use when moving to Signature Version 4, see Moving from Signature Version 2 to Signature Version 4.

For information about using Signature Version 4 with the Amazon S3 REST API, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

**What Happens if I Don't Make Updates?**

Requests signed with Signature Version 2 that are made after that will fail to authenticate with Amazon S3. Requesters will see errors stating that the request must be signed with Signature Version 4.

**Should I Make Changes Even if I'm Using a Presigned URL That Requires Me to Sign for More than 7 Days?**

If you are using a presigned URL that requires you to sign for more than 7 days, no action is currently needed. You can continue to use AWS Signature Version 2 to sign and authenticate the presigned URL. We will follow up and provide more details on how to migrate to Signature Version 4 for a presigned URL scenario.

**More Info**

- For more information about using Signature Version 4, see Signing AWS API Requests.
- View the list of changes between Signature Version 2 and Signature Version 4 in Changes in Signature Version 4.
- View the post AWS Signature Version 4 to replace AWS Signature Version 2 for signing Amazon S3 API requests in the AWS forums.
- If you have any questions or concerns, contact AWS Support.

## Moving from Signature Version 2 to Signature Version 4

If you currently use Signature Version 2 for Amazon S3 API request authentication, you should move to using Signature Version 4. Support is ending for Signature Version 2, as described in AWS Signature Version 2 Turned Off (Deprecated) for Amazon S3.

For information about using Signature Version 4 with the Amazon S3 REST API, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

The following table lists the SDKs with the necessary minimum version to use Signature Version 4 (SigV4). If you are using presigned URLs with the AWS Java, JavaScript (Node.js), or Python (Boto/CLI) SDKs, you must set the correct AWS Region and set Signature Version 4 in the client configuration. For information about setting `SigV4` in the client configuration, see Specifying the Signature Version in Request Authentication.

| If you use this SDK/ Product | Upgrade to this SDK version | Code change needed to the client to use Sigv4? | Link to SDK documentation |
|---|---|---|---|
| AWS SDK for Java v1 | Upgrade to Java 1.11.201+ or v2. | Yes | Specifying the Signature Version in Request Authentication |
| AWS SDK for Java v2 | No SDK upgrade is needed. | No | AWS SDK for Java |
| AWS SDK for .NET v1 | Upgrade to 3.1.10 or later. | Yes | AWS SDK for .NET |
| AWS SDK for .NET v2 | Upgrade to 3.1.10 or later. | No | AWS SDK for .NET v2 |
| AWS SDK for .NET v3 | Upgrade to 3.3.0.0 or later. | Yes | AWS SDK for .NET v3 |
| AWS SDK for JavaScript v1 | Upgrade to 2.68.0 or later. | Yes | AWS SDK for JavaScript |
| AWS SDK for JavaScript v2 | Upgrade to 2.68.0 or later. | Yes | AWS SDK for JavaScript |
| AWS SDK for JavaScript v3 | No action is currently needed. Upgrade to | No | AWS SDK for JavaScript |

| If you use this SDK/ Product | Upgrade to this SDK version | Code change needed to the client to use Sigv4? | Link to SDK documentation |
|---|---|---|---|
| | major version V3 in Q3 2019. | | |
| AWS SDK for PHP v1 | Recommend to upgrade to the most recent version of PHP or, at least to v2.7.4 with the signature parameter set to v4 in the S3 client's configura tion. | Yes | AWS SDK for PHP |

| If you use this SDK/ Product | Upgrade to this SDK version | Code change needed to the client to use Sigv4? | Link to SDK documentation |
|---|---|---|---|
| AWS SDK for PHP v2 | Recommend to upgrade to the most recent version of PHP or, at least to v2.7.4 with the signature parameter set to v4 in the S3 client's configuration. | No | AWS SDK for PHP |
| AWS SDK for PHP v3 | No SDK upgrade is needed. | No | AWS SDK for PHP |
| Boto2 | Upgrade to Boto2 v2.49.0. | Yes | Boto 2 Upgrade |
| Boto3 | Upgrade to 1.5.71 (Botocore), 1.4.6 (Boto3). | Yes | Boto 3 - AWS SDK for Python |
| AWS CLI | Upgrade to 1.11.108. | Yes | AWS Command Line Interface |

| If you use this SDK/ Product | Upgrade to this SDK version | Code change needed to the client to use Sigv4? | Link to SDK documentation |
|---|---|---|---|
| AWS CLI v2 (preview) | No SDK upgrade is needed. | No | AWS Command Line Interface version 2 |
| AWS SDK for Ruby v1 | Upgrade to Ruby V3. | Yes | Ruby V3 for AWS |
| AWS SDK for Ruby v2 | Upgrade to Ruby V3. | Yes | Ruby V3 for AWS |
| AWS SDK for Ruby v3 | No SDK upgrade is needed. | No | Ruby V3 for AWS |
| Go | No SDK upgrade is needed. | No | AWS SDK for Go |
| C++ | No SDK upgrade is needed. | No | AWS SDK for C++ |

**AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core**

If you are using module versions *earlier* than 3.3.0.0, you must upgrade to 3.3.0.0.

To get the version information, use the `Get-Module` cmdlet:

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

To update the 3.3.0.0 version, use the `Update-Module` cmdlet:

```
Update-Module -Name AWSPowershell
Update-Module -Name AWSPowershell.NetCore
```

You can use presigned URLs that are valid for more than 7 days that you will send Signature Version 2 traffic on.

# Developing with Amazon S3 using the REST API

The Amazon S3 architecture is designed to be programming language-neutral, using our supported interfaces to store and retrieve objects.

Amazon S3 currently provides a REST interface. With REST, metadata is returned in HTTP headers. Because we only support HTTP requests of up to 4 KB (not including the body), the amount of metadata you can supply is restricted. The REST API is an HTTP interface to Amazon S3. Using REST, you use standard HTTP requests to create, fetch, and delete buckets and objects.

You can use any toolkit that supports HTTP to use the REST API. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses the standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matched the style of standard HTTP usage.

For more information about sending requests using the REST API, see Making requests using the REST API. For some considerations you should keep in mind when using the REST API, see the topics below.

For more information about the Amazon S3 REST API, see the *Amazon Simple Storage Service API Reference*.

**Topics**

- Request routing

# Request routing

Programs that make requests against buckets created using the CreateBucket API that include a CreateBucketConfiguration must support redirects. Additionally, some clients that do not respect DNS TTLs might encounter issues.

This section describes routing and DNS issues to consider when designing your service or application for use with Amazon S3.

## Request redirection and the REST API

Amazon S3 uses the Domain Name System (DNS) to route requests to facilities that can process them. This system works effectively, but temporary routing errors can occur. If a request arrives at the wrong Amazon S3 location, Amazon S3 responds with a temporary redirect that tells the requester to resend the request to a new endpoint. If a request is incorrectly formed, Amazon S3 uses permanent redirects to provide direction on how to perform the request correctly.

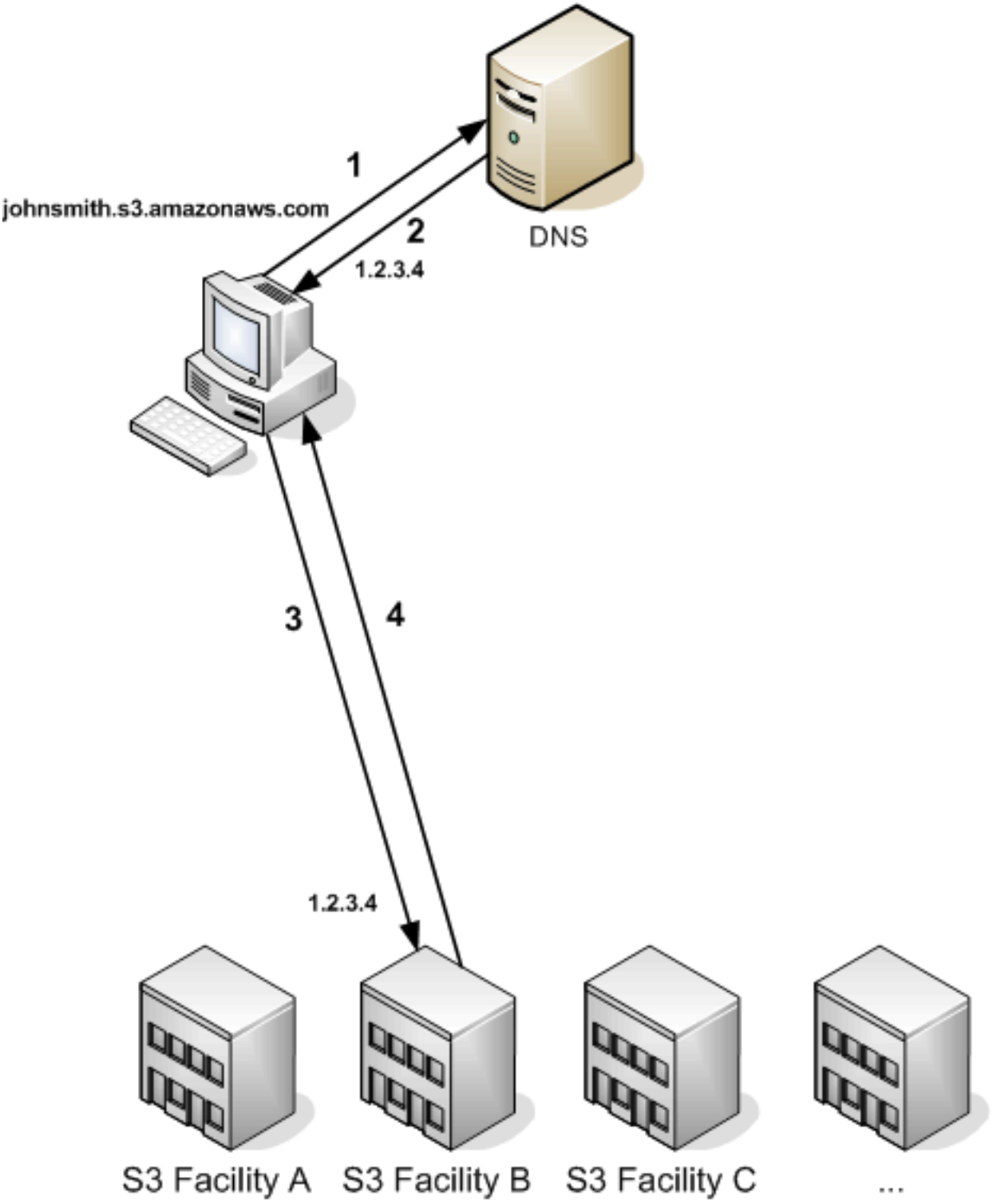
**Important**

To use this feature, you must have an application that can handle Amazon S3 redirect responses. The only exception is for applications that work exclusively with buckets that were created without `<CreateBucketConfiguration>`. For more information about location constraints, see Accessing and listing an Amazon S3 bucket.
For all Regions that launched after March 20, 2019, if a request arrives at the wrong Amazon S3 location, Amazon S3 returns an HTTP 400 Bad Request error.
For more information about enabling or disabling an AWS Region, see AWS Regions and Endpoints in the *AWS General Reference*.

### Topics

### DNS routing

DNS routing routes requests to appropriate Amazon S3 facilities. The following figure and procedure show an example of DNS routing.



### DNS routing request steps

1. The client makes a DNS request to get an object stored on Amazon S3.
2. The client receives one or more IP addresses for facilities that can process the request. In this example, the IP address is for Facility B.
3. The client makes a request to Amazon S3 Facility B.
4. Facility B returns a copy of the object to the client.

## Temporary request redirection

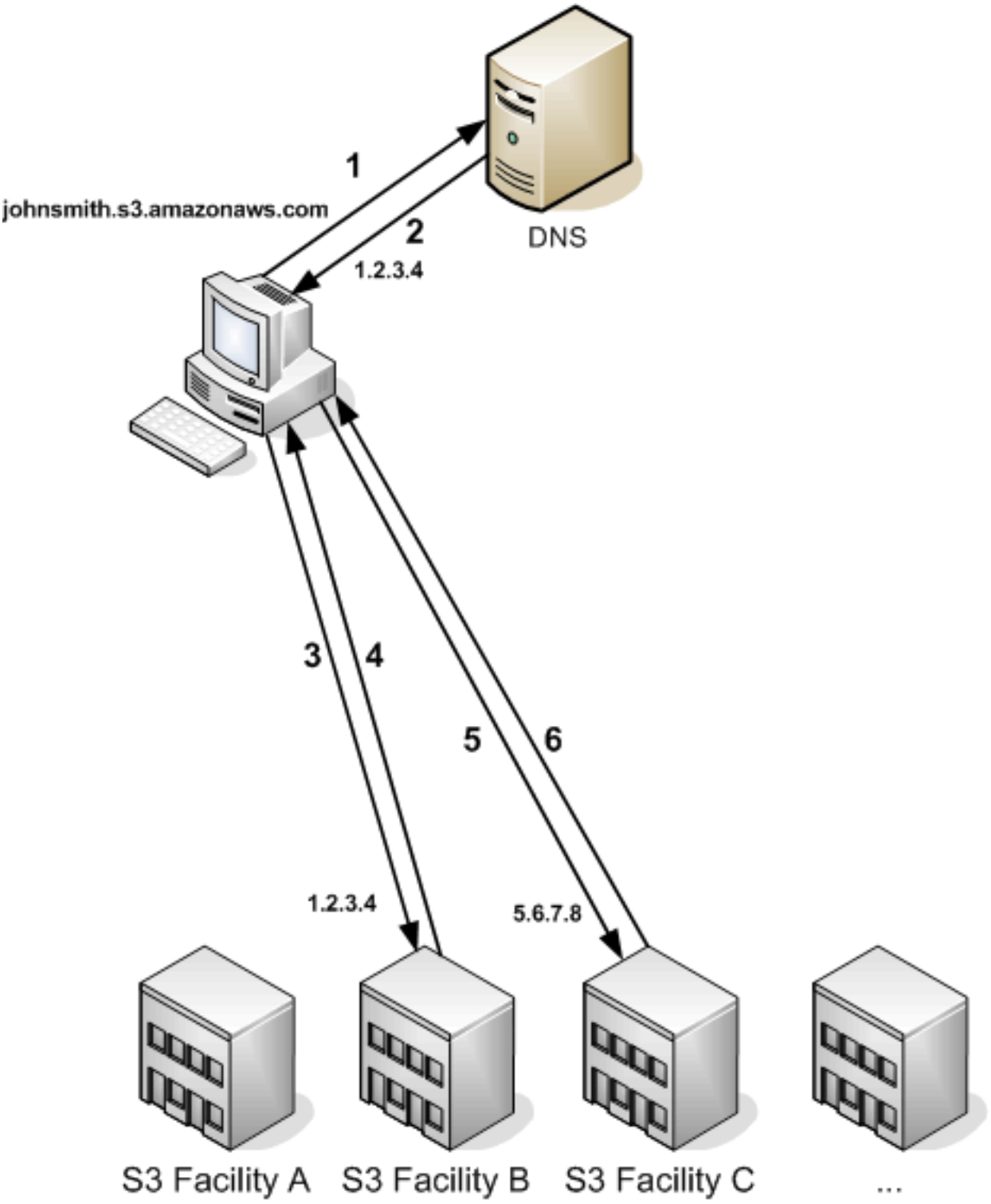
A temporary redirect is a type of error response that signals to the requester that they should resend the request to a different endpoint. Due to the distributed nature of Amazon S3, requests can be temporarily routed to the wrong facility. This is most likely to occur immediately after buckets are created or deleted.

For example, if you create a new bucket and immediately make a request to the bucket, you might receive a temporary redirect, depending on the location constraint of the bucket. If you created the bucket in the US East (N. Virginia) AWS Region, you will not see the redirect because this is also the default Amazon S3 endpoint.

However, if the bucket is created in any other Region, any requests for the bucket go to the default endpoint while the bucket's DNS entry is propagated. The default endpoint redirects the request to the correct endpoint with an HTTP 302 response. Temporary redirects contain a URI to the correct facility, which you can use to immediately resend the request.

> **Important**
>
> Don't reuse an endpoint provided by a previous redirect response. It might appear to work (even for long periods of time), but it might provide unpredictable results and will eventually fail without notice.

The following figure and procedure shows an example of a temporary redirect.




**Temporary request redirection steps**

1. The client makes a DNS request to get an object stored on Amazon S3.
2. The client receives one or more IP addresses for facilities that can process the request.

3. The client makes a request to Amazon S3 Facility B.
4. Facility B returns a redirect indicating the object is available from Location C.
5. The client resends the request to Facility C.
6. Facility C returns a copy of the object.

### Permanent request redirection

A permanent redirect indicates that your request addressed a resource inappropriately. For example, permanent redirects occur if you use a path-style request to access a bucket that was created using `<CreateBucketConfiguration>`. For more information, see Accessing and listing an Amazon S3 bucket.

To help you find these errors during development, this type of redirect does not contain a Location HTTP header that allows you to automatically follow the request to the correct location. Consult the resulting XML error document for help using the correct Amazon S3 endpoint.

### Request redirection examples

The following are examples of temporary request redirection responses.

#### REST API temporary redirect response

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?
rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
  Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

### SOAP API temporary redirect response

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

## DNS considerations

One of the design requirements of Amazon S3 is extremely high availability. One of the ways we meet this requirement is by updating the IP addresses associated with the Amazon S3 endpoint in DNS as needed. These changes are automatically reflected in short-lived clients, but not in some long-lived clients. Long-lived clients will need to take special action to re-resolve the Amazon S3 endpoint periodically to benefit from these changes. For more information about virtual machines (VMs), refer to the following:

- For Java, Sun's JVM caches DNS lookups forever by default; go to the "InetAddress Caching" section of the InetAddress documentation for information on how to change this behavior.
- For PHP, the persistent PHP VM that runs in the most popular deployment configurations caches DNS lookups until the VM is restarted. Go to the getHostByName PHP docs.

# Handling REST and SOAP errors

### Topics

This section describes REST and SOAP errors and how to handle them.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

## The REST error response

If a REST request results in an error, the HTTP reply has:

- An XML error document as the response body
- Content-Type: application/xml
- An appropriate 3xx, 4xx, or 5xx HTTP status code

Following is an example of a REST Error Response.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

For more information about Amazon S3 errors, go to ErrorCodeList.

### Response headers

Following are response headers returned by all operations:

- `x-amz-request-id:` A unique ID assigned to each request by the system. In the unlikely event that you have problems with Amazon S3, Amazon can use this to help troubleshoot the problem.
- `x-amz-id-2:` A special token that will help us to troubleshoot problems.

## Error response

When an Amazon S3 request is in error, the client receives an error response. The exact format of the error response is API specific: For example, the REST error response differs from the SOAP error response. However, all error responses have common elements.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

### Error code

The error code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. Many error codes are common across SOAP and REST APIs, but some are API-specific. For example, NoSuchKey is universal, but UnexpectedContent can occur only in response to an invalid REST request. In all cases, SOAP fault codes carry a prefix as indicated in the table of error codes, so that a NoSuchKey error is actually returned in SOAP as Client.NoSuchKey.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

### Error message

The error message contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they don't know how or don't care to handle. Sophisticated programs with

more exhaustive error handling and proper internationalization are more likely to ignore the error message.

**Further details**

Many error responses contain additional structured data meant to be read and understood by a developer diagnosing programming errors. For example, if you send a Content-MD5 header with a REST PUT request that doesn't match the digest calculated on the server, you receive a BadDigest error. The error response also includes as detail elements the digest we calculated, and the digest you told us to expect. During development, you can use this information to diagnose the error. In production, a well-behaved program might include this information in its error log.

## The SOAP error response

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

In SOAP, an error result is returned to the client as a SOAP fault, with the HTTP response code 500. If you do not receive a SOAP fault, then your request was successful. The Amazon S3 SOAP fault code is comprised of a standard SOAP 1.1 fault code (either "Server" or "Client") concatenated with the Amazon S3-specific error code. For example: "Server.InternalError" or "Client.NoSuchBucket". The SOAP fault string element contains a generic, human readable error message in English. Finally, the SOAP fault detail element contains miscellaneous information relevant to the error.

For example, if you attempt to delete the object "Fred", which does not exist, the body of the SOAP response contains a "NoSuchKey" SOAP fault.

**Example**

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
```

```
  </soapenv:Fault>
</soapenv:Body>
```

For more information about Amazon S3 errors, go to ErrorCodeList.

## Amazon S3 error best practices

When designing an application for use with Amazon S3, it is important to handle Amazon S3 errors appropriately. This section describes issues to consider when designing your application.

### Retry InternalErrors

Internal errors are errors that occur within the Amazon S3 environment.

Requests that receive an InternalError response might not have processed. For example, if a PUT request returns InternalError, a subsequent GET might retrieve the old value or the updated value.

If Amazon S3 returns an InternalError response, retry the request.

### Tune application for repeated SlowDown errors

As with any distributed system, S3 has protection mechanisms which detect intentional or unintentional resource over-consumption and react accordingly. SlowDown errors can occur when a high request rate triggers one of these mechanisms. Reducing your request rate will decrease or eliminate errors of this type. Generally speaking, most users will not experience these errors regularly; however, if you would like more information or are experiencing high or unexpected SlowDown errors, please post to our Amazon S3 developer forum or sign up for AWS Support https://aws.amazon.com/premiumsupport/.

### Isolate errors

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

Amazon S3 provides a set of error codes that are used by both the SOAP and REST API. The SOAP API returns standard Amazon S3 error codes. The REST API is designed to look like a standard HTTP server and interact with existing HTTP clients (e.g., browsers, HTTP client libraries, proxies, caches,

and so on). To ensure the HTTP clients handle errors properly, we map each Amazon S3 error to an HTTP status code.

HTTP status codes are less expressive than Amazon S3 error codes and contain less information about the error. For example, the `NoSuchKey` and `NoSuchBucket` Amazon S3 errors both map to the `HTTP 404 Not Found` status code.

Although the HTTP status codes contain less information about the error, clients that understand HTTP, but not the Amazon S3 API, will usually handle the error correctly.

Therefore, when handling errors or reporting Amazon S3 errors to end users, use the Amazon S3 error code instead of the HTTP status code as it contains the most information about the error. Additionally, when debugging your application, you should also consult the human readable <Details> element of the XML error response.

# Developer reference

This appendix include the following sections.

**Topics**



## Appendix a: Using the SOAP API

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

This section contains information specific to the Amazon S3 SOAP API.

> **Note**
>
> SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

**Topics**



## Common SOAP API elements

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

You can interact with Amazon S3 using SOAP 1.1 over HTTP. The Amazon S3 WSDL, which describes the Amazon S3 API in a machine-readable way, is available at: https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl. The Amazon S3 schema is available at https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd.

Most users will interact with Amazon S3 using a SOAP toolkit tailored for their language and development environment. Different toolkits will expose the Amazon S3 API in different ways. Please refer to your specific toolkit documentation to understand how to use it. This section illustrates the Amazon S3 SOAP operations in a toolkit-independent way by exhibiting the XML requests and responses as they appear "on the wire."

### Common elements

You can include the following authorization-related elements with any SOAP request:

- `AWSAccessKeyId:` The AWS Access Key ID of the requester
- `Timestamp:` The current time on your system
- `Signature:` The signature for the request

## Authenticating SOAP requests

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

Every non-anonymous request must contain authentication information to establish the identity of the principal making the request. In SOAP, the authentication information is put into the following elements of the SOAP request:

- Your AWS Access Key ID

  > **Note**
  >
  > When making authenticated SOAP requests, temporary security credentials are not supported. For more information about types of credentials, see Making requests.

- `Timestamp:` This must be a dateTime (go to http://www.w3.org/TR/xmlschema-2/#dateTime) in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as `2009-01-01T12:00:00.000Z`. Authorization will fail if this timestamp is more than 15 minutes away from the clock on Amazon S3 servers.
- `Signature:` The RFC 2104 HMAC-SHA1 digest (go to http://www.ietf.org/rfc/rfc2104.txt) of the concatenation of "AmazonS3" + OPERATION + Timestamp, using your AWS Secret Access Key as the key. For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

### Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
```

```
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

**Note**

SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

**Important**

Due to different interpretations regarding how extra time precision should be dropped, .NET users should take care not to send Amazon S3 overly specific time stamps. This can be accomplished by manually constructing `DateTime` objects with only millisecond precision.

## Setting access policy with SOAP

**Note**

SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

Access control can be set at the time a bucket or object is written by including the "AccessControlList" element with the request to `CreateBucket`, `PutObjectInline`, or `PutObject`. The AccessControlList element is described in Identity and Access Management for Amazon S3. If no access control list is specified with these operations, the resource is created with a default access policy that gives the requester FULL_CONTROL access (this is the case even if the request is a PutObjectInline or PutObject request for an object that already exists).

Following is a request that writes data to an object, makes the object readable by anonymous principals, and gives the specified user FULL_CONTROL rights to the bucket (Most developers will want to give themselves FULL_CONTROL access to their own bucket).

### Example

Following is a request that writes data to an object and makes the object readable by anonymous principals.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
```

```
</PutObjectInlineResponse>
```

The access control policy can be read or set for an existing bucket or object using the `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy`, and `SetObjectAccessControlPolicy` methods. For more information, see the detailed explanation of these methods.

## Appendix b: Authenticating requests (AWS signature version 2)

**Important**

This section describes how to authenticate requests using AWS Signature Version 2. Signature Version 2 is being turned off (deprecated), Amazon S3 will only accept API requests that are signed using Signature Version 4. For more information, see AWS Signature Version 2 Turned Off (Deprecated) for Amazon S3
Signature Version 4 is supported in all AWS Regions, and it is the only version that is supported for new Regions. For more information, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.
Amazon S3 offers you the ability to identify what API signature version was used to sign a request. It is important to identify if any of your workflows are utilizing Signature Version 2 signing and upgrading them to use Signature Version 4 to prevent impact to your business.

- If you are using CloudTrail event logs(recommended option), please see Identifying Amazon S3 Signature Version 2 requests by using CloudTrail on how to query and identify such requests.
- If you are using the Amazon S3 Server Access logs, see Identifying Signature Version 2 requests by using Amazon S3 access logs

**Topics**

## Authenticating requests using the REST API

When accessing Amazon S3 using REST, you must provide the following items in your request so the request can be authenticated:

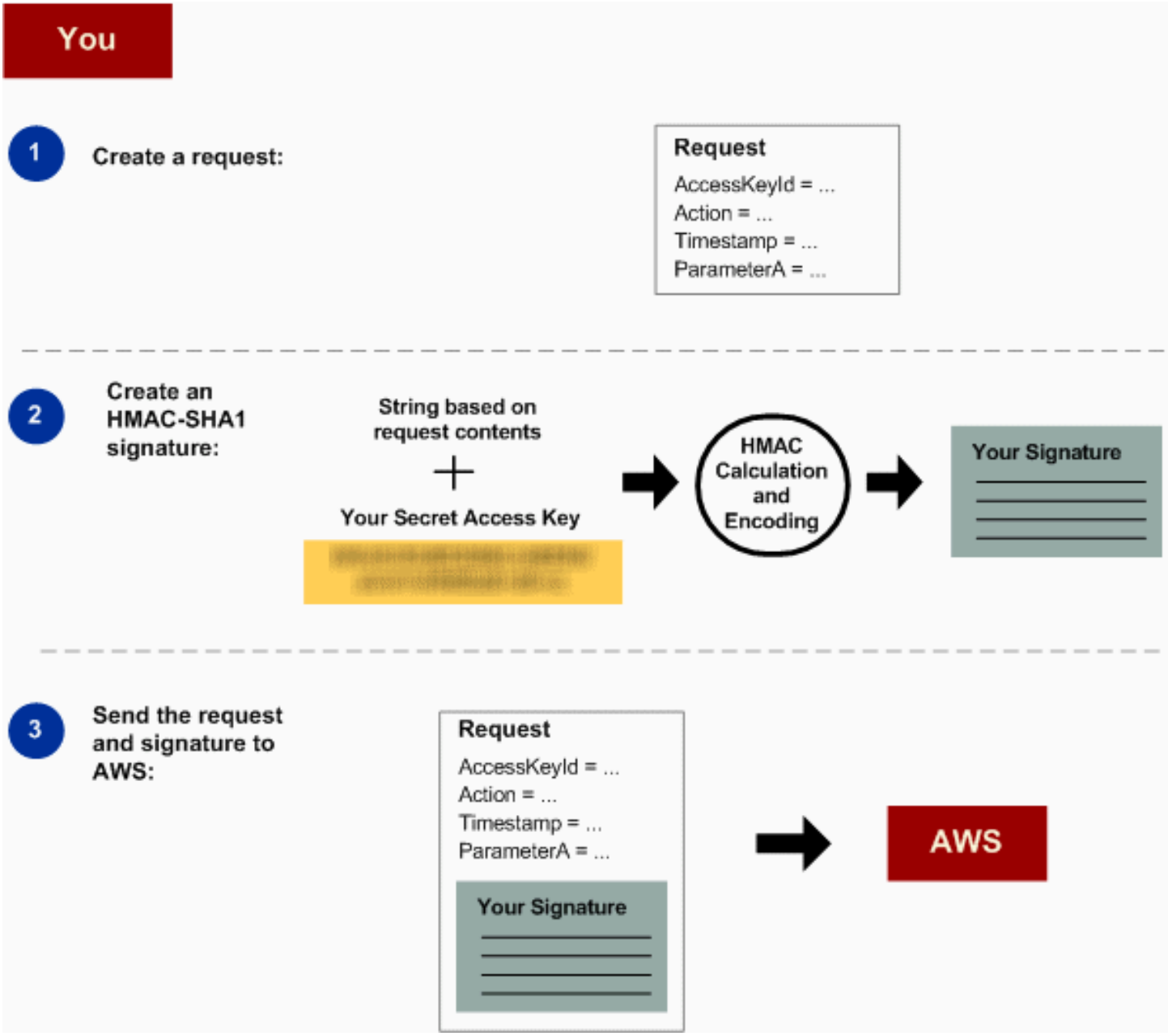
**Request elements**

- **AWS access key Id** – Each request must contain the access key ID of the identity you are using to send your request.
- **Signature** – Each request must contain a valid request signature, or the request is rejected.

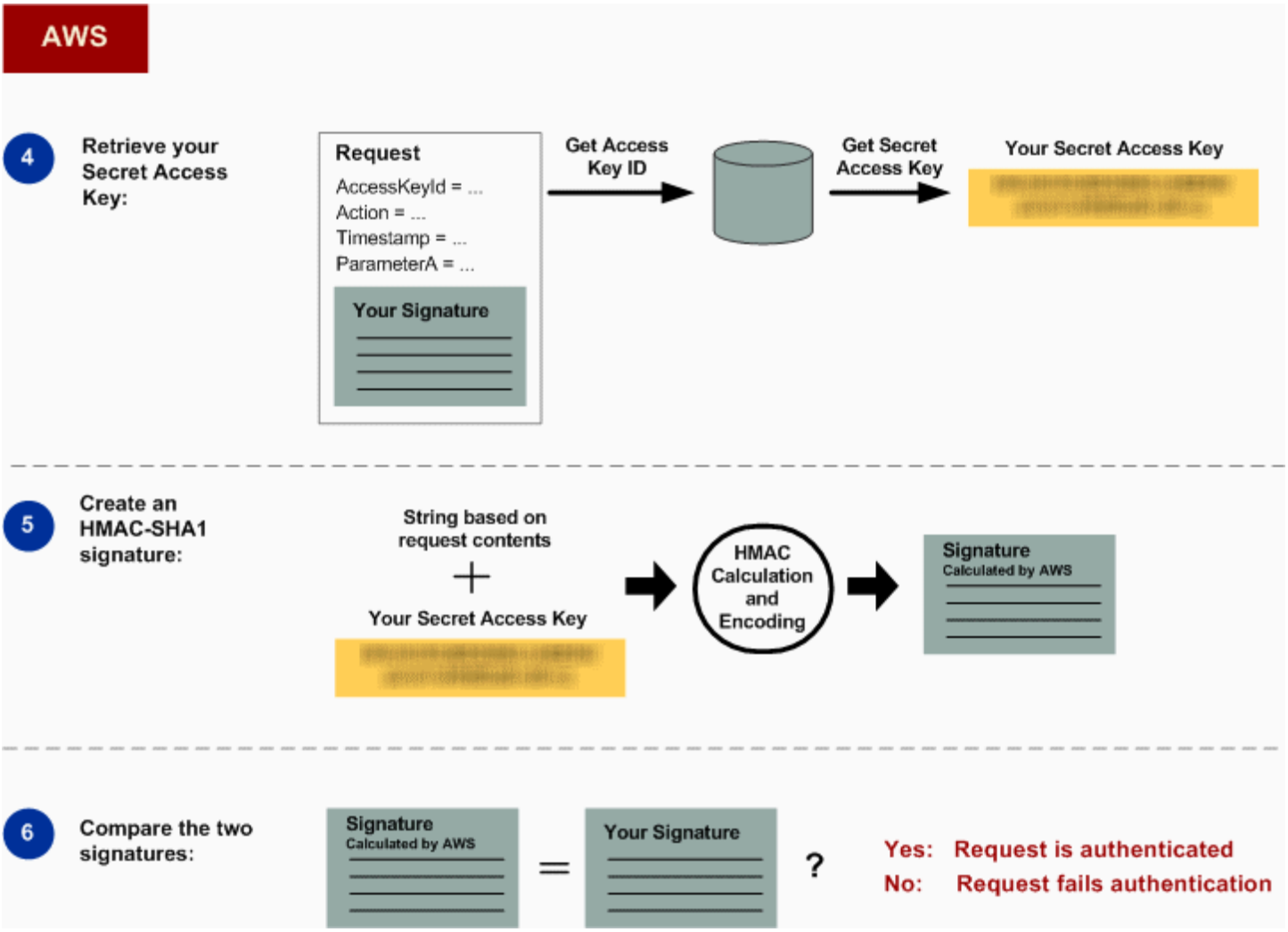
  A request signature is calculated using your secret access key, which is a shared secret known only to you and AWS.
- **Time stamp** – Each request must contain the date and time the request was created, represented as a string in UTC.
- **Date** – Each request must contain the time stamp of the request.

  Depending on the API action you're using, you can provide an expiration date and time for the request instead of or in addition to the time stamp. See the authentication topic for the particular action to determine what it requires.

Following are the general steps for authenticating requests to Amazon S3. It is assumed you have the necessary security credentials, access key ID and secret access key.




| | |
|---|---|
| 1 | Construct a request to AWS. |
| 2 | Calculate the signature using your secret access key. |
| 3 | Send the request to Amazon S3. Include your access key ID and the signature in your request. Amazon S3 performs the next three steps. |




| | |
|---|---|
| 4 | Amazon S3 uses the access key ID to look up your secret access key. |
| 5 | Amazon S3 calculates a signature from the request data and the secret access key using the same algorithm that you used to calculate the signature you sent in the request. |
| 6 | If the signature generated by Amazon S3 matches the one you sent in the request, the request is considered authentic. If the comparison fails, the request is discarded, and Amazon S3 returns an error response. |

### Detailed authentication information

For detailed information about REST authentication, see Signing and authenticating REST requests.

## Signing and authenticating REST requests

### Topics



> **Note**
>
> This topic explains authenticating requests using Signature Version 2. Amazon S3 now supports the latest Signature Version 4. This latest signature version is supported in all regions and any new regions after January 30, 2014 will support only Signature Version 4. For more information, go to Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

Authentication is the process of proving your identity to the system. Identity is an important factor in Amazon S3 access control decisions. Requests are allowed or denied in part based on the identity of the requester. For example, the right to create buckets is reserved for registered developers and (by default) the right to create objects in a bucket is reserved for the owner of the bucket in question. As a developer, you'll be making requests that invoke these privileges, so you'll need to prove your identity to the system by authenticating your requests. This section shows you how.

> **Note**
>
> The content in this section does not apply to HTTP POST. For more information, see Browser-based uploads using POST (AWS signature version 2).

The Amazon S3 REST API uses a custom HTTP scheme based on a keyed-HMAC (Hash Message Authentication Code) for authentication. To authenticate a request, you first concatenate selected elements of the request to form a string. You then use your AWS secret access key to calculate the HMAC of that string. Informally, we call this process "signing the request," and we call the output of the HMAC algorithm the signature, because it simulates the security properties of a real signature. Finally, you add this signature as a parameter of the request by using the syntax described in this section.

When the system receives an authenticated request, it fetches the AWS secret access key that you claim to have and uses it in the same way to compute a signature for the message it received. It then compares the signature it calculated against the signature presented by the requester. If the two signatures match, the system concludes that the requester must have access to the AWS secret access key and therefore acts with the authority of the principal to whom the key was issued. If the two signatures do not match, the request is dropped and the system responds with an error message.

**Example Authenticated Amazon S3 REST request**

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

**Using temporary security credentials**

If you are signing your request using temporary security credentials (see Making requests), you must include the corresponding security token in your request by adding the `x-amz-security-token` header.

When you obtain temporary security credentials using the AWS Security Token Service API, the response includes temporary security credentials and a session token. You provide the session token value in the `x-amz-security-token` header when you send requests to Amazon S3. For information about the AWS Security Token Service API provided by IAM, go to Action in the *AWS Security Token Service API Reference Guide* .

### The authentication header

The Amazon S3 REST API uses the standard HTTP `Authorization` header to pass authentication information. (The name of the standard header is unfortunate because it carries authentication information, not authorization.) Under the Amazon S3 authentication scheme, the Authorization header has the following form:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Developers are issued an AWS access key ID and AWS secret access key when they register. For request authentication, the `AWSAccessKeyId` element identifies the access key ID that was used to compute the signature and, indirectly, the developer making the request.

The `Signature` element is the RFC 2104 HMAC-SHA1 of selected elements from the request, and so the `Signature` part of the Authorization header will vary from request to request. If the request signature calculated by the system matches the `Signature` included with the request, the requester will have demonstrated possession of the AWS secret access key. The request will then be processed under the identity, and with the authority, of the developer to whom the key was issued.

Following is pseudogrammar that illustrates the construction of the `Authorization` request header. (In the example, \n means the Unicode code point `U+000A`, commonly called newline).

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-
Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
 Content-MD5 + "\n" +
 Content-Type + "\n" +
 Date + "\n" +
 CanonicalizedAmzHeaders +
 CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
 <HTTP-Request-URI, from the protocol name up to the query string> +
 [ subresource, if present. For example "?acl", "?location", or "?logging"];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 is an algorithm defined by RFC 2104 - Keyed-Hashing for Message Authentication. The algorithm takes as input two byte-strings, a key and a message. For Amazon S3 request authentication, use your AWS secret access key (`YourSecretAccessKey`) as the key, and the UTF-8 encoding of the `StringToSign` as the message. The output of HMAC-SHA1 is also a byte string, called the digest. The `Signature` request parameter is constructed by Base64 encoding this digest.

**Request canonicalization for signing**

Recall that when the system receives an authenticated request, it compares the computed request signature with the signature provided in the request in `StringToSign`. For that reason, you must compute the signature by using the same method used by Amazon S3. We call the process of putting a request in an agreed-upon form for signing *canonicalization*.

**Constructing the CanonicalizedResource element**

`CanonicalizedResource` represents the Amazon S3 resource targeted by the request. Construct it for a REST request as follows:

**Launch process**

| | |
|---|---|
| 1 | Start with an empty string (""). |
| 2 | If the request specifies a bucket using the HTTP Host header (virtual hosted-style), append the bucket name preceded by a "/" (e.g., "/bucketname"). For path-style requests and requests that don't address a bucket, do nothing. For more information about virtual hosted-style requests, see Virtual hosting of buckets.<br><br>For a virtual hosted-style request "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", the `CanonicalizedResource` is "/awsexamplebucket1".<br><br>For the path-style request, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", the `CanonicalizedResource` is "". |
| 3 | Append the path part of the un-decoded HTTP Request-URI, up-to but not including the query string.<br><br>For a virtual hosted-style request "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", the `CanonicalizedResource` is "/awsexamplebucket1/photos/puppy.jpg". |

| | |
|---|---|
| | For a path-style request, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", the `CanonicalizedResource` is "/awsexamplebucket1/photos/puppy.jpg". At this point, the `CanonicalizedResource` is the same for both the virtual hosted-style and path-style request.<br><br>For a request that does not address a bucket, such as GET Service, append "/". |
| 4 | If the request addresses a subresource, such as `?versioning`, `?location`, `?acl`, `?lifecycle`, or `?versionid`, append the subresource, its value if it has one, and the question mark. Note that in case of multiple subresources, subresources must be lexicographically sorted by subresource name and separated by '&', e.g., ?acl&versionId=*value*.<br><br>The subresources that must be included when constructing the CanonicalizedResource Element are acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, uploadId, uploads, versionId, versioning, versions, and website.<br><br>If the request specifies query string parameters overriding the response header values (see Get Object), append the query string parameters and their values. When signing, you do not encode these values; however, when making the request, you must encode these parameter values. The query string parameters in a GET request include `response-content-type`, `response-content-language`, `response-expires`, `response-cache-control`, `response-content-disposition`, and `response-content-encoding`.<br><br>The `delete` query string parameter must be included when you create the CanonicalizedResource for a multi-object Delete request. |

Elements of the CanonicalizedResource that come from the HTTP Request-URI should be signed literally as they appear in the HTTP request, including URL-Encoding meta characters.

The `CanonicalizedResource` might be different than the HTTP Request-URI. In particular, if your request uses the HTTP `Host` header to specify a bucket, the bucket does not appear in the HTTP Request-URI. However, the `CanonicalizedResource` continues to include the bucket. Query string parameters might also appear in the Request-URI but are not included in `CanonicalizedResource`. For more information, see Virtual hosting of buckets.

### Constructing the CanonicalizedAmzHeaders element

To construct the CanonicalizedAmzHeaders part of `StringToSign`, select all HTTP request headers that start with 'x-amz-' (using a case-insensitive comparison), and use the following process.

### CanonicalizedAmzHeaders process

| | |
|---|---|
| 1 | Convert each HTTP header name to lowercase. For example, '`X-Amz-Date `' becomes '`x-amz-date `'. |
| 2 | Sort the collection of headers lexicographically by header name. |
| 3 | Combine header fields with the same name into one "header-name:comma-separated-value-list" pair as prescribed by RFC 2616, section 4.2, without any spaces between values. For example, the two metadata headers '`x-amz-meta-username: fred  `' and '`x-amz-meta-username: barney  `' would be combined into the single header '`x-amz-meta-username: fred,barney  `'. |
| 4 | "Unfold" long headers that span multiple lines (as allowed by RFC 2616, section 4.2) by replacing the folding spaces (including new-line) by a single space. |
| 5 | Trim any spaces around the colon in the header. For example, the header '`x-amz-meta-username: fred,barney  `' would become '`x-amz-meta-username:fred,barney `' |
| 6 | Finally, append a newline character (`U+000A`) to each canonicalized header in the resulting list. Construct the CanonicalizedResource element by concatenating all headers in this list into a single string. |

### Positional versus named HTTP header StringToSign elements

The first few header elements of `StringToSign` (Content-Type, Date, and Content-MD5) are positional in nature. `StringToSign` does not include the names of these headers, only their values from the request. In contrast, the '`x-amz-`' elements are named. Both the header names and the header values appear in `StringToSign`.

If a positional header called for in the definition of `StringToSign` is not present in your request (for example, `Content-Type` or `Content-MD5` are optional for PUT requests and meaningless for GET requests), substitute the empty string ("") for that position.

**Time stamp requirement**

A valid time stamp (using either the HTTP `Date` header or an `x-amz-date` alternative) is mandatory for authenticated requests. Furthermore, the client timestamp included with an authenticated request must be within 15 minutes of the Amazon S3 system time when the request is received. If not, the request will fail with the `RequestTimeTooSkewed` error code. The intention of these restrictions is to limit the possibility that intercepted requests could be replayed by an adversary. For stronger protection against eavesdropping, use the HTTPS transport for authenticated requests.

> **Note**
>
> The validation constraint on request date applies only to authenticated requests that do not use query string authentication. For more information, see Query string request authentication alternative.

Some HTTP client libraries do not expose the ability to set the `Date` header for a request. If you have trouble including the value of the 'Date' header in the canonicalized headers, you can set the timestamp for the request by using an '`x-amz-date`' header instead. The value of the `x-amz-date` header must be in one of the RFC 2616 formats (http://www.ietf.org/rfc/rfc2616.txt). When an `x-amz-date` header is present in a request, the system will ignore any `Date` header when computing the request signature. Therefore, if you include the `x-amz-date` header, use the empty string for the `Date` when constructing the `StringToSign`. See the next section for an example.

**Authentication examples**

The examples in this section use the (non-working) credentials in the following table.

| Parameter | Value |
| --- | --- |
| AWSAccessKeyId | `AKIAIOSFODNN7EXAMPLE` |
| AWSSecretAccessKey | `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY` |

In the example StringToSigns, formatting is not significant, and \n means the Unicode code point U+000A, commonly called newline. Also, the examples use "+0000" to designate the time zone. You can use "GMT" to designate timezone instead, but the signatures shown in the examples will be different.

### Object GET

This example gets an object from the awsexamplebucket1 bucket.

| Request | StringToSign |
| --- | --- |
| `GET /photos/puppy.jpg HTTP/1.1`<br>`Host: awsexamplebucket1.us-west-1.s3.amazonaws.com`<br>`Date: Tue, 27 Mar 2007 19:36:42 +0000`<br><br>`Authorization: AWS AKIAIOSFODNN7EXAMPLE:qgk2+6Sv9/oM7G3qLEjTH1a1l1g=` | `GET\n`<br>`\n`<br>`\n`<br>`Tue, 27 Mar 2007 19:36:42 +0000\n`<br>`/awsexamplebucket1/photos/puppy.jpg` |

Note that the CanonicalizedResource includes the bucket name, but the HTTP Request-URI does not. (The bucket is specified by the Host header.)

**Note**

The following Python script calculates the preceding signature, using the provided parameters. You can use this script to construct your own signatures, replacing the keys and StringToSign as appropriate.

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
```

```
                                hmac.new(
                                        secret_key, string_to_sign, sha1
                                        ).digest()
                                ).strip()


print(f"AWS {access_key.decode()}:{signature.decode()}")
```

### Object PUT

This example puts an object into the awsexamplebucket1 bucket.

| Request | StringToSign |
| --- | --- |
| `PUT /photos/puppy.jpg HTTP/1.1`<br>`Content-Type: image/jpeg`<br>`Content-Length: 94328`<br>`Host: awsexamplebucket1.s3.us-wes`<br>`t-1.amazonaws.com`<br>`Date: Tue, 27 Mar 2007 21:15:45 +0000`<br><br>*`Authorization: AWS AKIAIOSFODNN7EXAMP`*<br>*`LE:`*<br>*`iqRzw+ileNPu1fhspnRs8nOjjIA=`* | `PUT\n`<br>`\n`<br>`image/jpeg\n`<br>`Tue, 27 Mar 2007 21:15:45 +0000\n`<br>`/awsexamplebucket1/photos/puppy.jpg` |

Note the Content-Type header in the request and in the StringToSign. Also note that the Content-MD5 is left blank in the StringToSign, because it is not present in the request.

### List

This example lists the content of the awsexamplebucket1 bucket.

| Request | StringToSign |
| --- | --- |
| `GET /?prefix=photos&max-keys=50&marker=puppy`<br>` HTTP/1.1`<br>`User-Agent: Mozilla/5.0` | `GET\n`<br>`\n`<br>`\n` |

| Request | StringToSign |
| --- | --- |
| `Host: awsexamplebucket1.s3.us-west-1.amazonaws.com`<br>`Date: Tue, 27 Mar 2007 19:42:41 +0000`<br><br>*`Authorization: AWS AKIAIOSFODNN7EXAMPLE:m0WP8eCtspQl5Ahe6L1SozdX9YA=`* | `Tue, 27 Mar 2007 19:42:41 +0000\n`<br>`/awsexamplebucket1/` |

Note the trailing slash on the CanonicalizedResource and the absence of query string parameters.

**Fetch**

This example fetches the access control policy subresource for the 'awsexamplebucket1' bucket.

| Request | StringToSign |
| --- | --- |
| `GET /?acl HTTP/1.1`<br>`Host: awsexamplebucket1.s3.us-west-1.amazonaws.com`<br>`Date: Tue, 27 Mar 2007 19:44:46 +0000`<br><br>*`Authorization: AWS AKIAIOSFODNN7EXAMPLE:82ZHiFIjc+WbcwFKGUVEQspPn+0=`* | `GET\n`<br>`\n`<br>`\n`<br>`Tue, 27 Mar 2007 19:44:46 +0000\n`<br>`/awsexamplebucket1/?acl` |

Notice how the subresource query string parameter is included in the CanonicalizedResource.

**Delete**

This example deletes an object from the 'awsexamplebucket1' bucket using the path-style and Date alternative.

| Request | StringToSign |
| --- | --- |
| `DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1`<br>`User-Agent: dotnet`<br>`Host: s3.us-west-1.amazonaws.com`<br>`Date: Tue, 27 Mar 2007 21:20:27 +0000` | `DELETE\n`<br>`\n`<br>`\n`<br>`Tue, 27 Mar 2007 21:20:26 +0000\n` |

| Request | StringToSign |
| --- | --- |
| x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000<br>*Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=* | /awsexamplebucket1/photos/puppy.jpg |

Note how we used the alternate 'x-amz-date' method of specifying the date (because our client library prevented us from setting the date, say). In this case, the `x-amz-date` takes precedence over the `Date` header. Therefore, date entry in the signature must contain the value of the `x-amz-date` header.

### Upload

This example uploads an object to a CNAME style virtual hosted bucket with metadata.

| Request | StringToSign |
| --- | --- |
| PUT /db-backup.dat.gz HTTP/1.1<br>User-Agent: curl/7.15.5<br>Host: static.example.com:8080<br>Date: Tue, 27 Mar 2007 21:06:08 +0000<br><br>x-amz-acl: public-read<br>content-type: application/x-download<br>Content-MD5: 4gJE4saaMU4BqNR0kLY+lw==<br>X-Amz-Meta-ReviewedBy: joe@example.com<br>X-Amz-Meta-ReviewedBy: jane@example.com<br>X-Amz-Meta-FileChecksum: 0x02661779<br>X-Amz-Meta-ChecksumAlgorithm: crc32<br>Content-Disposition: attachment; filename=database.dat<br>Content-Encoding: gzip<br>Content-Length: 5913339<br><br>*Authorization: AWS AKIAIOSFODNN7EXAMPLE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=* | PUT\n<br>4gJE4saaMU4BqNR0kLY+lw==\n<br>application/x-download\n<br>Tue, 27 Mar 2007 21:06:08 +0000\n<br><br>x-amz-acl:public-read\n<br>x-amz-meta-checksumalgorithm:crc32\n<br>x-amz-meta-filechecksum:0x02661779\n<br>x-amz-meta-reviewedby:joe@example.com,jane@example.com\n<br>/static.example.com/db-backup.dat.gz |

Notice how the 'x-amz-' headers are sorted, trimmed of extra spaces, and converted to lowercase. Note also that multiple headers with the same name have been joined using commas to separate values.

Note how only the `Content-Type` and `Content-MD5` HTTP entity headers appear in the `StringToSign`. The other `Content-*` entity headers do not.

Again, note that the `CanonicalizedResource` includes the bucket name, but the HTTP Request-URI does not. (The bucket is specified by the Host header.)

**List all my buckets**

| Request | StringToSign |
| --- | --- |
| `GET / HTTP/1.1`<br>`Host: s3.us-west-1.amazonaws.com`<br>`Date: Wed, 28 Mar 2007 01:29:59 +0000`<br><br>`Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=` | `GET\n`<br>`\n`<br>`\n`<br>`Wed, 28 Mar 2007 01:29:59 +0000\n`<br>`/` |

**Unicode keys**

| Request | StringToSign |
| --- | --- |
| `GET /dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re HTTP/1.1`<br>`Host: s3.us-west-1.amazonaws.com`<br>`Date: Wed, 28 Mar 2007 01:49:49 +0000`<br>`Authorization: AWS AKIAIOSFODNN7EXAMPLE:DNEZGsoieTZ92F3bUfSPQcbGmlM=` | `GET\n`<br>`\n`<br>`\n`<br>`Wed, 28 Mar 2007 01:49:49 +0000\n`<br>`/dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re` |

> **Note**
>
> The elements in `StringToSign` that were derived from the Request-URI are taken literally, including URL-Encoding and capitalization.

### REST request signing problems

When REST request authentication fails, the system responds to the request with an XML error document. The information contained in this error document is meant to help developers diagnose the problem. In particular, the `StringToSign` element of the `SignatureDoesNotMatch` error document tells you exactly what request canonicalization the system is using.

Some toolkits silently insert headers that you do not know about beforehand, such as adding the header `Content-Type` during a PUT. In most of these cases, the value of the inserted header remains constant, allowing you to discover the missing headers by using tools such as Ethereal or tcpmon.

### Query string request authentication alternative

You can authenticate certain types of requests by passing the required information as query-string parameters instead of using the `Authorization` HTTP header. This is useful for enabling direct third-party browser access to your private Amazon S3 data without proxying the request. The idea is to construct a "presigned" request and encode it as a URL that an end-user's browser can retrieve. Additionally, you can limit a presigned request by specifying an expiration time.

For more information on using query parameters to authenticate requests , see Authenticating Requests: Using Query Parameters (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*. For examples of using the AWS SDKs to generating presigned URLs, see Sharing objects with presigned URLs.

### Creating a signature

Following is an example query string authenticated Amazon S3 REST request.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa
%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

The query string request authentication method doesn't require any special HTTP headers. Instead, the required authentication elements are specified as query string parameters:

| Query string parameter name | Example value | Description |
|---|---|---|
| AWSAccessKeyId | AKIAIOSFODNN7EXAMPLE | Your AWS access key ID. Specifies the AWS secret access key used to sign the request and, indirectly, the identity of the developer making the request. |
| Expires | 1141889120 | The time when the signature expires, specified as the number of seconds since the epoch (00:00:00 UTC on January 1, 1970). A request received after this time (according to the server) will be rejected. |
| Signature | vjbyPxybdZaNmGa%2ByT272YEAiv4%3D | The URL encoding of the Base64 encoding of the HMAC-SHA1 of StringToSign. |

The query string request authentication method differs slightly from the ordinary method but only in the format of the Signature request parameter and the StringToSign element. Following is pseudo-grammar that illustrates the query string request authentication method.

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-
Of( StringToSign ) ) ) );

StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

YourSecretAccessKey is the AWS secret access key ID that Amazon assigns to you when you sign up to be an Amazon Web Service developer. Notice how the Signature is URL-Encoded to

make it suitable for placement in the query string. Note also that in `StringToSign`, the HTTP `Date` positional element has been replaced with `Expires`. The `CanonicalizedAmzHeaders` and `CanonicalizedResource` are the same.

> **Note**
>
> In the query string authentication method, you do not use the `Date` or the `x-amz-date request` header when calculating the string to sign.

**Query string request authentication**

| Request | StringToSign |
| --- | --- |
| `GET /photos/puppy.jpg?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&`<br>`    Signature=NpgCjnDzrM%2BWFzoENXmpNDUsSn8%3D&`<br>`    Expires=1175139620 HTTP/1.1`<br><br>`Host: awsexamplebucket1.s3.us-west-1.amazonaws.com` | `GET\n`<br>`\n`<br>`\n`<br>`1175139620\n`<br><br>`/awsexamplebucket1/photos/puppy.jpg` |

We assume that when a browser makes the GET request, it won't provide a Content-MD5 or a Content-Type header, nor will it set any x-amz- headers, so those parts of the `StringToSign` are left blank.

**Using Base64 encoding**

HMAC request signatures must be Base64 encoded. Base64 encoding converts the signature into a simple ASCII string that can be attached to the request. Characters that could appear in the signature string like plus (+), forward slash (/), and equals (=) must be encoded if used in a URI. For example, if the authentication code includes a plus (+) sign, encode it as %2B in the request. Encode a forward slash as %2F and equals as %3D.

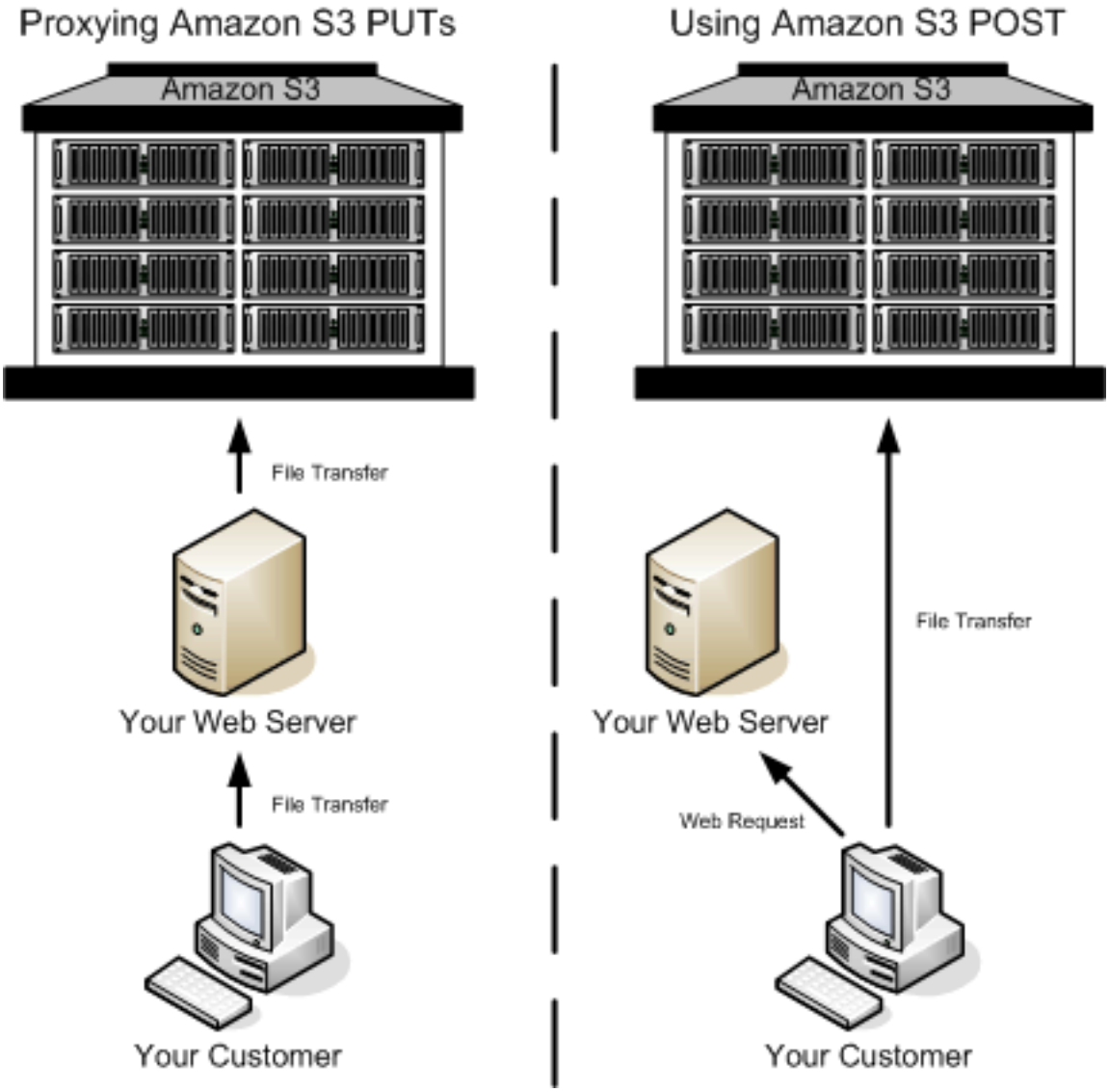
For examples of Base64 encoding, refer to the Amazon S3 Authentication examples.

## Browser-based uploads using POST (AWS signature version 2)

Amazon S3 supports POST, which allows your users to upload content directly to Amazon S3. POST is designed to simplify uploads, reduce upload latency, and save you money on applications where users upload data to store in Amazon S3.

> **Note**
>
> The request authentication discussed in this section is based on AWS Signature Version 2, a protocol for authenticating inbound API requests to AWS services.
> Amazon S3 now supports Signature Version 4, a protocol for authenticating inbound API requests to AWS services, in all AWS Regions. At this time, AWS Regions created before January 30, 2014 will continue to support the previous protocol, Signature Version 2. Any new regions after January 30, 2014 will support only Signature Version 4 and therefore all requests to those regions must be made with Signature Version 4. For more information, see Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

The following figure shows an upload using Amazon S3 POST.




**Uploading using POST**

| | |
|---|---|
| 1 | The user opens a web browser and accesses your web page. |
| 2 | Your web page contains an HTTP form that contains all the information necessary for the user to upload content to Amazon S3. |
| 3 | The user uploads content directly to Amazon S3. |

> **Note**
>
> Query string authentication is not supported for POST.

## HTML forms (AWS signature version 2)

### Topics



When you communicate with Amazon S3, you normally use the REST or SOAP API to perform put, get, delete, and other operations. With POST, users upload data directly to Amazon S3 through their browsers, which cannot process the SOAP API or create a REST PUT request.

> **Note**
>
> SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the AWS SDKs.

To allow users to upload content to Amazon S3 by using their browsers, you use HTML forms. HTML forms consist of a form declaration and form fields. The form declaration contains high-level information about the request. The form fields contain detailed information about the request, as well as the policy that is used to authenticate it and ensure that it meets the conditions that you specify.

> **Note**
>
> The form data and boundaries (excluding the contents of the file) cannot exceed 20 KB.

This section explains how to use HTML forms.

### HTML form encoding

The form and policy must be UTF-8 encoded. You can apply UTF-8 encoding to the form by specifying it in the HTML heading or as a request header.

> **Note**
>
> The HTML form declaration does not accept query string authentication parameters.

The following is an example of UTF-8 encoding in the HTML heading:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

The following is an example of UTF-8 encoding in a request header:

```
Content-Type: text/html; charset=UTF-8
```

### HTML form declaration

The form declaration has three components: the action, the method, and the enclosure type. If any of these values is improperly set, the request fails.

The action specifies the URL that processes the request, which must be set to the URL of the bucket. For example, if the name of your bucket is `awsexamplebucket1` and the Region is US West (N. California), the URL is `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

> **Note**
>
> The key name is specified in a form field.

The method must be POST.

The enclosure type (enctype) must be specified and must be set to multipart/form-data for both file uploads and text area uploads. For more information, go to RFC 1867.

### Example

The following example is a form declaration for the bucket "awsexamplebucket1".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"

enctype="multipart/form-data">
```

### HTML form fields

The following table describes fields that can be used within an HTML form.

> **Note**
>
> The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) will be used. For example, "C:\Program Files\directory1\file.txt" will be interpreted as "file.txt". If no file or file name is provided, the variable is replaced with an empty string.

| Field name | Description | Required |
|---|---|---|
| `AWSAccessKeyId` | The AWS Access Key ID of the owner of the bucket who grants an anonymous user access for a request that satisfies the set of constraints in the policy. This field is required if the request includes a policy document. | Conditional |
| `acl` | An Amazon S3 access control list (ACL). If an invalid access control list is specified, an error is generated. For more information on ACLs, see Access control lists (ACLs). | No |

| Field name | Description | Required |
| --- | --- | --- |
| | Type: String<br><br>Default: private<br><br>Valid Values: `private \| public-read \| public-read-write \| aws-exec-read \| authenticated-read \| bucket-owner-read \| bucket-owner-full-control` | |
| `Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires` | REST-specific headers. For more information, see PUT Object. | No |
| `key` | The name of the uploaded key.<br><br>To use the filename provided by the user, use the ${filename} variable. For example, if user Betty uploads the file lolcatz.jpg and you specify /user/betty/${filename}, the file is stored as /user/betty/lolcatz.jpg.<br><br>For more information, see Working with object metadata. | Yes |
| `policy` | Security policy describing what is permitted in the request. Requests without a security policy are considered anonymous and will succeed only on publicly writable buckets. | No |

| Field name | Description | Required |
| --- | --- | --- |
| `success_action_redirect, redirect` | The URL to which the client is redirected upon successful upload. Amazon S3 appends the bucket, key, and etag values as query string parameters to the URL.<br><br>If success_action_redirect is not specified, Amazon S3 returns the empty document type specified in the success_action_status field.<br><br>If Amazon S3 cannot interpret the URL, it ignores the field.<br><br>If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL.<br><br>For more information, see Redirection.<br><br>**Note**<br>The redirect field name is deprecated and support for the redirect field name will be removed in the future. | No |

| Field name | Description | Required |
| --- | --- | --- |
| `success_action_status` | The status code returned to the client upon successful upload if success_action_redirect is not specified.<br><br>Valid values are 200, 201, or 204 (default).<br><br>If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code.<br><br>If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information about the content of the XML document, see POST Object.<br><br>If the value is not set or if it is set to an invalid value, Amazon S3 returns an empty document with a 204 status code.<br><br>**Note**<br>Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting `success_action_status` to 201. | No |

| Field name | Description | Required |
| --- | --- | --- |
| `signature` | The HMAC signature constructed by using the secret access key that corresponds to the provided AWSAccessKeyId. This field is required if a policy document is included with the request.<br><br>For more information, see Identity and Access Management for Amazon S3. | Conditional |
| `x-amz-security-token` | A security token used by session credentials<br><br>If the request is using Amazon DevPay then it requires two `x-amz-security-token` form fields: one for the product token and one for the user token.<br><br>If the request is using session credentials, then it requires one `x-amz-security-token` form. For more information, see Temporary Security Credentials in the *IAM User Guide*. | No |
| Other field names prefixed with x-amz-meta- | User-specified metadata.<br><br>Amazon S3 does not validate or use this data.<br><br>For more information, see PUT Object. | No |

| Field name | Description | Required |
| --- | --- | --- |
| file | File or text content.<br><br>The file or content must be the last field in the form. Any fields below it are ignored.<br><br>You cannot upload more than one file at a time. | Yes |

### Policy construction

### Topics

- Expiration
- Conditions
- Condition matching
- Character escaping

The policy is a UTF-8 and Base64-encoded JSON document that specifies conditions that the request must meet and is used to authenticate the content. Depending on how you design your policy documents, you can use them per upload, per user, for all uploads, or according to other designs that meet your needs.

> **Note**
>
> Although the policy document is optional, we highly recommend it over making a bucket publicly writable.

The following is an example of a policy document:

```
{ "expiration": "2007-12-01T12:00:00.000Z",

  "conditions": [

    {"acl": "public-read" },
```

```
    {"bucket": "awsexamplebucket1" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

The policy document contains the expiration and conditions.

**Expiration**

The expiration element specifies the expiration date of the policy in ISO 8601 UTC date format. For example, "2007-12-01T12:00:00.000Z" specifies that the policy is not valid after midnight UTC on 2007-12-01. Expiration is required in a policy.

**Conditions**

The conditions in the policy document validate the contents of the uploaded object. Each form field that you specify in the form (except AWSAccessKeyId, signature, file, policy, and field names that have an x-ignore- prefix) must be included in the list of conditions.

> **Note**
>
> If you have multiple fields with the same name, the values must be separated by commas. For example, if you have two fields named "x-amz-meta-tag" and the first one has a value of "Ninja" and second has a value of "Stallman", you would set the policy document to `Ninja,Stallman`.
> All variables within the form are expanded before the policy is validated. Therefore, all condition matching should be performed against the expanded fields. For example, if you set the key field to `user/betty/${filename}`, your policy might be `[ "starts-with", "$key", "user/betty/" ]`. Do not enter `[ "starts-with", "$key", "user/betty/${filename}" ]`. For more information, see Condition matching.

The following table describes policy document conditions.

| Element name | Description |
| --- | --- |
| acl | Specifies conditions that the ACL must meet.<br><br>Supports exact matching and `starts-with` . |
| content-length-range | Specifies the minimum and maximum allowable size for the uploaded content.<br><br>Supports range matching. |
| Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires | REST-specific headers.<br><br>Supports exact matching and `starts-with` . |
| key | The name of the uploaded key.<br><br>Supports exact matching and `starts-with` . |
| success_action_redirect, redirect | The URL to which the client is redirected upon successful upload.<br><br>Supports exact matching and `starts-with` . |
| success_action_status | The status code returned to the client upon successful upload if success_action_redirect is not specified.<br><br>Supports exact matching. |
| x-amz-security-token | Amazon DevPay security token.<br><br>Each request that uses Amazon DevPay requires two `x-amz-security-token` form fields: one for the product token and one for the user token. As a result, the values must be separated by commas. For example, if the user token is `eW91dHViZQ==` and the product |

| Element name | Description |
| --- | --- |
| | token is `b0hnNVNKWVJIQTA=` , you set the policy entry to: `{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }` . |
| Other field names prefixed with x-amz-meta- | User-specified metadata.<br><br>Supports exact matching and `starts-with` . |

**Note**

If your toolkit adds additional fields (e.g., Flash adds filename), you must add them to the policy document. If you can control this functionality, prefix `x-ignore-` to the field so Amazon S3 ignores the feature and it won't affect future versions of this feature.

### Condition matching

The following table describes condition matching types. Although you must specify one condition for each form field that you specify in the form, you can create more complex matching criteria by specifying multiple conditions for a form field.

| Condition | Description |
| --- | --- |
| Exact Matches | Exact matches verify that fields match specific values. This example indicates that the ACL must be set to public-read:<br><br>`{"acl": "public-read" }`<br><br>This example is an alternate way to indicate that the ACL must be set to public-read:<br><br>`[ "eq", "$acl", "public-read" ]` |

| Condition | Description |
| --- | --- |
| Starts With | If the value must start with a certain value, use starts-with. This example indicates that the key must start with user/betty:<br>`["starts-with", "$key", "user/betty/"]` |
| Matching Any Content | To configure the policy to allow any content within a field, use starts-with with an empty value. This example allows any success_action_redirect:<br>`["starts-with", "$success_action_redirect", ""]` |
| Specifying Ranges | For fields that accept ranges, separate the upper and lower ranges with a comma. This example allows a file size from 1 to 10 megabytes:<br>`["content-length-range", 1048579, 10485760]` |

### Character escaping

The following table describes characters that must be escaped within a policy document.

| Escape sequence | Description |
| --- | --- |
| \\ | Backslash |
| \$ | Dollar sign |
| \b | Backspace |
| \f | Form feed |

| Escape sequence | Description |
|---|---|
| \n | New line |
| \r | Carriage return |
| \t | Horizontal tab |
| \v | Vertical tab |
| \u*xxxx* | All Unicode characters |

### Constructing a signature

| Step | Description |
|---|---|
| 1 | Encode the policy by using UTF-8. |
| 2 | Encode those UTF-8 bytes by using Base64. |
| 3 | Sign the policy with your secret access key by using HMAC SHA-1. |
| 4 | Encode the SHA-1 signature by using Base64. |

For general information about authentication, see Identity and Access Management for Amazon S3.

### Redirection

This section describes how to handle redirects.

### General redirection

On completion of the POST request, the user is redirected to the location that you specified in the `success_action_redirect` field. If Amazon S3 cannot interpret the URL, it ignores the `success_action_redirect` field.

If `success_action_redirect` is not specified, Amazon S3 returns the empty document type specified in the `success_action_status` field.

If the POST request fails, Amazon S3 displays an error and does not provide a redirect.

### Pre-upload redirection

If your bucket was created using <CreateBucketConfiguration>, your end users might require a redirect. If this occurs, some browsers might handle the redirect incorrectly. This is relatively rare but is most likely to occur right after a bucket is created.

### Upload examples (AWS signature version 2)

**Topics**

- File upload
- Text area upload

> **Note**
>
> The request authentication discussed in this section is based on AWS Signature Version 2, a protocol for authenticating inbound API requests to AWS services.
> Amazon S3 now supports Signature Version 4, a protocol for authenticating inbound API requests to AWS services, in all AWS Regions. At this time, AWS Regions created before January 30, 2014 will continue to support the previous protocol, Signature Version 2. Any new regions after January 30, 2014 will support only Signature Version 4 and therefore all requests to those regions must be made with Signature Version 4. For more information, see Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

### File upload

This example shows the complete process for constructing a policy and form that can be used to upload a file attachment.

### Policy and form construction

The following policy supports uploads to Amazon S3 for the awsexamplebucket1 bucket.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

This policy requires the following:

- The upload must occur before 12:00 UTC on December 1, 2007.
- The content must be uploaded to the awsexamplebucket1 bucket.
- The key must start with "user/eric/".
- The ACL is set to public-read.
- The success_action_redirect is set to https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html.
- The object is an image file.
- The x-amz-meta-uuid tag must be set to 14365123651274.
- The x-amz-meta-tag can contain any value.

The following is a Base64-encoded version of this policy.

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidW
```

Using your credentials create a signature, for example `0RavWzkygo6QX9caELEqKi9kDbU=` is the signature for the preceding policy document.

The following form supports a POST request to the amzn-s3-demo-bucket bucket that uses this policy.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
    Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    File: <input type="file" name="file" /> <br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

### Sample request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
```

```
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
```

```
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

### Sample response

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

## Text area upload

### Topics

- Policy and form construction
- Sample request
- Sample response

The following example shows the complete process for constructing a policy and form to upload a text area. Uploading a text area is useful for submitting user-created content, such as blog postings.

### Policy and form construction

The following policy supports text area uploads to Amazon S3 for the awsexamplebucket1 bucket.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
```

```
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

This policy requires the following:

- The upload must occur before 12:00 GMT on 2007-12-01.
- The content must be uploaded to the awsexamplebucket1 bucket.
- The key must start with "user/eric/".
- The ACL is set to public-read.
- The success_action_redirect is set to https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html.
- The object is HTML text.
- The x-amz-meta-uuid tag must be set to 14365123651274.
- The x-amz-meta-tag can contain any value.

Following is a Base64-encoded version of this policy.

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

Using your credentials, create a signature. For example, qA7FWXKq6VvU68lI9KdveT1cWgF= is the signature for the preceding policy document.

The following form supports a POST request to the amzn-s3-demo-bucket bucket that uses this policy.

```
<html>
```

```
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

### Sample request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
```

```
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN

--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
```

```
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

### Sample response

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

### POST with adobe flash

This section describes how to use POST with Adobe Flash.

### Adobe flash player security

By default, the Adobe Flash Player security model prohibits Adobe Flash Players from making network connections to servers outside the domain that serves the SWF file.

To override the default, you must upload a publicly readable crossdomain.xml file to the bucket that will accept POST uploads. The following is a sample crossdomain.xml file.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

**Note**

For more information about the Adobe Flash security model, go to the Adobe website.

Adding the crossdomain.xml file to your bucket allows any Adobe Flash Player to connect to the crossdomain.xml file within your bucket; however, it does not grant access to the actual Amazon S3 bucket.

### Adobe flash considerations

The FileReference API in Adobe Flash adds the `Filename` form field to the POST request. When you build Adobe Flash applications that upload to Amazon S3 by using the FileReference API action, include the following condition in your policy:

```
['starts-with', '$Filename', '']
```

Some versions of the Adobe Flash Player do not properly handle HTTP responses that have an empty body. To configure POST to return a response that does not have an empty body, set `success_action_status` to 201. Amazon S3 will then return an XML document with a 201 status code. For information about the content of the XML document, see POST Object. For information about form fields, see HTML form fields.

# Best practices design patterns: optimizing Amazon S3 performance

Your applications can easily achieve thousands of transactions per second in request performance when uploading and retrieving storage from Amazon S3. Amazon S3 automatically scales to high request rates. For example, your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per partitioned Amazon S3 prefix. There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by using parallelization. For example, if you create 10 prefixes in an Amazon S3 bucket to parallelize reads, you could scale your read performance to 55,000 read requests per second. Similarly, you can scale write operations by writing to multiple prefixes. The scaling, in the case of both read and write operations, happens gradually and is not instantaneous. While Amazon S3 is scaling to your new higher request rate, you may see some 503 (Slow Down) errors. These errors will dissipate when the scaling is complete. For more information about creating and using prefixes, see Organizing objects using prefixes.

Some data lake applications on Amazon S3 scan millions or billions of objects for queries that run over petabytes of data. These data lake applications achieve single-instance transfer rates that maximize the network interface use for their Amazon EC2 instance, which can be up to 100 Gb/s on a single instance. These applications then aggregate throughput across multiple instances to get multiple terabits per second.

Other applications are sensitive to latency, such as social media messaging applications. These applications can achieve consistent small object latencies (and first-byte-out latencies for larger objects) of roughly 100–200 milliseconds.

Other AWS services can also help accelerate performance for different application architectures. For example, if you want higher transfer rates over a single HTTP connection or single-digit millisecond latencies, use Amazon CloudFront or Amazon ElastiCache for caching with Amazon S3.

Additionally, if you want fast data transport over long distances between a client and an S3 bucket, use Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration. Transfer Acceleration uses the globally distributed edge locations in CloudFront to accelerate data transport over geographical distances. If your Amazon S3 workload uses server-side encryption with AWS KMS, see AWS KMS Limits in the AWS Key Management Service Developer Guide for information about the request rates supported for your use case.

The following topics describe best practice guidelines and design patterns for optimizing performance for applications that use Amazon S3. Refer to the Performance Guidelines for Amazon S3 and Performance Design Patterns for Amazon S3 for the most current information about performance optimization for Amazon S3.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see What is S3 Express One Zone? and Directory buckets.

**Topics**



# Performance Guidelines for Amazon S3

When building applications that upload and retrieve objects from Amazon S3, follow our best practices guidelines to optimize performance. We also offer more detailed Performance Design Patterns.

To obtain the best performance for your application on Amazon S3, we recommend the following guidelines.

**Topics**

## Measure Performance

When optimizing performance, look at network throughput, CPU, and DRAM requirements. Depending on the mix of demands for these different resources, it might be worth evaluating different Amazon EC2 instance types. For more information about instance types, see Instance Types in the *Amazon EC2 User Guide*.

It's also helpful to look at DNS lookup time, latency, and data transfer speed using HTTP analysis tools when measuring performance.

To understand the performance requirements and optimize the performance of your application, you can also monitor the 503 error responses that you receive. Monitoring certain performance metrics may incur additional expenses. For more information, see Amazon S3 pricing.

### Monitor the number of 503 (Slow Down) status error responses

To monitor the number of 503 status error responses that you get, you can use one of the following options:

- Use Amazon CloudWatch request metrics for Amazon S3. The CloudWatch request metrics include a metric for 5xx status responses. For more information about CloudWatch request metrics, see Monitoring metrics with Amazon CloudWatch.
- Use the 503 (Service Unavailable) error count available in the advanced metrics section of Amazon S3 Storage Lens. For more information, see Using S3 Storage Lens metrics to improve performance.
- Use Amazon S3 server access logging. With server access logging, you can filter and review all requests that receive 503 (Internal Error) responses. You can also use Amazon Athena to parse logs. For more information about server access logging, see Logging requests with server access logging.

By monitoring the number of HTTP 503 status error code, you can often gain valuable insights into which prefixes, keys, or buckets are getting the most throttling requests.

## Scale Storage Connections Horizontally

Spreading requests across many connections is a common design pattern to horizontally scale performance. When you build high performance applications, think of Amazon S3 as a very large distributed system, not as a single network endpoint like a traditional storage server. You can achieve the best performance by issuing multiple concurrent requests to Amazon S3. Spread

these requests over separate connections to maximize the accessible bandwidth from Amazon S3. Amazon S3 doesn't have any limits for the number of connections made to your bucket.

## Use Byte-Range Fetches

Using the Range HTTP header in a GET Object request, you can fetch a byte-range from an object, transferring only the specified portion. You can use concurrent connections to Amazon S3 to fetch different byte ranges from within the same object. This helps you achieve higher aggregate throughput versus a single whole-object request. Fetching smaller ranges of a large object also allows your application to improve retry times when requests are interrupted. For more information, see Downloading objects.

Typical sizes for byte-range requests are 8 MB or 16 MB. If objects are PUT using a multipart upload, it's a good practice to GET them in the same part sizes (or at least aligned to part boundaries) for best performance. GET requests can directly address individual parts; for example, `GET ?partNumber=N.`

## Retry Requests for Latency-Sensitive Applications

Aggressive timeouts and retries help drive consistent latency. Given the large scale of Amazon S3, if the first request is slow, a retried request is likely to take a different path and quickly succeed. The AWS SDKs have configurable timeout and retry values that you can tune to the tolerances of your specific application.

## Combine Amazon S3 (Storage) and Amazon EC2 (Compute) in the Same AWS Region

Although S3 bucket names are globally unique, each bucket is stored in a Region that you select when you create the bucket. To optimize performance, we recommend that you access the bucket from Amazon EC2 instances in the same AWS Region when possible. This helps reduce network latency and data transfer costs.

For more information about data transfer costs, see Amazon S3 Pricing.

## Use Amazon S3 Transfer Acceleration to Minimize Latency Caused by Distance

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration manages fast, easy, and secure transfers of files over long geographic distances between the client and an S3 bucket.

Transfer Acceleration takes advantage of the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, it is routed to Amazon S3 over an optimized network path. Transfer Acceleration is ideal for transferring gigabytes to terabytes of data regularly across continents. It's also useful for clients that upload to a centralized bucket from all over the world.

You can use the Amazon S3 Transfer Acceleration Speed Comparison tool to compare accelerated and non-accelerated upload speeds across Amazon S3 Regions. The Speed Comparison tool uses multipart uploads to transfer a file from your browser to various Amazon S3 Regions with and without using Amazon S3 Transfer Acceleration.

### Use the Latest Version of the AWS SDKs

The AWS SDKs provide built-in support for many of the recommended guidelines for optimizing Amazon S3 performance. The SDKs provide a simpler API for taking advantage of Amazon S3 from within an application and are regularly updated to follow the latest best practices. For example, the SDKs include logic to automatically retry requests on HTTP 503 errors and are investing in code to respond and adapt to slow connections.

The SDKs also provide the Transfer Manager, which automates horizontally scaling connections to achieve thousands of requests per second, using byte-range requests where appropriate. It's important to use the latest version of the AWS SDKs to obtain the latest performance optimization features.

You can also optimize performance when you are using HTTP REST API requests. When using the REST API, you should follow the same best practices that are part of the SDKs. Allow for timeouts and retries on slow requests, and multiple connections to allow fetching of object data in parallel. For information about using the REST API, see the Amazon Simple Storage Service API Reference.

## Performance Design Patterns for Amazon S3

When designing applications to upload and retrieve objects from Amazon S3, use our best practices design patterns for achieving the best performance for your application. We also offer Performance Guidelines for you to consider when planning your application architecture.

To optimize performance, you can use the following design patterns.

**Topics**

- Using Caching for Frequently Accessed Content

- Timeouts and Retries for Latency-Sensitive Applications
- Horizontal Scaling and Request Parallelization for High Throughput
- Using Amazon S3 Transfer Acceleration to Accelerate Geographically Disparate Data Transfers

## Using Caching for Frequently Accessed Content

Many applications that store data in Amazon S3 serve a "working set" of data that is repeatedly requested by users. If a workload is sending repeated GET requests for a common set of objects, you can use a cache such as Amazon CloudFront, Amazon ElastiCache, or AWS Elemental MediaStore to optimize performance. Successful cache adoption can result in low latency and high data transfer rates. Applications that use caching also send fewer direct requests to Amazon S3, which can help reduce request costs.

Amazon CloudFront is a fast content delivery network (CDN) that transparently caches data from Amazon S3 in a large set of geographically distributed points of presence (PoPs). When objects might be accessed from multiple Regions, or over the internet, CloudFront allows data to be cached close to the users that are accessing the objects. This can result in high performance delivery of popular Amazon S3 content. For information about CloudFront, see the Amazon CloudFront Developer Guide.

Amazon ElastiCache is a managed, in-memory cache. With ElastiCache, you can provision Amazon EC2 instances that cache objects in memory. This caching results in orders of magnitude reduction in GET latency and substantial increases in download throughput. To use ElastiCache, you modify application logic to both populate the cache with hot objects and check the cache for hot objects before requesting them from Amazon S3. For examples of using ElastiCache to improve Amazon S3 GET performance, see the blog post Turbocharge Amazon S3 with Amazon ElastiCache for Redis.

AWS Elemental MediaStore is a caching and content distribution system specifically built for video workflows and media delivery from Amazon S3. MediaStore provides end-to-end storage APIs specifically for video, and is recommended for performance-sensitive video workloads. For information about MediaStore, see the AWS Elemental MediaStore User Guide.

## Timeouts and Retries for Latency-Sensitive Applications

There are certain situations where an application receives a response from Amazon S3 indicating that a retry is necessary. Amazon S3 maps bucket and object names to the object data associated with them. If an application generates high request rates (typically sustained rates of over 5,000 requests per second to a small number of objects), it might receive HTTP 503 *slowdown* responses.

If these errors occur, each AWS SDK implements automatic retry logic using exponential backoff. If you are not using an AWS SDK, you should implement retry logic when receiving the HTTP 503 error. For information about back-off techniques, see Error Retries and Exponential Backoff in AWSin the *Amazon Web Services General Reference*.

Amazon S3 automatically scales in response to sustained new request rates, dynamically optimizing performance. While Amazon S3 is internally optimizing for a new request rate, you will receive HTTP 503 request responses temporarily until the optimization completes. After Amazon S3 internally optimizes performance for the new request rate, all requests are generally served without retries.

For latency-sensitive applications, Amazon S3 advises tracking and aggressively retrying slower operations. When you retry a request, we recommend using a new connection to Amazon S3 and performing a fresh DNS lookup.

When you make large variably sized requests (for example, more than 128 MB), we advise tracking the throughput being achieved and retrying the slowest 5 percent of the requests. When you make smaller requests (for example, less than 512 KB), where median latencies are often in the tens of milliseconds range, a good guideline is to retry a GET or PUT operation after 2 seconds. If additional retries are needed, the best practice is to back off. For example, we recommend issuing one retry after 2 seconds and a second retry after an additional 4 seconds.

If your application makes fixed-size requests to Amazon S3, you should expect more consistent response times for each of these requests. In this case, a simple strategy is to identify the slowest 1 percent of requests and to retry them. Even a single retry is frequently effective at reducing latency.

If you are using AWS Key Management Service (AWS KMS) for server-side encryption, see Limits in the *AWS Key Management Service Developer Guide* for information about the request rates that are supported for your use case.

## Horizontal Scaling and Request Parallelization for High Throughput

Amazon S3 is a very large distributed system. To help you take advantage of its scale, we encourage you to horizontally scale parallel requests to the Amazon S3 service endpoints. In addition to distributing the requests within Amazon S3, this type of scaling approach helps distribute the load over multiple paths through the network.

For high-throughput transfers, Amazon S3 advises using applications that use multiple connections to GET or PUT data in parallel. For example, this is supported by Amazon S3 Transfer Manager

in the AWS Java SDK, and most of the other AWS SDKs provide similar constructs. For some applications, you can achieve parallel connections by launching multiple requests concurrently in different application threads, or in different application instances. The best approach to take depends on your application and the structure of the objects that you are accessing.

You can use the AWS SDKs to issue GET and PUT requests directly rather than employing the management of transfers in the AWS SDK. This approach lets you tune your workload more directly, while still benefiting from the SDK's support for retries and its handling of any HTTP 503 responses that might occur. As a general rule, when you download large objects within a Region from Amazon S3 to Amazon EC2, we suggest making concurrent requests for byte ranges of an object at the granularity of 8–16 MB. Make one concurrent request for each 85–90 MB/s of desired network throughput. To saturate a 10 Gb/s network interface card (NIC), you might use about 15 concurrent requests over separate connections. You can scale up the concurrent requests over more connections to saturate faster NICs, such as 25 Gb/s or 100 Gb/s NICs.

Measuring performance is important when you tune the number of requests to issue concurrently. We recommend starting with a single request at a time. Measure the network bandwidth being achieved and the use of other resources that your application uses in processing the data. You can then identify the bottleneck resource (that is, the resource with the highest usage), and hence the number of requests that are likely to be useful. For example, if processing one request at a time leads to a CPU usage of 25 percent, it suggests that up to four concurrent requests can be accommodated. Measurement is essential, and it is worth confirming resource use as the request rate is increased.

If your application issues requests directly to Amazon S3 using the REST API, we recommend using a pool of HTTP connections and re-using each connection for a series of requests. Avoiding per-request connection setup removes the need to perform TCP slow-start and Secure Sockets Layer (SSL) handshakes on each request. For information about using the REST API, see the Amazon Simple Storage Service API Reference.

Finally, it's worth paying attention to DNS and double-checking that requests are being spread over a wide pool of Amazon S3 IP addresses. DNS queries for Amazon S3 cycle through a large list of IP endpoints. But caching resolvers or application code that reuses a single IP address do not benefit from address diversity and the load balancing that follows from it. Network utility tools such as the `netstat` command line tool can show the IP addresses being used for communication with Amazon S3, and we provide guidelines for DNS configurations to use. For more information about these guidelines, see Making requests.

# Using Amazon S3 Transfer Acceleration to Accelerate Geographically Disparate Data Transfers

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration is effective at minimizing or eliminating the latency caused by geographic distance between globally dispersed clients and a regional application using Amazon S3. Transfer Acceleration uses the globally distributed edge locations in CloudFront for data transport. The AWS edge network has points of presence in more than 50 locations. Today, it is used to distribute content through CloudFront and to provide rapid responses to DNS queries made to Amazon Route 53.

The edge network also helps to accelerate data transfers into and out of Amazon S3. It is ideal for applications that transfer data across or between continents, have a fast internet connection, use large objects, or have a lot of content to upload. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path. In general, the farther away you are from an Amazon S3 Region, the higher the speed improvement you can expect from using Transfer Acceleration.

You can set up Transfer Acceleration on new or existing buckets. You can use a separate Amazon S3 Transfer Acceleration endpoint to use the AWS edge locations. The best way to test whether Transfer Acceleration helps client request performance is to use the Amazon S3 Transfer Acceleration Speed Comparison tool. Network configurations and conditions vary from time to time and from location to location. So you are charged only for transfers where Amazon S3 Transfer Acceleration can potentially improve your upload performance. For information about using Transfer Acceleration with different AWS SDKs, see Enabling and using S3 Transfer Acceleration.

# What is Amazon S3 on Outposts?

AWS Outposts is a fully managed service that offers the same AWS infrastructure, AWS services, APIs, and tools to virtually any data center, co-location space, or on-premises facility for a truly consistent hybrid experience. AWS Outposts is ideal for workloads that require low-latency access to on-premises systems, local data processing, data residency, and migration of applications with local system interdependencies. For more information, see What is AWS Outposts? in the *AWS Outposts User Guide*.

With Amazon S3 on Outposts, you can create S3 buckets on your Outposts and easily store and retrieve objects on premises. S3 on Outposts provides a new storage class, OUTPOSTS, which uses the Amazon S3 APIs and is designed to store data durably and redundantly across multiple devices and servers on your Outposts. You communicate with your Outposts bucket by using an access point and endpoint connection over a virtual private cloud (VPC).

You can use the same APIs and features on Outposts buckets as you do on Amazon S3, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.



## How S3 on Outposts works

S3 on Outposts is an object storage service that stores data as objects within buckets on your Outpost. An *object* is a data file and any metadata that describes the file. A *bucket* is a container for objects.

To store your data in S3 on Outposts, you first create a bucket. When you create the bucket, you specify a bucket name and the Outpost that will hold the bucket. To access your S3 on Outposts bucket and perform object operations, you next create and configure an access point. You must also create an endpoint to route requests to your access point.

Access points simplify data access for any AWS service or customer application that stores data in S3. Access points are named network endpoints that are attached to buckets and can be used to perform object operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions and network controls.

You can create and manage your S3 on Outposts buckets, access points, and endpoints by using the AWS Management Console, AWS CLI, AWS SDKs, or REST API. To upload and manage objects in your S3 on Outposts bucket, you can use the AWS CLI, AWS SDKs, or REST API.

## Regions

During AWS Outposts provisioning, you or AWS creates a service link connection that connects your Outpost back to your chosen AWS Region or Outposts home Region for bucket operations and telemetry. An Outpost relies on connectivity to the parent AWS Region. The Outposts rack is not designed for disconnected operations or environments with limited to no connectivity. For more information, see Outpost connectivity to AWS Regions in the *AWS Outposts User Guide*.

## Buckets

A bucket is a container for objects stored in S3 on Outposts. You can store any number of objects in a bucket and can have up to 100 buckets per account per Outpost.

When you create a bucket, you enter a bucket name and choose the Outpost where the bucket will reside. After you create a bucket, you cannot change the bucket name or move the bucket to a different Outpost. Bucket names must follow Amazon S3 bucket naming rules. In S3 on Outposts, bucket names are unique to an Outpost and AWS account. S3 on Outposts buckets require the `outpost-id`, `account-id`, and bucket name to identify them.

The following example shows the Amazon Resource Name (ARN) format for S3 on Outposts buckets. The ARN is comprised of the Region your Outpost is homed to, your Outpost account, the Outpost ID, and the bucket name.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

Every object is contained in a bucket. You must use access points to access any object in an Outposts bucket. When you specify the bucket for object operations, you use the access point ARN or access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the access point ARN format for S3 on Outposts, which includes the `outpost-id`, `account-id`, and access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about buckets, see Working with S3 on Outposts buckets.

## Objects

Objects are the fundamental entities stored in S3 on Outposts. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as `Content-Type`. You can also specify custom metadata at the time that the object is stored. An object is uniquely identified within a bucket by a key (or name).

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

## Keys

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket and object key uniquely identifies each object.

The following example shows the ARN format for S3 on Outposts objects, which includes the AWS Region code for the Region that the Outpost is homed to, AWS account ID, Outpost ID, bucket name, and object key:

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

For more information about object keys, see Working with S3 on Outposts objects.

## S3 Versioning

You can use S3 Versioning on Outposts buckets to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets. S3 Versioning helps you recover from unintended user actions and application failures.

For more information, see Managing S3 Versioning for your S3 on Outposts bucket.

## Version ID

When you enable S3 Versioning in a bucket, S3 on Outposts generates a unique version ID for each object added to the bucket. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as PutObject, the new objects get a unique version ID.

For more information, see Managing S3 Versioning for your S3 on Outposts bucket.

## Storage class and encryption

S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS). The S3 Outposts storage class is available only for objects stored in buckets on AWS Outposts. If you try to use other S3 storage classes with S3 on Outposts, S3 on Outposts returns the `InvalidStorageClass` error.

By default, objects stored in the S3 Outposts (0UTP0STS) storage class are encrypted using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Data encryption in S3 on Outposts.

## Bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size.

Bucket policies use JSON-based IAM policy language that is standard across AWS. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies allow or deny requests based on the elements in the policy. These elements can include the requester, S3 on Outposts actions, resources, and aspects or conditions of the request (for example, the IP address used to make the request). For example, you can create a bucket policy that grants cross-account

permissions to upload objects to an S3 on Outposts bucket while ensuring that the bucket owner has full control of the uploaded objects. For more information, see Examples of Amazon S3 bucket policies.

In your bucket policy, you can use wildcard characters (*) in ARNs and other values to grant permissions to a subset of objects. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

### S3 on Outposts access points

S3 on Outposts access points are named network endpoints with dedicated access policies that describe how data can be accessed using that endpoint. Access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are attached to buckets that you can use to perform S3 object operations, such as `GetObject` and `PutObject`.

When you specify the bucket for object operations, you use the access point ARN or access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

Access points have distinct permissions and network controls that S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket.

For more information, see Accessing your S3 on Outposts buckets and objects.

## Features of S3 on Outposts

### Access management

S3 on Outposts provides features for auditing and managing access to your buckets and objects. By default, S3 on Outposts buckets and the objects in them are private. You have access only to the S3 on Outposts resources that you create.

To grant granular resource permissions that support your specific use case or to audit the permissions of your S3 on Outposts resources, you can use the following features.

- S3 Block Public Access – Block public access to buckets and objects. For buckets on Outposts, Block Public Access is always enabled by default.

- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources, including your S3 on Outposts resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- S3 on Outposts access points – Manage data access for shared datasets in S3 on Outposts. Access points are named network endpoints with dedicated access policies. Access points are attached to buckets and can be used to perform object operations, such as `GetObject` and `PutObject`.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- AWS Resource Access Manager (AWS RAM) – Securely share your S3 on Outposts capacity across AWS accounts, within your organization or organizational units (OUs) in AWS Organizations.

## Storage logging and monitoring

S3 on Outposts provides logging and monitoring tools that you can use to monitor and control how your S3 on Outposts resources are being used. For more information, see Monitoring tools.

- Amazon CloudWatch metrics for S3 on Outposts – Track the operational health of your resources and understand your capacity availability.
- Amazon CloudWatch Events events for S3 on Outposts – Create a rule for any S3 on Outposts API event to receive notifications through all supported CloudWatch Events targets, including Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), and AWS Lambda.
- AWS CloudTrail logs for S3 on Outposts – Record actions taken by a user, a role, or an AWS service in S3 on Outposts. CloudTrail logs provide you with detailed API tracking for S3 bucket-level and object-level operations.

## Strong consistency

S3 on Outposts provides strong read-after-write consistency for PUT and DELETE requests of objects in your S3 on Outposts bucket in all AWS Regions. This behavior applies to both writes of new objects and to PUT requests that overwrite existing objects and to DELETE requests. In addition, S3 on Outposts object tags and object metadata (for example, the HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model.

# Related services

After you load your data into S3 on Outposts, you can use it with other AWS services. The following are the services that you might use most frequently:

- Amazon Elastic Compute Cloud (Amazon EC2) – Provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- Amazon Elastic Block Store (Amazon EBS) on Outposts – Use Amazon EBS local snapshots on Outposts to store snapshots of volumes on an Outpost locally in S3 on Outposts.
- Amazon Relational Database Service (Amazon RDS) on Outposts – Use Amazon RDS local backups to store your Amazon RDS backups locally on your Outpost.
- AWS DataSync – Automate transferring data between your Outposts and AWS Regions, choosing what to transfer, when to transfer, and how much network bandwidth to use. S3 on Outposts is integrated with AWS DataSync. For on-premises applications that require high-throughput local processing, S3 on Outposts provides on-premises object storage to minimize data transfers and buffer from network variations, while providing you the ability to easily transfer data between Outposts and AWS Regions.

# Accessing S3 on Outposts

You can work with S3 on Outposts in any of the following ways:

## AWS Management Console

The console is a web-based user interface for managing S3 on Outposts and AWS resources. If you've signed up for an AWS account, you can access S3 on Outposts by signing into the AWS Management Console and choosing **S3** from the AWS Management Console home page. Then, choose **Outposts buckets** from the left navigation pane.

## AWS Command Line Interface

You can use the AWS command line tools to issue commands or build scripts at your system's command line to perform AWS (including S3) tasks.

The AWS Command Line Interface (AWS CLI) provides commands for a broad set of AWS services. The AWS CLI is supported on Windows, macOS, and Linux. To get started, see the AWS Command Line Interface User Guide. For more information about the commands that you can use with S3 on Outposts, see s3api, s3control, and s3outposts in the *AWS CLI Command Reference*.

### AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and so on). The AWS SDKs provide a convenient way to create programmatic access to S3 on Outposts and AWS. Because S3 on Outposts uses the same SDKs as Amazon S3, S3 on Outposts provides a consistent experience using the same S3 APIs, automation, and tools.

S3 on Outposts is a REST service. You can send requests to S3 on Outposts by using the AWS SDK libraries, which wrap the underlying REST API and simplify your programming tasks. For example, the SDKs take care of tasks such as calculating signatures, cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools to Build on AWS.

## Paying for S3 on Outposts

You can purchase a variety of AWS Outposts rack configurations featuring a combination of Amazon EC2 instance types, Amazon EBS General Purpose solid state drive (SSD) volumes (gp2), and S3 on Outposts. Pricing includes delivery, installation, infrastructure service maintenance, and software patches and upgrades.

For more information, see AWS Outposts rack pricing.

## Next steps

For more information about working with S3 on Outposts, see the following topics:

- Setting up your Outpost
- How is Amazon S3 on Outposts different from Amazon S3?
- Getting started with Amazon S3 on Outposts
- Networking for S3 on Outposts
- Working with S3 on Outposts buckets
- Working with S3 on Outposts objects

- Security in S3 on Outposts
- Managing S3 on Outposts storage
- Developing with Amazon S3 on Outposts

# Setting up your Outpost

To get started with Amazon S3 on Outposts, you will need an Outpost with Amazon S3 capacity deployed at your facility. For information about options for ordering an Outpost and S3 capacity, see AWS Outposts. To check if your Outposts has S3 capacity on it, you can use the ListOutpostsWithS3 API call. For specifications and to see how S3 on Outposts is different than Amazon S3, see How is Amazon S3 on Outposts different from Amazon S3?

For more information, see the following topics.

**Topics**

- Order a new Outpost

## Order a new Outpost

If you need to order a new Outpost with S3 capacity, see AWS Outposts rack pricing to understand the capacity options for Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Amazon S3.

After you select your configuration, follow the steps in Create an Outpost and order Outpost capacity in the *AWS Outposts User Guide.*

# How is Amazon S3 on Outposts different from Amazon S3?

Amazon S3 on Outposts delivers object storage to your on-premises AWS Outposts environment. Using S3 on Outposts helps you to meet local processing, data residency, and demanding performance needs by keeping data close to on-premises applications. Because it uses Amazon S3 APIs and features, S3 on Outposts makes it easy to store, secure, tag, report on, and control access to the data on your Outposts and extend AWS infrastructure to your on-premises facility for a consistent hybrid experience.

For more information about how S3 on Outposts is unique, see the following topics.

**Topics**

## S3 on Outposts specifications

- The maximum Outposts bucket size is 50 TB.
- The maximum number of Outposts buckets is 100 per AWS account.
- Outposts buckets can be accessed only by using access points and endpoints.
- The maximum number of access points per Outposts bucket is 10.
- Access point policies are limited to 20 KB in size.
- The Outpost owner can manage access within your organization in AWS Organizations by using AWS Resource Access Manager. All accounts that need access to the Outpost must be within the same organization as the owner account in AWS Organizations.
- The S3 on Outposts bucket owner account is always the owner of all objects in the bucket.
- Only the S3 on Outposts bucket owner account can perform operations on the bucket.
- Object size limitations are consistent with Amazon S3.
- All objects stored on S3 on Outposts are stored in the OUTPOSTS storage class.
- By default, all objects stored in the OUTPOSTS storage class are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C).
- If there is not enough space to store an object on your Outpost, the API returns an insufficient capacity exception (ICE).

## API operations supported by S3 on Outposts

For a list of API operations supported by S3 on Outposts, see Amazon S3 on Outposts API operations.

## Amazon S3 features not supported by S3 on Outposts

The following Amazon S3 features are currently not supported by Amazon S3 on Outposts. Any attempts to use them are rejected.

- Access control lists (ACLs)
- Cross-origin resource sharing (CORS)
- S3 Batch Operations
- S3 Inventory reports
- Changing the default bucket encryption
- Public buckets
- Multi-factor authentication (MFA) delete
- S3 Lifecycle transitions (aside from object deletion and stopping incomplete multipart uploads)
- S3 Object Lock legal hold
- Object Lock retention
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- S3 Replication Time Control (S3 RTC)
- Amazon CloudWatch request metrics
- Metrics configuration
- Transfer Acceleration
- S3 Event Notifications
- Requester Pays buckets
- S3 Select
- AWS Lambda events
- Server access logging
- HTTP POST requests
- SOAP
- Website access

## S3 on Outposts network requirements

- To route requests to an S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. The following limits apply to endpoints for S3 on Outposts:
  - Each virtual private cloud (VPC) on an Outpost can have one associated endpoint, and you can have up to 100 endpoints per Outpost.

- You can map multiple access points to the same endpoint.
- You can add endpoints only to VPCs with CIDR blocks in the subspaces of the following CIDR ranges:
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- You can create endpoints to an Outpost only from VPCs that have non-overlapping CIDR blocks.
- You can create an endpoint only from within its Outposts subnet.
- The subnet that you use to create an endpoint must contain four IP addresses for S3 on Outposts to use.
- If you specify the customer-owned IP address pool (CoIP pool), it must contain four IP addresses for S3 on Outposts to use.
- You can create only one endpoint per Outpost per VPC.

# Getting started with Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

With Amazon S3 on Outposts, you can use the Amazon S3 APIs and features, such as object storage, access policies, encryption, and tagging, on AWS Outposts as you do on Amazon S3. For information about S3 on Outposts, see What is Amazon S3 on Outposts?

**Topics**

- Setting up IAM with S3 on Outposts
- Getting started by using the AWS Management Console
- Getting started by using the AWS CLI and SDK for Java

# Setting up IAM with S3 on Outposts

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon S3 on Outposts resources. IAM is an AWS service that you can use with no additional charge. By default, users don't have permissions for S3 on Outposts resources and operations. To grant access permissions for S3 on Outposts resources and API operations, you can use IAM to create users, groups, or roles and attach permissions.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Creating a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

In addition to IAM identity-based policies, S3 on Outposts supports both bucket and access point policies. Bucket policies and access point policies are resource-based policies that are attached to the S3 on Outposts resource.

- A bucket policy is attached to the bucket and allows or denies requests to the bucket and the objects in it based on the elements in the policy.
- In contrast, an access point policy is attached to the access point and allows or denies requests to the access point.

The access point policy works with the bucket policy that is attached to the underlying S3 on Outposts bucket. For an application or user to access objects in an S3 on Outposts bucket through

an S3 on Outposts access point, both the access point policy and the bucket policy must permit the request.

Restrictions that you include in an access point policy apply only to requests made through that access point. For example, if an access point is attached to a bucket, you can't use the access point policy to allow or deny requests that are made directly to the bucket. However, restrictions that you apply to a bucket policy can allow or deny requests made directly to the bucket or through the access point.

In an IAM policy or a resource-based policy, you define which S3 on Outposts actions are allowed or denied. S3 on Outposts actions correspond to specific S3 on Outposts API operations. S3 on Outposts actions use the `s3-outposts:` namespace prefix. Requests made to the S3 on Outposts control API in an AWS Region and requests made to the object API endpoints on the Outpost are authenticated by using IAM and authorized against the `s3-outposts:` namespace prefix. To work with S3 on Outposts, configure your IAM users and authorize them against the `s3-outposts:` IAM namespace.

For more information, see Actions, resources, and condition keys for Amazon S3 on Outposts in the *Service Authorization Reference*.

**Note**

- Access control lists (ACLs) are not supported by S3 on Outposts.
- S3 on Outposts defaults to the bucket owner as object owner to help ensure that the owner of a bucket can't be prevented from accessing or deleting objects.
- S3 on Outposts always has S3 Block Public Access enabled to help ensure that objects can never have public access.

For more information about setting up IAM for S3 on Outposts, see the following topics.

**Topics**

- Principals for S3 on Outposts policies
- Resource ARNs for S3 on Outposts
- Example policies for S3 on Outposts
- Permissions for S3 on Outposts endpoints

- Service-linked roles for S3 on Outposts

## Principals for S3 on Outposts policies

When you create a resource-based policy to grant access to your S3 on Outposts bucket, you must use the `Principal` element to specify the person or application that can make a request for an action or operation on that resource. For S3 on Outposts policies, you can use one of the following principals:

- An AWS account
- An IAM user
- An IAM role
- All principals, by specifying a wildcard character (*) in a policy that uses a `Condition` element to limit access to a specific IP range

> **Important**
>
> You can't write a policy for an S3 on Outposts bucket that uses a wildcard character (*) in the `Principal` element unless the policy also includes a `Condition` that limits access to a specific IP address range. This restriction helps ensure that there is no public access to your S3 on Outposts bucket. For an example, see Example policies for S3 on Outposts.

For more information about the `Principal` element, see AWS JSON policy elements: Principal in the *IAM User Guide*.

## Resource ARNs for S3 on Outposts

Amazon Resource Names (ARNs) for S3 on Outposts contain the Outpost ID in addition to the AWS Region that the Outpost is homed to, the AWS account ID, and the resource name. To access and perform actions on your Outposts buckets and objects, you must use one of the ARN formats shown in the following table.

The *`partition`* value in the ARN refers to a group of AWS Regions. Each AWS account is scoped to one partition. The following are the supported partitions:

- `aws` – AWS Regions

- `aws-us-gov` – AWS GovCloud (US) Regions

### S3 on Outposts ARN formats

| Amazon S3 on Outposts ARN | ARN format | Example |
|---|---|---|
| Bucket ARN | arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* | arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-bucket1* |
| Access point ARN | arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name* | arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name* |
| Object ARN | arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* /object/*object_key* | arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-bucket1 /object/myobject* |
| S3 on Outposts access point object ARN (used in policies) | arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoi* | arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspo |

| Amazon S3 on Outposts ARN | ARN format | Example |
|---|---|---|
| | *nt_name* / object/*object_key* | int/ *access-point-name/object/myobject* |
| S3 on Outposts ARN | arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* | arn:*aws*:s3-outposts: *us-west-2* :*123456789012* : outpost/ *op-01ac5d28a6a232904* |

## Example policies for S3 on Outposts

### Example : S3 on Outposts bucket policy with an AWS account principal

The following bucket policy uses an AWS account principal to grant access to an S3 on Outposts bucket. To use this bucket policy, replace the *user input placeholders* with your own information.

```
{
   "Version":"2012-10-17",
   "Id":"ExampleBucketPolicy1",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket"
      }
   ]
}
```

**Example : S3 on Outposts bucket policy with a wildcard principal (*) and condition key to limit access to a specific IP address range**

The following bucket policy uses a wildcard principal (*) with the `aws:SourceIp` condition to limit access to a specific IP address range. To use this bucket policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
    "Id": "ExampleBucketPolicy2",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": { "AWS" : "*" },
            "Action":"s3-outposts:*",
            "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "198.51.100.0/24"
                }
            }
        }
    ]
}
```

## Permissions for S3 on Outposts endpoints

S3 on Outposts requires its own permissions in IAM to manage S3 on Outposts endpoint actions.

**Note**

- For endpoints that use the customer-owned IP address pool (CoIP pool) access type, you also must have permissions to work with IP addresses from your CoIP pool, as described in the following table.

- For shared accounts that access S3 on Outposts by using AWS Resource Access Manager, users in these shared accounts can't create their own endpoints on a shared subnet. If a user in a shared account wants to manage their own endpoints, the shared account must create its own subnet on the Outpost. For more information, see the section called "Sharing S3 on Outposts".

### S3 on Outposts endpoint-related IAM permissions

| Action | IAM permissions |
| --- | --- |
| `CreateEndpoint` | `s3-outposts:CreateEndpoint`<br>`ec2:CreateNetworkInterface`<br>`ec2:DescribeNetworkInterfaces`<br>`ec2:DescribeVpcs`<br>`ec2:DescribeSecurityGroups`<br>`ec2:DescribeSubnets`<br>`ec2:CreateTags`<br>`iam:CreateServiceLinkedRole`<br>For endpoints that are using the on-premises customer-owned IP address pool (CoIP pool) access type, the following additional permissio ns are required:<br>`s3-outposts:CreateEndpoint`<br>`ec2:DescribeCoipPools`<br>`ec2:GetCoipPoolUsage`<br>`ec2:AllocateAddress`<br>`ec2:AssociateAddress` |

| Action | IAM permissions |
| --- | --- |
| | `ec2:DescribeAddresses`<br><br>`ec2:DescribeLocalGatewayRouteTableVpcAssociations` |
| `DeleteEndpoint` | `s3-outposts:DeleteEndpoint`<br><br>`ec2:DeleteNetworkInterface`<br><br>`ec2:DescribeNetworkInterfaces`<br><br>For endpoints that are using the on-premises customer-owned IP address pool (CoIP pool) access type, the following additional permissions are required:<br><br>`s3-outposts:DeleteEndpoint`<br><br>`ec2:DisassociateAddress`<br><br>`ec2:DescribeAddresses`<br><br>`ec2:ReleaseAddress` |
| `ListEndpoints` | `s3-outposts:ListEndpoints` |

**Note**

You can use resource tags in an IAM policy to manage permissions.

## Service-linked roles for S3 on Outposts

S3 on Outposts uses IAM service-linked roles to create some network resources on your behalf. For more information, see Using service-linked roles for Amazon S3 on Outposts.

# Getting started by using the AWS Management Console

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

To get started with S3 on Outposts by using the console, see the following topics. To get started by using the AWS CLI or AWS SDK for Java, see Getting started by using the AWS CLI and SDK for Java.

**Topics**



## Create a bucket, an access point, and an endpoint

The following procedure shows you how to create your first bucket in S3 on Outposts. When you create a bucket using the console, you also create an access point and an endpoint associated with the bucket so that you can immediately begin storing objects in your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose **Create Outposts bucket**.
4. For **Bucket name**, enter a Domain Name System (DNS)-compliant name for your bucket.

   The bucket name must:

   - Be unique within the AWS account, the Outpost, and the AWS Region that the Outpost is homed to.
   - Be 3–63 characters long.

- Not contain uppercase characters.
- Start with a lowercase letter or number.

After you create the bucket, you can't change its name. For information about naming buckets, see Bucket naming rules.

> **Important**
>
> Avoid including sensitive information such as account numbers in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

5. For **Outpost**, choose the Outpost where you want the bucket to reside.
6. Under **Bucket Versioning**, set the S3 Versioning state for your S3 on Outposts bucket to one of the following options:

   - **Disable** (default) – The bucket remains unversioned.
   - **Enable** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID.

   For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.
7. (Optional) Add any **optional tags** that you would like to associate with the Outposts bucket. You can use tags to track criteria for individual projects or groups of projects, or to label your buckets by using cost-allocation tags.

   By default, all objects stored in your Outposts bucket are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C). To change the encryption type, you must use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.
8. In the **Outposts access point settings** section, enter the access point name.

   S3 on Outposts access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are named network endpoints that are attached to Outposts buckets that you can use to perform S3 object operations. For more information, see Access points.

Access point names must be unique within the account for this Region and Outpost, and comply with the Access points restrictions and limitations.

9. Choose the **VPC** for this Amazon S3 on Outposts access point.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

10. (Optional for an existing VPC) Choose an **Endpoint subnet** for your endpoint.

    A subnet is a range of IP addresses in your VPC. If you don't have the subnet that you want, choose **Create subnet**. For more information, see Networking for S3 on Outposts.

11. (Optional for an existing VPC) Choose an **Endpoint security group** for your endpoint.

    A security group acts as a virtual firewall to control inbound and outbound traffic.

12. (Optional for an existing VPC) Choose the **Endpoint access type**:

    - **Private** – To be used with the VPC.
    - **Customer owned IP** – To be used with a customer-owned IP address pool (CoIP pool) from within your on-premises network.

13. (Optional) Specify the **Outpost access point policy**. The console automatically displays the **Amazon Resource Name (ARN)** for the access point, which you can use in the policy.
14. Choose **Create Outposts bucket**.

> **Note**
>
> It can take up to 5 minutes for your Outpost endpoint to be created and your bucket to be ready to use. To configure additional bucket settings, choose **View details**.

## Next steps

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management

Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

After you create an S3 on Outposts bucket, access point, and endpoint, you can use the AWS CLI or SDK for Java to upload an object to your bucket. For more information, see Upload an object to an S3 on Outposts bucket.

## Getting started by using the AWS CLI and SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

To get started with S3 on Outposts, you must create a bucket, an access point, and an endpoint. Then, you can upload objects to your bucket. The following examples show you how to get started with S3 on Outposts by using the AWS CLI and SDK for Java. To get started by using the console, see Getting started by using the AWS Management Console.

**Topics**



### Step 1: Create a bucket

The following AWS CLI and SDK for Java examples show you how to create an S3 on Outposts bucket.

AWS CLI

**Example**

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

SDK for Java

**Example**

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the SDK for Java.

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

## Step 2: Create an access point

To access your Amazon S3 on Outposts bucket, you must create and configure an access point. These examples how you how to create an access point by using the AWS CLI and the SDK for Java.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

AWS CLI

**Example**

The following AWS CLI example creates an access point for an Outposts bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

SDK for Java

**Example**

The following SDK for Java example creates an access point for an Outposts bucket. To use this example, replace the *`user input placeholders`* with your own information.

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();
```

```
}
```

## Step 3: Create an endpoint

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

These examples show you how to create an endpoint by using the AWS CLI and the SDK for Java. For more information about the permissions required to create and manage endpoints, see Permissions for S3 on Outposts endpoints.

AWS CLI

**Example**

The following AWS CLI example creates an endpoint for an Outpost by using the VPC resource access type. The VPC is derived from the subnet. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

The following AWS CLI example creates an endpoint for an Outpost by using the customer-owned IP address pool (CoIP pool) access type. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

SDK for Java

**Example**

The following SDK for Java example creates an endpoint for an Outpost. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type
 is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

### Step 4: Upload an object to an S3 on Outposts bucket

To upload an object, see Upload an object to an S3 on Outposts bucket.

# Networking for S3 on Outposts

You can use Amazon S3 on Outposts to store and retrieve objects on-premises for applications that require local data access, data processing, and data residency. This section describes the networking requirements for accessing S3 on Outposts.

**Topics**

- Choosing your networking access type
- Accessing your S3 on Outposts buckets and objects
- Cross-account elastic network interfaces

## Choosing your networking access type

You can access S3 on Outposts from within a VPC or from your on-premises network. You communicate with your Outpost bucket by using an access point and endpoint connection. This connection keeps traffic between your VPC and your S3 on Outposts buckets within the AWS network. When you create an endpoint, you must specify the endpoint access type as either `Private` (for VPC routing) or `CustomerOwnedIp` (for a customer-owned IP address pool [CoIP pool]).

- `Private` (for VPC routing) – If you don't specify the access type, S3 on Outposts uses `Private` by default. With the `Private` access type, instances in your VPC don't require public IP addresses to communicate with resources in your Outpost. You can work with S3 on Outposts from within a VPC. This type of endpoint is accessible from your on-premises network through direct VPC routing. For more information, see Local gateway route tables in the *AWS Outposts User Guide*.
- `CustomerOwnedIp` (for CoIP pool) – If you don't default to the `Private` access type and choose `CustomerOwnedIp`, you must specify an IP address range. You can use this access type to work with S3 on Outposts from both your on-premises network and within a VPC. When accessing S3 on Outposts within a VPC, your traffic is limited to the bandwidth of the local gateway.

## Accessing your S3 on Outposts buckets and objects

To access your S3 on Outposts buckets and objects, you must have the following:

- An access point for the VPC.
- An endpoint for the same VPC.
- An active connection between your Outpost and your AWS Region. For more information about how to connect your Outpost to a Region, see Outpost connectivity to AWS Regions in the *AWS Outposts User Guide*.

For more information about accessing buckets and objects in S3 on Outposts, see Working with S3 on Outposts buckets and Working with S3 on Outposts objects.

## Cross-account elastic network interfaces

S3 on Outposts endpoints are named resources with Amazon Resource Names (ARNs). When these endpoints are created, AWS Outposts sets up multiple cross-account elastic network interfaces. S3

on Outposts cross-account elastic network interfaces are like other network interfaces with one exception: S3 on Outposts associates the cross-account elastic network interfaces to Amazon EC2 instances.

The S3 on Outposts Domain Name System (DNS) load balances your requests over the cross-account elastic network interface. S3 on Outposts creates the cross-account elastic network interface in your AWS account that is visible from the **Network interfaces** pane of the Amazon EC2 console.

For endpoints that use the CoIP pool access type, S3 on Outposts allocates and associates IP addresses with the cross-account elastic network interface from the configured CoIP pool.

# Working with S3 on Outposts buckets

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You can use the same APIs and features on Outpost buckets as you do on Amazon S3, including access policies, encryption, and tagging. For more information, see What is Amazon S3 on Outposts?

You communicate with your Outpost buckets by using an access point and endpoint connection over a virtual private cloud (VPC). To access your S3 on Outposts buckets and objects, you must have an access point for the VPC and an endpoint for the same VPC. For more information, see Networking for S3 on Outposts.

## Buckets

In S3 on Outposts, bucket names are unique to an Outpost and require the AWS Region code for the Region the Outpost is homed to, AWS account ID, Outpost ID, and the bucket name to identify them.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

For more information, see Resource ARNs for S3 on Outposts.

## Access points

Amazon S3 on Outposts supports virtual private cloud (VPC)-only access points as the only means to access your Outposts buckets.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following example shows the ARN format that you use for S3 on Outposts access points. The access point ARN includes the AWS Region code for the Region the Outpost is homed to, AWS account ID, Outpost ID, and access point name.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

## Endpoints

To route requests to an S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. With S3 on Outposts endpoints, you can privately connect your VPC to your Outpost bucket. S3 on Outposts endpoints are virtual uniform resource identifiers (URIs) of the entry point to your S3 on Outposts bucket. They are horizontally scaled, redundant, and highly available VPC components.

Each virtual private cloud (VPC) on your Outpost can have one associated endpoint, and you can have up to 100 endpoints per Outpost. You must create these endpoints to be able to access your Outpost buckets and perform object operations. Creating these endpoints also enables the API model and behaviors to be the same by allowing the same operations to work in S3 and S3 on Outposts.

## API operations on S3 on Outposts

To manage Outpost bucket API operations, S3 on Outposts hosts a separate endpoint that is distinct from the Amazon S3 endpoint. This endpoint is `s3-outposts.region.amazonaws.com`.

To use Amazon S3 API operations, you must sign the bucket and objects using the correct ARN format. You must pass ARNs to API operations so that Amazon S3 can determine whether the request is for Amazon S3 (`s3-control.region.amazonaws.com`) or for S3 on Outposts (`s3-`

outposts.*region*.amazonaws.com). Based on the ARN format, S3 can then sign and route the request appropriately.

Whenever a request is sent to the Amazon S3 control plane, the SDK extracts the components from the ARN and includes the additional header x-amz-*outpost-id*, with the *outpost-id* value extracted from the ARN. The service name from the ARN is used to sign the request before it is routed to the S3 on Outposts endpoint. This behavior applies to all API operations handled by the s3control client.

The following table lists the extended API operations for Amazon S3 on Outposts and their changes relative to Amazon S3.

| API | S3 on Outposts parameter value |
|---|---|
| CreateBucket | Bucket name as ARN, Outpost ID |
| ListRegionalBuckets | Outpost ID |
| DeleteBucket | Bucket name as ARN |
| DeleteBucketLifecycleConfiguration | Bucket name as ARN |
| GetBucketLifecycleConfiguration | Bucket name as ARN |
| PutBucketLifecycleConfiguration | Bucket name as ARN |
| GetBucketPolicy | Bucket name as ARN |
| PutBucketPolicy | Bucket name as ARN |
| DeleteBucketPolicy | Bucket name as ARN |
| GetBucketTagging | Bucket name as ARN |
| PutBucketTagging | Bucket name as ARN |
| DeleteBucketTagging | Bucket name as ARN |

| API | S3 on Outposts parameter value | |
|---|---|---|
| `CreateAccessPoint` | Access point name as ARN | |
| `DeleteAccessPoint` | Access point name as ARN | |
| `GetAccessPoint` | Access point name as ARN | |
| `GetAccessPoint` | Access point name as ARN | |
| `ListAccessPoints` | Access point name as ARN | |
| `PutAccessPointPolicy` | Access point name as ARN | |
| `GetAccessPointPolicy` | Access point name as ARN | |
| `DeleteAccessPointPolicy` | Access point name as ARN | |

## Creating and managing S3 on Outposts buckets

For more information about creating and managing S3 on Outposts buckets, see the following topics.

**Topics**

- Creating an S3 on Outposts bucket
- Adding tags for S3 on Outposts buckets
- Managing access to an Amazon S3 on Outposts bucket using a bucket policy
- Listing Amazon S3 on Outposts buckets
- Getting an S3 on Outposts bucket by using the AWS CLI and the SDK for Java
- Deleting your Amazon S3 on Outposts bucket
- Working with Amazon S3 on Outposts access points
- Working with Amazon S3 on Outposts endpoints

# Creating an S3 on Outposts bucket

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

**Note**

The AWS account that creates the bucket owns it and is the only one that can commit actions to it. Buckets have configuration properties, such as Outpost, tag, default encryption, and access point settings. The access point settings include the virtual private cloud (VPC), the access point policy for accessing the objects in the bucket, and other metadata. For more information, see S3 on Outposts specifications.

If you want to create a bucket that uses AWS PrivateLink to provide bucket and endpoint management access through *interface VPC endpoints* in your virtual private cloud (VPC), see AWS PrivateLink for S3 on Outposts.

The following examples show you how to create an S3 on Outposts bucket by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose **Create Outposts bucket**.
4. For **Bucket name**, enter a Domain Name System (DNS)-compliant name for your bucket.

   The bucket name must:

- Be unique within the AWS account, the Outpost, and the AWS Region that the Outpost is homed to.
- Be 3–63 characters long.
- Not contain uppercase characters.
- Start with a lowercase letter or number.

After you create the bucket, you can't change its name. For information about naming buckets, see Bucket naming rules.

> **Important**
>
> Avoid including sensitive information such as account numbers in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

5. For **Outpost**, choose the Outpost where you want the bucket to reside.
6. Under **Bucket Versioning**, set the S3 Versioning state for your S3 on Outposts bucket to one of the following options:

   - **Disable** (default) – The bucket remains unversioned.
   - **Enable** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID.

   For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.
7. (Optional) Add any **optional tags** that you would like to associate with the Outposts bucket. You can use tags to track criteria for individual projects or groups of projects, or to label your buckets by using cost-allocation tags.

   By default, all objects stored in your Outposts bucket are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C). To change the encryption type, you must use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.
8. In the **Outposts access point settings** section, enter the access point name.

S3 on Outposts access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are named network endpoints that are attached to Outposts buckets that you can use to perform S3 object operations. For more information, see Access points.

Access point names must be unique within the account for this Region and Outpost, and comply with the Access points restrictions and limitations.

9. Choose the **VPC** for this Amazon S3 on Outposts access point.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

10. (Optional for an existing VPC) Choose an **Endpoint subnet** for your endpoint.

    A subnet is a range of IP addresses in your VPC. If you don't have the subnet that you want, choose **Create subnet**. For more information, see Networking for S3 on Outposts.

11. (Optional for an existing VPC) Choose an **Endpoint security group** for your endpoint.

    A security group acts as a virtual firewall to control inbound and outbound traffic.

12. (Optional for an existing VPC) Choose the **Endpoint access type**:

    - **Private** – To be used with the VPC.
    - **Customer owned IP** – To be used with a customer-owned IP address pool (CoIP pool) from within your on-premises network.

13. (Optional) Specify the **Outpost access point policy**. The console automatically displays the **Amazon Resource Name (ARN)** for the access point, which you can use in the policy.

14. Choose **Create Outposts bucket**.

> **Note**
>
> It can take up to 5 minutes for your Outpost endpoint to be created and your bucket to be ready to use. To configure additional bucket settings, choose **View details**.

### Using the AWS CLI

#### Example

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

### Using the AWS SDK for Java

#### Example

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the SDK for Java.

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

## Adding tags for S3 on Outposts buckets

You can add tags for your Amazon S3 on Outposts buckets to track storage costs and other criteria for individual projects or groups of projects.

**Note**

The AWS account that creates the bucket owns it and is the only one that can change its tags.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose tags you want to edit.
4. Choose the **Properties** tab.
5. Under **Tags**, choose **Edit**.
6. Choose **Add new tag**, and enter the **Key** and optional **Value**.

   Add any tags that you would like to associate with an Outposts bucket to track other criteria for individual projects or groups of projects.
7. Choose **Save changes**.

### Using the AWS CLI

The following AWS CLI example applies a tagging configuration to an S3 on Outposts bucket by using a JSON document in the current folder that specifies tags (*tagging.json*). To use this example, replace each *user input placeholder* with your own information.

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging file://tagging.json

tagging.json

{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
```

```
    ]
}
```

The following AWS CLI example applies a tagging configuration to an S3 on Outposts bucket directly from the command line.

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

For more information about this command, see put-bucket-tagging in the *AWS CLI Reference*.

## Managing access to an Amazon S3 on Outposts bucket using a bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

You can update your bucket policy to manage access to your Amazon S3 on Outposts bucket. For more information, see the following topics.

**Topics**

- Adding or editing a bucket policy for an Amazon S3 on Outposts bucket
- Viewing the bucket policy for your Amazon S3 on Outposts bucket
- Deleting the bucket policy for your Amazon S3 on Outposts bucket
- Bucket policy examples

### Adding or editing a bucket policy for an Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to update your Amazon S3 on Outposts bucket policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS SDK for Java.

### Using the S3 console

#### To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose bucket policy you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, to create or edit new policy, choose **Edit**.

   You can now add or edit the S3 on Outposts bucket policy. For more information, see Setting up IAM with S3 on Outposts.

### Using the AWS CLI

The following AWS CLI example puts a policy on an Outposts bucket.

1. Save the following bucket policy to a JSON file. In this example, the file is named `policy1.json`. Replace the *`user input placeholders`* with your own information.

```
{
   "Version":"2012-10-17",
   "Id":"testBucketPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
      }
   ]
```

```
}
```

2. Submit the JSON file as part of the `put-bucket-policy` CLI command. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control put-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --policy file://policy1.json
```

### Using the AWS SDK for Java

The following SDK for Java example puts a policy on an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void putBucketPolicy(String bucketArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testBucketPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId+ "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + bucketArn + "\"}]}";

    PutBucketPolicyRequest reqPutBucketPolicy = new PutBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withPolicy(policy);

    PutBucketPolicyResult respPutBucketPolicy =
 s3ControlClient.putBucketPolicy(reqPutBucketPolicy);
    System.out.printf("PutBucketPolicy Response: %s%n",
 respPutBucketPolicy.toString());

}
```

## Viewing the bucket policy for your Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to view your Amazon S3 on Outposts bucket policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS SDK for Java.

### Using the S3 console

#### To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose permission you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, you can review your existing bucket policy. For more information, see Setting up IAM with S3 on Outposts.

### Using the AWS CLI

The following AWS CLI example gets a policy for an Outposts bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

### Using the AWS SDK for Java

The following SDK for Java example gets a policy for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void getBucketPolicy(String bucketArn) {

    GetBucketPolicyRequest reqGetBucketPolicy = new GetBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketPolicyResult respGetBucketPolicy =
 s3ControlClient.getBucketPolicy(reqGetBucketPolicy);
    System.out.printf("GetBucketPolicy Response: %s%n",
 respGetBucketPolicy.toString());
```

```
}
```

## Deleting the bucket policy for your Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to view your Amazon S3 on Outposts bucket policy by using the AWS Management Console or AWS Command Line Interface (AWS CLI).

### Using the S3 console

#### To delete a bucket policy

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose permission you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, choose **Delete**.
6. Confirm the deletion.

### Using the AWS CLI

The following example deletes the bucket policy for an S3 on Outposts bucket (`s3-outposts:DeleteBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket
```

## Bucket policy examples

With S3 on Outposts bucket policies, you can secure access to objects in your S3 on Outposts buckets, so that only users with the appropriate permissions can access them. You can even prevent

authenticated users without the appropriate permissions from accessing your S3 on Outposts resources.

This section presents examples of typical use cases for S3 on Outposts bucket policies. To test these policies, replace the *`user input placeholders`* with your own information (such as your bucket name).

To grant or deny permissions to a set of objects, you can use wildcard characters (*) in Amazon Resource Names (ARNs) and other values. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

For more information about AWS Identity and Access Management (IAM) policy language, see Setting up IAM with S3 on Outposts.

> **Note**
>
> When testing s3outposts permissions by using the Amazon S3 console, you must grant additional permissions that the console requires, such as `s3outposts:createendpoint`, `s3outposts:listendpoints`, and so on.

### Additional resources for creating bucket policies

- For a list of the IAM policy actions, resources, and condition keys you can use when creating an S3 on Outposts bucket policy, see Actions, resources, and condition keys for Amazon S3 on Outposts.
- For guidance on creating your S3 on Outposts policy, see Adding or editing a bucket policy for an Amazon S3 on Outposts bucket.

### Topics

- Managing access to an Amazon S3 on Outposts bucket based on specific IP addresses

### Managing access to an Amazon S3 on Outposts bucket based on specific IP addresses

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects

in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

**Restrict access to specific IP addresses**

The following example denies all users from performing any S3 on Outposts operations on objects in the specified buckets unless the request originates from the specified range of IP addresses.

> **Note**
>
> When restricting access to a specific IP address, make sure that you also specify which VPC endpoints, VPC source IP addresses, or external IP addresses can access the S3 on Outposts bucket. Otherwise, you might lose access to the bucket if your policy denies all users from performing any s3outposts operations on objects in your S3 on Outposts bucket without the proper permissions already in place.

This policy's `Condition` statement identifies *`192.0.2.0/24`* as the range of allowed IP version 4 (IPv4) IP addresses.

The `Condition` block uses the `NotIpAddress` condition and the `aws:SourceIp` condition key, which is an AWS wide condition key. The `aws:SourceIp` condition key can only be used for public IP address ranges. For more information about these condition keys, see Actions, resources, and condition keys for S3 on Outposts. The `aws:SourceIp` IPv4 values use standard CIDR notation. For more information, see IAM JSON policy elements reference in the *IAM User Guide*.

> **Warning**
>
> Before using this S3 on Outposts policy, replace the *`192.0.2.0/24`* IP address range in this example with an appropriate value for your use case. Otherwise, you'll lose the ability to access your bucket.

```
{
    "Version": "2012-10-17",
    "Id": "S3OutpostsPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
accesspoint/EXAMPLE-ACCESS-POINT-NAME"
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

### Allow both IPv4 and IPv6 addresses

When you start using IPv6 addresses, we recommend that you update all of your organization's policies with your IPv6 address ranges in addition to your existing IPv4 ranges. Doing this will help to make sure that the policies continue to work as you make the transition to IPv6.

The following S3 on Outposts example bucket policy shows how to mix IPv4 and IPv6 address ranges to cover all of your organization's valid IP addresses. The example policy allows access to the example IP addresses *192.0.2.1* and *2001:DB8:1234:5678::1* and denies access to the addresses *203.0.113.1* and *2001:DB8:1234:5678:ABCD::1*.

The `aws:SourceIp` condition key can only be used for public IP address ranges. The IPv6 values for `aws:SourceIp` must be in standard CIDR format. For IPv6, we support using `::` to represent a range of 0s (for example, `2001:DB8:1234:5678::/64`). For more information, see IP address condition operators in the *IAM User Guide*.

> **Warning**
>
> Replace the IP address ranges in this example with appropriate values for your use case before using this S3 on Outposts policy. Otherwise, you might lose the ability to access your bucket.

```
{
```

```
    "Id": "S3OutpostsPolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET",
                        "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-
ID/bucket/DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
            }
        }
    ]
}
```

## Listing Amazon S3 on Outposts buckets

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through

the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about working with buckets in S3 on Outposts, see Working with S3 on Outposts buckets.

The following examples show you how to return a list of your S3 on Outposts buckets by using the AWS Management Console, AWS CLI, and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Under **Outposts buckets**, review your list of S3 on Outposts buckets.

### Using the AWS CLI

The following AWS CLI example gets a list of buckets in an Outpost. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see list-regional-buckets in the *AWS CLI Reference*.

```
aws s3control list-regional-buckets --account-id 123456789012 --outpost-
id op-01ac5d28a6a232904
```

### Using the AWS SDK for Java

The following SDK for Java example gets a list of buckets in an Outpost. For more information, see ListRegionalBuckets in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void listRegionalBuckets() {

    ListRegionalBucketsRequest reqListBuckets = new ListRegionalBucketsRequest()
            .withAccountId(AccountId)
            .withOutpostId(OutpostId);

    ListRegionalBucketsResult respListBuckets =
 s3ControlClient.listRegionalBuckets(reqListBuckets);
```

```
    System.out.printf("ListRegionalBuckets Response: %s%n",
 respListBuckets.toString());

}
```

## Getting an S3 on Outposts bucket by using the AWS CLI and the SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following examples show you how to get an S3 on Outposts bucket by using the AWS CLI and AWS SDK for Java.

> **Note**
>
> When you're working with Amazon S3 on Outposts through the AWS CLI or AWS SDKs, you provide the access point ARN for the Outpost in place of the bucket name. The access point ARN takes the following form, where *region* is the AWS Region code for the Region that the Outpost is homed to:
> `arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point`
> For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

### Using the AWS CLI

The following S3 on Outposts example gets a bucket by using the AWS CLI. To use this command, replace each *user input placeholder* with your own information. For more information about this command, see get-bucket in the *AWS CLI Reference*.

```
aws s3control get-bucket --account-id 123456789012 --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
```

### Using the AWS SDK for Java

The following S3 on Outposts example gets a bucket by using the SDK for Java. For more information, see GetBucket in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketArn) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketArn)
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());

}
```

## Deleting your Amazon S3 on Outposts bucket

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about working with buckets in S3 on Outposts, see Working with S3 on Outposts buckets.

The AWS account that creates the bucket owns it and is the only one that can delete it.

**Note**

- Outposts buckets must be empty before they can be deleted.

  The Amazon S3 console doesn't support S3 on Outposts object actions. To delete objects in an S3 on Outposts bucket, you must use the REST API, AWS CLI, or AWS SDKs.
- Before you can delete an Outposts bucket, you must delete any Outposts access points for the bucket. For more information, see Deleting an access point.
- You cannot recover a bucket after it has been deleted.

The following examples show you how to delete an S3 on Outposts bucket by using the AWS Management Console and AWS Command Line Interface (AWS CLI).

**Using the S3 console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the bucket that you want to delete, and choose **Delete**.
4. Confirm the deletion.

**Using the AWS CLI**

The following example deletes an S3 on Outposts bucket (`s3-outposts:DeleteBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-bucket --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket
```

## Working with Amazon S3 on Outposts access points

To access your Amazon S3 on Outposts bucket, you must create and configure an access point.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform

Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

> **Note**
>
> The AWS account that creates the Outposts bucket owns it and is the only one that can assign access points to it.

The following sections describe how to create and manage access points for S3 on Outposts buckets.

**Topics**



## Creating an S3 on Outposts access point

To access your Amazon S3 on Outposts bucket, you must create and configure an access point.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following examples show you how to create an S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

> **Note**
>
> The AWS account that creates the Outposts bucket owns it and is the only one that can assign access points to it.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to create an Outposts access point for.
4. Choose the **Outposts access points** tab.
5. In the **Outposts access points** section, choose **Create Outposts access point**.
6. In **Outposts access point settings**, enter a name for the access point, and then choose the virtual private cloud (VPC) for the access point.
7. If you want to add a policy for your access point, enter it in the **Outposts access point policy** section.

   For more information, see Setting up IAM with S3 on Outposts.

### Using the AWS CLI

#### Example

The following AWS CLI example creates an access point for an Outposts bucket. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

### Using the AWS SDK for Java

#### Example

The following SDK for Java example creates an access point for an Outposts bucket. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

## Using a bucket-style alias for your S3 on Outposts bucket access point

With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Every time you create an access point for a bucket, S3 on Outposts automatically generates an access point alias. You can use this access point alias instead of an access point ARN for any data plane operation. For example, you can use an access point alias to perform object-level operations such as PUT, GET, LIST, and more. For a list of these operations, see Amazon S3 API operations for managing objects.

The following examples show an ARN and access point alias for an access point named *my-access-point*.

- **Access point ARN** – `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*my-access-point*
- **Access point alias** – *my-access-po-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10*`--op-s3`

For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

For more information about access point aliases, see the following topics.

**Topics**



### Access point aliases

An access point alias is created within the same namespace as an S3 on Outposts bucket. When you create an access point, S3 on Outposts automatically generates an access point alias that cannot be changed. An access point alias meets all the requirements of a valid S3 on Outposts bucket name and consists of the following parts:

```
access point name prefix-metadata--op-s3
```

> **Note**
>
> The `--op-s3` suffix is reserved for access point aliases, we recommend that you don't use it for bucket or access point names. For more information about S3 on Outposts bucket-naming rules, see Working with S3 on Outposts buckets.

### Finding the access point alias

The following examples show you how to find an access point alias by using the Amazon S3 console and the AWS CLI.

#### Example : Find and copy an access point alias in the Amazon S3 console

After you create an access point in the console, you can get the access point alias from the **Access Point alias** column in the **Access Points** list.

#### To copy an access point alias

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. To copy the access point alias, do one of the following:
   - In the **Access Points** list, select the option button next to the access point name, and then choose **Copy Access Point alias**.

- Choose the access point name. Then, under **Outposts access point overview**, copy the access point alias.

**Example : Create an access point by using the AWS CLI and find the access point alias in the response**

The following AWS CLI example for the `create-access-point` command creates the access point and returns the automatically generated access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --bucket example-outposts-bucket --name example-
outposts-access-point --account-id 123456789012

{
    "AccessPointArn":
    "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
accesspoint/example-outposts-access-point",
    "Alias": "example-outp-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

**Example : Get an access point alias by using the AWS CLI**

The following AWS CLI example for the `get-access-point` command returns information about the specified access point. This information includes the access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-access-point --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --name example-outposts-access-point --account-id 123456789012

{
    "Name": "example-outposts-access-point",
    "Bucket": "example-outposts-bucket",
    "NetworkOrigin": "Vpc",
    "VpcConfiguration": {
        "VpcId": "vpc-01234567890abcdef"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
```

```
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2022-09-18T17:49:15.584000+00:00",
    "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

**Example : List access points to find an access point alias by using the AWS CLI**

The following AWS CLI example for the `list-access-points` command lists information about the specified access point. This information includes the access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket

{
    "AccessPointList": [
        {
            "Name": "example-outposts-access-point",
            "NetworkOrigin": "Vpc",
            "VpcConfiguration": {
                "VpcId": "vpc-01234567890abcdef"
            },
            "Bucket": "example-outposts-bucket",
            "AccessPointArn": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point",
            "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
        }
    ]
}
```

**Using an access point alias in an S3 on Outposts object operation**

When adopting access points, you can use access point alias without requiring extensive code changes.

This AWS CLI example shows a `get-object` operation for an S3 on Outposts bucket. This example uses the access point alias as the value for `--bucket` instead of the full access point ARN.

```
aws s3api get-object --bucket my-access-po-
o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3 --key  testkey  sample-object.rtf
```

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

#### Limitations

- Aliases cannot be configured by customers.
- Aliases cannot be deleted or modified or disabled on an access point.
- You can't use an access point alias for S3 on Outposts control plane operations. For a list of S3 on Outposts control plane operations, see Amazon S3 Control API operations for managing buckets.
- Aliases cannot be used in AWS Identity and Access Management (IAM) policies.

## Viewing information about an access point configuration

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following topics show you how to return configuration information for an S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

#### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. Choose the Outposts access point that you want to view configuration details for.
4. Under **Outposts access point overview**, review the access point configuration details.

### Using the AWS CLI

The following AWS CLI example gets an access point for an Outposts bucket. Replace the *user input placeholders* with your own information.

```
aws s3control get-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

### Using the AWS SDK for Java

The following SDK for Java example gets an access point for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPoint(String accessPointArn) {

    GetAccessPointRequest reqGetAP = new GetAccessPointRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointResult respGetAP = s3ControlClient.getAccessPoint(reqGetAP);
    System.out.printf("GetAccessPoint Response: %s%n", respGetAP.toString());

}
```

## View a list of your Amazon S3 on Outposts access points

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following topics show you how to return a list of your S3 on Outposts access points by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Outposts access points**.
3. Under **Outposts access points**, review your list of S3 on Outposts access points.

### Using the AWS CLI

The following AWS CLI example lists the access points for an Outposts bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

### Using the AWS SDK for Java

The following SDK for Java example lists the access points for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void listAccessPoints(String bucketArn) {

    ListAccessPointsRequest reqListAPs = new ListAccessPointsRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    ListAccessPointsResult respListAPs = s3ControlClient.listAccessPoints(reqListAPs);
    System.out.printf("ListAccessPoints Response: %s%n", respListAPs.toString());

}
```

## Deleting an access point

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following examples show you how to delete an access point by using the AWS Management Console and the AWS Command Line Interface (AWS CLI).

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. In the **Outposts access points** section, choose the Outposts access point that you want to delete.
4. Choose **Delete**.
5. Confirm the deletion.

### Using the AWS CLI

The following AWS CLI example deletes an Outposts access point. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-access-point --account-id 123456789012 --name arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point
```

## Adding or editing an access point policy

Access points have distinct permissions and network controls that Amazon S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Access points.

The following topics show you how to add or edit the access point policy for your S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to edit the access point policy for.
4. Choose the **Outposts access points** tab.
5. In the **Outposts access points** section, choose the access point whose policy you want to edit, and choose **Edit policy**.

6. Add or edit the policy in the **Outposts access point policy** section. For more information, see Setting up IAM with S3 on Outposts.

### Using the AWS CLI

The following AWS CLI example puts a policy on an Outposts access point.

1. Save the following access point policy to a JSON file. In this example, the file is named `appolicy1.json`. Replace the *`user input placeholders`* with your own information.

```
{
   "Version":"2012-10-17",
   "Id":"exampleAccessPointPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point
      }
   ]
}
```

2. Submit the JSON file as part of the `put-access-point-policy` CLI command. Replace the *`user input placeholders`* with your own information.

```
aws s3control put-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --policy file://appolicy1.json
```

### Using the AWS SDK for Java

The following SDK for Java example puts a policy on an Outposts access point.

```
import com.amazonaws.services.s3control.model.*;
```

```
public void putAccessPointPolicy(String accessPointArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testAccessPointPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId + "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + accessPointArn +
 "\"}]}";

    PutAccessPointPolicyRequest reqPutAccessPointPolicy = new
 PutAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn)
            .withPolicy(policy);

    PutAccessPointPolicyResult respPutAccessPointPolicy =
 s3ControlClient.putAccessPointPolicy(reqPutAccessPointPolicy);
    System.out.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
    printWriter.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());

}
```

## Viewing an access point policy for an S3 on Outposts access point

Access points have distinct permissions and network controls that Amazon S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Access points.

For more information about working with access points in S3 on Outposts, see Working with S3 on Outposts buckets.

The following topics show you how to view your S3 on Outposts access point policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. Choose the Outposts access point that you want to view the policy for.
4. On the **Permissions** tab, review the S3 on Outposts access point policy.

5. To edit the access point policy, see Adding or editing an access point policy.

### Using the AWS CLI

The following AWS CLI example gets a policy for an Outposts access point. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control get-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

### Using the AWS SDK for Java

The following SDK for Java example gets a policy for an Outposts access point.

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPointPolicy(String accessPointArn) {

    GetAccessPointPolicyRequest reqGetAccessPointPolicy = new
 GetAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointPolicyResult respGetAccessPointPolicy =
 s3ControlClient.getAccessPointPolicy(reqGetAccessPointPolicy);
    System.out.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());
    printWriter.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());

}
```

## Working with Amazon S3 on Outposts endpoints

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

After you create an endpoint, you can use the 'Status' field, to understand the state of the endpoint. If your Outposts is offline, it will return a CREATE_FAILED. You can check your service link connection, delete the endpoint, and retry the create operation after your connection has resumed. For a list of additional error codes, see below. For more information, see Endpoints.

| API | Status | Failed Reason Error Code | Message - Failed Reason |
|---|---|---|---|
| `CreateEndpoint` | Create_Failed | OutpostNotReachable | Endpoint could not be created as the service link connection to your Outposts home Region is down. Check your connection, delete the endpoint, and try again. |
| `CreateEndpoint` | Create_Failed | InternalError | Endpoint could not be created due to Internal Error. Please delete the endpoint and create again. |
| `DeleteEndpoint` | Delete_Failed | OutpostNotReachable | Endpoint could not be deleted as the service link connection to your Outposts home Region is down. Check your connection and please try again. |
| `DeleteEndpoint` | Delete_Failed | InternalError | Endpoint could not be deleted due to Internal Error. Please try again. |

For more information about working with buckets on S3 on Outposts, see Working with S3 on Outposts buckets.

The following sections describe how to create and manage endpoints for S3 on Outposts.

**Topics**

- Creating an endpoint on an Outpost
- Viewing a list of your Amazon S3 on Outposts endpoints
- Deleting an Amazon S3 on Outposts endpoint

## Creating an endpoint on an Outpost

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

### Permissions

For more information about the permissions that are required to create an endpoint, see Permissions for S3 on Outposts endpoints.

When you create an endpoint, S3 on Outposts also creates a service-linked role in your AWS account. For more information, see Using service-linked roles for Amazon S3 on Outposts.

The following examples show you how to create an S3 on Outposts endpoint by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. Choose the **Outposts endpoints** tab.
4. Choose **Create Outposts endpoint**.
5. Under **Outpost**, choose the Outpost to create this endpoint on.
6. Under **VPC**, choose a VPC that does not yet have an endpoint and that also complies with the rules for Outposts endpoints.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud.
7. Choose **Create Outposts endpoint**.

### Using the AWS CLI

#### Example

The following AWS CLI example creates an endpoint for an Outpost by using the VPC resource access type. The VPC is derived from the subnet. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

The following AWS CLI example creates an endpoint for an Outpost by using the customer-owned IP address pool (CoIP pool) access type. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

### Using the AWS SDK for Java

#### Example

The following SDK for Java example creates an endpoint for an Outpost. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type is
```

```
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

## Viewing a list of your Amazon S3 on Outposts endpoints

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

The following examples show you how to return a list of your S3 on Outposts endpoints by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. On the **Outposts access points** page, choose the **Outposts endpoints** tab.
4. Under **Outposts endpoints**, you can view a list of your S3 on Outposts endpoints.

### Using the AWS CLI

The following AWS CLI example lists the endpoints for the AWS Outposts resources that are associated with your account. For more information about this command, see list-endpoints in the *AWS CLI Reference*.

```
aws s3outposts list-endpoints
```

### Using the AWS SDK for Java

The following SDK for Java example lists the endpoints for an Outpost. For more information, see ListEndpoints in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
```

```
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.ListEndpointsRequest;
import com.amazonaws.services.s3outposts.model.ListEndpointsResult;

public void listEndpoints() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    ListEndpointsRequest listEndpointsRequest = new ListEndpointsRequest();
    ListEndpointsResult listEndpointsResult =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
    System.out.println("List endpoints result is " + listEndpointsResult);
}
```

## Deleting an Amazon S3 on Outposts endpoint

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

The following examples show you how to delete your S3 on Outposts endpoints by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

### Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. On the **Outposts access points** page, choose the **Outposts endpoints** tab.
4. Under **Outposts endpoints**, choose the endpoint that you want to delete, and choose **Delete**.

### Using the AWS CLI

The following AWS CLI example deletes an endpoint for an Outpost. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3outposts delete-endpoint --endpoint-id example-endpoint-id --outpost-
id op-01ac5d28a6a232904
```

**Using the AWS SDK for Java**

The following SDK for Java example deletes an endpoint for an Outpost. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.arn.Arn;
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.DeleteEndpointRequest;

public void deleteEndpoint(String endpointArnInput) {
    String outpostId = "op-01ac5d28a6a232904";
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    Arn endpointArn = Arn.fromString(endpointArnInput);
    String[] resourceParts = endpointArn.getResource().getResource().split("/");
    String endpointId = resourceParts[resourceParts.length - 1];
    DeleteEndpointRequest deleteEndpointRequest = new DeleteEndpointRequest()
                .withEndpointId(endpointId)
                .withOutpostId(outpostId);
    s3OutpostsClient.deleteEndpoint(deleteEndpointRequest);
    System.out.println("Endpoint with id " + endpointId + " is deleted.");
}
```

# Working with S3 on Outposts objects

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the

access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

Object ARNs use the following format, which includes the AWS Region that the Outpost is homed to, AWS account ID, Outpost ID, bucket name, and object key:

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

**Topics**

- Upload an object to an S3 on Outposts bucket
- Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java
- Getting an object from an Amazon S3 on Outposts bucket
- Listing the objects in an Amazon S3 on Outposts bucket
- Deleting objects in Amazon S3 on Outposts buckets
- Using HeadBucket to determine if an S3 on Outposts bucket exists and you have access permissions
- Performing and managing a multipart upload with the SDK for Java
- Using presigned URLs for S3 on Outposts
- Amazon S3 on Outposts with local Amazon EMR on Outposts
- Authorization and authentication caching

# Upload an object to an S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following AWS CLI and AWS SDK for Java examples show you how to upload an object to an S3 on Outposts bucket by using an access point.

AWS CLI

**Example**

The following example puts an object named `sample-object.xml` into an S3 on Outposts bucket (`s3-outposts:PutObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see put-object in the *AWS CLI Reference*.

```
aws s3api put-object --bucket arn:aws:s3-
outposts:Region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml --body sample-object.xml
```

SDK for Java

### Example

The following example puts an object into an S3 on Outposts bucket by using the SDK for Java. To use this example, replace each *user input placeholder* with your own information. For more information, see Uploading objects.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

public class PutObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(accessPointArn, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(accessPointArn,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
```

```
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following example shows you how to copy an object in an S3 on Outposts bucket by using the AWS SDK for Java.

## Using the AWS SDK for Java

The following S3 on Outposts example copies an object into a new object in the same bucket by using the SDK for Java. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(accessPointArn,
 sourceKey, accessPointArn, destinationKey);
            s3Client.copyObject(copyObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

# Getting an object from an Amazon S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to download (get) an object by using the AWS Command Line Interface (AWS CLI) and AWS SDK for Java.

### Using the AWS CLI

The following example gets an object named `sample-object.xml` from an S3 on Outposts bucket (`s3-outposts:GetObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see get-object in the *AWS CLI Reference*.

```
aws s3api get-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point --key testkey sample-object.xml
```

### Using the AWS SDK for Java

The following S3 on Outposts example gets an object by using the SDK for Java. To use this example, replace each *user input placeholder* with your own information. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String accessPointArn = "*** access point ARN ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(accessPointArn, key));
            System.out.println("Content-Type: " +
 fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
```

```
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(accessPointArn,
key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
print the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
GetObjectRequest(accessPointArn, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
```

```
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

## Listing the objects in an Amazon S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

> **Note**
>
> With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to list the objects in an S3 on Outposts bucket using the AWS CLI and AWS SDK for Java.

### Using the AWS CLI

The following example lists the objects in an S3 on Outposts bucket (`s3-outposts:ListObjectsV2`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see list-objects-v2 in the *AWS CLI Reference*.

```
aws s3api list-objects-v2 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

**Note**

When using this action with Amazon S3 on Outposts through the AWS SDKs, you provide the Outposts access point ARN in place of the bucket name, in the following form: `arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-Outposts-Access-Point`. For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

### Using the AWS SDK for Java

The following S3 on Outposts example lists objects in a bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

**Important**

This example uses ListObjectsV2, which is the latest revision of the `ListObjects` API operation. We recommend that you use this revised API operation for application development. For backward compatibility, Amazon S3 continues to support the prior version of this API operation.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
```

```
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(accessPointArn).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

## Deleting objects in Amazon S3 on Outposts buckets

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to delete a single object or multiple objects in an S3 on Outposts bucket by using the AWS Command Line Interface (AWS CLI) and AWS SDK for Java.

### Using the AWS CLI

The following examples show you how to delete a single object or multiple objects from an S3 on Outposts bucket.

delete-object

The following example deletes an object named `sample-object.xml` from an S3 on Outposts bucket (`s3-outposts:DeleteObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see delete-object in the *AWS CLI Reference*.

```
aws s3api delete-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml
```

delete-objects

The following example deletes two objects named `sample-object.xml` and `test1.text` from an S3 on Outposts bucket (`s3-outposts:DeleteObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see delete-objects in the *AWS CLI Reference*.

```
aws s3api delete-objects --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --delete file://delete.json

delete.json
{
  "Objects": [
    {
      "Key": "test1.txt"
    },
    {
      "Key": "sample-object.xml"
    }
  ],
  "Quiet": false
}
```

### Using the AWS SDK for Java

The following examples show you how to delete a single object or multiple objects from an S3 on Outposts bucket.

### DeleteObject

The following S3 on Outposts example deletes an object in a bucket by using the SDK for Java. To use this example, specify the access point ARN for the Outpost and the key name for the object that you want to delete. For more information, see DeleteObject in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** key name ****";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(accessPointArn, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

### DeleteObjects

The following S3 on Outposts example uploads and then deletes objects in a bucket by using the SDK for Java. To use this example, specify the access point ARN for the Outpost. For more information, see DeleteObjects in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

import java.util.ArrayList;

public class DeleteObjects {

    public static void main(String[] args) {
       String accessPointArn = "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point";


        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(accessPointArn, keyName, "Object number " + i + "
 to be deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");

            // Delete the sample objects.
```

```
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest(accessPointArn)
                    .withKeys(keys)
                    .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
 s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully
 deleted.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Using HeadBucket to determine if an S3 on Outposts bucket exists and you have access permissions

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

**Note**

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following AWS Command Line Interface (AWS CLI) and AWS SDK for Java examples show you how to use the HeadBucket API operation to determine if an Amazon S3 on Outposts bucket exists and whether you have permission to access it. For more information, see HeadBucket in the *Amazon Simple Storage Service API Reference*.

### Using the AWS CLI

The following S3 on Outposts AWS CLI example uses the `head-bucket` command to determine if a bucket exists and you have permissions to access it. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see head-bucket in the *AWS CLI Reference*.

```
aws s3api head-bucket --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

### Using the AWS SDK for Java

The following S3 on Outposts example shows how to determine if a bucket exists and if you have permission to access it. To use this example, specify the access point ARN for the Outpost. For more information, see HeadBucket in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.HeadBucketRequest;

public class HeadBucket {
```

```
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.headBucket(new HeadBucketRequest(accessPointArn));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Performing and managing a multipart upload with the SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following examples show how you can use S3 on Outposts with the AWS SDK for Java to perform and manage a multipart upload.

### Topics

- Perform a multipart upload of an object in an S3 on Outposts bucket
- Copy a large object in an S3 on Outposts bucket by using multipart upload
- List parts of an object in an S3 on Outposts bucket
- Retrieve a list of in-progress multipart uploads in an S3 on Outposts bucket

## Perform a multipart upload of an object in an S3 on Outposts bucket

The following S3 on Outposts example initiates, uploads, and finishes a multipart upload of an object to a bucket by using the SDK for Java. To use this example, replace each *user input placeholder* with your own information. For more information, see Uploading an object using multipart upload.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();
```

```
            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
```

## Copy a large object in an S3 on Outposts bucket by using multipart upload

The following S3 on Outposts example uses the SDK for Java to copy an object in a bucket. To use this example, replace each *user input placeholder* with your own information. This example is adapted from Copying an object using multipart upload.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
```

```
            InitiateMultipartUploadResult initResult =
s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

## List parts of an object in an S3 on Outposts bucket

The following S3 on Outposts example lists the parts of an object in a bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.List;

public class ListParts {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** Key name ***";
        String uploadId = "*** Upload ID ***";

        try {
            // This code expects that you have AWS credentials set up per:
```

```
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            ListPartsRequest listPartsRequest = new ListPartsRequest(accessPointArn,
 keyName, uploadId);
            PartListing partListing = s3Client.listParts(listPartsRequest);
            List<PartSummary> partSummaries = partListing.getParts();

            System.out.println(partSummaries.size() + " multipart upload parts");
            for (PartSummary p : partSummaries) {
                System.out.println("Upload part: Part number = \"" + p.getPartNumber()
 + "\", ETag = " + p.getETag());
            }

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

## Retrieve a list of in-progress multipart uploads in an S3 on Outposts bucket

The following S3 on Outposts example shows how to retrieve a list of the in-progress multipart uploads from an Outposts bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information. This example is adapted from the Listing multipart uploads example for Amazon S3.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;
```

```
import java.util.List;

public class ListMultipartUploads {
    public static void main(String[] args) {
                String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(accessPointArn);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads =
 multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\",
 id = " + u.getUploadId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

# Using presigned URLs for S3 on Outposts

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

## Limiting presigned URL capabilities

The capabilities of a presigned URL are limited by the permissions of the user who created it. In essence, presigned URLs are bearer tokens that grant access to those who possess them. As such, we recommend that you protect them appropriately.

### AWS Signature Version 4 (SigV4)

To enforce specific behavior when presigned URL requests are authenticated by using AWS Signature Version 4 (SigV4), you can use condition keys in bucket policies and access point policies. For example, you can create a bucket policy that uses the `s3-outposts:signatureAge` condition to deny any Amazon S3 on Outposts presigned URL request on objects in the `example-outpost-bucket` bucket if the signature is more than 10 minutes old. To use this example, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
```

```
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

For a list of condition keys and additional example policies that you can use to enforce specific behavior when presigned URL requests are authenticated by using Signature Version 4, see AWS Signature Version 4 (SigV4) authentication-specific policy keys.

**Network path restriction**

If you want to restrict the use of presigned URLs and all S3 on Outposts access to particular network paths, you can write policies that require a particular network path. To set the restriction on the IAM principal that makes the call, you can use identity-based AWS Identity and Access Management (IAM) policies (for example, user, group, or role policies). To set the restriction on the S3 on Outposts resource, you can use resource-based policies (for example, bucket and access point policies).

A network-path restriction on the IAM principal requires the user of those credentials to make requests from the specified network. A restriction on the bucket or access point requires that all requests to that resource originate from the specified network. These restrictions also apply outside of the presigned URL scenario.

The IAM global condition that you use depends on the type of endpoint. If you are using the public endpoint for S3 on Outposts, use `aws:SourceIp`. If you are using a VPC endpoint for S3 on Outposts, use `aws:SourceVpc` or `aws:SourceVpce`.

The following IAM policy statement requires the principal to access AWS only from the specified network range. With this policy statement, all access must originate from that range. This includes the case of someone who's using a presigned URL for S3 on Outposts. To use this example, replace the *`user input placeholders`* with your own information.

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
```

```
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

For an example bucket policy that uses the `aws:SourceIP` AWS global condition key to restrict access to an S3 on Outposts bucket to a specific network range, see Setting up IAM with S3 on Outposts.

## Who can create a presigned URL

Anyone with valid security credentials can create a presigned URL. But for a user in the VPC to successfully access an object, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon.

You can use the following credentials to create a presigned URL:

- **IAM instance profile** – Valid up to 6 hours.
- **AWS Security Token Service** – Valid up to 36 hours when signed with permanent credentials, such as the credentials of the AWS account root user or an IAM user.
- **IAM user** – Valid up to 7 days when you're using AWS Signature Version 4.

  To create a presigned URL that's valid for up to 7 days, first delegate IAM user credentials (the access key and secret key) to the SDK that you're using. Then, generate a presigned URL by using AWS Signature Version 4.

**Note**

- If you created a presigned URL by using a temporary token, the URL expires when the token expires, even if you created the URL with a later expiration time.
- Because presigned URLs grant access to your S3 on Outposts buckets to whoever has the URL, we recommend that you protect them appropriately. For more information about protecting presigned URLs, see Limiting presigned URL capabilities.

## When does S3 on Outposts check the expiration date and time of a presigned URL?

At the time of the HTTP request, S3 on Outposts checks the expiration date and time of a signed URL. For example, if a client begins to download a large file immediately before the expiration time, the download continues even if the expiration time passes during the download. However, if the connection drops and the client tries to restart the download after the expiration time passes, the download fails.

For more information about using a presigned URL to share or upload objects, see the following topics.

**Topics**

- Sharing objects by using presigned URLs
- Generating a presigned URL to upload an object to an S3 on Outposts bucket

## Sharing objects by using presigned URLs

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An access point Amazon Resource Name (ARN) for the Amazon S3 on Outposts bucket
- An object key
- An HTTP method (GET for downloading objects)
- An expiration date and time

A presigned URL is valid only for the specified duration. That is, you must start the action that's allowed by the URL before the expiration date and time. You can use a presigned URL multiple times, up to the expiration date and time. If you created a presigned URL by using a temporary token, then the URL expires when the token expires, even if you created the URL with a later expiration time.

Users in the virtual private cloud (VPC) who have access to the presigned URL can access the object. For example, if you have a video in your bucket and both the bucket and the object are private, you can share the video with others by generating a presigned URL. Because presigned URLs grant access to your S3 on Outposts buckets to whoever has the URL, we recommend that you protect these URLs appropriately. For more details about protecting presigned URLs, see Limiting presigned URL capabilities.

Anyone with valid security credentials can create a presigned URL. However, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon. For more information, see Who can create a presigned URL.

You can generate a presigned URL to share an object in an S3 on Outposts bucket by using the AWS SDKs and the AWS CLI. For more information, see the following examples.

**Using the AWS SDKs**

You can use the AWS SDKs to generate a presigned URL that you can give to others so that they can retrieve an object.

> **Note**
>
> When you use the AWS SDKs to generate a presigned URL, the maximum expiration time for a presigned URL is 7 days from the time of creation.

Java

**Example**

The following example generates a presigned URL that you can give to others so that they can retrieve an object from an S3 on Outposts bucket. For more information, see Using presigned URLs for S3 on Outposts. To use this example, replace the *`user input placeholders`* with your own information.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

import java.io.IOException;
import java.net.URL;
import java.time.Instant;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accessPointArn = "*** access point ARN ***";
        String objectKey = "*** object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = Instant.now().toEpochMilli();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(accessPointArn, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);
```

```
            System.out.println("Pre-Signed URL: " + url.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't
 process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

**Example**

The following example generates a presigned URL that you can give to others so that they can retrieve an object from an S3 on Outposts bucket. For more information, see Using presigned URLs for S3 on Outposts. To use this example, replace the *`user input placeholders`* with your own information.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string accessPointArn = "*** access point ARN ***";
        private const string objectKey = "*** object key ***";
        // Specify how long the presigned URL lasts, in hours.
        private const double timeoutDuration = 12;
        // Specify your bucket Region (an example Region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
```

```
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL(timeoutDuration);
        }
        static string GeneratePreSignedURL(double duration)
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = accessPointArn,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            return urlString;
        }
    }
}
```

Python

The following example generates a presigned URL to share an object by using the SDK for Python (Boto3). For example, use a Boto3 client and the `generate_presigned_url` function to generate a presigned URL that allows you to `GET` an object.

```
import boto3
    url = boto3.client('s3').generate_presigned_url(
    ClientMethod='get_object',
```

```
    Params={'Bucket': 'ACCESS_POINT_ARN', 'Key': 'OBJECT_KEY'},
    ExpiresIn=3600)
```

For more information about using the SDK for Python (Boto3) to generate a presigned URL, see Python in the *AWS SDK for Python (Boto) API Reference*.

#### Using the AWS CLI

The following example AWS CLI command generates a presigned URL for an S3 on Outposts bucket. To use this example, replace the *`user input placeholders`* with your own information.

> **Note**
>
> When you use the AWS CLI to generate a presigned URL, the maximum expiration time for a presigned URL is 7 days from the time of creation.

```
aws s3 presign s3://arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point/mydoc.txt --expires-in 604800
```

For more information, see presign in the *AWS CLI Command Reference*.

### Generating a presigned URL to upload an object to an S3 on Outposts bucket

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An access point Amazon Resource Name (ARN) for the Amazon S3 on Outposts bucket
- An object key
- An HTTP method (PUT for uploading objects)
- An expiration date and time

A presigned URL is valid only for the specified duration. That is, you must start the action that's allowed by the URL before the expiration date and time. You can use a presigned URL multiple times, up to the expiration date and time. If you created a presigned URL by using a temporary token, then the URL expires when the token expires, even if you created the URL with a later expiration time.

If the action allowed by a presigned URL consists of multiple steps, such as a multipart upload, you must start all steps before the expiration time. If S3 on Outposts tries to start a step with an expired URL, you receive an error.

Users in the virtual private cloud (VPC) who have access to the presigned URL can upload objects. For example, a user in the VPC who has access to the presigned URL can upload an object to your bucket. Because presigned URLs grant access to your S3 on Outposts bucket to any user in the VPC who has access to the presigned URL, we recommend that you protect these URLs appropriately. For more details about protecting presigned URLs, see Limiting presigned URL capabilities.

Anyone with valid security credentials can create a presigned URL. However, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon. For more information, see Who can create a presigned URL.

**Using the AWS SDKs to generate a presigned URL for an S3 on Outposts object operation**

Java

**SDK for Java 2.x**

This example shows how to generate a presigned URL that you can use to upload an object to an S3 on Outposts bucket for a limited time. For more information, see Using presigned URLs for S3 on Outposts.

```
   public static void signBucket(S3Presigner presigner, String
outpostAccessPointArn, String keyName) {

       try {
```

```
            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(accessPointArn)
                    .key(keyName)
                    .contentType("text/plain")
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))
                    .putObjectRequest(objectRequest)
                    .build();

            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);


            String myURL = presignedRequest.url().toString();
            System.out.println("Presigned URL to upload a file to: " +myURL);
            System.out.println("Which HTTP method must be used when uploading a
file: " +
                    presignedRequest.httpRequest().method());

            // Upload content to the S3 on Outposts bucket by using this URL.
            URL url = presignedRequest.url();

            // Create the connection and use it to upload the new object by using
the presigned URL.
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned
URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " +
connection.getResponseCode());

        } catch (S3Exception e) {
            e.getStackTrace();
```

```
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
```

Python

**SDK for Python (Boto3)**

This example shows how to generate a presigned URL that can perform an S3 on Outposts action for a limited time. For more information, see Using presigned URLs for S3 on Outposts. To make a request with the URL, use the `Requests` package.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned S3 on Outposts URL that can be used to perform an
 action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds that the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
```

```
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.",
 client_method)
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('-'*88)
    print("Welcome to the Amazon S3 on Outposts presigned URL demo.")
    print('-'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('accessPointArn', help="The name of the S3 on Outposts
 access point ARN.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in S3 on
 Outposts. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.accessPointArn, 'Key':
 args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must
 be the "
```

```
                    f"name of a file that exists on your computer.")

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('-'*88)


if __name__ == '__main__':
    usage_demo()
```

## Amazon S3 on Outposts with local Amazon EMR on Outposts

Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Amazon EMR also helps you transform and move large amounts of data into and out of other AWS data stores and databases, and supports Amazon S3 on Outposts. For more information about Amazon EMR, see Amazon EMR on Outposts in the *Amazon EMR Management Guide*.

For Amazon S3 on Outposts, Amazon EMR started to support the Apache Hadoop S3A connector in version 7.0.0. Earlier versions of Amazon EMR don't support local S3 on Outposts, and the EMR File System (EMRFS) is not supported.

### Supported applications

Amazon EMR with Amazon S3 on Outposts supports the following applications:

- Hadoop
- Spark
- Hue
- Hive
- Sqoop
- Pig
- Hudi

- Flink

For more information, see the Amazon EMR Release Guide.

## Create and configure an Amazon S3 on Outposts bucket

Amazon EMR uses the AWS SDK for Java with Amazon S3 on Outposts to store input data and output data. Your Amazon EMR log files are stored in a Regional Amazon S3 location that you select and aren't stored locally on the Outpost. For more information, see Amazon EMR logs in the *Amazon EMR Management Guide*.

To conform with Amazon S3 and DNS requirements, S3 on Outposts buckets have naming restrictions and limitations. For more information, see Creating an S3 on Outposts bucket.

With Amazon EMR version 7.0.0 and later, you can use Amazon EMR with S3 on Outposts and the S3A file system.

**Prerequisites**

**S3 on Outposts permissions** – When you create your Amazon EMR instance profile, your role must contain the AWS Identity and Access Management (IAM) namespace for S3 on Outposts. S3 on Outposts has its own namespace, `s3-outposts*`. For an example policy that uses this namespace, see Setting up IAM with S3 on Outposts.

**S3A connector** – To configure your EMR cluster to access data from an Amazon S3 on Outposts bucket, you must use the Apache Hadoop S3A connector. To use the connector, ensure that all of your S3 URIs use the `s3a` scheme. If they don't, you can configure the file system implementation that you use for your EMR cluster so that your S3 URIs work with the S3A connector.

To configure the file system implementation to work with the S3A connector, you use the `fs.`*`file_scheme`*`.impl` and `fs.AbstractFileSystem.`*`file_scheme`*`.impl` configuration properties for your EMR cluster, where *`file_scheme`* corresponds to the type of S3 URIs that you have. To use the following example, replace the *`user input placeholders`* with your own information. For example, to change the file system implementation for S3 URIs that use the `s3` scheme, specify the following cluster configuration properties:

```
[
  {
"Classification": "core-site",
    "Properties": {
    "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
```

```
    "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

To use S3A, set the `fs.file_scheme.impl` configuration property to `org.apache.hadoop.fs.s3a.S3AFileSystem`, and set the `fs.AbstractFileSystem.file_scheme.impl` property to `org.apache.hadoop.fs.s3a.S3A`.

For example, if you are accessing the path `s3a://bucket/...`, set the `fs.s3a.impl` property to `org.apache.hadoop.fs.s3a.S3AFileSystem`, and set the `fs.AbstractFileSystem.s3a.impl` property to `org.apache.hadoop.fs.s3a.S3A`.

## Getting started using Amazon EMR with Amazon S3 on Outposts

The following topics explain how to get started using Amazon EMR with Amazon S3 on Outposts.

**Topics**



### Create a permissions policy

Before you can create an EMR cluster that uses Amazon S3 on Outposts, you must create an IAM policy to attach to the Amazon EC2 instance profile for the cluster. The policy must have permissions to access the S3 on Outposts access point Amazon Resource Name (ARN). For more information about creating IAM policies for S3 on Outposts, see Setting up IAM with S3 on Outposts.

The following example policy shows how to grant the required permissions. After you create the policy, attach the policy to the instance profile role that you use to create your EMR cluster, as described in the the section called “Create and configure your cluster” section. To use this example, replace the *user input placeholders* with your own information.

```
{
```

```
"Version":"2012-10-17",
  "Statement": [
        {
  "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name,
            "Action": [
                "s3-outposts:*"
            ]
        }
    ]

 }
```

### Create and configure your cluster

To create a cluster that runs Spark with S3 on Outposts, complete the following steps in the console.

#### To create a cluster that runs Spark with S3 on Outposts

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the left navigation pane, choose **Clusters**.
3. Choose **Create cluster**.
4. For **Amazon EMR release**, choose **emr-7.0.0** or later.
5. For Application bundle, choose **Spark interactive**. Then select any other supported applications that you want to be included in your cluster.
6. To enable Amazon S3 on Outposts, enter your configuration settings.

   **Sample configuration settings**

   To use the following sample configuration settings, replace the *user input placeholders* with your own information.

   ```
   [
    {
      "Classification": "core-site",
      "Properties": {
        "fs.s3a.bucket.DOC-EXAMPLE-BUCKET.accesspoint.arn": "arn:aws:s3-outposts:us-
   west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name"
   ```

```
     "fs.s3a.committer.name": "magic",
     "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "hadoop-env",
    "Configurations": [
      {
        "Classification": "export",
        "Properties": {
          "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
       }
     ],
     "Properties": {}
   },
   {
     "Classification": "spark-env",
     "Configurations": [
       {
         "Classification": "export",
         "Properties": {
           "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
       }
      ],
      "Properties": {}
     },
     {
      "Classification": "spark-defaults",
      "Properties": {
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-11-amazon-
corretto.x86_64",
        "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
      }
     }
  ]
```

7. In the **Networking** section, choose a virtual private cloud (VPC) and subnet that are on your AWS Outposts rack. For more information about Amazon EMR on Outposts, see EMR clusters on AWS Outposts in the *Amazon EMR Management Guide*.

8. In the **EC2 instance profile for Amazon EMR** section, choose the IAM role that has the permissions policy that you created earlier attached.
9. Configure your remaining cluster settings, and then choose **Create cluster**.

### Configurations overview

The following tables describe the S3A and Spark configurations and the values to specify for their parameters when you set up a cluster that uses S3 on Outposts with Amazon EMR.

### S3A configurations

| Parameter | Default value | Required value for S3 on Outposts | Explanation |
|---|---|---|---|
| `fs.s3a.aws.credentials.provider` | If not specified, S3A will look for S3 in Region bucket with the Outposts bucket name. | The access point ARN of the S3 on Outposts bucket | Amazon S3 on Outposts supports virtual private cloud (VPC)-only access points as the only means to access your Outposts buckets. |
| `fs.s3a.committer.name` | `file` | `magic` | Magic committer is the only supported committer for S3 on Outposts. |
| `fs.s3a.select.enabled` | `TRUE` | `FALSE` | S3 Select is not supported on Outposts. |
| `JAVA_HOME` | `/usr/lib/jvm/java-8` | `/usr/lib/jvm/java-11-amazon-corretto.x86_64` | S3 on Outposts on S3A requires Java version 11. |

**Spark configurations**

| Parameter | Default value | Required value for S3 on Outposts | Explanation |
|---|---|---|---|
| `spark.sql.sources.fastS3PartitionDiscovery.enabled` | `TRUE` | `FALSE` | S3 on Outposts doesn't support fast partition. |
| `spark.executorEnv.JAVA_HOME` | `/usr/lib/jvm/java-8` | `/usr/lib/jvm/java-11-amazon-corretto.x86_64` | S3 on Outposts on S3A requires Java version 11. |

**Considerations**

Consider the following when you integrate Amazon EMR with S3 on Outposts buckets:

- Amazon S3 on Outposts is supported with Amazon EMR version 7.0.0 and later.
- The S3A connector is required to use S3 on Outposts with Amazon EMR. Only S3A has the features required to interact with S3 on Outposts buckets. For S3A connector setup information, see Prerequisites.
- Amazon S3 on Outposts supports only server-side encryption with Amazon S3 managed keys (SSE-S3) with Amazon EMR. For more information, see the section called “Data encryption”.
- Amazon S3 on Outposts doesn't support writes with the S3A FileOutputCommitter. Writes with the S3A FileOutputCommitter on S3 on Outposts buckets result in the following error: InvalidStorageClass: The storage class you specified is not valid.
- Amazon S3 on Outposts isn't supported with Amazon EMR Serverless or Amazon EMR on EKS.
- Amazon EMR logs are stored in a Regional Amazon S3 location that you select, and are not stored locally in the S3 on Outposts bucket.

# Authorization and authentication caching

S3 on Outposts securely caches authentication and authorization data locally on Outposts racks. The cache removes round trips to the parent AWS Region for every request. This eliminates the variability that is introduced by network round trips. With the authentication and authorization cache in S3 on Outposts, you get consistent latencies that are independent from the latency of the connection between the Outposts and the AWS Region.

When you make an S3 on Outposts API request, the authentication and authorization data is securely cached. The cached data is then used to authenticate subsequent S3 object API requests. S3 on Outposts only caches authentication and authorization data when the request is signed using Signature Version 4A (SigV4A). The cache is stored locally on the Outposts within the S3 on Outposts service. It asynchronously refreshes when you make an S3 API request. The cache is encrypted, and no plaintext cryptographic keys are stored on Outposts.

The cache is valid for up to 10 minutes when the Outpost is connected to the AWS Region. It is refreshed asynchronously when you make an S3 on Outposts API request, to ensure that the latest policies are used. If the Outpost is disconnected from the AWS Region, the cache will be valid for up to 12 hours.

## Configuring the authorization and authentication cache

S3 on Outposts automatically caches authentication and authorization data for requests signed with the SigV4A algorithm. For more information, see Signing AWS API requests in the *AWS Identity and Access Management User Guide*. The SigV4A algorithm is available in the latest versions of the AWS SDKs. You can obtain it through a dependency on the AWS Common Runtime (CRT) libraries.

You need to use the latest version of the AWS SDK and install the latest version of the CRT. For example, you can run `pip install awscrt` to obtain the latest version of the CRT with Boto3.

S3 on Outposts does not cache authentication and authorization data for requests signed with the SigV4 algorithm.

## Validating SigV4A signing

You can use AWS CloudTrail to validate that requests were signed with SigV4A. For more information on setting up CloudTrail for S3 on Outposts, see Monitoring S3 on Outposts with AWS CloudTrail logs.

After you have configured CloudTrail, you can verify how a request was signed in the `SignatureVersion` field of the CloudTrail logs. Requests that were signed with SigV4A will have a `SignatureVersion` set to `AWS4-ECDSA-P256-SHA256`. Requests that were signed with SigV4 will have `SignatureVersion` set to `AWS4-HMAC-SHA256`.

# Security in S3 on Outposts

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon S3 on Outposts, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using S3 on Outposts. The following topics show you how to configure S3 on Outposts to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your S3 on Outposts resources.

**Topics**

## Data encryption in S3 on Outposts

By default, all data stored in Amazon S3 on Outposts is encrypted by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

You can optionally use server-side encryption with customer-provided encryption keys (SSE-C). To use SSE-C, specify an encryption key as part of your object API requests. Server-side encryption encrypts only the object data, not the object metadata. For more information, see Using server-side encryption with customer-provided keys (SSE-C).

> **Note**
>
> S3 on Outposts doesn't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

## AWS PrivateLink for S3 on Outposts

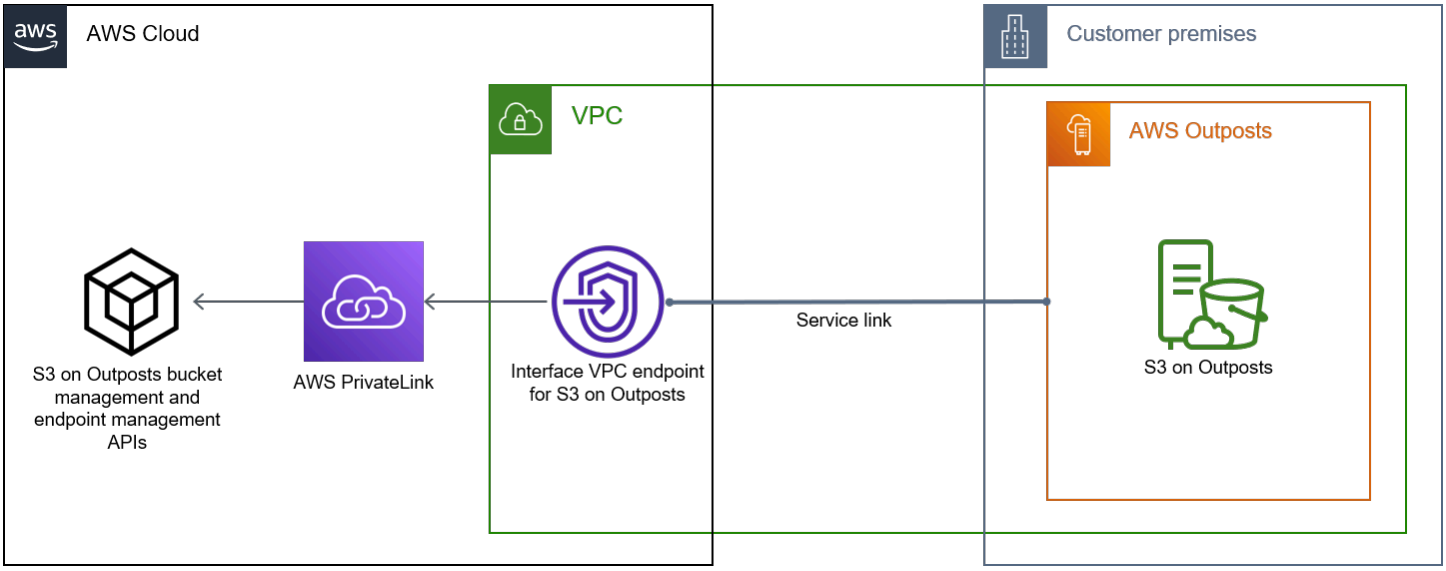
S3 on Outposts supports AWS PrivateLink, which provides direct management access to your S3 on Outposts storage through a private endpoint within your virtual private network. This allows you to simplify your internal network architecture and perform management operations on your Outposts object storage by using private IP addresses in your Virtual Private Cloud (VPC). Using AWS PrivateLink eliminates the need to use public IP addresses or proxy servers.

With AWS PrivateLink for Amazon S3 on Outposts, you can provision *interface VPC endpoints* in your virtual private cloud (VPC) to access your S3 on Outposts bucket management and endpoint management APIs. Interface VPC endpoints are directly accessible from applications deployed in your VPC or on premises over your virtual private network (VPN) or AWS Direct Connect. You can access the bucket and endpoint management APIs through AWS PrivateLink. AWS PrivateLink doesn't support data transfer API operations, such as GET, PUT, and similar APIs. These operations are already transferred privately through the S3 on Outposts endpoint and access point configuration. For more information, see Networking for S3 on Outposts.

Interface endpoints are represented by one or more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests made to interface endpoints for S3 on Outposts are automatically routed to S3 on Outposts bucket and endpoint management APIs on the AWS network. You can also access interface endpoints in your VPC from on-premises

applications through AWS Direct Connect or AWS Virtual Private Network (AWS VPN). For more information about how to connect your VPC with your on-premises network, see the *AWS Direct Connect User Guide* and the *AWS Site-to-Site VPN User Guide*.

Interface endpoints route requests for S3 on Outposts bucket and endpoint management APIs over the AWS network and through AWS PrivateLink, as illustrated in the following diagram.




For general information about interface endpoints, see Interface VPC endpoints (AWS PrivateLink) in the *AWS PrivateLink Guide*.

**Topics**

- Restrictions and limitations
- Accessing S3 on Outposts interface endpoints
- Updating an on-premises DNS configuration
- Creating a VPC endpoint for S3 on Outposts
- Creating bucket policies and VPC endpoint policies for S3 on Outposts

## Restrictions and limitations

When you access S3 on Outposts bucket and endpoint management APIs through AWS PrivateLink, VPC limitations apply. For more information, see Interface endpoint properties and limitations and AWS PrivateLink quotas in the *AWS PrivateLink Guide*.

In addition, AWS PrivateLink doesn't support the following:

- Federal Information Processing Standard (FIPS) endpoints
- S3 on Outposts data transfer APIs, for example, GET, PUT, and similar object API operations.
- Private DNS

## Accessing S3 on Outposts interface endpoints

To access S3 on Outposts bucket and endpoint management APIs using AWS PrivateLink, you *must* update your applications to use endpoint-specific DNS names. When you create an interface endpoint, AWS PrivateLink generates two types of endpoint-specific S3 on Outposts names: *Regional* and *zonal*.

- **Regional DNS names** – include a unique VPC endpoint ID, a service identifier, the AWS Region, and `vpce.amazonaws.com`, for example, *`vpce-1a2b3c4d-5e6f`*`.s3-outposts.us-east-1.vpce.amazonaws.com`.
- **Zonal DNS names** – include a unique VPC endpoint ID, the Availability Zone, a service identifier, the AWS Region, and `vpce.amazonaws.com`, for example, *`vpce-1a2b3c4d-5e6f`***`-us-east-1a.`**`s3-outposts.us-east-1.vpce.amazonaws.com`. You might use this option if your architecture isolates Availability Zones. For example, you could use zonal DNS names for fault containment or to reduce Regional data transfer costs.

> **Important**
>
> S3 on Outposts interface endpoints are resolved from the public DNS domain. S3 on Outposts does not support private DNS. Use the `--endpoint-url` parameter for all bucket and endpoint management APIs.

### AWS CLI examples

Use the `--region` and `--endpoint-url` parameters to access bucket management and endpoint management APIs through S3 on Outposts interface endpoints.

**Example : Use the endpoint URL to list buckets with the S3 control API**

In the following example, replace the Region *`us-east-1`*, VPC endpoint URL *`vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com`*, and account ID *`111122223333`* with appropriate information.

```
aws s3control list-regional-buckets --region us-east-1 --endpoint-url
 https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com --account-
id 111122223333
```

### AWS SDK examples

Update your SDKs to the latest version, and configure your clients to use an endpoint URL for accessing the S3 control API for S3 on Outposts interface endpoints. For more information, see AWS SDK examples for AWS PrivateLink.

SDK for Python (Boto3)

**Example : Use an endpoint URL to access the S3 control API**

In the following example, replace the Region *us-east-1* and VPC endpoint URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* with appropriate information.

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com'
)
```

For more information, see AWS PrivateLink for Amazon S3 in the *Boto3 developer guide*.

SDK for Java 2.x

**Example : Use an endpoint URL to access the S3 control API**

In the following example, replace the VPC endpoint URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with appropriate information.

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://vpce-1a2b3c4d-5e6f.s3-outposts.us-
east-1.vpce.amazonaws.com"))
                                  .build()
```

For more information, see `S3ControlClient` in the *AWS SDK for Java API Reference*.

## Updating an on-premises DNS configuration

When using endpoint-specific DNS names to access the interface endpoints for S3 on Outposts bucket management and endpoint management APIs, you don't have to update your on-premises DNS resolver. You can resolve the endpoint-specific DNS name with the private IP address of the interface endpoint from the public S3 on Outposts DNS domain.

## Creating a VPC endpoint for S3 on Outposts

To create a VPC interface endpoint for S3 on Outposts, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

## Creating bucket policies and VPC endpoint policies for S3 on Outposts

You can attach an endpoint policy to your VPC endpoint that controls access to S3 on Outposts. You can also use the `aws:sourceVpce` condition in S3 on Outposts bucket policies to restrict access to specific buckets from a specific VPC endpoint. With VPC endpoint policies, you can control access to S3 on Outposts bucket management APIs and endpoint management APIs. With bucket policies, you can control access to the S3 on Outposts bucket management APIs. However, you can't manage access to object actions for S3 on Outposts using `aws:sourceVpce`.

Access policies for S3 on Outposts specify the following information:

- The AWS Identity and Access Management (IAM) principal for which actions are allowed or denied.
- The S3 control actions that are allowed or denied.
- The S3 on Outposts resources on which actions are allowed or denied.

The following examples show policies that restrict access to a bucket or to an endpoint. For more information about VPC connectivity, see Network-to-VPC connectivity options in the AWS whitepaper Amazon Virtual Private Cloud Connectivity Options.

**Important**

- When you apply the example policies for VPC endpoints described in this section, you might block your access to the bucket without intending to do so. Bucket permissions

that limit bucket access to connections originating from your VPC endpoint can block all connections to the bucket. For information about how to fix this issue, see My bucket policy has the wrong VPC or VPC endpoint ID. How can I fix the policy so that I can access the bucket? in the *AWS Support Knowledge Center*.

- Before using the following example bucket policies, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
- If your policy only allows access to an S3 on Outposts bucket from a specific VPC endpoint, it disables console access for that bucket because console requests don't originate from the specified VPC endpoint.

**Topics**

- Example: Restricting access to a specific bucket from a VPC endpoint
- Example: Denying access from a specific VPC endpoint in an S3 on Outposts bucket policy

### Example: Restricting access to a specific bucket from a VPC endpoint

You can create an endpoint policy that restricts access to specific S3 on Outposts buckets only. The following policy restricts access for the GetBucketPolicy action only to the *example-outpost-bucket*. To use this policy, replace the example values with your own.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": {"AWS":"111122223333"},
      "Action": "s3-outposts:GetBucketPolicy",
      "Effect": "Allow",
      "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket"
    }
  ]
}
```

**Example: Denying access from a specific VPC endpoint in an S3 on Outposts bucket policy**

The following S3 on Outposts bucket policy denies access to GetBucketPolicy on the *example-outpost-bucket* bucket through the *vpce-1a2b3c4d* VPC endpoint.

The `aws:sourceVpce` condition specifies the endpoint and does not require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the endpoint ID. To use this policy, replace the example values with your own.

```
{
    "Version": "2012-10-17",
    "Id": "Policy1415115909152",
    "Statement": [
        {
            "Sid": "Deny-access-to-specific-VPCE",
            "Principal": {"AWS":"111122223333"},
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Deny",
            "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket",
            "Condition": {
                "StringEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}
            }
        }
    ]
}
```

## AWS Signature Version 4 (SigV4) authentication-specific policy keys

The following table shows the condition keys related to AWS Signature Version 4 (SigV4) authentication that you can use with Amazon S3 on Outposts. In a bucket policy, you can add these conditions to enforce specific behavior when requests are authenticated by using Signature Version 4. For example policies, see Bucket policy examples that use Signature Version 4-related condition keys. For more information about authenticating requests using Signature Version 4, see Authenticating requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*

**Applicable keys for `s3-outposts:*` actions or any of the S3 on Outposts actions**

| Applicable keys | Description |
| --- | --- |
| `s3-outposts:authType` | S3 on Outposts supports various methods of authentication. To restrict incoming requests to use a specific authentication method, you can use this optional condition key. For example, you can use this condition key to allow only the HTTP `Authorization` header to be used in request authentication.<br><br>Valid values:<br><br>`REST-HEADER`<br><br>`REST-QUERY-STRING` |
| `s3-outposts:signatureAge` | The length of time, in milliseconds, that a signature is valid in an authenticated request.<br><br>This condition works only for presigned URLs.<br><br>In Signature Version 4, the signing key is valid for up to seven days. Therefore, the signatures are also valid for up to seven days. For more information, see Introduction to signing requests in the *Amazon Simple Storage Service API Reference*. You can use this condition to further limit the signature age.<br><br>Example value: `600000` |
| `s3-outposts:x-amz-content-sha256` | You can use this condition key to disallow unsigned content in your bucket.<br><br>When you use Signature Version 4, for requests that use the `Authorization` header, you add the `x-amz-content-sha256` header in the signature calculation and then set its value to the hash payload.<br><br>You can use this condition key in your bucket policy to deny any uploads where the payloads are not signed. For example: |

| Applicable keys | Description |
| --- | --- |
| | • Deny uploads that use the `Authorization` header to authenticate requests but don't sign the payload. For more information, see Transferring payload in a single chunk in the *Amazon Simple Storage Service API Reference*.<br>• Deny uploads that use presigned URLs. Presigned URLs always have an `UNSIGNED_PAYLOAD`. For more information, see Authenticating requests and Authentication methods in the *Amazon Simple Storage Service API Reference*.<br><br>Valid value: `UNSIGNED-PAYLOAD` |

## Bucket policy examples that use Signature Version 4-related condition keys

To use the following examples, replace the *`user input placeholders`* with your own information.

### Example : `s3-outposts:signatureAge`

The following bucket policy denies any S3 on Outposts presigned URL request on objects in `example-outpost-bucket` if the signature is more than 10 minutes old.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
```

```
        }
    ]
}
```

### Example : `s3-outposts:authType`

The following bucket policy allows only requests that use the `Authorization` header for request authentication. Any presigned URL requests will be denied since presigned URLs use query parameters to provide request and authentication information. For more information, see Authentication methods in the *Amazon Simple Storage Service API Reference*.

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use the Authorization header for
 request authentication. Deny presigned URL requests.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringNotEquals": {
                           "s3-outposts:authType": "REST-HEADER"
                     }
               }
         }
   ]
}
```

### Example : `s3-outposts:x-amz-content-sha256`

The following bucket policy denies any uploads with unsigned payloads, such as uploads that are using presigned URLs. For more information, see Authenticating requests and Authentication methods in the *Amazon Simple Storage Service API Reference*.

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
```

```
                "Sid": "Deny uploads with unsigned payloads.",
                "Effect": "Deny",
                "Principal": {"AWS":"111122223333"},
                "Action": "s3-outposts:*",
                "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
                "Condition": {
                      "StringEquals": {
                            "s3-outposts:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                      }
                }
          }
   ]
}
```

## AWS managed policies for Amazon S3 on Outposts

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

### AWS managed policy: AWSS3OnOutpostsServiceRolePolicy

Helps manage network resources for you as part of the service-linked role `AWSServiceRoleForS3OnOutposts`.

To view the permissions for this policy, see AWSS3OnOutpostsServiceRolePolicy.

### S3 on Outposts updates to AWS managed policies

View details about updates to AWS managed policies for S3 on Outposts since this service began tracking these changes.

| Change | Description | Date |
| --- | --- | --- |
| S3 on Outposts added `AWSS3OnOutpostsServiceRolePolicy` | S3 on Outposts added `AWSS3OnOutpostsServiceRolePolicy` as part of the service-linked role `AWSServiceRoleForS3OnOutposts`, which helps manage network resources for you. | October 3, 2023 |
| S3 on Outposts started tracking changes | S3 on Outposts started tracking changes for its AWS managed policies. | October 3, 2023 |

## Using service-linked roles for Amazon S3 on Outposts

Amazon S3 on Outposts uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to S3 on Outposts. Service-linked roles are predefined by S3 on Outposts and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up S3 on Outposts easier because you don’t have to manually add the necessary permissions. S3 on Outposts defines the permissions of its service-linked roles, and unless defined otherwise, only S3 on Outposts can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your S3 on Outposts resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

## Service-linked role permissions for S3 on Outposts

S3 on Outposts uses the service-linked role named **AWSServiceRoleForS3OnOutposts** to help manage network resources for you.

The `AWSServiceRoleForS3OnOutposts` service-linked role trusts the following services to assume the role:

- `s3-outposts.amazonaws.com`

The role permissions policy named `AWSS3OnOutpostsServiceRolePolicy` allows S3 on Outposts to complete the following actions on the specified resources:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeCoipPools",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeAddresses",
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
            ],
            "Resource": "*",
            "Sid": "DescribeVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
```

```
            ],
            "Sid": "CreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForCreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:ipv4pool-ec2/*"
            ],
            "Sid": "AllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:elastic-ip/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForAllocateIpAddress"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:AssociateAddress"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "ReleaseVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateNetworkInterface",
                        "AllocateAddress"
                    ],
                    "aws:RequestTag/CreatedBy": [
                        "S3 On Outposts"
                    ]
                }
            },
            "Sid": "CreateTags"
        }
    ]
}
```

You must configure permissions to allow an IAM entity (such as a role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

## Creating a service-linked role for S3 on Outposts

You don't need to manually create a service-linked role. When you create an S3 on Outposts endpoint in the AWS Management Console, the AWS CLI, or the AWS API, S3 on Outposts creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create an S3 on Outposts endpoint, S3 on Outposts creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **S3 on Outposts** use case. In the AWS CLI or the AWS API, create a service-linked role with the `s3-outposts.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

## Editing a service-linked role for S3 on Outposts

S3 on Outposts does not allow you to edit the `AWSServiceRoleForS3OnOutposts` service-linked role. This includes the name of the role because various entities might reference it. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

## Deleting a service-linked role for S3 on Outposts

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the S3 on Outposts service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

**To delete S3 on Outposts resources used by the AWSServiceRoleForS3OnOutposts role**

1. Delete the S3 on Outposts endpoints in your AWS account across all AWS Regions.
2. Delete the service-linked role using IAM.

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForS3OnOutposts` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

### Supported Regions for S3 on Outposts service-linked roles

S3 on Outposts supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see S3 on Outposts Regions and endpoints.

## Managing S3 on Outposts storage

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about managing and sharing your Amazon S3 on Outposts storage capacity, see the following topics.

**Topics**

- Managing S3 Versioning for your S3 on Outposts bucket
- Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket
- Replicating objects for S3 on Outposts
- Sharing S3 on Outposts by using AWS RAM
- Other AWS services that use S3 on Outposts

## Managing S3 Versioning for your S3 on Outposts bucket

When enabled, S3 Versioning saves multiple distinct copies of an object in the same bucket. You can use S3 Versioning to preserve, retrieve, and restore every version of every object stored in your

Outposts buckets. S3 Versioning helps you recover from unintended user actions and application failures.

Amazon S3 on Outposts buckets have three versioning states:

- **Unversioned** – If you've never enabled or suspended S3 Versioning on your bucket, it is unversioned and returns no S3 Versioning status. For more information about S3 Versioning, see Using versioning in S3 buckets.
- **Enabled** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as PutObject, the new objects get a unique version ID.
- **Suspended** – Suspends S3 Versioning for the objects in the bucket. All objects added to the bucket after versioning is suspended receive the version ID `null`. For more information, see Adding objects to versioning-suspended buckets.

After you enable S3 Versioning for an S3 on Outposts bucket, it can never return to an unversioned state. However, you can suspend versioning. For more information about S3 Versioning, see Using versioning in S3 buckets.

For each object in your bucket, you have a current version and zero or more noncurrent versions. To reduce storage costs, you can configure your bucket S3 Lifecycle rules to expire noncurrent versions after a specified time period. For more information, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

The following examples show you how to enable or suspend versioning for an existing S3 on Outposts bucket by using the AWS Management Console and the AWS Command Line Interface (AWS CLI). To create a bucket with S3 Versioning enabled, see Creating an S3 on Outposts bucket.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can commit actions to it. Buckets have configuration properties, such as Outpost, tag, default encryption, and access point settings. The access point settings include the virtual private cloud (VPC), the access point policy for accessing the objects in the bucket, and other metadata. For more information, see S3 on Outposts specifications.

### Using the S3 console

#### To edit the S3 Versioning settings for your bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to enable S3 Versioning for.
4. Choose the **Properties** tab.
5. Under **Bucket Versioning**, choose **Edit**.
6. Edit the S3 Versioning settings for the bucket by choosing one of the following options:
   - To suspend S3 Versioning and stop the creation of new object versions, choose **Suspend**.
   - To enable S3 Versioning and save multiple distinct copies of each object, choose **Enable**.
7. Choose **Save changes**.

### Using the AWS CLI

To enable or suspend S3 Versioning for your bucket by using the AWS CLI, use the `put-bucket-versioning` command, as shown in the following examples. To use these examples, replace each *`user input placeholder`* with your own information.

For more information, see put-bucket-versioning in the *AWS CLI Reference*.

#### Example : To enable S3 Versioning

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Enabled
```

#### Example : To suspend S3 Versioning

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Suspended
```

# Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Managing your storage lifecycle.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage the lifecycle configuration for your S3 on Outposts bucket, see the following topics.

**Topics**

- Creating and managing a lifecycle rule by using the AWS Management Console
- Creating and managing a lifecycle configuration by using the AWS CLI and SDK for Java

## Creating and managing a lifecycle rule by using the AWS Management Console

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Managing your storage lifecycle.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage a lifecycle rule for an S3 on Outposts by using the AWS Management Console, see the following topics.

**Topics**



### Creating a lifecycle rule

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to create a lifecycle rule for.
4. Choose the **Management** tab, and then choose **Create Lifecycle rule**.
5. Enter a value for **Lifecycle rule name**.
6. Under **Rule scope**, choose one of the following options:
   - To limit the scope to specific filters, choose **Limit the scope of this rule using one or more filters**. Then, add a prefix filter, tags, or object size.
   - To apply the rule to all objects in the bucket, choose **Apply to all objects in the bucket**.
7. Under **Lifecycle rule actions**, choose one of the following options:
   - **Expire current versions of objects** – For versioning-enabled buckets, S3 on Outposts adds a delete marker and retains the objects as noncurrent versions. For buckets that don't use S3 Versioning, S3 on Outposts permanently deletes the objects.
   - **Permanently delete noncurrent versions of objects** – S3 on Outposts permanently deletes noncurrent versions of objects.
   - **Delete expired object delete markers or incomplete multipart uploads** – S3 on Outposts permanently deletes expired object delete markers or incomplete multipart uploads.

     If you limit the scope of your Lifecycle rule by using object tags, you can't choose **Delete expired object delete markers**. You also can't choose **Delete expired object delete markers** if you choose **Expire current object versions**.

> **Note**
>
> Size-based filters can't be used with delete markers and incomplete multipart uploads.

8. If you chose **Expire current versions of objects** or **Permanently delete noncurrent versions of objects**, configure the rule trigger based on a specific date or the object's age.
9. If you chose **Delete expired object delete markers**, to confirm that you want to delete expired object delete markers, select **Delete expired object delete markers**.
10. Under **Timeline Summary**, review your Lifecycle rule, and choose **Create rule**.

### Enabling a lifecycle rule

#### To enable or disable a bucket lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to enable or disable a lifecycle rule for.
4. Choose the **Management** tab, and then under **Lifecycle rule**, choose the rule that you want to enable or disable.
5. For **Action**, choose **Enable or disable rule**.

### Editing a lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to edit a lifecycle rule for.
4. Choose the **Management** tab, and then choose the **Lifecycle rule** that you want to edit.
5. (Optional) Update the value for **Lifecycle rule name**.
6. Under **Rule scope**, edit the scope as needed:

   - To limit the scope to specific filters, choose **Limit the scope of this rule using one or more filters**. Then, add a prefix filter, tags, or object size.
   - To apply the rule to all objects in the bucket, choose **Apply to all objects in the bucket**.

7. Under **Lifecycle rule actions**, choose one of the following options:

   - **Expire current versions of objects** – For versioning-enabled buckets, S3 on Outposts adds a delete marker and retains the objects as noncurrent versions. For buckets that don't use S3 Versioning, S3 on Outposts permanently deletes the objects.
   - **Permanently delete noncurrent versions of objects** – S3 on Outposts permanently deletes noncurrent versions of objects.
   - **Delete expired object delete markers or incomplete multipart uploads** – S3 on Outposts permanently deletes expired object delete markers or incomplete multipart uploads.

     If you limit the scope of your Lifecycle rule by using object tags, you can't choose **Delete expired object delete markers**. You also can't choose **Delete expired object delete markers** if you choose **Expire current object versions**.

   > **Note**
   >
   > Size-based filters can't be used with delete markers and incomplete multipart uploads.

8. If you chose **Expire current versions of objects** or **Permanently delete noncurrent versions of objects**, configure the rule trigger based on a specific date or the object age.
9. If you chose **Delete expired object delete markers**, to confirm that you want to delete expired object delete markers, select **Delete expired object delete markers**.
10. Choose **Save**.

### Deleting a lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to delete a lifecycle rule for.
4. Choose the **Management** tab, and then under **Lifecycle rule**, choose the rule that you want to delete.
5. Choose **Delete**.

## Creating and managing a lifecycle configuration by using the AWS CLI and SDK for Java

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Managing your storage lifecycle.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage a lifecycle configuration for an S3 on Outposts bucket by using the AWS Command Line Interface (AWS CLI) and the AWS SDK for Java, see the following examples.

**Topics**

- PUT a lifecycle configuration
- GET the lifecycle configuration on an S3 on Outposts bucket

### PUT a lifecycle configuration

AWS CLI

The following AWS CLI example puts a lifecycle configuration policy on an Outposts bucket. This policy specifies that all objects that have the flagged prefix (*myprefix*) and tags expire after 10 days. To use this example, replace each *user input placeholder* with your own information.

1. Save the lifecycle configuration policy to a JSON file. In this example, the file is named `lifecycle1.json`.

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
```

```
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ],
                    "ObjectSizeGreaterThan": 1000,
                    "ObjectSizeLessThan": 5000
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 10
            }
        }
    ]
}
```

2. Submit the JSON file as part of the `put-bucket-lifecycle-configuration` CLI command. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see put-bucket-lifecycle-configuration in the *AWS CLI Reference*.

```
aws s3control put-bucket-lifecycle-configuration --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket --lifecycle-configuration file://lifecycle1.json
```

SDK for Java

The following SDK for Java example puts a lifecycle configuration on an Outposts bucket. This lifecycle configuration specifies that all objects that have the flagged prefix (*`myprefix`*) and tags expire after 10 days. To use this example, replace each *`user input placeholder`* with your own information. For more information, see PutBucketLifecycleConfiguration in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void putBucketLifecycleConfiguration(String bucketArn) {

    S3Tag tag1 = new S3Tag().withKey("mytagkey1").withValue("mytagkey1");
    S3Tag tag2 = new S3Tag().withKey("mytagkey2").withValue("mytagkey2");

    LifecycleRuleFilter lifecycleRuleFilter = new LifecycleRuleFilter()
            .withAnd(new LifecycleRuleAndOperator()
                    .withPrefix("myprefix")
                    .withTags(tag1, tag2))
                    .withObjectSizeGreaterThan(1000)
                    .withObjectSizeLessThan(5000);

    LifecycleExpiration lifecycleExpiration = new LifecycleExpiration()
            .withExpiredObjectDeleteMarker(false)
            .withDays(10);

    LifecycleRule lifecycleRule = new LifecycleRule()
            .withStatus("Enabled")
            .withFilter(lifecycleRuleFilter)
            .withExpiration(lifecycleExpiration)
            .withID("id-1");

    LifecycleConfiguration lifecycleConfiguration = new LifecycleConfiguration()
            .withRules(lifecycleRule);

    PutBucketLifecycleConfigurationRequest reqPutBucketLifecycle = new
 PutBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withLifecycleConfiguration(lifecycleConfiguration);

    PutBucketLifecycleConfigurationResult respPutBucketLifecycle =
 s3ControlClient.putBucketLifecycleConfiguration(reqPutBucketLifecycle);
    System.out.printf("PutBucketLifecycleConfiguration Response: %s%n",
 respPutBucketLifecycle.toString());

}
```

**GET the lifecycle configuration on an S3 on Outposts bucket**

AWS CLI

The following AWS CLI example gets a lifecycle configuration on an Outposts bucket. To use this command, replace each *user input placeholder* with your own information. For more information about this command, see get-bucket-lifecycle-configuration in the *AWS CLI Reference*.

```
aws s3control get-bucket-lifecycle-configuration --account-id 123456789012 --bucket
 arn:aws:s3-outposts:<your-region>:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket
```

SDK for Java

The following SDK for Java example gets a lifecycle configuration for an Outposts bucket. For more information, see GetBucketLifecycleConfiguration in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void getBucketLifecycleConfiguration(String bucketArn) {

    GetBucketLifecycleConfigurationRequest reqGetBucketLifecycle = new
 GetBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketLifecycleConfigurationResult respGetBucketLifecycle =
 s3ControlClient.getBucketLifecycleConfiguration(reqGetBucketLifecycle);
    System.out.printf("GetBucketLifecycleConfiguration Response: %s%n",
 respGetBucketLifecycle.toString());

}
```

## Replicating objects for S3 on Outposts

With S3 Replication on AWS Outposts, you can configure Amazon S3 on Outposts to automatically replicate S3 objects across different Outposts, or between buckets on the same Outpost. You can use S3 Replication on Outposts to maintain multiple replicas of your data in the same or

different Outposts, or across different accounts, to help meet data-residency needs. S3 Replication on Outposts helps power your compliant storage needs and data sharing across accounts. If you need to ensure that your replicas are identical to the source data, you can use S3 Replication on Outposts to make replicas of your objects that retain all metadata, such as the original object creation time, tags, and version IDs.

S3 Replication on Outposts also provides detailed metrics and notifications to monitor the status of object replication between buckets. You can use Amazon CloudWatch to monitor replication progress by tracking bytes pending replication, operations pending replication, and replication latency between your source and destination buckets. To quickly diagnose and correct configuration issues, you can also set up Amazon EventBridge to receive notifications about replication object failures. To learn more, see Managing your replication.

**Topics**



## Replication configuration

S3 on Outposts stores a replication configuration as XML. In the replication configuration XML file, you specify an AWS Identity and Access Management (IAM) role and one or more rules.

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

S3 on Outposts can't replicate objects without your permission. You grant S3 on Outposts permissions with the IAM role that you specify in the replication configuration. S3 on Outposts assumes that IAM role to replicate objects on your behalf. You must grant the required permissions to the IAM role before starting replication. For more information about these permissions for S3 on Outposts, see Creating an IAM role.

You add one rule in a replication configuration in the following scenarios:

- You want to replicate all objects.
- You want to replicate one subset of objects. You identify the object subset by adding a filter in the rule. In the filter, you specify an object key prefix, tags, or a combination of both, to identify the subset of objects that the rule applies to.

You add multiple rules in a replication configuration if you want to replicate different subsets of objects. In each rule, you specify a filter that selects a different subset of objects. For example, you might choose to replicate objects that have either `tax/` or `document/` key prefixes. To do this, you add two rules, one that specifies the `tax/` key prefix filter and another that specifies the `document/` key prefix.

For more information about S3 on Outposts replication configuration and replication rules, see ReplicationConfiguration in the *Amazon Simple Storage Service API Reference*.

## Requirements for S3 Replication on Outposts

Replication requires the following:

- The destination Outpost CIDR range must be associated in your source Outpost subnet table. For more information, see Prerequisites for creating replication rules.
- Both the source and destination buckets must have S3 Versioning enabled. For more information about versioning, see Managing S3 Versioning for your S3 on Outposts bucket.
- Amazon S3 on Outposts must have permission to replicate objects from the source bucket to the destination bucket on your behalf. That means you must create a service role to delegate `GET` and `PUT` permissions to S3 on Outposts.
  1. Before creating the service role, you must have `GET` permission on the source bucket and `PUT` permission on the destination bucket.
  2. To create the service role to delegate permissions to S3 on Outposts, you must first configure permissions to allow an IAM entity (a user or role) to perform the `iam:CreateRole` and

`iam:PassRole` actions. Then, you allow the IAM entity to create a service role. To make S3 on Outposts assume the service role on your behalf and delegate GET and PUT permissions to S3 on Outposts, you must assign the necessary trust and permissions policies to the role. For more information about these permissions for S3 on Outposts, see Creating an IAM role. For more information about creating a service role, see Creating a service role.

## What is replicated?

By default, S3 on Outposts replicates the following:

- Objects created after you add a replication configuration.
- Object metadata from the source objects to the replicas. For information about how to replicate metadata from the replicas to the source objects, see Replication status if Amazon S3 replica modification sync on Outposts is enabled.
- Object tags, if there are any.

### How delete operations affect replication

If you delete an object from the source bucket, the following actions occur by default:

- If you make a `DELETE` request without specifying an object version ID, S3 on Outposts adds a delete marker. S3 on Outposts deals with the delete marker as follows:
  - S3 on Outposts does not replicate the delete marker by default.
  - However, you can add *delete marker replication* to non-tag-based rules. For more information about how to enable delete marker replication in your replication configuration, see Using the S3 console.
- If you specify an object version ID to delete in a `DELETE` request, S3 on Outposts permanently deletes that object version in the source bucket. However, it doesn't replicate the deletion in the destination buckets. In other words, it doesn't delete the same object version from the destination buckets. This behavior protects data from malicious deletions.

## What isn't replicated?

By default, S3 on Outposts doesn't replicate the following:

- Objects in the source bucket that are replicas that were created by another replication rule. For example, suppose you configure replication where bucket A is the source and bucket B is the destination. Now suppose that you add another replication configuration where bucket B is the source and bucket C is the destination. In this case, objects in bucket B that are replicas of objects in bucket A are not replicated to bucket C.
- Objects in the source bucket that have already been replicated to a different destination. For example, if you change the destination bucket in an existing replication configuration, S3 on Outposts won't replicate the objects again.
- Objects that are created with server-side encryption with customer-provided encryption keys (SSE-C).
- Updates to bucket-level subresources.

  For example, if you change the lifecycle configuration or add a notification configuration to your source bucket, these changes are not applied to the destination bucket. This feature makes it possible to have different configurations on the source and destination buckets.
- Actions performed by lifecycle configuration.

  For example, if you enable a lifecycle configuration only on your source bucket and configure expiration actions, S3 on Outposts creates delete markers for expired objects in the source bucket but doesn't replicate those markers to the destination buckets. If you want the same lifecycle configuration applied to both the source and destination buckets, enable the same lifecycle configuration on both. For more information about lifecycle configuration, see Managing your storage lifecycle.

## What isn't supported by S3 Replication on Outposts?

The following S3 Replication features are currently not supported by S3 on Outposts:

- S3 Replication Time Control (S3 RTC). S3 RTC is not supported because the object traffic in S3 Replication on Outposts travels over your on-premises network (the local gateway). For more information about local gateways, see Working with the local gateway in the *AWS Outposts User Guide*.
- S3 Replication for Batch Operations.

## Setting up replication

> **Note**
>
> Objects that existed in your bucket before you set up a replication rule aren't replicated automatically. In other words, Amazon S3 on Outposts doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration, you can use the `CopyObject` API operation to copy them to the same bucket. After the objects are copied, they appear as "new" objects in the bucket and the replication configuration will apply to them. For more information about copying an object, see Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java and CopyObject in the *Amazon Simple Storage Service API Reference*.

To enable S3 Replication on Outposts, add a replication rule to your source Outposts bucket. The replication rule tells S3 on Outposts to replicate objects as specified. In the replication rule, you must provide the following:

- **The source Outposts bucket access point** – The access point Amazon Resource Name (ARN) or access point alias of the bucket from which you want S3 on Outposts to replicate the objects. For more information about using access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.
- **The objects that you want to replicate** – You can replicate all of the objects in the source Outposts bucket or a subset. You identify a subset by providing a key name prefix, one or more object tags, or both in the configuration.

  For example, if you configure a replication rule to replicate only objects with the key name prefix `Tax/`, S3 on Outposts replicates objects with keys such as `Tax/doc1` or `Tax/doc2`. But it doesn't replicate objects with the key `Legal/doc3`. If you specify both a prefix and one or more tags, S3 on Outposts replicates only objects that have the specific key prefix and tags.
- **The destination Outposts bucket** – The ARN or access point alias of the bucket to which you want S3 on Outposts to replicate the objects.

You can configure the replication rule by using the REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or Amazon S3 console.

S3 on Outposts also provides API operations to support setting up replication rules. For more information, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

**Topics**



### Prerequisites for creating replication rules

**Topics**



#### Connecting your source and destination Outpost subnets

To have your replication traffic go from your source Outpost to your destination Outpost over your local gateway, you must add a new route to set up networking. You must connect the Classless Inter-Domain Routing (CIDR) networking ranges of your access points together. For each pair of access points, you need to set up this connection only once.

Some steps to set up the connection are different, depending on the access type of your Outposts endpoints that are associated with your access points. The access type for endpoints is either **Private** (direct virtual private cloud [VPC] routing for AWS Outposts) or **Customer owned IP** (a customer-owned IP address pool [CoIP pool] within your on-premises network).

**Step 1: Find the CIDR range of your source Outposts endpoint**

**To find the CIDR range of your source endpoint that's associated with your source access point**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.

3. In the **Outposts buckets** list, choose the source bucket that you want for replication.
4. Choose the **Outposts access points** tab, and choose the Outposts access point for the source bucket for your replication rule.
5. Choose the Outposts endpoint.
6. Copy the subnet ID for use in Step 5.
7. The method that you use to find the CIDR range of the source Outposts endpoint depends on the access type of your endpoint.

   In the **Outposts endpoint overview** section, see the **Access Type**.

   - If the access type is **Private**, copy the **Classless inter-domain routing (CIDR)** value to use in Step 6.
   - If the access type is **Customer Owned IP**, do the following:
     1. Copy the **Customer owned IPv4 pool** value to use as the ID of the address pool later on.
     2. Open the AWS Outposts console at https://console.aws.amazon.com/outposts/.
     3. In the navigation pane, choose **Local gateway route tables**.
     4. Choose the **Local gateway route table ID** value of your source Outpost.
     5. In the details pane, choose the **CoIP pools** tab. Paste the value of your CoIP pool ID that you copied previously in the search box.
     6. For the matched CoIP pool, copy the corresponding **CIDRs** value of your source Outposts endpoint for use in Step 6.

### Step 2: Find the subnet ID and the CIDR range of your destination Outposts endpoint

To find the subnet ID and the CIDR range of your destination endpoint that's associated with your destination access point, follow the same substeps in Step 1 and change your source Outposts endpoint to your destination Outposts endpoint when you apply those substeps. Copy the subnet ID value of your destination Outposts endpoint for use in Step 6. Copy the CIDR value of your destination Outposts endpoint for use in Step 5.

### Step 3: Find the local gateway ID of your source Outpost

#### To find the local gateway ID of your source Outpost

1. Open the AWS Outposts console at https://console.aws.amazon.com/outposts/.
2. In the left navigation pane, choose **Local gateways**.

3. On the **Local gateways** page, find the Outpost ID of your source Outpost that you want to use for replication.
4. Copy the local gateway ID value of your source Outpost for use in Step 5.

For more information about local gateway, see Local gateway in the *AWS Outposts User Guide*.

**Step 4: Find the local gateway ID of your destination Outpost**

To find the local gateway ID of your destination Outpost, follow the same substeps in Step 3, except look for the Outpost ID for your destination Outpost. Copy the local gateway ID value of your destination Outpost for use in Step 6.

**Step 5: Set up the connection from your source Outpost subnet to your destination Outpost subnet**

**To connect from your source Outpost subnet to your destination Outpost subnet**

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose **Subnets**.
3. In the search box, enter the subnet ID for your source Outposts endpoint that you found in Step 1. Choose the subnet with the matched subnet ID.
4. For the matched subnet item, choose the **Route table** value of this subnet.
5. On the page with a selected route table, choose **Actions**, and then choose **Edit routes**.
6. On the **Edit routes** page, choose **Add route**.
7. Under **Destination**, enter the CIDR range of your destination Outposts endpoint that you found in Step 2.
8. Under **Target**, choose **Outpost Local Gateway**, and enter the local gateway ID of your source Outpost that you found in Step 3.
9. Choose **Save changes**.
10. Make sure the **Status** for the route is **Active**.

**Step 6: Set up the connection from your destination Outpost subnet to your source Outpost subnet**

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose **Subnets**.
3. In the search box, enter the subnet ID for your destination Outposts endpoint that you found in Step 2. Choose the subnet with the matched subnet ID.
4. For the matched subnet item, choose the **Route table** value of this subnet.
5. On the page with a selected route table, choose **Actions**, and then choose **Edit routes**.
6. On the **Edit routes** page, choose **Add route**.
7. Under **Destination**, enter the CIDR range of your source Outposts endpoint that you found in Step 1.
8. Under **Target**, choose **Outpost Local Gateway**, and enter the local gateway ID of your destination Outpost that you found in Step 4.
9. Choose **Save changes**.
10. Make sure the **Status** for the route is **Active**.

After you connect the CIDR networking ranges of your source and destination access points, you must create an AWS Identity and Access Management (IAM) role.

**Creating an IAM role**

By default, all S3 on Outposts resources—buckets, objects, and related subresources—are private, and only the resource owner can access the resource. S3 on Outposts needs permissions to read and replicate objects from the source Outposts bucket. You grant these permissions by creating an IAM *service role* and specifying that role in your replication configuration.

This section explains the trust policy and minimum required permissions policy. The example walkthroughs provide step-by-step instructions to create an IAM role. For more information, see Creating replication rules on Outposts. For more information about IAM roles, see IAM roles in the *IAM User Guide*.

- The following example shows a *trust policy*, where you identify S3 on Outposts as the service principal that can assume the role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- The following example shows an *access policy*, where you grant the role permissions to perform replication tasks on your behalf. When S3 on Outposts assumes the role, it has the permissions that you specify in this policy. To use this policy, replace the *`user input placeholders`* with your own information. Make sure to replace them with the Outpost IDs of your source and destination Outposts and the bucket names and access point names of your source and destination Outposts buckets.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
```

```
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

The access policy grants permissions for the following actions:

- `s3-outposts:GetObjectVersionForReplication` – Permission for this action is granted on all objects to allow S3 on Outposts to get a specific object version that's associated with each object.

- `s3-outposts:GetObjectVersionTagging` – Permission for this action on objects in the *SOURCE-OUTPOSTS-BUCKET* bucket (the source bucket) allows S3 on Outposts to read object tags for replication. For more information, see Adding tags for S3 on Outposts buckets. If S3 on Outposts doesn't have this permission, it replicates the objects, but not the object tags.
- `s3-outposts:ReplicateObject` and `s3-outposts:ReplicateDelete` – Permissions for these actions on all objects in the *DESTINATION-OUTPOSTS-BUCKET* bucket (the destination bucket) allow S3 on Outposts to replicate objects or delete markers to the destination Outposts bucket. For information about delete markers, see How delete operations affect replication.

  **Note**

  - Permission for the `s3-outposts:ReplicateObject` action on the *DESTINATION-OUTPOSTS-BUCKET* bucket (the destination bucket) also allows replication of object tags. Therefore, you don't need to explicitly grant permission for the `s3-outposts:ReplicateTags` action.
  - For cross-account replication, the owner of the destination Outposts bucket must update its bucket policy to grant permission for the `s3-outposts:ReplicateObject` action on the*DESTINATION-OUTPOSTS-BUCKET*. The `s3-outposts:ReplicateObject` action allows S3 on Outposts to replicate objects and object tags to the destination Outposts bucket.

For a list of S3 on Outposts actions, see Actions defined by S3 on Outposts.

> **Important**
>
> The AWS account that owns the IAM role must have permissions for the actions that it grants to the IAM role.
> For example, suppose that the source Outposts bucket contains objects owned by another AWS account. The owner of the objects must explicitly grant the AWS account that owns the IAM role the required permissions through the bucket policy and the access point policy. Otherwise, S3 on Outposts can't access the objects, and replication of the objects fails.
> The permissions described here are related to the minimum replication configuration. If you choose to add optional replication configurations, you must grant additional permissions to S3 on Outposts.

**Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts**

When the source and destination Outposts buckets aren't owned by the same accounts, the owner of the destination Outposts bucket must update the bucket and access point policies for the destination bucket. These policies must grant the owner of the source Outposts bucket and the IAM service role permissions to perform replication actions, as shown in the following policy examples, or replication will fail. In these policy examples, *`DESTINATION-OUTPOSTS-BUCKET`* is the destination bucket. To use these policy examples, replace the *`user input placeholders`* with your own information.

If you're creating the IAM service role manually, set the role path as `role/service-role/`, as shown in the following policy examples. For more information, see IAM ARNs in the *IAM User Guide*.

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
```

```
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*"
         ]
      }

   ]
}
```

```
{
"Version":"2012-10-17",
   "Id":"PolicyForDestinationAccessPoint",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource" :[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/
object/*"
         ]
      }
   ]
}
```

**Note**

If objects in the source Outposts bucket are tagged, note the following:

If the source Outposts bucket owner grants S3 on Outposts permission for the `s3-outposts:GetObjectVersionTagging` and `s3-outposts:ReplicateTags` actions to replicate object tags (through the IAM role), Amazon S3 replicates the tags along with the objects. For information about the IAM role, see Creating an IAM role.

### Creating replication rules on Outposts

S3 Replication on Outposts is the automatic, asynchronous replication of objects across buckets in the same or different AWS Outposts. Replication copies newly created objects and object updates from a source Outposts bucket to a destination Outposts bucket or buckets. For more information, see Replicating objects for S3 on Outposts.

**Note**

Objects that existed in the source Outposts bucket before you set up replication rules aren't replicated. In other words, S3 on Outposts doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration, you can use the `CopyObject` API operation to copy them to the same bucket. After the objects are copied, they appear as "new" objects in the bucket and the replication configuration will apply to them. For more information about copying an object, see Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java and CopyObject in the *Amazon Simple Storage Service API Reference*.

When you configure replication, you add replication rules to the source Outposts bucket. Replication rules define which source Outposts bucket objects to replicate and the destination Outposts bucket or buckets where the replicated objects will be stored. You can create a rule to replicate all the objects in a bucket or a subset of objects with a specific key name prefix, one or more object tags, or both. A destination Outposts bucket can be in the same Outpost as the source Outposts bucket, or it can be in a different Outpost.

For S3 on Outposts replication rules, you must provide both the source Outposts bucket's access point Amazon Resource Name (ARN) and the destination Outposts bucket's access point ARN instead of the source and destination Outposts bucket names.

If you specify an object version ID to delete, S3 on Outposts deletes that object version in the source Outposts bucket. But it doesn't replicate the deletion to the destination Outposts bucket. In

other words, it doesn't delete the same object version from the destination Outposts bucket. This behavior protects data from malicious deletions.

When you add a replication rule to an Outposts bucket, the rule is enabled by default, so the rule starts working as soon as you save it.

In this example, you set up replication for source and destination Outposts buckets that are on different Outposts and are owned by the same AWS account. Examples are provided for using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), and the AWS SDK for Java and AWS SDK for .NET. For information about cross-account S3 Replication on Outposts permissions, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.

For prerequisites to set up S3 on Outposts replication rules, see Prerequisites for creating replication rules.

**Using the S3 console**

Follow these steps to configure a replication rule when the destination Amazon S3 on Outposts bucket is in a different Outpost from the source Outposts bucket.

If the destination Outposts bucket is in a different account from the source Outposts bucket, you must add a bucket policy to the destination Outposts bucket to grant the owner of the source Outposts bucket account permission to replicate objects in the destination Outposts bucket. For more information, see Granting permissions when the source and destination buckets are owned by different AWS accounts.

**To create a replication rule**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Outposts Buckets** list, choose the name of the bucket that you want to use as the source bucket.
3. Choose the **Management** tab, scroll down to the **Replication rules** section, and then choose **Create replication rule**.
4. For **Replication rule name**, enter a name for your rule to help identify the rule later. The name is required and must be unique within the bucket.
5. Under **Status**, **Enabled** is chosen by default. An enabled rule starts to work as soon as you save it. If you want to enable the rule later, choose **Disabled**.

6. Under **Priority**, the rule's priority value determines which rule to apply if there are overlapping rules. When objects are included in the scope of more than one replication rule, S3 on Outposts uses these priority values to avoid conflicts. By default, new rules are added to the replication configuration at the highest priority. The higher the number, the higher the priority.

   To change the priority for the rule, after you save the rule, choose the rule name from the replication rule list, choose **Actions**, and then choose **Edit priority**.

7. Under **Source bucket**, you have the following options for setting the replication source:

   - To replicate the whole bucket, choose **Apply to *all* objects in the bucket**.
   - To apply prefix or tag filtering to the replication source, choose **Limit the scope of this rule by using one or more filters**. You can combine a prefix and tags.
     - To replicate all objects that have the same prefix, under **Prefix**, enter a prefix in the box. Using the **Prefix** filter limits replication to all objects that have names that begin with the same string (for example, `pictures`).

       If you enter a prefix that is the name of a folder, you must use a **/** (forward slash) as the last character (for example, `pictures/`).
     - To replicate all objects that have one or more of the same object tags, choose **Add tag**, and then enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Adding tags for S3 on Outposts buckets.

8. To access your S3 on Outposts source bucket for replication, under **Source access point name**, choose an access point that is attached to the source bucket.

9. Under **Destination**, choose the access point ARN of the destination Outposts bucket where you want S3 on Outposts to replicate objects. The destination Outposts bucket can be in the same or a different AWS account as the source Outposts bucket.

   If the destination bucket is in a different account from the source Outposts bucket, you must add a bucket policy to the destination Outposts bucket to grant the owner of the source Outposts bucket account permission to replicate objects to the destination Outposts bucket. For more information, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.

> **Note**
>
> If versioning is not enabled on the destination Outposts bucket, you get a warning that contains an **Enable versioning** button. Choose this button to enable versioning on the bucket.

10. Set up an AWS Identity and Access Management (IAM) service role that S3 on Outposts can assume to replicate objects on your behalf.

    To set up an IAM role, under **IAM role**, do one of the following:

    - To have S3 on Outposts create a new IAM role for your replication configuration, choose **Choose from existing IAM roles**, and then choose **Create new role**. When you save the rule, a new policy is generated for the IAM role that matches the source and destination Outposts buckets that you choose. We recommend that you choose **Create new role**.
    - You can also choose to use an existing IAM role. If you do, you must choose a role that grants S3 on Outposts the necessary permissions for replication. If this role doesn't grant S3 on Outposts sufficient permissions to follow your replication rule, replication fails.

      To choose an existing role, choose **Choose from existing IAM roles**, and then choose the role from the dropdown menu. You can also choose **Enter an IAM role ARN** and then enter the IAM role's Amazon Resource Name (ARN).

    > **Important**
    >
    > When you add a replication rule to an S3 on Outposts bucket, you must have the `iam:CreateRole` and `iam:PassRole` permissions to be able to create and pass the IAM role that grants S3 on Outposts replication permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

11. All objects in Outposts buckets are encrypted by default. For more information about S3 on Outposts encryption, see Data encryption in S3 on Outposts. Only objects that are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) can be replicated. The replication of objects that are encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) or server-side encryption with customer-provided encryption keys (SSE-C) is not supported.

12. As needed, enable the following additional options while setting the replication rule configuration:

    - If you want to enable S3 on Outposts replication metrics in your replication configuration, select **Replication metrics**. For more information, see Monitoring progress with replication metrics.
    - If you want to enable delete marker replication in your replication configuration, select **Delete marker replication**. For more information, see How delete operations affect replication.
    - If you want to replicate metadata changes made to the replicas back to the source objects, select **Replica modification sync**. For more information, see Replication status if Amazon S3 replica modification sync on Outposts is enabled.

13. To finish, choose **Create rule**.

After you save your rule, you can edit, enable, disable, or delete your rule. To do so, go to the **Management** tab for the source Outposts bucket, scroll down to the **Replication rules** section, choose your rule, and then choose **Edit rule**.

**Using the AWS CLI**

To use the AWS CLI to set up replication when the source and destination Outposts buckets are owned by the same AWS account, you do the following:

- Create source and destination Outposts buckets.
- Enable versioning on both of the buckets.
- Create an IAM role that gives S3 on Outposts permission to replicate objects.
- Add the replication configuration to the source Outposts bucket.

To verify your setup, you test it.

**To set up replication when the source and destination Outposts buckets are owned by the same AWS account**

1. Set a credentials profile for the AWS CLI. In this example, we use the profile name `acctA`. For information about setting credential profiles, see Named profiles in the *AWS Command Line Interface User Guide*.

> **Important**
>
> The profile that you use for this exercise must have the necessary permissions. For example, in the replication configuration, you specify the IAM service role that S3 on Outposts can assume. You can do this only if the profile that you use has the `iam:CreateRole` and `iam:PassRole` permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*. If you use administrator credentials to create a named profile, the named profile will have the necessary permission to perform all the tasks.

2. Create a source bucket and enable versioning on it. The following `create-bucket` command creates a *SOURCE-OUTPOSTS-BUCKET* bucket in the US East (N. Virginia) (`us-east-1`) Region. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control create-bucket --bucket SOURCE-OUTPOSTS-BUCKET --outpost-id SOURCE-OUTPOST-ID --profile acctA --region us-east-1
```

   The following `put-bucket-versioning` command enables versioning on the *SOURCE-OUTPOSTS-BUCKET* bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

3. Create a destination bucket and enable versioning on it. The following `create-bucket` command creates a *DESTINATION-OUTPOSTS-BUCKET* bucket in the US West (Oregon) (`us-west-2`) Region. To use this command, replace the *user input placeholders* with your own information.

> **Note**
>
> To set up a replication configuration when both the source and destination Outposts buckets are in the same AWS account, you use the same named profile. This example uses `acctA`. To test the replication configuration when the buckets are owned by different AWS accounts, you specify different profiles for each bucket.

```
aws s3control create-bucket --bucket DESTINATION-OUTPOSTS-BUCKET --create-bucket-
configuration LocationConstraint=us-west-2 --outpost-id DESTINATION-OUTPOST-ID --
profile acctA --region us-west-2
```

The following `put-bucket-versioning` command enables versioning on the *DESTINATION-OUTPOSTS-BUCKET* bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-
OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

4. Create an IAM service role. Later in the replication configuration, you add this service role to the *SOURCE-OUTPOSTS-BUCKET* bucket. S3 on Outposts assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   a. Create an IAM role.

      i. Copy the following trust policy and save it to a file named `s3-on-outposts-role-trust-policy.json` in the current directory on your local computer. This policy grants S3 on Outposts service principal permissions to assume the service role.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. Run the following command to create the role. Replace the *user input placeholders* with your own information.

```
aws iam create-role --role-name replicationRole --assume-role-policy-
document file://s3-on-outposts-role-trust-policy.json --profile acctA
```

b. Attach a permissions policy to the service role.

i. Copy the following permissions policy and save it to a file named `s3-on-outposts-role-permissions-policy.json` in the current directory on your local computer. This policy grants permissions for various S3 on Outposts bucket and object actions. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
```

```
}
```

ii. Run the following command to create a policy and attach it to the role. Replace the *user input placeholders* with your own information.

```
aws iam put-role-policy --role-name replicationRole --policy-
document file://s3-on-outposts-role-permissions-policy.json --policy-
name replicationRolePolicy --profile acctA
```

5. Add a replication configuration to the *SOURCE-OUTPOSTS-BUCKET* bucket.

a. Although the S3 on Outposts API requires a replication configuration in XML format, the AWS CLI requires that you specify the replication configuration in JSON format. Save the following JSON in a file called `replication.json` to the local directory on your computer. To use this configuration, replace the *user input placeholders* with your own information.

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket":
        "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-
ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT"
      }
    }
  ]
}
```

b. Run the following `put-bucket-replication` command to add the replication configuration to your source Outposts bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-replication --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-
```

```
ID/bucket/SOURCE-OUTPOSTS-BUCKET --replication-configuration file://
replication.json --profile acctA
```

c. To retrieve the replication configuration, use the `get-bucket-replication` command. To use this command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-bucket-replication --account-id 123456789012 --bucket
 arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET --profile acctA
```

6. Test the setup in the Amazon S3 console:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. In the *`SOURCE-OUTPOSTS-BUCKET`* bucket, create a folder named `Tax`.

   c. Add sample objects to the `Tax` folder in the *`SOURCE-OUTPOSTS-BUCKET`* bucket.

   d. In the *`DESTINATION-OUTPOSTS-BUCKET`* bucket, verify the following:

   - S3 on Outposts replicated the objects.

     **Note**

     The amount of time that it takes for S3 on Outposts to replicate an object depends on the size of the object. For information about how to see the status of replication, see Getting replication status information.

   - On the object's **Properties** tab, the **Replication status** is set to **Replica** (identifying this as a replica object).

## Managing your replication

This section describes additional replication configuration options that are available in S3 on Outposts, how to determine the replication status, and how to troubleshot replication. For information about core replication configuration, see Setting up replication.

### Topics

- Monitoring progress with replication metrics

- Getting replication status information
- Troubleshooting replication
- Using EventBridge for S3 Replication on Outposts

### Monitoring progress with replication metrics

S3 Replication on Outposts provides detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor in 5-minute intervals the progress of replication by tracking bytes pending replication, replication latency replication, and operations pending. To assist in troubleshooting any configuration issues, you can also set up Amazon EventBridge to receive notifications about replication failures.

When replication metrics are enabled, S3 Replication on Outposts publishes the following metrics to Amazon CloudWatch:

- **Bytes Pending Replication** – The total number of bytes of objects that are pending replication for a given replication rule.
- **Replication Latency** – The maximum number of seconds by which the replication destination bucket is behind the source bucket for a given replication rule.
- **Operations Pending Replication** – The number of operations that are pending replication for a given replication rule. Operations include objects, delete markers, and tags.

> **Note**
>
> S3 Replication on Outposts metrics are billed at the same rate as CloudWatch custom metrics. For more information, see CloudWatch pricing.

### Getting replication status information

The replication status can help you determine the current state of an object that's being replicated by Amazon S3 on Outposts. The replication status of a source object will return either `PENDING`, `COMPLETED`, or `FAILED`. The replication status of a replica will return `REPLICA`.

### Replication status overview

In a replication scenario, you have a source bucket on which you configure replication and a destination bucket to which S3 on Outposts replicates objects. When you request an object (using

GetObject) or object metadata (using HeadObject) from these buckets, S3 on Outposts returns the x-amz-replication-status header in the response as follows:

- When you request an object from the source bucket, S3 on Outposts returns the x-amz-replication-status header if the object in your request is eligible for replication.

  For example, suppose that you specify the object prefix TaxDocs in your replication configuration to tell S3 on Outposts to replicate only objects with the key name prefix TaxDocs. Any objects that you upload that have this key name prefix—for example, TaxDocs/document1.pdf—will be replicated. For object requests with this key name prefix, S3 on Outposts returns the x-amz-replication-status header with one of the following values for the object's replication status: PENDING, COMPLETED, or FAILED.

  > **Note**
  >
  > If object replication fails after you upload an object, you can't retry replication. You must upload the object again. Objects transition to a FAILED state for issues such as missing replication role permissions or missing bucket permissions. For temporary failures, such as if a bucket or your Outpost is unavailable, the replication status doesn't transition to FAILED, but remains PENDING. After the resource is back online, S3 on Outposts resumes replicating those objects.

- When you request an object from a destination bucket, if the object in your request is a replica that S3 on Outposts created, S3 on Outposts returns the x-amz-replication-status header with the value REPLICA.

> **Note**
>
> Before deleting an object from a source bucket that has replication enabled, check the object's replication status to ensure that the object has been replicated.

### Replication status if Amazon S3 replica modification sync on Outposts is enabled

When your replication rules enable S3 on Outposts replica modification sync, replicas can report statuses other than REPLICA. If metadata changes are in the process of replicating, the x-amz-replication-status header for the replica returns PENDING. If replica modification sync fails to

replicate metadata, the header for the replica returns `FAILED`. If metadata is replicated correctly, the header for the replica returns the value `REPLICA`.

**Troubleshooting replication**

If object replicas don't appear in the destination Amazon S3 on Outposts bucket after you configure replication, use these troubleshooting tips to identify and fix issues.

- The time it takes S3 on Outposts to replicate an object depends on several factors, including the distance between the source and destination Outposts, and the size of the object.

  You can check the source object's replication status. If the object's replication status is `PENDING`, S3 on Outposts hasn't completed the replication. If the object's replication status is `FAILED`, check the replication configuration that you set on the source bucket.
- In the replication configuration on the source bucket, verify the following:
  - The access point Amazon Resource Name (ARN) of the destination bucket is correct.
  - The key name prefix is correct. For example, if you set the configuration to replicate objects with the prefix `Tax`, then only objects with key names such as `Tax/document1` or `Tax/document2` are replicated. An object with the key name `document3` is not replicated.
  - The status is `Enabled`.
- Verify that versioning hasn't been suspended on either bucket. Both the source and destination buckets must have versioning enabled.
- If the destination bucket is owned by another AWS account, verify that the bucket owner has a bucket policy on the destination bucket that allows the source bucket owner to replicate objects. For an example, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.
- If an object replica doesn't appear in the destination bucket, the following issues might prevent replication:
  - S3 on Outposts doesn't replicate an object in a source bucket that is a replica created by another replication configuration. For example, if you set a replication configuration from bucket A to bucket B to bucket C, S3 on Outposts doesn't replicate object replicas in bucket B to bucket C.

    If you want to replicate objects in bucket A to bucket B and bucket C, set multiple bucket destinations in different replication rules for your source bucket replication configuration. For example, create two replication rules on source bucket A, with one rule to replicate to destination bucket B and the other rule to replicate to destination bucket C.

  - A source bucket owner can grant other AWS accounts permission to upload objects. By default, the source bucket owner doesn't have permissions for the objects created by other accounts. The replication configuration replicates only the objects for which the source bucket owner has access permissions. To avoid replication issues, the source bucket owner can grant other AWS accounts permissions to create objects conditionally, requiring explicit access permissions on those objects. For an example policy, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.
- Suppose that in the replication configuration, you add a rule to replicate a subset of objects that have a specific tag. In this case, you must assign the specific tag key and value at the time that the object is created in order for S3 on Outposts to replicate the object. If you first create an object and then add the tag to that existing object, S3 on Outposts doesn't replicate the object.
- Replication fails if the bucket policy denies access to the replication role for any of the following actions:

  Source bucket:

  ```
  "s3-outposts:GetObjectVersionForReplication",
  "s3-outposts:GetObjectVersionTagging"
  ```

  Destination buckets:

  ```
  "s3-outposts:ReplicateObject",
  "s3-outposts:ReplicateDelete",
  "s3-outposts:ReplicateTags"
  ```
- Amazon EventBridge can notify you when objects don't replicate to their destination Outposts. For more information, see Using EventBridge for S3 Replication on Outposts.

### Using EventBridge for S3 Replication on Outposts

Amazon S3 on Outposts is integrated with Amazon EventBridge and uses the `s3-outposts` namespace. EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. For more information, see What is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

To assist in troubleshooting any replication configuration issues, you can set up Amazon EventBridge to receive notifications about replication failure events. EventBridge can notify you in

instances when objects don't replicate to their destination Outposts. For more information about the current state of an object that's being replicated, see Replication status overview.

Whenever certain events happen in your Outposts bucket, S3 on Outposts can send events to EventBridge. Unlike other destinations, you don't need to select which event types that you want to deliver. You can also use EventBridge rules to route events to additional targets. After EventBridge is enabled, S3 on Outposts sends all of the following events to EventBridge.

| Event type | Description | Namespace |
| --- | --- | --- |
| `Operation FailedReplication` | The replication of an object within a replication rule failed. For more information about S3 Replication on Outposts failure reasons, see Using EventBridge to view S3 Replication on Outposts failure reasons. | `s3-outposts` |

### Using EventBridge to view S3 Replication on Outposts failure reasons

The following table lists S3 Replication on Outposts failure reasons. You can configure an EventBridge rule to publish and view the failure reason through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), AWS Lambda, or Amazon CloudWatch Logs. For more information about the permissions that are required to use these resources for EventBridge, see Using resource-based policies for EventBridge.

| Replication failure reason | Description |
| --- | --- |
| `AssumeRoleNotPermitted` | S3 on Outposts can't assume the AWS Identity and Access Management (IAM) role that's specified in the replication configuration. |
| `DstBucketNotFound` | S3 on Outposts can't find the destination bucket that's specified in the replication configuration. |
| `DstBucketUnversioned` | Versioning isn't enabled on the Outposts destination bucket. To replicate objects with |

| Replication failure reason | Description |
| --- | --- |
| | S3 Replication on Outposts, you must enable versioning on the destination bucket. |
| DstDelObjNotPermitted | S3 on Outposts can't replicate deletes to the destination bucket. The s3-outposts:ReplicateDelete permission might be missing for the destination bucket. |
| DstMultipartCompleteNotPermitted | S3 on Outposts can't complete a multipart upload of objects in the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket. |
| DstMultipartInitNotPermitted | S3 on Outposts can't initiate a multipart upload of objects to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket. |
| DstMultipartPartUploadNotPermitted | S3 on Outposts can't upload multipart objects in the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket. |
| DstOutOfCapacity | S3 on Outposts can't replicate to the destination Outpost because the Outpost is out of S3 storage capacity. |
| DstPutObjNotPermitted | S3 on Outposts can't replicate objects to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket. |

| Replication failure reason | Description |
| --- | --- |
| DstPutTaggingNotPermitted | S3 on Outposts can't replicate object tags to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket. |
| DstVersionNotFound | S3 on Outposts can't find the required object version in the destination bucket in order to replicate that object version's metadata. |
| SrcBucketReplicationConfigMissing | S3 on Outposts can't find a replication configuration for the access point that's associated with the source Outposts bucket. |
| SrcGetObjNotPermitted | S3 on Outposts can't access the object in the source bucket for replication. The s3-outposts:GetObjectVersionForReplication permission might be missing for the source bucket. |
| SrcGetTaggingNotPermitted | S3 on Outposts can't access the object tag information from the source bucket. The s3-outposts:GetObjectVersionTagging permission might be missing for the source bucket. |
| SrcHeadObjectNotPermitted | S3 on Outposts can't retrieve object metadata from the source bucket. The s3-outposts:GetObjectVersionForReplication permission might be missing for the source bucket. |
| SrcObjectNotEligible | The object isn't eligible for replication. The object or its object tags don't match the replication configuration. |

For more information about troubleshooting replication, see the following topics:

- Creating an IAM role
- Troubleshooting replication

**Monitoring EventBridge with CloudWatch**

For monitoring, Amazon EventBridge integrates with Amazon CloudWatch. EventBridge automatically sends metrics to CloudWatch every minute. These metrics include the number of events that have been matched by a rule and the number of times a target is invoked by a rule. When a rule runs in EventBridge, all of the targets associated with the rule are invoked. You can monitor your EventBridge behavior through CloudWatch in the following ways.

- You can monitor the available EventBridge metrics for your EventBridge rules from the CloudWatch dashboard. Then, you can use CloudWatch features, such as CloudWatch alarms, to set alarms on certain metrics. If those metrics reach the custom threshold values that you've specified in the alarms, you receive notifications and can take action accordingly.
- You can set Amazon CloudWatch Logs as a target of your EventBridge rule. Then, EventBridge creates log streams and CloudWatch Logs stores the text from the events as log entries. For more information, see EventBridge and CloudWatch Logs.

For more information about debugging EventBridge event delivery and archiving events, see the following topics:

- Event retry policy and using dead-letter queues
- Archiving EventBridge events

## Sharing S3 on Outposts by using AWS RAM

Amazon S3 on Outposts supports sharing S3 capacity across multiple accounts within an organization by using AWS Resource Access Manager (AWS RAM). With S3 on Outposts sharing, you can allow others to create and manage buckets, endpoints, and access points on your Outpost.

This topic demonstrates how to use AWS RAM to share S3 on Outposts and related resources with another AWS account in your AWS organization.

## Prerequisites

- The Outpost owner account has an organization configured in AWS Organizations. For more information, see Creating an organization in the *AWS Organizations User Guide*.
- The organization includes the AWS account that you want to share your S3 on Outposts capacity with. For more information, see Sending invitations to AWS accounts in the *AWS Organizations User Guide*.
- Select one of the following options that you want to share. The second resource (either **Subnets** or **Outposts**) must be selected so that endpoints are also accessible. Endpoints are a networking requirement in order to access data stored in S3 on Outposts.

| Option 1 | Option 2 |
| --- | --- |
| **S3 on Outposts**<br><br>Allows the user to create buckets on your Outposts and access points and to add objects to those buckets. | **S3 on Outposts**<br><br>Allows the user to create buckets on your Outposts and access points and to add objects to those buckets. |
| **Subnets**<br><br>Allows the user to use your virtual private cloud (VPC) and the endpoints that are associated with your subnet. | **Outposts**<br><br>Allows the user to see S3 capacity charts and the AWS Outposts console home page. Also allows users to create subnets on shared Outposts and create endpoints. |

## Procedure

1. Sign in to the AWS Management Console by using the AWS account that owns the Outpost, and then open the AWS RAM console at https://console.aws.amazon.com/ram.
2. Make sure that you have enabled sharing with AWS Organizations in AWS RAM. For information, see Enable resource sharing within AWS Organizations in the *AWS RAM User Guide*.
3. Use either Option 1 or Option 2 in the prerequisites to create a resource share. If you have multiple S3 on Outposts resources, select the Amazon Resource Names (ARNs) of the resources that you want to share. To enable endpoints, share either your subnet or Outpost.

For more information about how to create a resource share, see Create a resource share in the *AWS RAM User Guide*.

4. The AWS account that you shared your resources with should now be able to use S3 on Outposts. Depending on the option that you selected in the prerequisites, provide the following information to the account user:

| Option 1 | Option 2 |
| --- | --- |
| The Outpost ID | The Outpost ID |
| The VPC ID | |
| The subnet ID | |
| The security group ID | |

> **Note**
>
> The user can confirm that the resources have been shared with them by using the AWS RAM console, the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. The user can view their existing resource shares by using the get-resource-shares CLI command.

## Usage examples

After you have shared your S3 on Outposts resources with another account, that account can manage buckets and objects on your Outpost. If you shared the **Subnets** resource, then that account can use the endpoint that you created. The following examples demonstrate how a user can use the AWS CLI to interact with your Outpost after you share these resources.

**Example : Create a bucket**

The following example creates a bucket named *amzn-s3-demo-bucket1* on the Outpost *op-01ac5d28a6a232904*. Before using this command, replace each *user input placeholder* with the appropriate values for your use case.

```
aws s3control create-bucket --bucket amzn-s3-demo-bucket1 --outpost-
id op-01ac5d28a6a232904
```

For more information about this command, see create-bucket in the *AWS CLI Reference*.

**Example : Create an access point**

The following example creates an access point on an Outpost by using the example parameters in the following table. Before using this command, replace these *user input placeholder* values and the AWS Region code with the appropriate values for your use case.

| Parameter | Value |
| --- | --- |
| Account ID | *111122223333* |
| Access point name | *example-outpost-access-point* |
| Outpost ID | *op-01ac5d28a6a232904* |
| Outpost bucket name | *amzn-s3-demo-bucket1* |
| VPC ID | *vpc-1a2b3c4d5e6f7g8h9* |

**Note**

The Account ID parameter must be the AWS account ID of the bucket owner, which is the shared user.

```
aws s3control create-access-point --account-id 111122223333 --name example-outpost-
access-point \
--bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/
bucket/amzn-s3-demo-bucket1 \
--vpc-configuration VpcId=vpc-1a2b3c4d5e6f7g8h9
```

For more information about this command, see create-access-point in the *AWS CLI Reference*.

### Example : Upload an object

The following example uploads the file *my_image.jpg* from the user's local file system to an object named *images/my_image.jpg* through the access point *example-outpost-access-point* on the Outpost *op-01ac5d28a6a232904*, owned by the AWS account *111122223333*. Before using this command, replace these *user input placeholder* values and the AWS Region code with the appropriate values for your use case.

```
aws s3api put-object --bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-point \
--body my_image.jpg --key images/my_image.jpg
```

For more information about this command, see put-object in the *AWS CLI Reference*.

> **Note**
>
> If this operation results in a Resource not found error or is unresponsive, your VPC might not have a shared endpoint.
> To check whether there is a shared endpoint, use the list-shared-endpoints AWS CLI command. If there is no shared endpoint, work with the Outpost owner to create one.
> For more information, see ListSharedEndpoints in the *Amazon Simple Storage Service API Reference*.

### Example : Create an endpoint

The following example creates an endpoint on a shared Outpost. Before using this command, replace the *user input placeholder* values for the Outpost ID, subnet ID, and security group ID with the appropriate values for your use case.

> **Note**
>
> The user can perform this operation only if the resource share includes the **Outposts** resource.

```
aws s3outposts create-endpoint --outposts-id op-01ac5d28a6a232904 --subnet-id XXXXXX --security-group-id XXXXXXX
```

For more information about this command, see create-endpoint in the *AWS CLI Reference*.

## Other AWS services that use S3 on Outposts

Other AWS services that run local to your AWS Outposts can also use your Amazon S3 on Outposts capacity. In Amazon CloudWatch the `S3Outposts` namespace shows detailed metrics for buckets within S3 on Outposts, but these metrics don't include usage for other AWS services. To manage your S3 on Outposts capacity that is consumed by other AWS services, see the information in the following table.

| AWS service | Description | Learn more |
| --- | --- | --- |
| Amazon S3 | All direct S3 on Outposts usage has a matching account and bucket CloudWatch metric. | See metrics |
| Amazon Elastic Block Store (Amazon EBS) | For Amazon EBS on Outposts, you can choose an AWS Outpost as your snapshot destination and store locally in your S3 on Outpost. | Learn more |
| Amazon Relationa l Database Service (Amazon RDS) | You can use Amazon RDS local backups to store your RDS backups locally on your Outpost. | Learn more |

# Monitoring S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about monitoring your Amazon S3 on Outposts storage capacity, see the following topics.

**Topics**



# Managing S3 on Outposts capacity with Amazon CloudWatch metrics

To help manage the fixed S3 capacity on your Outpost, we recommend that you create CloudWatch alerts that tell you when your storage utilization exceeds a certain threshold. For more information about the CloudWatch metrics for S3 on Outposts, see CloudWatch metrics. If there is not enough space to store an object on your Outpost, the API returns an insufficient capacity exemption (ICE). To free up space, you can create CloudWatch alarms that trigger explicit data deletion, or use a lifecycle expiration policy to expire objects. To save data before deletion, you can use AWS DataSync to copy data from your Amazon S3 on Outposts bucket to an S3 bucket in an AWS Region. For more information about using DataSync, see Getting Started with AWS DataSync in the *AWS DataSync User Guide*.

## CloudWatch metrics

The `S3Outposts` namespace includes the following metrics for Amazon S3 on Outposts buckets. You can monitor the total number of S3 on Outposts bytes provisioned, the total free bytes available for objects, and the total size of all objects for a given bucket. Bucket or account-related metrics exist for all direct S3 usage. Indirect S3 usage, such as storing Amazon Elastic Block Store local snapshots or Amazon Relational Database Service backups on an Outpost, consumes S3 capacity, but is not included in bucket or account-related metrics. For more information about Amazon EBS local snapshots, see Amazon EBS local snapshots on Outposts. To see your Amazon EBS cost report, visit https://console.aws.amazon.com/billing/.

> **Note**
>
> S3 on Outposts supports only the following metrics, and no other Amazon S3 metrics. Because S3 on Outposts has a fixed capacity limit, we recommend creating CloudWatch alarms to notify you when your storage utilization exceeds a certain threshold.

| Metric | Description | Time Period | Units | Type |
|---|---|---|---|---|
| `Outpost`<br>`talByte` | The total provisioned capacity in bytes for an Outpost. | 5 minutes | Bytes | S3 on Outposts |
| `Outpostl`<br>`eeBytes` | The count of free bytes available on an Outpost to store customer data. | 5 minutes | Bytes | S3 on Outposts |
| `BucketU`<br>`dBytes` | The total size of all objects for the given bucket. | 5 minutes | Bytes | S3 on Outposts. Direct S3 usage only. |
| `Accountl`<br>`edBytes` | The total size of all objects for the specified Outposts account. | 5 minutes | Bytes | S3 on Outposts. Direct S3 usage only. |
| `BytesPe`<br>`ingRepl`<br>`ation` | The total number of bytes of objects that are pending replication for a given replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts. | 5 minutes | Bytes | Optional. For S3 Replication on Outposts. |
| `Operati`<br>`sPendin`<br>`eplicat`<br>`n` | The total number of operations that are pending replication for a given replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts. | 5 minutes | Counts | Optional. For S3 Replication on Outposts. |
| `Replica`<br>`onLaten` | The current number of seconds delay by which the replication destination bucket | 5 minutes | Seconds | Optional. For S3 Replication on Outposts. |

| Metric | Description | Time Period | Units | Type |
| --- | --- | --- | --- | --- |
| | is behind the source bucket for a given replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts. | | | |

## Receiving S3 on Outposts event notifications by using Amazon CloudWatch Events

You can use CloudWatch Events to create a rule for any Amazon S3 on Outposts API event. When you create a rule, you can choose to get notified through all supported CloudWatch targets, including Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), and AWS Lambda. For more information, see the list of AWS services that can be targets for CloudWatch Events in the *Amazon CloudWatch Events User Guide*. To choose a target service to work with your S3 on Outposts, see Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail in the *Amazon CloudWatch Events User Guide*.

> **Note**
>
> For S3 on Outposts object operations, AWS API call events sent by CloudTrail will match your rules only if you have trails (optionally with event selectors) configured to receive those events. For more information, see Working with CloudTrail log files in the *AWS CloudTrail User Guide*.

### Example

The following is a sample rule for the `DeleteObject` operation. To use this sample rule, replace *`amzn-s3-demo-bucket1`* with the name of your S3 on Outposts bucket.

```
{
  "source": [
    "aws.s3-outposts"
```

```
  ],
  "detail-type": [
    "AWS API call through CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3-outposts.amazonaws.com"
    ],
    "eventName": [
      "DeleteObject"
    ],
    "requestParameters": {
      "bucketName": [
        "amzn-s3-demo-bucket1"
      ]
    }
  }
}
```

## Monitoring S3 on Outposts with AWS CloudTrail logs

Amazon S3 on Outposts is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in S3 on Outposts. You can use AWS CloudTrail to get information about S3 on Outposts bucket-level and object-level requests to audit and log your S3 on Outposts event activity. To enable CloudTrail data events for all your Outposts buckets or for a list of specific Outposts buckets, you must create a trail manually in CloudTrail. For more information about CloudTrail log file entries, see S3 on Outposts log file entries.

**Note**

- It's a best practice to create a lifecycle policy for your AWS CloudTrail data event Outposts bucket. Configure the lifecycle policy to periodically remove log files after the period of time that you need to audit them. Doing so reduces the amount of data that Amazon Athena analyzes for each query. For more information, see Setting a lifecycle configuration on a bucket.
- For examples of how to query CloudTrail logs, see the *AWS Big Data Blog* post Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena.

## Enable CloudTrail logging for objects in an S3 on Outposts bucket

You can use the Amazon S3 console to configure an AWS CloudTrail trail to log data events for objects in an Amazon S3 on Outposts bucket. CloudTrail supports logging S3 on Outposts object-level API operations such as `GetObject`, `DeleteObject`, and `PutObject`. These events are called *data events*.

By default, CloudTrail trails don't log data events. However, you can configure trails to log data events for S3 on Outposts buckets that you specify, or to log data events for all the S3 on Outposts buckets in your AWS account. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

CloudTrail does not populate data events in the CloudTrail event history. Additionally, not all S3 on Outposts bucket–level API operations are populated in the CloudTrail event history. For more information about how to query CloudTrail logs, see Using Amazon CloudWatch Logs filter patterns and Amazon Athena to query CloudTrail logs on the AWS Knowledge Center.

To configure a trail to log data events for an S3 on Outposts bucket, you can use either the AWS CloudTrail console or the Amazon S3 console. If you are configuring a trail to log data events for all the S3 on Outposts buckets in your AWS account, it's easier to use the CloudTrail console. For information about using the CloudTrail console to configure a trail to log S3 on Outposts data events, see Data events in the *AWS CloudTrail User Guide*.

**Important**

Additional charges apply for data events. For more information, see AWS CloudTrail pricing.

The following procedure shows how to use the Amazon S3 console to configure a CloudTrail trail to log data events for an S3 on Outposts bucket.

**Note**

The AWS account that creates the bucket owns it and is the only one that can configure S3 on Outposts data events to be sent to AWS CloudTrail.

**To enable CloudTrail data events logging for objects in an S3 on Outposts bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the name of the Outposts bucket whose data events you want to log by using CloudTrail.
4. Choose **Properties**.
5. In the **AWS CloudTrail data events** section, choose **Configure in CloudTrail**.

   The AWS CloudTrail console opens.

   You can create a new CloudTrail trail or reuse an existing trail and configure S3 on Outposts data events to be logged in your trail.
6. On the CloudTrail console **Dashboard** page, choose **Create trail**.
7. On the **Step 1 Choose trail attributes** page, provide a name for the trail, choose an S3 bucket to store trail logs, specify any other settings that you want, and then choose **Next**.
8. On the **Step 2 Choose log events** page, under **Event type**, choose **Data events**.

   For **Data event type**, choose **S3 Outposts**. Choose **Next**.

   **Note**

   - When you create a trail and configure data event logging for S3 on Outposts, you must specify the data event type correctly.
     - If you use the CloudTrail console, choose **S3 Outposts** for **Data event type**. For information about how to create trails in the CloudTrail console, see Creating and updating a trail with the console in the *AWS CloudTrail User Guide*. For information about how to configure S3 on Outposts data event logging in the CloudTrail console, see Logging data events for Amazon S3 Objects in the *AWS CloudTrail User Guide*.
     - If you use the AWS Command Line Interface (AWS CLI) or the AWS SDKs, set the `resources.type` field to `AWS::S3Outposts::Object`. For more information about how to log S3 on Outposts data events with the AWS CLI, see Log S3 on Outposts events in the *AWS CloudTrail User Guide*.

- If you use the CloudTrail console or the Amazon S3 console to configure a trail to log data events for an S3 on Outposts bucket, the Amazon S3 console shows that object-level logging is enabled for the bucket.

9. On the **Step 3 Review and create** page, review the trail attributes and the log events that you configured. Then, choose **Create trail**.

**To disable CloudTrail data events logging for objects in an S3 on Outposts bucket**

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the left navigation pane, choose **Trails**.
3. Choose the name of the trail that you created to log events for your S3 on Outposts bucket.
4. On the details page for your trail, choose **Stop logging** in the upper-right corner.
5. In the dialog box that appears, choose **Stop logging**.

# Developing with Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following topics provide information about developing with S3 on Outposts.

**Topics**

- Amazon S3 on Outposts API operations
- Configure the S3 control client for S3 on Outposts by using the SDK for Java
- Making requests to S3 on Outposts over IPv6

# Amazon S3 on Outposts API operations

This topic lists the Amazon S3, Amazon S3 Control, and Amazon S3 on Outposts API operations that you can use with Amazon S3 on Outposts.

**Topics**



## Amazon S3 API operations for managing objects

S3 on Outposts is designed to use the same object API operations as Amazon S3. You must use access points to access any object in an Outpost bucket. When you use an object API operation with S3 on Outposts, you provide either the Outposts access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

Amazon S3 on Outposts supports the following Amazon S3 API operations:

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2

- ListObjectVersions
- ListParts
- PutObject
- PutObjectTagging
- UploadPart
- UploadPartCopy

## Amazon S3 Control API operations for managing buckets

S3 on Outposts supports the following Amazon S3 Control API operations for working with buckets.

- CreateAccessPoint
- CreateBucket
- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteBucket
- DeleteBucketLifecycleConfiguration
- DeleteBucketPolicy
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccessPoint
- GetAccessPointPolicy
- GetBucket
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- GetBucketReplication
- GetBucketTagging
- GetBucketVersioning
- ListAccessPoints
- ListRegionalBuckets
- PutAccessPointPolicy

- PutBucketLifecycleConfiguration
- PutBucketPolicy
- PutBucketReplication
- PutBucketTagging
- PutBucketVersioning

### S3 on Outposts API operations for managing Outposts

S3 on Outposts supports the following Amazon S3 on Outposts API operations for managing endpoints.

- CreateEndpoint
- DeleteEndpoint
- ListEndpoints
- ListOutpostsWithS3
- ListSharedEndpoints

## Configure the S3 control client for S3 on Outposts by using the SDK for Java

The following example configures the Amazon S3 control client for Amazon S3 on Outposts by using the AWS SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;

public AWSS3Control createS3ControlClient() {

    String accessKey = AWSAccessKey;
    String secretKey = SecretAccessKey;
    BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);

    return AWSS3ControlClient.builder().enableUseArnRegion()
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
```

```
            .build();

}
```

## Making requests to S3 on Outposts over IPv6

Amazon S3 on Outposts and S3 on Outposts dual-stack endpoints support requests to S3 on Outposts buckets using either the IPv6 or IPv4 protocol. With IPv6 support for S3 on Outposts, you can access and operate your buckets and control plane resources through S3 on Outposts APIs over IPv6 networks.

> **Note**
>
> S3 on Outposts object actions (such as `PutObject` or `GetObject`) aren't supported over IPv6 networks.

There are no additional charges for accessing S3 on Outposts over IPv6 networks. For more information about S3 on Outposts, see S3 on Outposts pricing.

**Topics**

- Getting started with IPv6
- Using dual-stack endpoints to make requests over an IPv6 network
- Using IPv6 addresses in IAM policies
- Testing IP address compatibility
- Using IPv6 with AWS PrivateLink
- Using S3 on Outposts dual-stack endpoints

### Getting started with IPv6

To make a request to an S3 on Outposts bucket over IPv6, you must use a dual-stack endpoint. The next section describes how to make requests over IPv6 by using dual-stack endpoints.

The following are important considerations before trying to access an S3 on Outposts bucket over IPv6:

- The client and the network accessing the bucket must be enabled to use IPv6.

- Both virtual hosted-style and path style requests are supported for IPv6 access. For more information, see Using S3 on Outposts dual-stack endpoints.
- If you use source IP address filtering in your AWS Identity and Access Management (IAM) user or S3 on Outposts bucket policies, you must update the policies to include IPv6 address ranges.

> **Note**
>
> This requirement only applies to S3 on Outposts bucket operations and control plane resources across IPv6 networks. Amazon S3 on Outposts object actions aren't supported across IPv6 networks.

- When using IPv6, server access log files output IP addresses in an IPv6 format. You must update existing tools, scripts, and software that you use to parse S3 on Outposts log files, so that they can parse the IPv6 formatted remote IP addresses. The updated tools, scripts, and software will then correctly parse the IPv6 formatted remote IP addresses.

## Using dual-stack endpoints to make requests over an IPv6 network

To make requests with S3 on Outposts API calls over IPv6, you can use dual-stack endpoints via AWS CLI or AWS SDK. The Amazon S3 control API operations and S3 on Outposts API operations work the same way whether you're accessing S3 on Outposts over an IPv6 protocol or IPv4 protocol. However, be aware that S3 on Outposts object actions (such as `PutObject` or `GetObject`) aren't supported over IPv6 networks.

When using the AWS Command Line Interface (AWS CLI) and AWS SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the S3 on Outposts endpoint in the configuration file.

You can use a dual-stack endpoint to access an S3 on Outposts bucket over IPv6 from any of the following:

- The AWS CLI, see Using dual-stack endpoints from the AWS CLI.
- The AWS SDKs, see Using S3 on Outposts dual-stack endpoints from the AWS SDKs.

## Using IPv6 addresses in IAM policies

Before trying to access an S3 on Outposts bucket using an IPv6 protocol, make sure that IAM users or S3 on Outposts bucket policies used for IP address filtering are updated to include IPv6 address

ranges. If IP address filtering policies aren't updated to handle IPv6 addresses, you can lose access to an S3 on Outposts bucket while trying to use the IPv6 protocol.

IAM policies that filter IP addresses use IP address condition operators. The following S3 on Outposts bucket policy identifies the 54.240.143.* IP range of allowed IPv4 addresses by using IP address condition operators. Any IP addresses outside of this range will be denied access to the S3 on Outposts bucket (DOC-EXAMPLE-BUCKET). Since all IPv6 addresses are outside of the allowed range, this policy prevents IPv6 addresses from being able to access DOC-EXAMPLE-BUCKET.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3outposts:*",
      "Resource": "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

You can modify the S3 on Outposts bucket policy's Condition element to allow both IPv4 (54.240.143.0/24) and IPv6 (2001:DB8:1234:5678::/64) address ranges as shown in the following example. You can use the same type of Condition block shown in the example to update both your IAM user and bucket policies.

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

Before using IPv6 you must update all relevant IAM user and bucket policies that use IP address filtering to allow IPv6 address ranges. We recommend that you update your IAM policies with your organization's IPv6 address ranges in addition to your existing IPv4 address ranges. For an example of a bucket policy that allows access over both IPv6 and IPv4, see Restrict access to specific IP addresses.

You can review your IAM user policies using the IAM console at https://console.aws.amazon.com/iam/. For more information about IAM, see the IAM User Guide. For information about editing S3 on Outposts bucket policies, see Adding or editing a bucket policy for an Amazon S3 on Outposts bucket.

### Testing IP address compatibility

If you're using a Linux or Unix instance, or macOS X platform, you can test your access to a dual-stack endpoint over IPv6. For example, to test the connection to Amazon S3 on Outposts endpoints over IPv6, use the `dig` command:

```
dig s3-outposts.us-west-2.api.aws AAAA +short
```

If your dual-stack endpoint over an IPv6 network is properly set up, the `dig` command returns the connected IPv6 addresses. For example:

```
dig s3-outposts.us-west-2.api.aws AAAA +short

2600:1f14:2588:4800:b3a9:1460:159f:ebce

2600:1f14:2588:4802:6df6:c1fd:ef8a:fc76

2600:1f14:2588:4801:d802:8ccf:4e04:817
```

### Using IPv6 with AWS PrivateLink

S3 on Outposts supports the IPv6 protocol for AWS PrivateLink services and endpoints. With AWS PrivateLink support for the IPv6 protocol, you can connect to service endpoints within your VPC over IPv6 networks, from either on-premises or other private connections. The IPv6 support for AWS PrivateLink for S3 on Outposts also allows you to integrate AWS PrivateLink with dual-stack endpoints. For steps on how to enable IPv6 for AWS PrivateLink, see Expedite your IPv6 adoption with AWS PrivateLink services and endpoints.

> **Note**
>
> To update the supported IP address type from IPv4 to IPv6, see Modify the supported IP address type in the *AWS PrivateLink User Guide*.

### Using IPv6 with AWS PrivateLink

If you're using AWS PrivateLink with IPv6, you must create an IPv6 or dual-stack VPC interface endpoint. For general steps on how to create a VPC endpoint using the AWS Management Console, see Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink User Guide*.

AWS Management Console

Use the following procedure to create an interface VPC endpoint that connects to S3 on Outposts.

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. For **Service category**, choose **AWS services**.
5. For **Service name**, choose the S3 on Outposts service (**com.amazonaws.us-east-1.s3-outposts**).
6. For VPC, choose the VPC from which you'll access S3 on Outposts.
7. For **Subnets**, choose one subnet per Availability Zone from which you'll access S3 on Outposts. You can't select multiple subnets from the same Availability Zone. For each subnet that you select, a new endpoint network interface is created. By default, IP addresses from the subnet IP address ranges are assigned to the endpoint network interfaces. To designate an IP address for an endpoint network interface, choose **Designate IP addresses** and enter an IPv6 address from the subnet address range.
8. For **IP address type**, choose **Dualstack**. Assign both IPv4 and IPv6 addresses to your endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges.
9. For **Security groups**, choose the security groups to associate with the endpoint network interfaces for the VPC endpoint. By default, the default security group is associated with the VPC.

10. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the permissions that principals have toperform actions on resources over the VPC endpoint. This option is available only if the service supports VPC endpoint policies. For more information, see Endpoint policies.
11. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
12. Choose **Create endpoint**.

#### Example – S3 on Outposts bucket policy

To allow S3 on Outposts to interact with your VPC endpoints, you can then update your S3 on Outposts policy like this:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3-outposts:*",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

AWS CLI

> **Note**
>
> To enable the IPv6 network on your VPC endpoint, you must have `IPv6` set for the `SupportedIpAddressType` filter for S3 on Outposts.

The following example uses the `create-vpc-endpoint` command to create a new dual-stack interface endpoint.

```
aws ec2 create-vpc-endpoint \
--vpc-id vpc-12345678 \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.us-east-1.s3-outposts \
--subnet-id subnet-12345678 \
```

```
--security-group-id sg-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

Depending on the AWS PrivateLink service configuration, newly created endpoint connections might need to be accepted by the VPC endpoint service provider before they can be used. For more information, see Accept and reject endpoint connection requests in the *AWS PrivateLink User Guide*.

The following example uses the `modify-vpc-endpoint` command to update the IPv-only VPC endpoint to a dual-stack endpoint. The dual-stack endpoint allows access to both the IPv4 and IPv6 networks.

```
aws ec2 modify-vpc-endpoint \
--vpc-endpoint-id vpce-12345678 \
--add-subnet-ids subnet-12345678 \
--remove-subnet-ids subnet-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

For more information about how to enable the IPv6 network for AWS PrivateLink, see Expedite your IPv6 adoption with AWS PrivateLink services and endpoints.

## Using S3 on Outposts dual-stack endpoints

S3 on Outposts dual-stack endpoints support requests to S3 on Outposts buckets over IPv6 and IPv4. This section describes how to use S3 on Outposts dual-stack endpoints.

**Topics**

- S3 on Outposts dual-stack endpoints
- Using dual-stack endpoints from the AWS CLI
- Using S3 on Outposts dual-stack endpoints from the AWS SDKs

### S3 on Outposts dual-stack endpoints

When you make a request to a dual-stack endpoint, the S3 on Outposts bucket URL resolves to an IPv6 or an IPv4 address. For more information about accessing an S3 on Outposts bucket over IPv6, see Making requests to S3 on Outposts over IPv6.

To access an S3 on Outposts bucket through a dual-stack endpoint, use a path-style endpoint name. S3 on Outposts supports only Regional dual-stack endpoint names, which means that you must specify the Region as part of the name.

For a dual-stack path-style FIPs endpoint, use the following naming convention:

```
s3-outposts-fips.region.api.aws
```

For dual-stack non-FIPS endpoint, use the following naming convention:

```
s3-outposts.region.api.aws
```

> **Note**
>
> Virtual hosted-style endpoint names aren't supported in S3 on Outposts.

### Using dual-stack endpoints from the AWS CLI

This section provides examples of AWS CLI commands used to make requests to a dual-stack endpoint. For instructions on setting up the AWS CLI, see Getting started by using the AWS CLI and SDK for Java.

You set the configuration value `use_dualstack_endpoint` to `true` in a profile in your AWS Config file to direct all Amazon S3 requests made by the `s3` and `s3api` AWS CLI commands to the dual-stack endpoint for the specified Region. You specify the Region in the configuration file or in a command using the `--region` option.

When using dual-stack endpoints with the AWS CLI, only `path` addressing style is supported. The addressing style, set in the configuration file, determines whether the bucket name is in the hostname or in the URL. For more information, see s3outposts in the *AWS CLI User Guide*.

To use a dual-stack endpoint via the AWS CLI, use the `--endpoint-url` parameter with the `http://s3.dualstack.region.amazonaws.com` or `https://s3-outposts-fips.region.api.aws` endpoint for any `s3control` or `s3outposts` commands.

For example:

```
$  aws s3control list-regional-buckets --endpoint-url https://s3-
outposts.region.api.aws
```

### Using S3 on Outposts dual-stack endpoints from the AWS SDKs

This section provides examples of how to access a dual-stack endpoint by using the AWS SDKs.

### AWS SDK for Java 2.x dual-stack endpoint example

The following examples show how to use the `S3ControlClient` and `S3OutpostsClient` classes to enable dual-stack endpoints when creating an S3 on Outposts client using the AWS SDK for Java 2.x. For instructions on creating and testing a working Java example for Amazon S3 on Outposts, see Getting started by using the AWS CLI and SDK for Java.

### Example – Create an `S3ControlClient` class with dual-stack endpoints enabled

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsRequest;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsResponse;
import software.amazon.awssdk.services.s3control.model.S3ControlException;


public class DualStackEndpointsExample1 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");
        String accountId = "111122223333";
        String navyId = "9876543210";

        try {
            // Create an S3ControlClient with dual-stack endpoints enabled.
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                                                             .region(clientRegion)
                                                             .dualstackEnabled(true)
                                                             .build();
            ListRegionalBucketsRequest listRegionalBucketsRequest =
 ListRegionalBucketsRequest.builder()

       .accountId(accountId)

       .outpostId(navyId)

       .build();
```

```
            ListRegionalBucketsResponse listBuckets =
 s3ControlClient.listRegionalBuckets(listRegionalBucketsRequest);
            System.out.printf("ListRegionalBuckets Response: %s%n",
 listBuckets.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3ControlException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

### Example – Create an `S3OutpostsClient` with dual-stack endpoints enabled

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3outposts.S3OutpostsClient;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsRequest;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsResponse;
import software.amazon.awssdk.services.s3outposts.model.S3OutpostsException;


public class DualStackEndpointsExample2 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");

        try {
            // Create an S3OutpostsClient with dual-stack endpoints enabled.
            S3OutpostsClient s3OutpostsClient = S3OutpostsClient.builder()
                                                                .region(clientRegion)
```

```
                                                                .dualstackEnabled(true)
                                                                .build();
            ListEndpointsRequest listEndpointsRequest =
 ListEndpointsRequest.builder().build();

            ListEndpointsResponse listEndpoints =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
            System.out.printf("ListEndpoints Response: %s%n",
 listEndpoints.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3OutpostsException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

If you're using the AWS SDK for Java 2.x on Windows, you might have to set the following Java virtual machine (JVM) property:

```
java.net.preferIPv6Addresses=true
```

# Code examples for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon S3 with an AWS software development kit (SDK).

*Actions* are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

*Cross-service examples* are sample applications that work across multiple AWS services.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

**Get started**

## Hello Amazon S3

The following code examples show how to get started using Amazon S3.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Code for the CMakeLists.txt CMake file.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)
```

```
# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code for the hello_s3.cpp source file.

```
#include <aws/core/Aws.h>
```

```
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- For API details, see ListBuckets in *AWS SDK for C++ API Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
```

```
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- For API details, see ListBuckets in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListBuckets in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

export const helloS3 = async () => {
  const command = new ListBucketsCommand({});

  const { Buckets } = await client.send(command);
  console.log("Buckets: ");
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
  return Buckets;
};
```

- For API details, see ListBuckets in *AWS SDK for JavaScript API Reference*.

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
use Aws\S3\S3Client;
```

```
$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- For API details, see ListBuckets in *AWS SDK for PHP API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource("s3")
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")


if __name__ == "__main__":
    hello_s3()
```

- For API details, see ListBuckets in *AWS SDK for Python (Boto3) API Reference*.

Ruby

### SDK for Ruby

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
```

```
  manager.list_buckets
end
```

- For API details, see ListBuckets in *AWS SDK for Ruby API Reference*.

### Code examples

- Actions for Amazon S3 using AWS SDKs
  - Use AbortMultipartUpload with an AWS SDK or CLI
  - Use AbortMultipartUploads with an AWS SDK or CLI
  - Use CompleteMultipartUpload with an AWS SDK or CLI
  - Use CopyObject with an AWS SDK or CLI
  - Use CreateBucket with an AWS SDK or CLI
  - Use CreateMultiRegionAccessPoint with an AWS SDK or CLI
  - Use CreateMultipartUpload with an AWS SDK or CLI
  - Use DeleteBucket with an AWS SDK or CLI
  - Use DeleteBucketAnalyticsConfiguration with an AWS SDK or CLI
  - Use DeleteBucketCors with an AWS SDK or CLI
  - Use DeleteBucketEncryption with an AWS SDK or CLI
  - Use DeleteBucketInventoryConfiguration with an AWS SDK or CLI
  - Use DeleteBucketLifecycle with an AWS SDK or CLI
  - Use DeleteBucketMetricsConfiguration with an AWS SDK or CLI
  - Use DeleteBucketPolicy with an AWS SDK or CLI
  - Use DeleteBucketReplication with an AWS SDK or CLI
  - Use DeleteBucketTagging with an AWS SDK or CLI
  - Use DeleteBucketWebsite with an AWS SDK or CLI
  - Use DeleteObject with an AWS SDK or CLI
  - Use DeleteObjectTagging with an AWS SDK or CLI
  - Use DeleteObjects with an AWS SDK or CLI
  - Use DeletePublicAccessBlock with an AWS SDK or CLI
  - Use GetBucketAccelerateConfiguration with an AWS SDK or CLI

- Use GetBucketAcl with an AWS SDK or CLI
- Use GetBucketAnalyticsConfiguration with an AWS SDK or CLI
- Use GetBucketCors with an AWS SDK or CLI
- Use GetBucketEncryption with an AWS SDK or CLI
- Use GetBucketInventoryConfiguration with an AWS SDK or CLI
- Use GetBucketLifecycleConfiguration with an AWS SDK or CLI
- Use GetBucketLocation with an AWS SDK or CLI
- Use GetBucketLogging with an AWS SDK or CLI
- Use GetBucketMetricsConfiguration with an AWS SDK or CLI
- Use GetBucketNotification with an AWS SDK or CLI
- Use GetBucketPolicy with an AWS SDK or CLI
- Use GetBucketPolicyStatus with an AWS SDK or CLI
- Use GetBucketReplication with an AWS SDK or CLI
- Use GetBucketRequestPayment with an AWS SDK or CLI
- Use GetBucketTagging with an AWS SDK or CLI
- Use GetBucketVersioning with an AWS SDK or CLI
- Use GetBucketWebsite with an AWS SDK or CLI
- Use GetObject with an AWS SDK or CLI
- Use GetObjectAcl with an AWS SDK or CLI
- Use GetObjectAttributes with an AWS SDK or CLI
- Use GetObjectLegalHold with an AWS SDK or CLI
- Use GetObjectLockConfiguration with an AWS SDK or CLI
- Use GetObjectRetention with an AWS SDK or CLI
- Use GetObjectTagging with an AWS SDK or CLI
- Use GetPublicAccessBlock with an AWS SDK or CLI
- Use HeadBucket with an AWS SDK or CLI
- Use HeadObject with an AWS SDK or CLI
- Use ListBucketAnalyticsConfigurations with an AWS SDK or CLI
- Use ListBucketInventoryConfigurations with an AWS SDK or CLI
- Use ListBuckets with an AWS SDK or CLI

- Use ListMultipartUploads with an AWS SDK or CLI
- Use ListObjectVersions with an AWS SDK or CLI
- Use ListObjects with an AWS SDK or CLI
- Use ListObjectsV2 with an AWS SDK or CLI
- Use PutBucketAccelerateConfiguration with an AWS SDK or CLI
- Use PutBucketAcl with an AWS SDK or CLI
- Use PutBucketCors with an AWS SDK or CLI
- Use PutBucketEncryption with an AWS SDK or CLI
- Use PutBucketLifecycleConfiguration with an AWS SDK or CLI
- Use PutBucketLogging with an AWS SDK or CLI
- Use PutBucketNotification with an AWS SDK or CLI
- Use PutBucketNotificationConfiguration with an AWS SDK or CLI
- Use PutBucketPolicy with an AWS SDK or CLI
- Use PutBucketReplication with an AWS SDK or CLI
- Use PutBucketRequestPayment with an AWS SDK or CLI
- Use PutBucketTagging with an AWS SDK or CLI
- Use PutBucketVersioning with an AWS SDK or CLI
- Use PutBucketWebsite with an AWS SDK or CLI
- Use PutObject with an AWS SDK or CLI
- Use PutObjectAcl with an AWS SDK or CLI
- Use PutObjectLegalHold with an AWS SDK or CLI
- Use PutObjectLockConfiguration with an AWS SDK or CLI
- Use PutObjectRetention with an AWS SDK or CLI
- Use RestoreObject with an AWS SDK or CLI
- Use SelectObjectContent with an AWS SDK or CLI
- Use UploadPart with an AWS SDK or CLI

- Scenarios for Amazon S3 using AWS SDKs
  - Create a presigned URL for Amazon S3 using an AWS SDK
  - A web page that lists Amazon S3 objects using an AWS SDK
  - Delete incomplete multipart uploads to Amazon S3 using an AWS SDK

- Download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory
- Get an Amazon S3 object from a Multi-Region Access Point by using an AWS SDK
- Get an object from an Amazon S3 bucket using an AWS SDK, specifying an If-Modified-Since header
- Get started with Amazon S3 buckets and objects using an AWS SDK
- Get started with encryption for Amazon S3 objects using an AWS SDK
- Get started with tags for Amazon S3 objects using an AWS SDK
- Get the legal hold configuration of an Amazon S3 object using an AWS SDK
- Work with Amazon S3 object lock features using an AWS SDK
- Manage access control lists (ACLs) for Amazon S3 buckets using an AWS SDK
- Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK
- Parse Amazon S3 URIs using an AWS SDK
- Perform a multipart copy of an Amazon S3 object using an AWS SDK
- Perform a multipart upload of an Amazon S3 object using an AWS SDK
- Receive and process Amazon S3 event notifications by using an AWS SDK.
- Send S3 event notifications to Amazon EventBridge using an AWS SDK
- Track an Amazon S3 object upload or download using an AWS SDK
- Example approaches for unit and integration testing with an AWS SDK
- Recursively upload a local directory to an Amazon Simple Storage Service (Amazon S3) bucket
- Upload or download large files to and from Amazon S3 using an AWS SDK
- Upload a stream of unknown size to an Amazon S3 object using an AWS SDK
- Use checksums to work with an Amazon S3 object using an AWS SDK
- Work with Amazon S3 object integrity features using an AWS SDK
- Work with Amazon S3 versioned objects using an AWS SDK

- Serverless examples for Amazon S3 using AWS SDKs
  - Invoke a Lambda function from an Amazon S3 trigger
- Cross-service examples for Amazon S3 using AWS SDKs
  - Build an Amazon Transcribe app
  - Convert text to speech and back to text using an AWS SDK
  - Create a photo asset management application that lets users manage photos using labels

- Create an Amazon Textract explorer application
- Detect PPE in images with Amazon Rekognition using an AWS SDK
- Detect entities in text extracted from an image using an AWS SDK
- Detect faces in an image using an AWS SDK
- Detect objects in images with Amazon Rekognition using an AWS SDK
- Detect people and objects in a video with Amazon Rekognition using an AWS SDK
- Save EXIF and other image information using an AWS SDK
- Transform data for your application with S3 Object Lambda

## Actions for Amazon S3 using AWS SDKs

The following code examples demonstrate how to perform individual Amazon S3 actions with AWS SDKs. These excerpts call the Amazon S3 API and are code excerpts from larger programs that must be run in context. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the Amazon Simple Storage Service (Amazon S3) API Reference.

**Examples**

- Use AbortMultipartUpload with an AWS SDK or CLI
- Use AbortMultipartUploads with an AWS SDK or CLI
- Use CompleteMultipartUpload with an AWS SDK or CLI
- Use CopyObject with an AWS SDK or CLI
- Use CreateBucket with an AWS SDK or CLI
- Use CreateMultiRegionAccessPoint with an AWS SDK or CLI
- Use CreateMultipartUpload with an AWS SDK or CLI
- Use DeleteBucket with an AWS SDK or CLI
- Use DeleteBucketAnalyticsConfiguration with an AWS SDK or CLI
- Use DeleteBucketCors with an AWS SDK or CLI
- Use DeleteBucketEncryption with an AWS SDK or CLI
- Use DeleteBucketInventoryConfiguration with an AWS SDK or CLI
- Use DeleteBucketLifecycle with an AWS SDK or CLI

- Use DeleteBucketMetricsConfiguration with an AWS SDK or CLI
- Use DeleteBucketPolicy with an AWS SDK or CLI
- Use DeleteBucketReplication with an AWS SDK or CLI
- Use DeleteBucketTagging with an AWS SDK or CLI
- Use DeleteBucketWebsite with an AWS SDK or CLI
- Use DeleteObject with an AWS SDK or CLI
- Use DeleteObjectTagging with an AWS SDK or CLI
- Use DeleteObjects with an AWS SDK or CLI
- Use DeletePublicAccessBlock with an AWS SDK or CLI
- Use GetBucketAccelerateConfiguration with an AWS SDK or CLI
- Use GetBucketAcl with an AWS SDK or CLI
- Use GetBucketAnalyticsConfiguration with an AWS SDK or CLI
- Use GetBucketCors with an AWS SDK or CLI
- Use GetBucketEncryption with an AWS SDK or CLI
- Use GetBucketInventoryConfiguration with an AWS SDK or CLI
- Use GetBucketLifecycleConfiguration with an AWS SDK or CLI
- Use GetBucketLocation with an AWS SDK or CLI
- Use GetBucketLogging with an AWS SDK or CLI
- Use GetBucketMetricsConfiguration with an AWS SDK or CLI
- Use GetBucketNotification with an AWS SDK or CLI
- Use GetBucketPolicy with an AWS SDK or CLI
- Use GetBucketPolicyStatus with an AWS SDK or CLI
- Use GetBucketReplication with an AWS SDK or CLI
- Use GetBucketRequestPayment with an AWS SDK or CLI
- Use GetBucketTagging with an AWS SDK or CLI
- Use GetBucketVersioning with an AWS SDK or CLI
- Use GetBucketWebsite with an AWS SDK or CLI
- Use GetObject with an AWS SDK or CLI
- Use GetObjectAcl with an AWS SDK or CLI
- Use GetObjectAttributes with an AWS SDK or CLI

- Use GetObjectLegalHold with an AWS SDK or CLI
- Use GetObjectLockConfiguration with an AWS SDK or CLI
- Use GetObjectRetention with an AWS SDK or CLI
- Use GetObjectTagging with an AWS SDK or CLI
- Use GetPublicAccessBlock with an AWS SDK or CLI
- Use HeadBucket with an AWS SDK or CLI
- Use HeadObject with an AWS SDK or CLI
- Use ListBucketAnalyticsConfigurations with an AWS SDK or CLI
- Use ListBucketInventoryConfigurations with an AWS SDK or CLI
- Use ListBuckets with an AWS SDK or CLI
- Use ListMultipartUploads with an AWS SDK or CLI
- Use ListObjectVersions with an AWS SDK or CLI
- Use ListObjects with an AWS SDK or CLI
- Use ListObjectsV2 with an AWS SDK or CLI
- Use PutBucketAccelerateConfiguration with an AWS SDK or CLI
- Use PutBucketAcl with an AWS SDK or CLI
- Use PutBucketCors with an AWS SDK or CLI
- Use PutBucketEncryption with an AWS SDK or CLI
- Use PutBucketLifecycleConfiguration with an AWS SDK or CLI
- Use PutBucketLogging with an AWS SDK or CLI
- Use PutBucketNotification with an AWS SDK or CLI
- Use PutBucketNotificationConfiguration with an AWS SDK or CLI
- Use PutBucketPolicy with an AWS SDK or CLI
- Use PutBucketReplication with an AWS SDK or CLI
- Use PutBucketRequestPayment with an AWS SDK or CLI
- Use PutBucketTagging with an AWS SDK or CLI
- Use PutBucketVersioning with an AWS SDK or CLI
- Use PutBucketWebsite with an AWS SDK or CLI
- Use PutObject with an AWS SDK or CLI
- Use PutObjectAcl with an AWS SDK or CLI

- Use PutObjectLegalHold with an AWS SDK or CLI
- Use PutObjectLockConfiguration with an AWS SDK or CLI
- Use PutObjectRetention with an AWS SDK or CLI
- Use RestoreObject with an AWS SDK or CLI
- Use SelectObjectContent with an AWS SDK or CLI
- Use UploadPart with an AWS SDK or CLI

## Use `AbortMultipartUpload` with an AWS SDK or CLI

The following code examples show how to use `AbortMultipartUpload`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Delete incomplete multipart uploads
- Work with Amazon S3 object integrity

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
```

```
                                    const Aws::String &uploadID,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see AbortMultipartUpload in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

**To abort the specified multipart upload**

The following `abort-multipart-upload` command aborts a multipart upload for the key `multipart/01` in the bucket `my-bucket`.

```
aws s3api abort-multipart-upload \
    --bucket my-bucket \
    --key multipart/01 \
    --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

The upload ID required by this command is output by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`.

- For API details, see AbortMultipartUpload in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command aborts multipart uploads created earlier than 5 days ago.**

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

**Example 2: This command aborts multipart uploads created earlier than January 2nd, 2014.**

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

**Example 3: This command aborts multipart uploads created earlier than January 2nd, 2014, 10:45:37.**

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02
 10:45:37"
```

- For API details, see AbortMultipartUpload in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `AbortMultipartUploads` with an AWS SDK or CLI

The following code example shows how to use `AbortMultipartUploads`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;

    /// <summary>
    /// This example shows how to use the Amazon Simple Storage Service
    /// (Amazon S3) to stop a multi-part upload process using the Amazon S3
    /// TransferUtility.
    /// </summary>
    public class AbortMPU
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await AbortMPUAsync(client, bucketName);
        }

        /// <summary>
        /// Cancels the multi-part copy process.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// the TransferUtility object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the
        /// multi-part copy operation is in progress.</param>
        public static async Task AbortMPUAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var transferUtility = new TransferUtility(client);

                // Cancel all in-progress uploads initiated before the specified
date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
```

```
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
    }
```

- For API details, see AbortMultipartUploads in *AWS SDK for .NET API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `CompleteMultipartUpload` with an AWS SDK or CLI

The following code examples show how to use `CompleteMultipartUpload`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Perform a multipart copy
- Perform a multipart upload
- Use checksums
- Work with Amazon S3 object integrity

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! Complete a multipart upload to an S3 bucket.
```

```
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- For API details, see CompleteMultipartUpload in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command completes a multipart upload for the key `multipart/01` in the bucket `my-bucket`:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket my-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

The upload ID required by this command is output by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`.

The multipart upload option in the above command takes a JSON structure that describes the parts of the multipart upload that should be reassembled into the complete file. In this example, the `file://` prefix is used to load the JSON structure from a file in the local folder named `mpustruct`.

mpustruct:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

The ETag value for each part is upload is output each time you upload a part using the `upload-part` command and can also be retrieved by calling `list-parts` or calculated by taking the MD5 checksum of each part.

Output:

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "my-bucket",
    "Location": "https://my-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
}
```

- For API details, see CompleteMultipartUpload in *AWS CLI Command Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();
```

- For API details, see CompleteMultipartUpload in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `CopyObject` with an AWS SDK or CLI

The following code examples show how to use `CopyObject`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Get started with buckets and objects
- Get started with encryption

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class CopyObject
    {
        public static async Task Main()
        {
            // Specify the AWS Region where your buckets are located if it is
            // different from the AWS Region of the default user.
            IAmazonS3 s3Client = new AmazonS3Client();

            // Remember to change these values to refer to your Amazon S3
objects.
            string sourceBucketName = "doc-example-bucket1";
            string destinationBucketName = "doc-example-bucket2";
            string sourceObjectKey = "testfile.txt";
            string destinationObjectKey = "testfilecopy.txt";
```

```
            Console.WriteLine($"Copying {sourceObjectKey} from {sourceBucketName}
 to ");
            Console.WriteLine($"{destinationBucketName} as
 {destinationObjectKey}");

            var response = await CopyingObjectAsync(
                s3Client,
                sourceObjectKey,
                destinationObjectKey,
                sourceBucketName,
                destinationBucketName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("\nCopy complete.");
            }
        }

        /// <summary>
        /// This method calls the AWS SDK for .NET to copy an
        /// object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The Amazon S3 client object.</param>
        /// <param name="sourceKey">The name of the object to be copied.</param>
        /// <param name="destinationKey">The name under which to save the copy.</
param>
        /// <param name="sourceBucketName">The name of the Amazon S3 bucket
        /// where the file is located now.</param>
        /// <param name="destinationBucketName">The name of the Amazon S3
        /// bucket where the copy should be saved.</param>
        /// <returns>Returns a CopyObjectResponse object with the results from
        /// the async call.</returns>
        public static async Task<CopyObjectResponse> CopyingObjectAsync(
            IAmazonS3 client,
            string sourceKey,
            string destinationKey,
            string sourceBucketName,
            string destinationBucketName)
        {
            var response = new CopyObjectResponse();
            try
            {
                var request = new CopyObjectRequest
                {
```

```
                    SourceBucket = sourceBucketName,
                    SourceKey = sourceKey,
                    DestinationBucket = destinationBucketName,
                    DestinationKey = destinationKey,
                };
                response = await client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error copying object: '{ex.Message}'");
            }

            return response;
        }
    }
```

- For API details, see CopyObject in *AWS SDK for .NET API Reference*.

Bash

**AWS CLI with Bash script**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_item_in_bucket
#
```

```
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- For API details, see CopyObject in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see CopyObject in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command copies an object from `bucket-1` to `bucket-2`:

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

Output:

```
{
    "CopyObjectResult": {
```

```
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- For API details, see CopyObject in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
```

```
 }
 return err
}
```

- For API details, see CopyObject in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Copy an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <objectKey> <fromBucket> <toBucket>
```

```
                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
bucket2).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

Use an S3TransferManager to copy an object from one bucket to another. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- For API details, see CopyObject in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Copy the object.

```
import { S3Client, CopyObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CopyObjectCommand({
    CopySource: "SOURCE_BUCKET/SOURCE_OBJECT_KEY",
    Bucket: "DESTINATION_BUCKET",
    Key: "NEW_OBJECT_KEY",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see CopyObject in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- For API details, see CopyObject in *AWS SDK for Kotlin API reference*.

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Simple copy of an object.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- For API details, see CopyObject in *AWS SDK for PHP API Reference*.

PowerShell

### Tools for PowerShell

**Example 1: This command copies the object "sample.txt" from bucket "test-files" to the same bucket but with a new key of "sample-copy.txt".**

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt
```

**Example 2: This command copies the object "sample.txt" from bucket "test-files" to the bucket "backup-files" with a key of "sample-copy.txt".**

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt -DestinationBucket backup-files
```

**Example 3: This command downloads the object "sample.txt" from bucket "test-files" to a local file with name "local-sample.txt".**

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

**Example 4: Downloads the single object to the specified file. The downloaded file will be found at c:\downloads\data\archive.zip**

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:
\downloads
```

**Example 5: Downloads all objects that match the specified key prefix to the local folder. The relative key hierarchy will be preserved as subfolders in the overall download location.**

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- For API details, see CopyObject in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
```

```
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

- For API details, see CopyObject in *AWS SDK for Python (Boto3) API Reference*.

Ruby

### SDK for Ruby

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Copy an object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copy an object and add server-side encryption to the destination object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see CopyObject in *AWS SDK for Ruby API Reference*.

Rust

### SDK for Rust

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}
```

- For API details, see CopyObject in *AWS SDK for Rust API reference*.

SAP ABAP

### SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
TRY.
    lo_s3->copyobject(
      iv_bucket = iv_dest_bucket
      iv_key = iv_dest_object
      iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|
    ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see CopyObject in *AWS SDK for SAP ABAP API reference*.

Swift

### SDK for Swift

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

- For API details, see CopyObject in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `CreateBucket` with an AWS SDK or CLI

The following code examples show how to use `CreateBucket`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Get started with buckets and objects
- Work with versioned objects

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Shows how to create a new Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value representing the success or failure of
        /// the bucket creation process.</returns>
        public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var request = new PutBucketRequest
                {
                    BucketName = bucketName,
                    UseClientRegion = true,
                };

                var response = await client.PutBucketAsync(request);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error creating bucket: '{ex.Message}'");
                return false;
            }
        }
```

Create a bucket with object lock enabled.

```
   /// <summary>
   /// Create a new Amazon S3 bucket with object lock actions.
   /// </summary>
   /// <param name="bucketName">The name of the bucket to create.</param>
   /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
   {
       Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
       try
       {
           var request = new PutBucketRequest
           {
               BucketName = bucketName,
               UseClientRegion = true,
               ObjectLockEnabledForBucket = enableObjectLock,
           };

           var response = await _amazonS3.PutBucketAsync(request);

           return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"Error creating bucket: '{ex.Message}'");
           return false;
       }
   }
```

- For API details, see CreateBucket in *AWS SDK for .NET API Reference*.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
```

```
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi
```

```
  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- For API details, see CreateBucket in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

**Example 1: To create a bucket**

The following `create-bucket` example creates a bucket named `my-bucket`:

```
aws s3api create-bucket \
```

```
    --bucket my-bucket \
    --region us-east-1
```

Output:

```
{
    "Location": "/my-bucket"
}
```

For more information, see Creating a bucket in the *Amazon S3 User Guide*.

**Example 2: To create a bucket with owner enforced**

The following `create-bucket` example creates a bucket named `my-bucket` that uses the bucket owner enforced setting for S3 Object Ownership.

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

Output:

```
{
    "Location": "/my-bucket"
}
```

For more information, see Controlling ownership of objects and disabling ACLs in the *Amazon S3 User Guide*.

**Example 3: To create a bucket outside of the ``us-east-1`` region**

The following `create-bucket` example creates a bucket named `my-bucket` in the `eu-west-1` region. Regions outside of `us-east-1` require the appropriate `LocationConstraint` to be specified in order to create the bucket in the desired region.

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

Output:

```
{
    "Location": "http://my-bucket.s3.amazonaws.com/"
}
```

For more information, see Creating a bucket in the *Amazon S3 User Guide*.

- For API details, see CreateBucket in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a bucket with default configuration.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
```

```
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}
```

Create a bucket with object locking and wait for it to exist.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
```

```
    err = owned
   } else if errors.As(err, &exists) {
    log.Printf("Bucket %s already exists.\n", bucket)
    err = exists
   }
  } else {
   err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
   }
  }

  return bucket, err
}
```

- For API details, see CreateBucket in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a bucket.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the bucket to create. The bucket
 name must be unique, or an error occurs.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
```

```
            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Create a bucket with object lock enabled.

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
 bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- For API details, see CreateBucket in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the bucket.

```
import { CreateBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CreateBucketCommand({
    // The name of the bucket. Bucket names are unique and have several other
 constraints.
    // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
    Bucket: "bucket-name",
  });

  try {
    const { Location } = await client.send(command);
    console.log(`Bucket created with location ${Location}`);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see CreateBucket in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- For API details, see CreateBucket in *AWS SDK for Kotlin API reference*.

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
```

```
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- For API details, see CreateBucket in *AWS SDK for PHP API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a bucket with default settings.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
```

```
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

Create a versioned bucket with a lifecycle configuration.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.
```

```
    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
```

```
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- For API details, see CreateBucket in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
```

```
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see CreateBucket in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- For API details, see CreateBucket in *AWS SDK for Rust API reference*.

SAP ABAP

### SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
  CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
  CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```

- For API details, see CreateBucket in *AWS SDK for SAP ABAP API reference*.

Swift

### SDK for Swift

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }
```

- For API details, see CreateBucket in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `CreateMultiRegionAccessPoint` with an AWS SDK or CLI

The following code example shows how to use `CreateMultiRegionAccessPoint`.

Kotlin

**SDK for Kotlin**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Configure the S3 control client to send request to the us-west-2 Region.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
```

```
            return s3Control
        }
```

Create the Multi-Region Access Point.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
 CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
```

```
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Wait for the Multi-Region Access Point to become available.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- For more information, see AWS SDK for Kotlin developer guide.
- For API details, see CreateMultiRegionAccessPoint in *AWS SDK for Kotlin API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `CreateMultipartUpload` with an AWS SDK or CLI

The following code examples show how to use `CreateMultipartUpload`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Perform a multipart copy
- Perform a multipart upload
- Use checksums
- Work with Amazon S3 object integrity

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
```

```
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}
```

- For API details, see CreateMultipartUpload in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command creates a multipart upload in the bucket `my-bucket` with the key `multipart/01`:

```
aws s3api create-multipart-upload --bucket my-bucket --key 'multipart/01'
```

Output:

```
{
    "Bucket": "my-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

The completed file will be named `01` in a folder called `multipart` in the bucket `my-bucket`. Save the upload ID, key and bucket name for use with the `upload-part` command.

- For API details, see CreateMultipartUpload in *AWS CLI Command Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await
    .unwrap();
```

- For API details, see CreateMultipartUpload in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucket` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucket`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with buckets and objects

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Shows how to delete an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to
delete.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// the delete operation.</returns>
        public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string
bucketName)
        {
            var request = new DeleteBucketRequest
            {
                BucketName = bucketName,
            };

            var response = await client.DeleteBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- For API details, see DeleteBucket in *AWS SDK for .NET API Reference*.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
```

```
  fi
}
```

- For API details, see DeleteBucket in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see DeleteBucket in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command deletes a bucket named `my-bucket`:

```
aws s3api delete-bucket --bucket my-bucket --region us-east-1
```

- For API details, see DeleteBucket in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
```

```
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

- For API details, see DeleteBucket in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

- For API details, see DeleteBucket in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the bucket.

```
import { DeleteBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Delete a bucket.
export const main = async () => {
  const command = new DeleteBucketCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see DeleteBucket in *AWS SDK for JavaScript API Reference*.

PHP

**SDK for PHP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an empty bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- For API details, see DeleteBucket in *AWS SDK for PHP API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command removes all objects and object versions from the bucket 'test-files' and then deletes the bucket. The command will prompt for confirmation before proceeding. Add the -Force switch to suppress confirmation. Note that buckets that are not empty cannot be deleted.**

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- For API details, see DeleteBucket in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
```

```
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- For API details, see DeleteBucket in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
 # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
 #
 # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
 def delete_bucket(bucket)
   puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
   answer = gets.chomp.downcase
   if answer == "y"
     bucket.objects.batch_delete!
     bucket.delete
     puts("Emptied and deleted bucket #{bucket.name}.\n")
```

```
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- For API details, see DeleteBucket in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- For API details, see DeleteBucket in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see DeleteBucket in *AWS SDK for SAP ABAP API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }
```

- For API details, see DeleteBucket in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketAnalyticsConfiguration` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketAnalyticsConfiguration`.

CLI

**AWS CLI**

**To delete an analytics configuration for a bucket**

The following `delete-bucket-analytics-configuration` example removes the analytics configuration for the specified bucket and ID.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

This command produces no output.

- For API details, see DeleteBucketAnalyticsConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The command removes the analytics filter with name 'testfilter' in the given S3 bucket.**

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- For API details, see DeleteBucketAnalyticsConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketCors` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketCors`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- For API details, see DeleteBucketCors in *AWS SDK for .NET API Reference*.

CLI

### AWS CLI

The following command deletes a Cross-Origin Resource Sharing configuration from a bucket named `my-bucket`:

```
aws s3api delete-bucket-cors --bucket my-bucket
```

- For API details, see DeleteBucketCors in *AWS CLI Command Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
```

```
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- For API details, see DeleteBucketCors in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
```

```
    false
  end

end
```

- For API details, see DeleteBucketCors in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketEncryption` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketEncryption`.

CLI

### AWS CLI

#### To delete the server-side encryption configuration of a bucket

The following `delete-bucket-encryption` example deletes the server-side encryption configuration of the specified bucket.

```
aws s3api delete-bucket-encryption \
    --bucket my-bucket
```

This command produces no output.

- For API details, see DeleteBucketEncryption in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: This disables the encryption enabled for the S3 bucket provided.

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

**Output:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- For API details, see DeleteBucketEncryption in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketInventoryConfiguration` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketInventoryConfiguration`.

CLI

### AWS CLI

#### To delete the inventory configuration of a bucket

The following `delete-bucket-inventory-configuration` example deletes the inventory configuration with ID 1 for the specified bucket.

```
aws s3api delete-bucket-inventory-configuration \
    --bucket my-bucket \
    --id 1
```

This command produces no output.

- For API details, see DeleteBucketInventoryConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command removes the invventory named 'testInventoryName' corresponding to the given S3 bucket.**

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

**Output:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- For API details, see DeleteBucketInventoryConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketLifecycle` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketLifecycle`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- For API details, see DeleteBucketLifecycle in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

The following command deletes a lifecycle configuration from a bucket named `my-bucket`:

```
aws s3api delete-bucket-lifecycle --bucket my-bucket
```

- For API details, see DeleteBucketLifecycle in *AWS CLI Command Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
 self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- For API details, see DeleteBucketLifecycle in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketMetricsConfiguration` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketMetricsConfiguration`.

CLI

**AWS CLI**

**To delete a metrics configuration for a bucket**

The following `delete-bucket-metrics-configuration` example removes the metrics configuration for the specified bucket and ID.

```
aws s3api delete-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

This command produces no output.

- For API details, see DeleteBucketMetricsConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The command removes the metrics filter with name 'testmetrics' in the given S3 bucket.**

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- For API details, see DeleteBucketMetricsConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketPolicy` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketPolicy`.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see DeleteBucketPolicy in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command deletes a bucket policy from a bucket named `my-bucket`:

```
aws s3api delete-bucket-policy --bucket my-bucket
```

- For API details, see DeleteBucketPolicy in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
 (for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the bucket policy.

```
import { DeleteBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// This will remove the policy from the bucket.
export const main = async () => {
  const command = new DeleteBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see DeleteBucketPolicy in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Kotlin API reference*.

PowerShell

### Tools for PowerShell

#### Example 1: The command removes the bucket policy associated with the given S3 bucket.

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- For API details, see DeleteBucketPolicy in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketReplication` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketReplication`.

CLI

**AWS CLI**

The following command deletes a replication configuration from a bucket named `my-bucket`:

```
aws s3api delete-bucket-replication --bucket my-bucket
```

- For API details, see DeleteBucketReplication in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: Deletes the replication configuration associated with the bucket named 'mybucket'. Note that this operation requires permission for the s3:DeleteReplicationConfiguration action. You will be prompted for confirmation before the operation proceeds - to suppress confirmation, use the -Force switch.**

```
Remove-S3BucketReplication -BucketName mybucket
```

- For API details, see DeleteBucketReplication in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketTagging` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketTagging`.

CLI

**AWS CLI**

The following command deletes a tagging configuration from a bucket named `my-bucket`:

```
aws s3api delete-bucket-tagging --bucket my-bucket
```

- For API details, see DeleteBucketTagging in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command removes all the tags associated with the given S3 bucket.**

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

**Output:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- For API details, see DeleteBucketTagging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteBucketWebsite` with an AWS SDK or CLI

The following code examples show how to use `DeleteBucketWebsite`.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
```

```
                                        const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see DeleteBucketWebsite in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command deletes a website configuration from a bucket named `my-bucket`:

```
aws s3api delete-bucket-website --bucket my-bucket
```

- For API details, see DeleteBucketWebsite in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
 configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- For API details, see [DeleteBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the website configuration from the bucket.

```
import { DeleteBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Disable static website hosting on the bucket.
export const main = async () => {
  const command = new DeleteBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see DeleteBucketWebsite in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command disables the static website hosting property of the given S3 bucket.**

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

**Output:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- For API details, see DeleteBucketWebsite in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteObject` with an AWS SDK or CLI

The following code examples show how to use `DeleteObject`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Work with Amazon S3 object integrity
- Work with versioned objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an object in a non-versioned S3 bucket.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
```

```
        /// The Main method initializes the necessary variables and then calls
        /// the DeleteObjectNonVersionedBucketAsync method to delete the object
        /// named by the keyName parameter.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket";
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
            }
            catch (AmazonS3Exception ex)
```

```
            {
                Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

Delete an object in a versioned S3 bucket.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
```

```
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
 client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
 keyName);

                // Delete the object by specifying an object key and a version
 ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
```

```
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- For API details, see DeleteObject in *AWS SDK for .NET API Reference*.

Bash

**AWS CLI with Bash script**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
```

```
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- For API details, see DeleteObject in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
```

```
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see DeleteObject in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command deletes an object named `test.txt` from a bucket named `my-bucket`:

```
aws s3api delete-object --bucket my-bucket --key test.txt
```

If bucket versioning is enabled, the output will contain the version ID of the delete marker:

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

For more information about deleting objects, see Deleting Objects in the *Amazon S3 Developer Guide*.

- For API details, see DeleteObject in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
```

```
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}
```

- For API details, see DeleteObject in *AWS SDK for Go API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an object.

```
import { DeleteObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectCommand({
    Bucket: "test-bucket",
```

```
    Key: "test-key.txt",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see DeleteObject in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an object.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
```

```
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

Roll an object back to a previous version by deleting later versions of the object.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
```

```
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

Revive a deleted object by removing the object's active delete marker.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
```

```
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Create a Lambda handler that removes a delete marker from an S3 object. This handler can be used to efficiently clean up extraneous delete markers in a versioned bucket.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
 the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
```

```
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
 marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
 obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
 retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
```

```
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- For API details, see DeleteObject in *AWS SDK for Python (Boto3) API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(),
 Error> {
    client
```

```
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    println!("Object deleted.");

    Ok(())
}
```

- For API details, see DeleteObject in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key
        ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see DeleteObject in *AWS SDK for SAP ABAP API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    } catch {
        throw error
    }
}
```

- For API details, see DeleteObject in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteObjectTagging` with an AWS SDK or CLI

The following code examples show how to use `DeleteObjectTagging`.

CLI

### AWS CLI

#### To delete the tag sets of an object

The following `delete-object-tagging` example deletes the tag with the specified key from the object `doc1.rtf`.

```
aws s3api delete-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

This command produces no output.

- For API details, see DeleteObjectTagging in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: This command removes all the tags associated with the object with key 'testfile.txt' in the given S3 Bucket.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select
 '^Key'
```

#### Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- For API details, see DeleteObjectTagging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeleteObjects` with an AWS SDK or CLI

The following code examples show how to use `DeleteObjects`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with buckets and objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete all objects in an S3 bucket.

```
        /// <summary>
        /// Delete all of the objects stored in an existing Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// contents will be deleted.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// deleting all of the objects in the bucket.</returns>
        public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3
client, string bucketName)
        {
            // Iterate over the contents of the bucket and delete all objects.
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
```

```
            };

            try
            {
                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);
                    response.S3Objects
                        .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

                    // If the response is truncated, set the request
ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error deleting objects: {ex.Message}");
                return false;
            }
        }
```

Delete multiple objects in a non-versioned S3 bucket.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
```

```
   public class DeleteMultipleObjects
   {
       /// <summary>
       /// The Main method initializes the Amazon S3 client and the name of
       /// the bucket and then passes those values to MultiObjectDeleteAsync.
       /// </summary>
       public static async Task Main()
       {
           const string bucketName = "doc-example-bucket";

           // If the Amazon S3 bucket from which you wish to delete objects is
not
           // located in the same AWS Region as the default user, define the
           // AWS Region for the Amazon S3 bucket as a parameter to the client
           // constructor.
           IAmazonS3 s3Client = new AmazonS3Client();

           await MultiObjectDeleteAsync(s3Client, bucketName);
       }

       /// <summary>
       /// This method uses the passed Amazon S3 client to first create and then
       /// delete three files from the named bucket.
       /// </summary>
       /// <param name="client">The initialized Amazon S3 client object used to
call
       /// Amazon S3 methods.</param>
       /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
       /// will be created and then deleted.</param>
       public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
       {
           // Create three sample objects which we will then delete.
           var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

           // Now perform the multi-object delete, passing the key names and
           // version IDs. Since we are working with a non-versioned bucket,
           // the object keys collection includes null version IDs.
           DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
           {
               BucketName = bucketName,
               Objects = keysAndVersions,
```

```
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
```

```
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

Delete multiple objects in a versioned S3 bucket.

```
    using System;
    using System.Collections.Generic;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
```

```
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
```

```
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
```

```
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }
```

```
            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
```

```
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
 bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
 in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
```

```
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- For API details, see DeleteObjects in *AWS SDK for .NET API Reference*.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
```

```
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
```

```
  fi
}
```

- For API details, see DeleteObjects in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
```

```
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see DeleteObjects in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command deletes an object from a bucket named `my-bucket`:

```
aws s3api delete-objects --bucket my-bucket --delete file://delete.json
```

`delete.json` is a JSON document in the current directory that specifies the object to delete:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Output:

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
```

```
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
}
```

- For API details, see DeleteObjects in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
```

```
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 }
 return err
}
```

- For API details, see DeleteObjects in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName>

                Where:
                   bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;
```

```
        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                    .key(keyName)
                    .build();

            putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
                .objects(keys)
                .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DeleteObjects in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete multiple objects.

```
import { DeleteObjectsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectsCommand({
    Bucket: "test-bucket",
    Delete: {
      Objects: [{ Key: "object1.txt" }, { Key: "object2.txt" }],
    },
  });

  try {
    const { Deleted } = await client.send(command);
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see DeleteObjects in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- For API details, see DeleteObjects in *AWS SDK for Kotlin API reference*.

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete a set of objects from a list of keys.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- For API details, see DeleteObjects in *AWS SDK for PHP API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command removes the object "sample.txt" from bucket "test-files". You are prompted for confirmation before the command executes; to suppress the prompt use the -Force switch.**

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

**Example 2: This command removes the specified version of object "sample.txt" from bucket "test-files", assuming the bucket has been configured to enable object versions.**

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

**Example 3: This command removes objects "sample1.txt", "sample2.txt" and "sample3.txt" from bucket "test-files" as a single batch operation. The service response will list all keys processed, regardless of the success or error status of the deletion. To obtain only errors for keys that were not able to be processed by the service add the -ReportErrorsOnly parameter (this parameter can also be specified with the alias -Quiet.**

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

**Example 4: This example uses an inline expression with the -KeyCollection parameter to obtain the keys of the objects to delete. Get-S3Object returns a collection of Amazon.S3.Model.S3Object instances, each of which has a Key member of type string identifying the object.**

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-
files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

**Example 5: This example obtains all objects that have a key prefix "prefix/subprefix" in the bucket and deletes them. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's -InputObject (alias -S3ObjectCollection) parameter to enable the deletion to occur as a batch with a single call to the service.**

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

**Example 6: This example pipes a collection of Amazon.S3.Model.S3ObjectVersion instances that represent delete markers to the cmdlet for deletion. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's -InputObject (alias -S3ObjectCollection) parameter to enable the deletion to occur as a batch with a single call to the service.**

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

**Example 7: This script shows how to perform a batch delete of a set of objects (in this case delete markers) by constructing an array of objects to be used with the -KeyAndVersionCollection parameter.**

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- For API details, see DeleteObjects in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete a set of objects by using a list of object keys.

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
```

```
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

Delete all objects in a bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

Permanently delete a versioned object by deleting all of its versions.

```
def permanently_delete_object(bucket, object_key):
```

```
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- For API details, see DeleteObjects in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
 # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
 #
 # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
 def delete_bucket(bucket)
   puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
   answer = gets.chomp.downcase
   if answer == "y"
     bucket.objects.batch_delete!
     bucket.delete
     puts("Emptied and deleted bucket #{bucket.name}.\n")
```

```
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- For API details, see DeleteObjects in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
```

```
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}
```

- For API details, see DeleteObjects in *AWS SDK for Rust API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public func deleteObjects(bucket: String, keys: [String]) async throws {
```

```
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
                quiet: true
            )
        )

        do {
            let output = try await client.deleteObjects(input: input)

            // As of the last update to this example, any errors are returned
            // in the `output` object's `errors` property. If there are any
            // errors in this array, throw an exception. Once the error
            // handling is finalized in later updates to the AWS SDK for
            // Swift, this example will be updated to handle errors better.

            guard let errors = output.errors else {
                return  // No errors.
            }
            if errors.count != 0 {
                throw ServiceHandlerError.deleteObjectsError
            }
        } catch {
            throw error
        }
    }
```

- For API details, see DeleteObjects in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `DeletePublicAccessBlock` with an AWS SDK or CLI

The following code examples show how to use `DeletePublicAccessBlock`.

CLI

**AWS CLI**

**To delete the block public access configuration for a bucket**

The following `delete-public-access-block` example removes the block public access configuration on the specified bucket.

```
aws s3api delete-public-access-block \
    --bucket my-bucket
```

This command produces no output.

- For API details, see DeletePublicAccessBlock in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command turns off the block public access setting for the given bucket.**

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select
 '^BucketName'
```

**Output:**

```
s3testbucket
```

- For API details, see DeletePublicAccessBlock in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketAccelerateConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetBucketAccelerateConfiguration`.

CLI

**AWS CLI**

**To retrieve the accelerate configuration of a bucket**

The following `get-bucket-accelerate-configuration` example retrieves the accelerate configuration for the specified bucket.

```
aws s3api get-bucket-accelerate-configuration \
    --bucket my-bucket
```

Output:

```
{
    "Status": "Enabled"
}
```

- For API details, see GetBucketAccelerateConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the value Enabled, if the transfer acceleration settings is enabled for the bucket specified.**

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

**Output:**

```
Value
-----
Enabled
```

- For API details, see GetBucketAccelerateConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketAcl` with an AWS SDK or CLI

The following code examples show how to use `GetBucketAcl`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Manage access control lists (ACLs)

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
```

```
            });

            return getACLResponse.AccessControlList;
        }
```

- For API details, see GetBucketAcl in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();
```

```
            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
```

```
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
```

```
    return "Permission unknown";
}
```

- For API details, see GetBucketAcl in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command retrieves the access control list for a bucket named `my-bucket`:

```
aws s3api get-bucket-acl --bucket my-bucket
```

Output:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- For API details, see GetBucketAcl in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey>

                Where:
                  bucketName - The Amazon S3 bucket to get the access control
 list (ACL) for.
                  objectKey - The object to get the ACL for.\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
 bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- For API details, see GetBucketAcl in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get the ACL permissions.

```
import { GetBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketAclCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see GetBucketAcl in *AWS SDK for JavaScript API Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
 acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- For API details, see GetBucketAcl in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketAnalyticsConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetBucketAnalyticsConfiguration`.

CLI

**AWS CLI**

**To retrieve the analytics configuration for a bucket with a specific ID**

The following `get-bucket-analytics-configuration` example displays the analytics configuration for the specified bucket and ID.

```
aws s3api get-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

Output:

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- For API details, see GetBucketAnalyticsConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the details of the analytics filter with the name 'testfilter' in the given S3 bucket.**

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- For API details, see GetBucketAnalyticsConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketCors` with an AWS SDK or CLI

The following code examples show how to use `GetBucketCors`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
```

```
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- For API details, see GetBucketCors in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

The following command retrieves the Cross-Origin Resource Sharing configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-cors --bucket my-bucket
```

Output:

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
        {
            "AllowedHeaders": [
                "Authorization"
```

```
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
}
```

- For API details, see GetBucketCors in *AWS CLI Command Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get the CORS policy for the bucket.

```
import { GetBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketCorsCommand({
    Bucket: "test-bucket",
  });

  try {
    const { CORSRules } = await client.send(command);
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders.join(" ")}`,
```

```
        `\nAllowedMethods: ${cr.AllowedMethods.join(" ")}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins.join(" ")}`,
        `\nExposeHeaders: ${cr.ExposeHeaders.join(" ")}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see GetBucketCors in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.
```

```
        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
 self.bucket.name))
            raise
        else:
            return cors
```

- For API details, see GetBucketCors in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end
```

```
  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- For API details, see GetBucketCors in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketEncryption` with an AWS SDK or CLI

The following code examples show how to use `GetBucketEncryption`.

CLI

### AWS CLI

#### To retrieve the server-side encryption configuration for a bucket

The following `get-bucket-encryption` example retrieves the server-side encryption configuration for the bucket `my-bucket`.

```
aws s3api get-bucket-encryption \
    --bucket my-bucket
```

Output:

```
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
```

```
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- For API details, see GetBucketEncryption in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns all the server side encryption rules associated with the given bucket.**

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- For API details, see GetBucketEncryption in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketInventoryConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetBucketInventoryConfiguration`.

CLI

**AWS CLI**

**To retrieve the inventory configuration for a bucket**

The following `get-bucket-inventory-configuration` example retrieves the inventory configuration for the specified bucket with ID 1.

```
aws s3api get-bucket-inventory-configuration \
    --bucket my-bucket \
```

```
    --id 1
```

Output:

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::my-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    }
}
```

- For API details, see GetBucketInventoryConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the details of the inventory named 'testinventory' for the given S3 bucket.**

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- For API details, see GetBucketInventoryConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `GetBucketLifecycleConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetBucketLifecycleConfiguration`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- For API details, see GetBucketLifecycleConfiguration in *AWS SDK for .NET API Reference*.

CLI

### AWS CLI

The following command retrieves the lifecycle configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-lifecycle-configuration --bucket my-bucket
```

Output:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- For API details, see GetBucketLifecycleConfiguration in *AWS CLI Command Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
```

```
            return config.rules


```

- For API details, see GetBucketLifecycleConfiguration in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketLocation` with an AWS SDK or CLI

The following code examples show how to use `GetBucketLocation`.

CLI

### AWS CLI

The following command retrieves the location constraint for a bucket named `my-bucket`, if a constraint exists:

```
aws s3api get-bucket-location --bucket my-bucket
```

Output:

```
{
    "LocationConstraint": "us-west-2"
}
```

- For API details, see GetBucketLocation in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

**Example 1: This command returns the location constraint for the bucket 's3testbucket', if a constraint exists.**

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

**Output:**

```
Value
-----
ap-south-1
```

- For API details, see GetBucketLocation in *AWS Tools for PowerShell Cmdlet Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
```

```
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- For API details, see GetBucketLocation in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketLogging` with an AWS SDK or CLI

The following code examples show how to use `GetBucketLogging`.

CLI

### AWS CLI

#### To retrieve the logging status for a bucket

The following `get-bucket-logging` example retrieves the logging status for the specified bucket.

```
aws s3api get-bucket-logging \
    --bucket my-bucket
```

Output:

```
{
```

```
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "my-bucket-logs"
          }
}
```

- For API details, see GetBucketLogging in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the logging status for the specified bucket.**

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

**Output:**

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- For API details, see GetBucketLogging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketMetricsConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetBucketMetricsConfiguration`.

CLI

**AWS CLI**

**To retrieve the metrics configuration for a bucket with a specific ID**

The following `get-bucket-metrics-configuration` example displays the metrics configuration for the specified bucket and ID.

```
aws s3api get-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

Output:

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- For API details, see GetBucketMetricsConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the details about the metrics filter named 'testfilter' for the given S3 bucket.**

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testfilter'
```

- For API details, see GetBucketMetricsConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketNotification` with an AWS SDK or CLI

The following code examples show how to use `GetBucketNotification`.

CLI

### AWS CLI

The following command retrieves the notification configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-notification --bucket my-bucket
```

Output:

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
}
```

- For API details, see GetBucketNotification in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: This example retrieves notification configuration of the given bucket

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

**Output:**

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- For API details, see GetBucketNotification in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketPolicy` with an AWS SDK or CLI

The following code examples show how to use `GetBucketPolicy`.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;
```

```
        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see GetBucketPolicy in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command retrieves the bucket policy for a bucket named `my-bucket`:

```
aws s3api get-bucket-policy --bucket my-bucket
```

Output:

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::my-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":
\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::my-bucket/secret/*
\"}]}"
}
```

Get and put a bucket policyThe following example shows how you can download an Amazon S3 bucket policy, make modifications to the file, and then use `put-bucket-policy` to apply the modified bucket policy. To download the bucket policy to a file, you can run:

```
aws s3api get-bucket-policy --bucket mybucket --query Policy --output text >
 policy.json
```

You can then modify the `policy.json` file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

`policy.json` file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

```
aws s3api put-bucket-policy --bucket mybucket --policy file://policy.json
```

- For API details, see GetBucketPolicy in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to get the policy from.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- For API details, see GetBucketPolicy in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get the bucket policy.

```
import { GetBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const { Policy } = await client.send(command);
    console.log(JSON.parse(Policy));
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see GetBucketPolicy in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- For API details, see GetBucketPolicy in *AWS SDK for Kotlin API reference*.

PowerShell

### Tools for PowerShell

#### Example 1: This command outputs the bucket policy associated with the given S3 bucket.

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- For API details, see GetBucketPolicy in *AWS Tools for PowerShell Cmdlet Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- For API details, see GetBucketPolicy in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- For API details, see GetBucketPolicy in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketPolicyStatus` with an AWS SDK or CLI

The following code examples show how to use `GetBucketPolicyStatus`.

CLI

### AWS CLI

#### To retrieve the policy status for a bucket indicating whether the bucket is public

The following `get-bucket-policy-status` example retrieves the policy status for the bucket `my-bucket`.

```
aws s3api get-bucket-policy-status \
    --bucket my-bucket
```

Output:

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- For API details, see GetBucketPolicyStatus in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: This command returns policy status for the given S3 bucket, indicating whether the bucket is public.

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- For API details, see GetBucketPolicyStatus in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketReplication` with an AWS SDK or CLI

The following code examples show how to use `GetBucketReplication`.

CLI

### AWS CLI

The following command retrieves the replication configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-replication --bucket my-bucket
```

Output:

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::my-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- For API details, see GetBucketReplication in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: Returns the replication configuration information set on the bucket named 'mybucket'.**

```
Get-S3BucketReplication -BucketName mybucket
```

- For API details, see GetBucketReplication in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketRequestPayment` with an AWS SDK or CLI

The following code examples show how to use `GetBucketRequestPayment`.

CLI

**AWS CLI**

**To retrieve the request payment configuration for a bucket**

The following `get-bucket-request-payment` example retrieves the requester pays configuration for the specified bucket.

```
aws s3api get-bucket-request-payment \
    --bucket my-bucket
```

Output:

```
{
    "Payer": "BucketOwner"
}
```

- For API details, see GetBucketRequestPayment in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: Returns the request payment configuration for the bucket named 'mybucket'. By default, the bucket owner pays for downloads from the bucket.**

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- For API details, see GetBucketRequestPayment in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketTagging` with an AWS SDK or CLI

The following code examples show how to use `GetBucketTagging`.

CLI

**AWS CLI**

The following command retrieves the tagging configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-tagging --bucket my-bucket
```

Output:

```
{
    "TagSet": [
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- For API details, see GetBucketTagging in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns all the tags associated with the given bucket.**

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- For API details, see GetBucketTagging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketVersioning` with an AWS SDK or CLI

The following code examples show how to use `GetBucketVersioning`.

CLI

**AWS CLI**

The following command retrieves the versioning configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-versioning --bucket my-bucket
```

Output:

```
{
    "Status": "Enabled"
}
```

- For API details, see GetBucketVersioning in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the status of versioning with respect to the given bucket.**

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- For API details, see GetBucketVersioning in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetBucketWebsite` with an AWS SDK or CLI

The following code examples show how to use `GetBucketWebsite`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- For API details, see GetBucketWebsite in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
```

```
                    << "Error page: "
                    << websiteResult.GetErrorDocument().GetKey()
                    << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see GetBucketWebsite in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command retrieves the static website configuration for a bucket named `my-bucket`:

```
aws s3api get-bucket-website --bucket my-bucket
```

Output:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- For API details, see GetBucketWebsite in *AWS CLI Command Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get the website configuration.

```
import { GetBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const { ErrorDocument, IndexDocument } = await client.send(command);
    console.log(
      `Your bucket is set up to host a website. It has an error document:`,
      `${ErrorDocument.Key}, and an index document: ${IndexDocument.Suffix}.`,
    );
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see GetBucketWebsite in *AWS SDK for JavaScript API Reference*.

PowerShell

### Tools for PowerShell

**Example 1: This command returns the details of the static website configurations of the given S3 bucket.**

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- For API details, see GetBucketWebsite in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObject` with an AWS SDK or CLI

The following code examples show how to use `GetObject`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Get an object from a bucket if it has been modified
- Get an object from a Multi-Region Access Point
- Get started with buckets and objects
- Get started with encryption
- Track uploads and downloads

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Shows how to download an object from an Amazon S3 bucket to the
        /// local computer.
        /// </summary>
```

```
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket where the object is
        /// currently stored.</param>
        /// <param name="objectName">The name of the object to download.</param>
        /// <param name="filePath">The path, including filename, where the
        /// downloaded object will be stored.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// download process.</returns>
        public static async Task<bool> DownloadObjectFromBucketAsync(
            IAmazonS3 client,
            string bucketName,
            string objectName,
            string filePath)
        {
            // Create a GetObject request
            var request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
            };

            // Issue request and remember to dispose of the response
            using GetObjectResponse response = await
 client.GetObjectAsync(request);

            try
            {
                // Save object to local file
                await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error saving {objectName}: {ex.Message}");
                return false;
            }
        }
```

- For API details, see GetObject in *AWS SDK for .NET API Reference*.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
```

```
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- For API details, see GetObject in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- For API details, see GetObject in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following example uses the `get-object` command to download an object from Amazon S3:

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Note that the outfile parameter is specified without an option name such as "--outfile". The name of the output file must be the last parameter in the command.

The example below demonstrates the use of `--range` to download a specific byte range from an object. Note the byte ranges needs to be prefixed with "bytes=":

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

For more information about retrieving objects, see Getting Objects in the *Amazon S3 Developer Guide*.

- For API details, see GetObject in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

- For API details, see GetObject in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Read data as a byte array using an S3Client.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>
```

```
                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Use an S3TransferManager to download an object in an S3 bucket to a local file. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
```

```
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Read tags that belong to an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            GetObjectTaggingResponse tags =
 s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Get a URL for an object using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();
```

```
            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Get an object by using the S3Presigner client object using an S3Client.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
```

```
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents a text file.\s
                """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
                .region(region)
                .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
```

```
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
}
```

Get an object by using a ResponseTransformer object and S3Client.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }
```

```
    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see GetObject in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Download the object.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
  });

  try {
    const response = await client.send(command);
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see GetObject in *AWS SDK for JavaScript API Reference*.

Kotlin

**SDK for Kotlin**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
```

```
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- For API details, see GetObject in *AWS SDK for Kotlin API reference*.

PHP

**SDK for PHP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get an object.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- For API details, see GetObject in *AWS SDK for PHP API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command retrieves item "sample.txt" from bucket "test-files" and saves it to a file named "local-sample.txt" in the current location. The file "local-sample.txt" does not have to exist before this command is called.**

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

**Example 2: This command retrieves virtual directory "DIR" from bucket "test-files" and saves it to a folder named "Local-DIR" in the current location. The folder "Local-DIR" does not have to exist before this command is called.**

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

**Example 3: Downloads all objects with keys ending in '.json' from buckets with 'config' in the bucket name to files in the specified folder. The object keys are used to set the filenames.**

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- For API details, see GetObject in *AWS Tools for PowerShell Cmdlet Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
            raise
        else:
            return body
```

- For API details, see GetObject in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get an object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
```

```
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Get an object and report its server-side encryption state.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see GetObject in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, anyhow::Error> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone())?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
```

```
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await? {
        let bytes_len = bytes.len();
        file.write_all(&bytes)?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
}
```

- For API details, see GetObject in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                    iv_bucket = iv_bucket_name
                    iv_key = iv_object_key
                ).
        DATA(lv_object_data) = oo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see GetObject in *AWS SDK for SAP ABAP API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Download an object from a bucket to a local file.

```
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

Read an object into a Swift Data object.

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }
```

- For API details, see GetObject in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObjectAcl` with an AWS SDK or CLI

The following code examples show how to use `GetObjectAcl`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Manage access control lists (ACLs)

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```

```
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
```

```
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- For API details, see GetObjectAcl in *AWS SDK for C++ API Reference*.

CLI

### AWS CLI

The following command retrieves the access control list for an object in a bucket named `my-bucket`:

```
aws s3api get-object-acl --bucket my-bucket --key index.html
```

Output:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

- For API details, see GetObjectAcl in *AWS CLI Command Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- For API details, see GetObjectAcl in *AWS SDK for Kotlin API reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- For API details, see GetObjectAcl in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `GetObjectAttributes` with an AWS SDK or CLI

The following code examples show how to use `GetObjectAttributes`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Work with Amazon S3 object integrity

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
```

```
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
```

```
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
```

```
                            return false;
                        }
                    }
                } else {
                    std::cerr << "Error retrieving object attributes for object
 parts." <<
                              outcome.GetError().GetMessage() << std::endl;
                    return false;
                }
            }
        }

        return true;
}
```

- For API details, see GetObjectAttributes in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

**To retrieves metadata from an object without returning the object itself**

The following `get-object-attributes` example retrieves metadata from the object `doc1.rtf`.

```
aws s3api get-object-attributes \
    --bucket my-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"
```

Output:

```
{
    "LastModified": "2022-03-15T19:37:31+00:00",
    "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
    "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
    "StorageClass": "STANDARD",
    "ObjectSize": 405
}
```

For more information, see GetObjectAttributes in the Amazon S3 API Reference.

- For API details, see GetObjectAttributes in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObjectLegalHold` with an AWS SDK or CLI

The following code examples show how to use `GetObjectLegalHold`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Get the legal hold configuration of an object
- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
```

```
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
 {bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- For API details, see GetObjectLegalHold in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**Retrieves the Legal Hold status of an object**

The following `get-object-legal-hold` example retrieves the Legal Hold status for the specified object.

```
aws s3api get-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

Output:

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- For API details, see GetObjectLegalHold in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
```

```
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- For API details, see GetObjectLegalHold in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
```

```
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
 ex.getMessage() + "'");
        }

        return null;
    }
```

- For API details, see GetObjectLegalHold in *AWS SDK for Java 2.x API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put an object legal hold.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
 bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
```

```
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- For API details, see GetObjectLegalHold in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObjectLockConfiguration` with an AWS SDK or CLI

The following code examples show how to use `GetObjectLockConfiguration`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To retrieve an object lock configuration for a bucket**

The following `get-object-lock-configuration` example retrieves the object lock configuration for the specified bucket.

```
aws s3api get-object-lock-configuration \
    --bucket my-bucket-with-object-lock
```

Output:

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
            }
        }
    }
}
```

- For API details, see GetObjectLockConfiguration in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Get the object lock configuration details for an S3 bucket.
   public void getBucketObjectLockConfiguration(String bucketName) {
       GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
           .bucket(bucketName)
           .build();

       GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
       System.out.println("Bucket object lock config for "+bucketName +":  ");
       System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
       System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
   }
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  GetObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
```

```
export const main = async (client, bucketName) => {
  const command = new GetObjectLockConfigurationCommand({
    Bucket: bucketName,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
  });

  try {
    const { ObjectLockConfiguration } = await client.send(command);
    console.log(`Object Lock Configuration: ${ObjectLockConfiguration}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the value 'Enabled' if Object lock configuration is enabled for the given S3 bucket.**

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

**Output:**

```
Value
-----
Enabled
```

- For API details, see GetObjectLockConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get the object lock configuration.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
 "Enabled"
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- For API details, see GetObjectLockConfiguration in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObjectRetention` with an AWS SDK or CLI

The following code examples show how to use `GetObjectRetention`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
```

```
            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- For API details, see GetObjectRetention in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To retrieve the object retention configuration for an object**

The following `get-object-retention` example retrieves the object retention configuration for the specified object.

```
aws s3api get-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

Output:

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- For API details, see GetObjectRetention in *AWS CLI Command Reference*.

Go

### SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
```

```
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- For API details, see GetObjectRetention in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            return null;
        }
    }
```

- For API details, see GetObjectRetention in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { GetObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const { Retention } = await client.send(command);
    console.log(`Object Retention Settings: ${Retention.Status}`);
  } catch (err) {
    console.error(err);
```

```
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- For API details, see GetObjectRetention in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The command returns the mode and date till the object would be retained.**

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- For API details, see GetObjectRetention in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `GetObjectTagging` with an AWS SDK or CLI

The following code examples show how to use `GetObjectTagging`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with tags

CLI

**AWS CLI**

**To retrieve the tags attached to an object**

The following `get-object-tagging` example retrieves the values for the specified key from the specified object.

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

Output:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
}
```

The following `get-object-tagging` example tries to retrieve the tag sets of the object `doc2.rtf`, which has no tags.

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc2.rtf
```

Output:

```
{
    "TagSet": []
}
```

The following `get-object-tagging` example retrieves the tag sets of the object `doc3.rtf`, which has multiple tags.

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc3.rtf
```

Output:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
            "Key": "team"
        }
    ]
}
```

- For API details, see GetObjectTagging in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The sample returns the tags associated with the object present on the given S3 bucket.**

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

**Output:**

```
Key  Value
---  -----
test value
```

- For API details, see GetObjectTagging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `GetPublicAccessBlock` with an AWS SDK or CLI

The following code examples show how to use `GetPublicAccessBlock`.

CLI

### AWS CLI

#### To set or modify the block public access configuration for a bucket

The following `get-public-access-block` example displays the block public access configuration for the specified bucket.

```
aws s3api get-public-access-block \
    --bucket my-bucket
```

Output:

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- For API details, see GetPublicAccessBlock in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: The command returns the public access block configuration of the given S3 bucket.

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- For API details, see GetPublicAccessBlock in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use HeadBucket with an AWS SDK or CLI

The following code examples show how to use HeadBucket.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
###############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
###############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
```

```
    return 1 # 1 in Bash script means false.
  fi
}
```

- For API details, see HeadBucket in *AWS CLI Command Reference*.

CLI

**AWS CLI**

The following command verifies access to a bucket named `my-bucket`:

```
aws s3api head-bucket --bucket my-bucket
```

If the bucket exists and you have access to it, no output is returned. Otherwise, an error message will be shown. For example:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- For API details, see HeadBucket in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
```

```
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- For API details, see HeadBucket in *AWS SDK for Go API Reference*.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- For API details, see HeadBucket in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `HeadObject` with an AWS SDK or CLI

The following code examples show how to use `HeadObject`.

CLI

### AWS CLI

The following command retrieves metadata for an object in a bucket named `my-bucket`:

```
aws s3api head-object --bucket my-bucket --key index.html
```

Output:

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- For API details, see HeadObject in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Determine the content type of an object.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }
```

```
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Get the restore status of an object.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see HeadObject in *AWS SDK for Java 2.x API Reference*.

Ruby

### SDK for Ruby

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see HeadObject in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListBucketAnalyticsConfigurations` with an AWS SDK or CLI

The following code examples show how to use `ListBucketAnalyticsConfigurations`.

CLI

### AWS CLI

#### To retrieve a list of analytics configurations for a bucket

The following `list-bucket-analytics-configurations` retrieves a list of analytics configurations for the specified bucket.

```
aws s3api list-bucket-analytics-configurations \
    --bucket my-bucket
```

Output:

```
{
    "AnalyticsConfigurationList": [
        {
            "StorageClassAnalysis": {},
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- For API details, see ListBucketAnalyticsConfigurations in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the first 100 analytics configurations of the given S3 bucket.**

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- For API details, see ListBucketAnalyticsConfigurations in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListBucketInventoryConfigurations` with an AWS SDK or CLI

The following code examples show how to use `ListBucketInventoryConfigurations`.

CLI

**AWS CLI**

**To retrieve a list of inventory configurations for a bucket**

The following `list-bucket-inventory-configurations` example lists the inventory configurations for the specified bucket.

```
aws s3api list-bucket-inventory-configurations \
    --bucket my-bucket
```

Output:

```
{
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
```

```
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- For API details, see ListBucketInventoryConfigurations in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns the first 100 inventory configurations of the given S3 bucket.**

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- For API details, see ListBucketInventoryConfigurations in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListBuckets` with an AWS SDK or CLI

The following code examples show how to use `ListBuckets`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
```

```
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- For API details, see ListBuckets in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- For API details, see ListBuckets in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command uses the `list-buckets` command to display the names of all your Amazon S3 buckets (across all regions):

```
aws s3api list-buckets --query "Buckets[].Name"
```

The query option filters the output of `list-buckets` down to only the bucket names.

For more information about buckets, see Working with Amazon S3 Buckets in the *Amazon S3 Developer Guide*.

- For API details, see ListBuckets in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}
```

- For API details, see ListBuckets in *AWS SDK for Go API Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
```

```
}
```

- For API details, see ListBuckets in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List the buckets.

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new ListBucketsCommand({});

  try {
    const { Owner, Buckets } = await client.send(command);
    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see ListBuckets in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command returns all S3 buckets.**

```
Get-S3Bucket
```

**Example 2: This command returns bucket named "test-files"**

```
Get-S3Bucket -BucketName test-files
```

- For API details, see ListBuckets in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
```

```
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
 Boto3
                            that contains collections and factory methods to
 create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- For API details, see ListBuckets in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end
```

```
  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see ListBuckets in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
```

```
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- For API details, see ListBuckets in *AWS SDK for Rust API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// Return an array containing information about every available bucket.
///
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
///   each bucket.
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
    let output = try await client.listBuckets(input: ListBucketsInput())

    guard let buckets = output.buckets else {
        return []
    }
    return buckets
}
```

- For API details, see ListBuckets in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListMultipartUploads` with an AWS SDK or CLI

The following code examples show how to use `ListMultipartUploads`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Delete incomplete multipart uploads

CLI

**AWS CLI**

The following command lists all of the active multipart uploads for a bucket named `my-bucket`:

```
aws s3api list-multipart-uploads --bucket my-bucket
```

Output:

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
    "CommonPrefixes": []
}
```

In progress multipart uploads incur storage costs in Amazon S3. Complete or abort an active multipart upload to remove its parts from your account.

- For API details, see ListMultipartUploads in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListMultipartUploads in *AWS SDK for Java 2.x API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `ListObjectVersions` with an AWS SDK or CLI

The following code examples show how to use `ListObjectVersions`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Work with versioned objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
            // for the AWS Region to the client constructor like this:
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
```

```
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size:
 {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
```

```
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- For API details, see ListObjectVersions in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

The following command retrieves version information for an object in a bucket named `my-bucket`:

```
aws s3api list-object-versions --bucket my-bucket --prefix index.html
```

Output:

```
{
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": true,
```

```
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
```

```
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- For API details, see ListObjectVersions in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
```

```
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- For API details, see ListObjectVersions in *AWS SDK for Go API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
```

```
    }

    Ok(())
}
```

- For API details, see ListObjectVersions in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListObjects` with an AWS SDK or CLI

The following code examples show how to use `ListObjects`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create a web page that lists Amazon S3 objects

CLI

**AWS CLI**

The following example uses the `list-objects` command to display the names of all the objects in the specified bucket:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

The example uses the `--query` argument to filter the output of `list-objects` down to the key value and size for each object

For more information about objects, see Working with Amazon S3 Objects in the *Amazon S3 Developer Guide*.

- For API details, see ListObjects in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command retrieves the information about all of the items in the bucket "test-files".**

```
Get-S3Object -BucketName test-files
```

**Example 2: This command retrieves the information about the item "sample.txt" from bucket "test-files".**

```
Get-S3Object -BucketName test-files -Key sample.txt
```

**Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "test-files".**

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- For API details, see ListObjects in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `ListObjectsV2` with an AWS SDK or CLI

The following code examples show how to use `ListObjectsV2`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with buckets and objects

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Shows how to list the objects in an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket for which to list
        /// the contents.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// copy operation.</returns>
        public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client,
string bucketName)
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 5,
                };

                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");

                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);

                    response.S3Objects
```

```
                        .ForEach(obj => Console.WriteLine($"{obj.Key,-35}
{obj.LastModified.ToShortDateString(),10}{obj.Size,10}"));

                    // If the response is truncated, set the request
 ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' getting list of objects.");
                return false;
            }
        }
```

List objects with a paginator.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "doc-example-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
```

```
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- For API details, see ListObjectsV2 in *AWS SDK for .NET API Reference*.

Bash

### AWS CLI with Bash script

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
```

```
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- For API details, see ListObjectsV2 in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- For API details, see ListObjectsV2 in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

**To get a list of objects in a bucket**

The following `list-objects-v2` example lists the objects in the specified bucket.

```
aws s3api list-objects-v2 \
    --bucket my-bucket
```

Output:

```
{
    "Contents": [
```

```
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
    ]
}
```

- For API details, see ListObjectsV2 in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}
```

- For API details, see ListObjectsV2 in *AWS SDK for Go API Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + "
 KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
    }
}
```

List objects using pagination.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .maxKeys(1)
                    .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())
```

```
                    .forEach(content -> System.out.println(" Key: " +
 content.key() + " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListObjectsV2 in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List all of the objects in your bucket. If there is more than one object, IsTruncated and NextContinuationToken will be used to iterate over the full list.

```
import {
  S3Client,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new ListObjectsV2Command({
    Bucket: "my-bucket",
    // The default and maximum number of keys returned is 1000. This limits it to
    // one for demonstration purposes.
    MaxKeys: 1,
  });
```

```
  try {
    let isTruncated = true;

    console.log("Your bucket contains the following objects:\n");
    let contents = "";

    while (isTruncated) {
      const { Contents, IsTruncated, NextContinuationToken } =
        await client.send(command);
      const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
      contents += contentsList + "\n";
      isTruncated = IsTruncated;
      command.input.ContinuationToken = NextContinuationToken;
    }
    console.log(contents);
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see ListObjectsV2 in *AWS SDK for JavaScript API Reference*.

Kotlin

**SDK for Kotlin**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
```

```
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- For API details, see ListObjectsV2 in *AWS SDK for Kotlin API reference*.

PHP

**SDK for PHP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List objects in a bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- For API details, see ListObjectsV2 in *AWS SDK for PHP API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
 are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
```

```
                "Got objects %s from bucket '%s'", [o.key for o in objects],
 bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
 bucket.name)
            raise
        else:
            return objects
```

- For API details, see ListObjectsV2 in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
```

```
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see ListObjectsV2 in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
```

```
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- For API details, see ListObjectsV2 in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see ListObjectsV2 in *AWS SDK for SAP ABAP API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public func listBucketFiles(bucket: String) async throws -> [String] {
    let input = ListObjectsV2Input(
        bucket: bucket
    )
    let output = try await client.listObjectsV2(input: input)
    var names: [String] = []

    guard let objList = output.contents else {
        return []
    }

    for obj in objList {
        if let objName = obj.key {
            names.append(objName)
        }
    }

    return names
}
```

- For API details, see ListObjectsV2 in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketAccelerateConfiguration` with an AWS SDK or CLI

The following code examples show how to use `PutBucketAccelerateConfiguration`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
```

```
            const string bucketName = "doc-example-bucket";

            await EnableAccelerationAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method sets the configuration to enable transfer acceleration
        /// for the bucket referred to in the bucketName parameter.
        /// </summary>
        /// <param name="client">An Amazon S3 client used to enable the
        /// acceleration on an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
```

```
    }
```

- For API details, see PutBucketAccelerateConfiguration in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To set the accelerate configuration of a bucket**

The following `put-bucket-accelerate-configuration` example enables the accelerate configuration for the specified bucket.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket my-bucket \
    --accelerate-configuration Status=Enabled
```

This command produces no output.

- For API details, see PutBucketAccelerateConfiguration in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command enables the transfer acceleration for the given S3 bucket.**

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- For API details, see PutBucketAccelerateConfiguration in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `PutBucketAcl` with an AWS SDK or CLI

The following code examples show how to use `PutBucketAcl`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Manage access control lists (ACLs)

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
```

```
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- For API details, see PutBucketAcl in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
```

```
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- For API details, see PutBucketAcl in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

This example grants `full control` to two AWS users (*user1@example.com* and *user2@example.com*) and `read` permission to everyone:

```
aws s3api put-bucket-acl --bucket MyBucket --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

See http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html for details on custom ACLs (the s3api ACL commands, such as `put-bucket-acl`, use the same shorthand argument notation).

- For API details, see PutBucketAcl in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <id>\s

                Where:
                  bucketName - The Amazon S3 bucket to grant permissions on.\s
                  id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                    .grantee(builder -> builder.id(id)
                            .type(Type.CANONICAL_USER))
```

```
                    .permission(Permission.FULL_CONTROL)
                    .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                    .owner(builder -> builder.id(id))
                    .grants(grantList2)
                    .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                    .bucket(bucketName)
                    .accessControlPolicy(acl)
                    .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketAcl in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put the bucket ACL.

```
import { PutBucketAclCommand, S3Client } from "@aws-sdk/client-s3";
```

```
const client = new S3Client({});

// Most Amazon S3 use cases don't require the use of access control lists (ACLs).
// We recommend that you disable ACLs, except in unusual circumstances where
// you need to control access for each object individually.
// Consider a policy instead. For more information see https://
docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.
export const main = async () => {
  // Grant a user READ access to a bucket.
  const command = new PutBucketAclCommand({
    Bucket: "test-bucket",
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: "canonical-id-1",
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "FULL_CONTROL",
        },
      ],
      Owner: {
        ID: "canonical-id-2",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see PutBucketAcl in *AWS SDK for JavaScript API Reference*.

Kotlin

**SDK for Kotlin**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
```

```
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- For API details, see PutBucketAcl in *AWS SDK for Kotlin API reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
```

```
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- For API details, see PutBucketAcl in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketCors` with an AWS SDK or CLI

The following code examples show how to use `PutBucketCors`.

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- For API details, see PutBucketCors in *AWS SDK for .NET API Reference*.

CLI

### AWS CLI

The following example enables PUT, POST, and DELETE requests from *www.example.com*, and enables GET requests from any domain:

```
aws s3api put-bucket-cors --bucket MyBucket --cors-configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- For API details, see PutBucketCors in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <accountId>\s

                Where:
                    bucketName - The Amazon S3 bucket to upload an object into.
                    accountId - The id of the account that owns the Amazon S3
 bucket.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }
```

```
   public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
       try {
           DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                   .bucket(bucketName)
                   .expectedBucketOwner(accountId)
                   .build();

           s3.deleteBucketCors(bucketCorsRequest);

       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
       try {
           GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
                   .bucket(bucketName)
                   .expectedBucketOwner(accountId)
                   .build();

           GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
           List<CORSRule> corsRules = corsResponse.corsRules();
           for (CORSRule rule : corsRules) {
               System.out.println("allowOrigins: " + rule.allowedOrigins());
               System.out.println("AllowedMethod: " + rule.allowedMethods());
           }

       } catch (S3Exception e) {

           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
       List<String> allowMethods = new ArrayList<>();
```

```
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                    .allowedMethods(allowMethods)
                    .allowedOrigins(allowOrigins)
                    .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                    .corsRules(corsRules)
                    .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .corsConfiguration(configuration)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketCors in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Add a CORS rule.

```
import { PutBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// By default, Amazon S3 doesn't allow cross-origin requests. Use this command
// to explicitly allow cross-origin requests.
export const main = async () => {
  const command = new PutBucketCorsCommand({
    Bucket: "test-bucket",
    CORSConfiguration: {
      CORSRules: [
        {
          // Allow all headers to be sent to this bucket.
          AllowedHeaders: ["*"],
          // Allow only GET and PUT methods to be sent to this bucket.
          AllowedMethods: ["GET", "PUT"],
          // Allow only requests from the specified origin.
          AllowedOrigins: ["https://www.example.com"],
          // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
          // The entity tag represents a specific version of the object. The ETag
 reflects
          // changes only to the contents of an object, not its metadata.
          ExposeHeaders: ["ETag"],
          // How long the requesting browser should cache the preflight response.
 After
          // this time, the preflight request will have to be made again.
          MaxAgeSeconds: 3600,
        },
      ],
    },
```

```
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see PutBucketCors in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
 are
```

```
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- For API details, see PutBucketCors in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
```

```
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- For API details, see PutBucketCors in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketEncryption` with an AWS SDK or CLI

The following code examples show how to use `PutBucketEncryption`.

CLI

### AWS CLI

#### To configure server-side encryption for a bucket

The following `put-bucket-encryption` example sets AES256 encryption as the default for the specified bucket.

```
aws s3api put-bucket-encryption \
    --bucket my-bucket \
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

This command produces no output.

- For API details, see PutBucketEncryption in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command enables default AES256 server side encryption with Amazon S3 Managed Keys(SSE-S3) on the given bucket.**

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- For API details, see PutBucketEncryption in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketLifecycleConfiguration` with an AWS SDK or CLI

The following code examples show how to use `PutBucketLifecycleConfiguration`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Delete incomplete multipart uploads
- Work with versioned objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- For API details, see PutBucketLifecycleConfiguration in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

The following command applies a lifecycle configuration to a bucket named `my-bucket`:

```
aws s3api put-bucket-lifecycle-configuration --bucket my-bucket --lifecycle-
configuration  file://lifecycle.json
```

The file `lifecycle.json` is a JSON document in the current folder that specifies two rules:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

The first rule moves files with the prefix `rotated` to Glacier on the specified date. The second rule moves old object versions to Glacier when they are no longer current. For information on acceptable timestamp formats, see Specifying Parameter Values in the *AWS CLI User Guide*.

- For API details, see PutBucketLifecycleConfiguration in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
        public static void main(String[] args) {
                final String usage = """
```

```
                                Usage:
                                  <bucketName> <accountId>\s

                                Where:
                                  bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                                  accountId - The id of the account that owns the
Amazon S3 bucket.
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String bucketName = args[0];
                String accountId = args[1];
                Region region = Region.US_EAST_1;
                S3Client s3 = S3Client.builder()
                                .region(region)
                                .build();

                setLifecycleConfig(s3, bucketName, accountId);
                getLifecycleConfig(s3, bucketName, accountId);
                deleteLifecycleConfig(s3, bucketName, accountId);
                System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
                s3.close();
        }

        public static void setLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
                        // S3 Glacier.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        Transition transition = Transition.builder()
```

```
.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("Archive immediately rule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Create a second rule.
                        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        List<Transition> transitionList = new ArrayList<>();
                        transitionList.add(transition2);

                        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        LifecycleRule rule2 = LifecycleRule.builder()
                                        .id("Archive and then delete rule")
                                        .filter(ruleFilter2)
                                        .transitions(transitionList)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the LifecycleRule objects to an ArrayList.
                        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
                        ruleList.add(rule1);
                        ruleList.add(rule2);

                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(ruleList)
                                        .build();
```

```
                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Retrieve the configuration and add a new rule.
        public static void getLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
                        List<LifecycleRule> newList = new ArrayList<>();
                        List<LifecycleRule> rules = response.rules();
                        for (LifecycleRule rule : rules) {
                                newList.add(rule);
                        }

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("YearlyDocuments/")
                                        .build();
```

```
                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(3650)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("NewRule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the new rule to the list.
                        newList.add(rule1);
                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(newList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Delete the configuration from the Amazon S3 bucket.
        public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
```

```
                        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest
 = DeleteBucketLifecycleRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- For API details, see PutBucketLifecycleConfiguration in *AWS SDK for Java 2.x API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
configuration can
        be used to archive or delete the objects in the bucket according to
specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- For API details, see PutBucketLifecycleConfiguration in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketLogging` with an AWS SDK or CLI

The following code examples show how to use `PutBucketLogging`.

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();
```

```
            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
```

```
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
```

```
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
```

```
        }
    }
```

- For API details, see PutBucketLogging in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**Example 1: To set bucket policy logging**

The following `put-bucket-logging` example sets the logging policy for *MyBucket*. First, grant the logging service principal permission in your bucket policy using the `put-bucket-policy` command.

```
aws s3api put-bucket-policy \
    --bucket MyBucket \
    --policy file://policy.json
```

Contents of `policy.json`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyBucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

To apply the logging policy, use `put-bucket-logging`.

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

Contents of `logging.json`:

```
{
     "LoggingEnabled": {
         "TargetBucket": "MyBucket",
         "TargetPrefix": "Logs/"
     }
 }
```

The `put-bucket-policy` command is required to grant `s3:PutObject` permissions to the logging service principal.

For more information, see Amazon S3 Server Access Logging in the *Amazon S3 User Guide*.

**Example 2: To set a bucket policy for logging access to only a single user**

The following `put-bucket-logging` example sets the logging policy for *MyBucket*. The AWS user *bob@example.com* will have full control over the log files, and no one else has any access. First, grant S3 permission with `put-bucket-acl`.

```
aws s3api put-bucket-acl \
    --bucket MyBucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

Then apply the logging policy using `put-bucket-logging`.

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

Contents of `logging.json`:

```
{
    "LoggingEnabled": {
```

```
        "TargetBucket": "MyBucket",
        "TargetPrefix": "MyBucketLogs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

the `put-bucket-acl` command is required to grant S3's log delivery system the necessary permissions (write and read-acp permissions).

For more information, see Amazon S3 Server Access Logging in the *Amazon S3 Developer Guide*.

- For API details, see PutBucketLogging in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketNotification` with an AWS SDK or CLI

The following code examples show how to use `PutBucketNotification`.

CLI

### AWS CLI

The applies a notification configuration to a bucket named `my-bucket`:

```
aws s3api put-bucket-notification --bucket my-bucket --notification-
configuration file://notification.json
```

The file `notification.json` is a JSON document in the current folder that specifies an SNS topic and an event type to monitor:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

The SNS topic must have an IAM policy attached to it that allows Amazon S3 to publish to it:

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:my-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
    }
   }
  }
 ]
}
```

- For API details, see PutBucketNotification in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This example configures the SNS topic configuration for the S3 event ObjectRemovedDelete and enables notification for the given s3 bucket**

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
```

```
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName kt-tools -TopicConfiguration $topic
```

**Example 2: This example enables notifications of ObjectCreatedAll for the given bucket sending it to Lambda function.**

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $lambdaConfig
```

**Example 3: This example creates 2 different Lambda configuration on the basis of different key-suffix and configured both in a single command.**

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
```

```
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $firstLambdaConfig,$secondlambdaConfig
```

- For API details, see PutBucketNotification in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketNotificationConfiguration` with an AWS SDK or CLI

The following code examples show how to use `PutBucketNotificationConfiguration`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Process S3 event notifications
- Send event notifications to EventBridge

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic,
 sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
```

```
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
```

```
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- For API details, see PutBucketNotificationConfiguration in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To enable the specified notifications to a bucket**

The following `put-bucket-notification-configuration` example applies a notification configuration to a bucket named `my-bucket`. The file `notification.json` is a JSON document in the current folder that specifies an SNS topic and an event type to monitor.

```
aws s3api put-bucket-notification-configuration \
    --bucket my-bucket \
    --notification-configuration file://notification.json
```

Contents of `notification.json`:

```
{
    "TopicConfigurations": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
}
```

The SNS topic must have an IAM policy attached to it that allows Amazon S3 to publish to it.

```
{
```

```
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
                }
            }
        }
    ]
}
```

- For API details, see PutBucketNotificationConfiguration in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketPolicy` with an AWS SDK or CLI

The following code examples show how to use `PutBucketPolicy`.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
```

```
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- For API details, see PutBucketPolicy in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

This example allows all users to retrieve any object in *MyBucket* except those in the *MySecretFolder*. It also grants `put` and `delete` permission to the root user of the AWS account `1234-5678-9012`:

```
aws s3api put-bucket-policy --bucket MyBucket --policy file://policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/*"
      },
      {
         "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::MyBucket/MySecretFolder/*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::MyBucket/*"
        }

    ]
}
```

- For API details, see PutBucketPolicy in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <polFile>

                Where:
                    bucketName - The Amazon S3 bucket to set the policy on.
                    polFile - A JSON file containing the policy (see the Amazon
 S3 Readme for an example).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String
 policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
```

```
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
```

```
        }
        return fileText.toString();
    }
}
```

- For API details, see PutBucketPolicy in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Add the policy.

```
import { PutBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutBucketPolicyCommand({
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "AllowGetObject",
          // Allow this particular user to call GetObject on any object in this
 bucket.
          Effect: "Allow",
          Principal: {
            AWS: "arn:aws:iam::ACCOUNT-ID:user/USERNAME",
          },
          Action: "s3:GetObject",
          Resource: "arn:aws:s3:::BUCKET-NAME/*",
        },
      ],
    }),
```

```
    // Apply the preceding policy to this bucket.
    Bucket: "BUCKET-NAME",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see PutBucketPolicy in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
```

```
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- For API details, see PutBucketPolicy in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
```

```
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- For API details, see PutBucketPolicy in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketReplication` with an AWS SDK or CLI

The following code examples show how to use `PutBucketReplication`.

CLI

### AWS CLI

#### To configure replication for an S3 bucket

The following `put-bucket-replication` example applies a replication configuration to the specified S3 bucket.

```
aws s3api put-bucket-replication \
    --bucket AWSDOC-EXAMPLE-BUCKET1 \
    --replication-configuration file://replication.json
```

Contents of `replication.json`:

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
```

```
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2"
            }
        }
    ]
}
```

The destination bucket must have versioning enabled. The specified role must have permission to write to the destination bucket and have a trust relationship that allows Amazon S3 to assume the role.

Example role permission policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
```

```
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2/*"
        }
    ]
}
```

Example trust relationship policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

This command produces no output.

For more information, see This is the topic title in the *Amazon Simple Storage Service Console User Guide*.

- For API details, see PutBucketReplication in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This example sets a replication configuration with a single rule enabling replication to the 'exampletargetbucket' bucket any new objects created with the key name prefix "TaxDocs" in the bucket 'examplebucket'.**

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
```

```
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

**Example 2: This example sets a replication configuration with multiple rules enabling replication to the 'exampletargetbucket' bucket any new objects created with either the key name prefix "TaxDocs" or "OtherDocs". The key prefixes must not overlap.**

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

**Example 3: This example updates the replication configuration on the specified bucket to disable the rule controlling replication of objects with the key name prefix "TaxDocs" to the bucket 'exampletargetbucket'.**

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
```

```
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- For API details, see PutBucketReplication in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketRequestPayment` with an AWS SDK or CLI

The following code examples show how to use `PutBucketRequestPayment`.

CLI

### AWS CLI

#### Example 1: To enable ``requester pays`` configuration for a bucket

The following `put-bucket-request-payment` example enables `requester pays` for the specified bucket.

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

This command produces no output.

#### Example 2: To disable ``requester pays`` configuration for a bucket

The following `put-bucket-request-payment` example disables `requester pays` for the specified bucket.

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

This command produces no output.

- For API details, see PutBucketRequestPayment in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: Updates the request payment configuration for the bucket named 'mybucket' so that the person requesting downloads from the bucket will be charged for the download. By default the bucket owner pays for downloads. To set the request payment back to the default use 'BucketOwner' for the RequestPaymentConfiguration_Payer parameter.**

```
Write-S3BucketRequestPayment -BucketName mybucket -
RequestPaymentConfiguration_Payer Requester
```

- For API details, see PutBucketRequestPayment in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use PutBucketTagging with an AWS SDK or CLI

The following code examples show how to use `PutBucketTagging`.

CLI

**AWS CLI**

The following command applies a tagging configuration to a bucket named `my-bucket`:

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging file://tagging.json
```

The file `tagging.json` is a JSON document in the current folder that specifies tags:

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

Or apply a tagging configuration to `my-bucket` directly from the command line:

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- For API details, see PutBucketTagging in *AWS CLI Command Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: This command applies two tags to a bucket named `cloudtrail-test-2018`: a tag with a key of Stage and a value of Test, and a tag with a key of Environment and a value of Alpha. To verify that the tags were added to the bucket, run `Get-S3BucketTagging -BucketName bucket_name`. The results should show the tags that you applied to the bucket in the first command. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket. To add or delete individual tags, run the Resource Groups and Tagging API cmdlets, `Add-RGTResourceTag` and `Remove-RGTResourceTag`. Alternatively, use Tag Editor in the AWS Management Console to manage S3 bucket tags.**

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

**Example 2: This command pipes a bucket named `cloudtrail-test-2018` into the `Write-S3BucketTagging` cmdlet. It applies tags Stage:Production and Department:Finance to the bucket. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket.**

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- For API details, see PutBucketTagging in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketVersioning` with an AWS SDK or CLI

The following code examples show how to use `PutBucketVersioning`.

CLI

### AWS CLI

The following command enables versioning on a bucket named `my-bucket`:

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-
configuration Status=Enabled
```

The following command enables versioning, and uses an mfa code

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-
configuration Status=Enabled --mfa "SERIAL 123456"
```

- For API details, see PutBucketVersioning in *AWS CLI Command Reference*.

PowerShell

### Tools for PowerShell

#### Example 1: The command enables versioning for the given S3 bucket.

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status
 Enabled
```

- For API details, see PutBucketVersioning in *AWS Tools for PowerShell Cmdlet Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutBucketWebsite` with an AWS SDK or CLI

The following code examples show how to use `PutBucketWebsite`.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- For API details, see PutBucketWebsite in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- For API details, see PutBucketWebsite in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The applies a static website configuration to a bucket named `my-bucket`:

```
aws s3api put-bucket-website --bucket my-bucket --website-configuration file://
website.json
```

The file `website.json` is a JSON document in the current folder that specifies index and error pages for the website:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- For API details, see PutBucketWebsite in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName> [indexdoc]\s

                Where:
                   bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
                   indexdoc - The index document, ex. 'index.html'
                              If not specified, 'index.html' will be set.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }
```

```
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()

 .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                    .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                    .bucket(bucketName)
                    .websiteConfiguration(websiteConfig)
                    .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutBucketWebsite in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the website configuration.

```
import { PutBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";
```

```
const client = new S3Client({});

// Set up a bucket as a static website.
// The bucket needs to be publicly accessible.
export const main = async () => {
  const command = new PutBucketWebsiteCommand({
    Bucket: "test-bucket",
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request that is for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see PutBucketWebsite in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The command enables website hosting for the given bucket with the index document as 'index.html' and error document as 'error.html'.**

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- For API details, see PutBucketWebsite in *AWS Tools for PowerShell Cmdlet Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see PutBucketWebsite in *AWS SDK for Ruby API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutObject` with an AWS SDK or CLI

The following code examples show how to use `PutObject`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Get started with buckets and objects
- Track uploads and downloads
- Work with Amazon S3 object integrity

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        /// <summary>
        /// Shows how to upload a file from the local computer to an Amazon S3
        /// bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The Amazon S3 bucket to which the object
        /// will be uploaded.</param>
        /// <param name="objectName">The object to upload.</param>
        /// <param name="filePath">The path, including file name, of the object
        /// on the local computer to upload.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// upload procedure.</returns>
        public static async Task<bool> UploadFileAsync(
            IAmazonS3 client,
            string bucketName,
            string objectName,
            string filePath)
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await client.PutObjectAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully uploaded {objectName} to
{bucketName}.");
                return true;
```

```
            }
            else
            {
                Console.WriteLine($"Could not upload {objectName} to
{bucketName}.");
                return false;
            }
        }
```

Upload an object with server-side encryption.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
```

```
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

- For API details, see PutObject in *AWS SDK for .NET API Reference*.

Bash

**AWS CLI with Bash script**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3
```

```
  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- For API details, see PutObject in *AWS CLI Command Reference*.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
```

```
                                            std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- For API details, see PutObject in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following example uses the `put-object` command to upload an object to Amazon S3:

```
aws s3api put-object --bucket text-content --key dir-1/my_images.tar.bz2 --
body my_images.tar.bz2
```

The following example shows an upload of a video file (The video file is specified using Windows file system syntax.):

```
aws s3api put-object --bucket text-content --key dir-1/big-video-file.mp4 --body
 e:\media\videos\f-sharp-3-data-services.mp4
```

For more information about uploading objects, see Uploading Objects in the *Amazon S3 Developer Guide*.

- For API details, see PutObject in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put an object in a bucket by using the low-level API.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
```

```
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}
```

Upload an object to a bucket by using a transfer manager.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
```

```
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- For API details, see PutObject in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a file to a bucket using an S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
```

```
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Use an S3TransferManager to upload a file to a bucket. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
```

```
        return uploadResult.response().eTag();
    }
```

Upload an object to a bucket and set tags using an S3Client.

```
   public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
       try {
           Tag tag1 = Tag.builder()
                   .key("Tag 1")
                   .value("This is tag 1")
                   .build();

           Tag tag2 = Tag.builder()
                   .key("Tag 2")
                   .value("This is tag 2")
                   .build();

           List<Tag> tags = new ArrayList<>();
           tags.add(tag1);
           tags.add(tag2);

           Tagging allTags = Tagging.builder()
                   .tagSet(tags)
                   .build();

           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .tagging(allTags)
                   .build();

           s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

```
   public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
       try {
           GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
           List<Tag> obTags = getTaggingRes.tagSet();
           for (Tag sinTag : obTags) {
               System.out.println("The tag key is: " + sinTag.key());
               System.out.println("The tag value is: " + sinTag.value());
           }

           // Replace the object's tags with two new tags.
           Tag tag3 = Tag.builder()
                   .key("Tag 3")
                   .value("This is tag 3")
                   .build();

           Tag tag4 = Tag.builder()
                   .key("Tag 4")
                   .value("This is tag 4")
                   .build();

           List<Tag> tags = new ArrayList<>();
           tags.add(tag3);
           tags.add(tag4);

           Tagging updatedTags = Tagging.builder()
                   .tagSet(tags)
                   .build();

           PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .tagging(updatedTags)
                   .build();

           s3.putObjectTagging(taggingRequest1);
```

```
            GetObjectTaggingResponse getTaggingRes2 =
 s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Upload an object to a bucket and set metadata using an S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
```

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Upload an object to a bucket and set an object retention value using an S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <key> <bucketName>\s

                Where:
                    key - The name of the object (for example, book.pdf).\s
                    bucketName - The Amazon S3 bucket name that contains the
 object (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }
```

```
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                    .mode("COMPLIANCE")
                    .retainUntilDate(instant)
                    .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
                    .retention(lockRetention)
                    .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutObject in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload the object.

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see PutObject in *AWS SDK for JavaScript API Reference*.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- For API details, see PutObject in *AWS SDK for Kotlin API reference*.

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload an object to a bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- For API details, see PutObject in *AWS SDK for PHP API Reference*.

PowerShell

### Tools for PowerShell

**Example 1: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files".**

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

**Example 2: This command uploads the single file "sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files". If the -Key parameter is not supplied, the filename is used as the S3 object key.**

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

**Example 3: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "prefix/to/sample.txt" in bucket "test-files".**

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

**Example 4: This command uploads all files in the subdirectory "Scripts" to the bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename" where 'filename' varies.**

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

**Example 5: This command uploads all *.ps1 files in the local director "Scripts" to bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename.ps1" where 'filename' varies.**

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
 -SearchPattern *.ps1
```

**Example 6: This command creates a new S3 object containing the specified content string with key 'sample.txt'.**

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object
 contents"
```

**Example 7: This command uploads the specified file (the filename is used as the key) and applies the specified tags to the new object.**

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

**Example 8: This command recursively uploads the specified folder and applies the specified tags to all the new objects.**

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse
 -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- For API details, see PutObject in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
 When this
                     argument is a string, it is interpreted as a file name,
 which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
```

```
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

- For API details, see PutObject in *AWS SDK for Python (Boto3) API Reference*.

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a file using a managed uploader (Object.upload_file).

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Upload a file using Object.put.

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Upload a file using Object.put and add server-side encryption.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
```

```
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see PutObject in *AWS SDK for Ruby API Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}
```

- For API details, see PutObject in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
            iv_body = lv_body
        ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see PutObject in *AWS SDK for SAP ABAP API reference*.

Swift

**SDK for Swift**

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a file from local storage to a bucket.

```
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

Upload the contents of a Swift Data object to a bucket.

```
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

- For API details, see PutObject in *AWS SDK for Swift API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutObjectAcl` with an AWS SDK or CLI

The following code examples show how to use `PutObjectAcl`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Manage access control lists (ACLs)

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                               const Aws::String &granteePermission, const
 Aws::String &granteeType,
                               const Aws::String &granteeID, const Aws::String
 &granteeEmailAddress,
                               const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
```

```
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
            s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
```

```
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- For API details, see PutObjectAcl in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command grants `full control` to two AWS users (*user1@example.com* and *user2@example.com*) and `read` permission to everyone:

```
aws s3api put-object-acl --bucket MyBucket --key file.txt --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

See http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html for details on custom ACLs (the s3api ACL commands, such as `put-object-acl`, use the same shorthand argument notation).

- For API details, see PutObjectAcl in *AWS CLI Command Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
 identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
```

```
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- For API details, see PutObjectAcl in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutObjectLegalHold` with an AWS SDK or CLI

The following code examples show how to use `PutObjectLegalHold`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
```

- For API details, see PutObjectLegalHold in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To apply a Legal Hold to an object**

The following `put-object-legal-hold` example sets a Legal Hold on the object `doc1.rtf`.

```
aws s3api put-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

This command produces no output.

- For API details, see PutObjectLegalHold in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
```

```
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- For API details, see PutObjectLegalHold in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
```

```
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
 PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }
```

- For API details, see PutObjectLegalHold in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
```

```
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: "ON",
    },
    // Optionally, you can provide additional parameters
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object legal hold status: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- For API details, see PutObjectLegalHold in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put an object legal hold.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
 bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
 bucket, e
        )
```

- For API details, see PutObjectLegalHold in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutObjectLockConfiguration` with an AWS SDK or CLI

The following code examples show how to use `PutObjectLockConfiguration`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the object lock configuration of a bucket.

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
```

```
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

Set the default retention period of a bucket.

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
```

```
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- For API details, see [PutObjectLockConfiguration](#) in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To set an object lock configuration on a bucket**

The following `put-object-lock-configuration` example sets a 50-day object lock on the specified bucket.

```
aws s3api put-object-lock-configuration \
    --bucket my-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
 { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

This command produces no output.

- For API details, see PutObjectLockConfiguration in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the object lock configuration of a bucket.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
```

```
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

Set the default retention period of a bucket.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- For API details, see PutObjectLockConfiguration in *AWS SDK for Go API Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the object lock configuration of a bucket.

```
   // Enable object lock on an existing bucket.
   public void enableObjectLockOnBucket(String bucketName) {
       try {
           VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

           PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

           // Enable versioning on the bucket.
           getClient().putBucketVersioning(putBucketVersioningRequest);
           PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

           getClient().putObjectLockConfiguration(request);
           System.out.println("Successfully enabled object lock on
"+bucketName);

       } catch (S3Exception ex) {
           System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
```

```
        }
    }
```

Set the default retention period of a bucket.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();
```

```
getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
       System.out.println("Added a default retention to bucket "+bucketName
+".");
   }
```

- For API details, see PutObjectLockConfiguration in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Set the object lock configuration of a bucket.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
    // Optionally, you can provide additional parameters
```

```
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Lock Configuration updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

Set the default retention period of a bucket.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
      Rule: {
```

```
        DefaultRetention: {
          Mode: "GOVERNANCE",
          Years: 3,
        },
      },
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Default Object Lock Configuration updated: ${response.
$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- For API details, see PutObjectLockConfiguration in *AWS SDK for JavaScript API Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put object lock configuration.

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}},
        )
```

- For API details, see PutObjectLockConfiguration in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `PutObjectRetention` with an AWS SDK or CLI

The following code examples show how to use `PutObjectRetention`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Lock Amazon S3 objects

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
```

```
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
```

- For API details, see [PutObjectRetention](#) in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To set an object retention configuration for an object**

The following `put-object-retention` example sets an object retention configuration for the specified object until 2025-01-01.

```
aws s3api put-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
 "2025-01-01T00:00:00" }'
```

This command produces no output.

- For API details, see PutObjectRetention in *AWS CLI Command Reference*.

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
```

```
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- For API details, see PutObjectRetention in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Set or modify a retention period on an object in an S3 bucket.
   public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
       // Calculate the instant one day from now.
       Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

       // Convert the Instant to a ZonedDateTime object with a specific time
zone.
       ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

       // Define a formatter for human-readable output.
       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
```

```
        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }
```

- For API details, see PutObjectRetention in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
```

```
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    BypassGovernanceRetention: false,
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    Retention: {
      Mode: "GOVERNANCE", // or "COMPLIANCE"
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Retention settings updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- For API details, see PutObjectRetention in *AWS SDK for JavaScript API Reference*.

PowerShell

**Tools for PowerShell**

**Example 1: The command enables governance retention mode untill the date '31st Dec 2019 00:00:00' for 'testfile.txt' object in the given S3 bucket.**

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- For API details, see PutObjectRetention in *AWS Tools for PowerShell Cmdlet Reference*.

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Put an object retention.

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
                BypassGovernanceRetention=True,
            )
```

- For API details, see PutObjectRetention in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Use `RestoreObject` with an AWS SDK or CLI

The following code examples show how to use `RestoreObject`.

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "doc-example-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
```

```
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- For API details, see RestoreObject in *AWS SDK for .NET API Reference*.

CLI

**AWS CLI**

**To create a restore request for an object**

The following `restore-object` example restores the specified Amazon S3 Glacier object for the bucket `my-glacier-bucket` for 10 days.

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

This command produces no output.

- For API details, see RestoreObject in *AWS CLI Command Reference*.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class
 value of Glacier.\s
```

```
                    expectedBucketOwner - The account that owns the bucket (you
can obtain this value from the AWS Management Console).\s
                """";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                    .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                    .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                    .expectedBucketOwner(expectedBucketOwner)
                    .bucket(bucketName)
                    .key(keyName)
                    .restoreRequest(restoreRequest)
                    .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- For API details, see RestoreObject in *AWS SDK for Java 2.x API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `SelectObjectContent` with an AWS SDK or CLI

The following code examples show how to use `SelectObjectContent`.

CLI

**AWS CLI**

**To filter the contents of an Amazon S3 object based on an SQL statement**

The following `select-object-content` example filters the object `my-data-file.csv` with the specified SQL statement and sends output to a file.

```
aws s3api select-object-content \
    --bucket my-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

This command produces no output.

- For API details, see SelectObjectContent in *AWS CLI Command Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The following example shows a query using a JSON object. The complete example also shows the use of a CSV object.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" +
 UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
 c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
```

```
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
```

```
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
```

```
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- For API details, see SelectObjectContent in *AWS SDK for Java 2.x API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use `UploadPart` with an AWS SDK or CLI

The following code examples show how to use `UploadPart`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Perform a multipart upload
- Use checksums
- Work with Amazon S3 object integrity

C++

### SDK for C++

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
 &calculatedHash,
                                                         const
 std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
```

```
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- For API details, see UploadPart in *AWS SDK for C++ API Reference*.

CLI

**AWS CLI**

The following command uploads the first part in a multipart upload initiated with the `create-multipart-upload` command:

```
aws s3api upload-part --bucket my-bucket --key 'multipart/01' --part-number 1 --
body part01 --upload-id
  "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

The body option takes the name or path of a local file for upload (do not use the file:// prefix). The minimum part size is 5 MB. Upload ID is returned by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`. Bucket and key are specified when you create the multipart upload.

Output:

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Save the ETag value of each part for later. They are required to complete the multipart upload.

- For API details, see UploadPart in *AWS CLI Command Reference*.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
```

```
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );

    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();
```

- For API details, see UploadPart in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Scenarios for Amazon S3 using AWS SDKs

The following code examples show you how to implement common scenarios in Amazon S3 with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon S3. Each scenario includes a link to GitHub, where you can find instructions on how to set up and run the code.

**Examples**

- Create a presigned URL for Amazon S3 using an AWS SDK
- A web page that lists Amazon S3 objects using an AWS SDK
- Delete incomplete multipart uploads to Amazon S3 using an AWS SDK
- Download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory
- Get an Amazon S3 object from a Multi-Region Access Point by using an AWS SDK
- Get an object from an Amazon S3 bucket using an AWS SDK, specifying an If-Modified-Since header
- Get started with Amazon S3 buckets and objects using an AWS SDK
- Get started with encryption for Amazon S3 objects using an AWS SDK

- Get started with tags for Amazon S3 objects using an AWS SDK
- Get the legal hold configuration of an Amazon S3 object using an AWS SDK
- Work with Amazon S3 object lock features using an AWS SDK
- Manage access control lists (ACLs) for Amazon S3 buckets using an AWS SDK
- Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK
- Parse Amazon S3 URIs using an AWS SDK
- Perform a multipart copy of an Amazon S3 object using an AWS SDK
- Perform a multipart upload of an Amazon S3 object using an AWS SDK
- Receive and process Amazon S3 event notifications by using an AWS SDK.
- Send S3 event notifications to Amazon EventBridge using an AWS SDK
- Track an Amazon S3 object upload or download using an AWS SDK
- Example approaches for unit and integration testing with an AWS SDK
- Recursively upload a local directory to an Amazon Simple Storage Service (Amazon S3) bucket
- Upload or download large files to and from Amazon S3 using an AWS SDK
- Upload a stream of unknown size to an Amazon S3 object using an AWS SDK
- Use checksums to work with an Amazon S3 object using an AWS SDK
- Work with Amazon S3 object integrity features using an AWS SDK
- Work with Amazon S3 versioned objects using an AWS SDK

## Create a presigned URL for Amazon S3 using an AWS SDK

The following code examples show how to create a presigned URL for Amazon S3 and upload an object.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Generate a presigned URL that can perform an Amazon S3 action for a limited time.

```
    using System;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class GenPresignedUrl
    {
        public static void Main()
        {
            const string bucketName = "doc-example-bucket";
            const string objectKey = "sample.txt";

            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName,
 objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
```

```
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

Generate a presigned URL and perform an upload using that URL.

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
```

```
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
 passed in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
```

```
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Generate a pre-signed URL to download an object.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                      const Aws::String &key,
```

```
                                                        uint64_t expirationSeconds,
                                                        const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

Download using libcurl.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

Generate a pre-signed URL to upload an object.

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
```

```
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

Upload using libcurl.

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
```

```
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
```

```
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that wrap S3 presigning actions.

```
// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(context.TODO(),
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
```

```
 request, err := presigner.PresignClient.PresignPutObject(context.TODO(),
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(bucketName string, objectKey string)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(context.TODO(),
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}
```

Run an interactive example that generates and uses presigned URLs to upload, download, and delete an S3 object.

```


// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
```

```
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 presigning demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
 presignClient := s3.NewPresignClient(s3Client)
 presigner := actions.Presigner{PresignClient: presignClient}

 bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
  "you own or one you want to create:", demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(bucketName)
```

```
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(bucketName, uploadKey)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
 presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
 panic(err)
}
```

```
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Define an HTTP request wrapper used by the example to make HTTP requests.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
```

```
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Generate a pre-signed URL for an object, then download it (GET request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generate the URL.

```
   /* Create a pre-signed URL to download an object in a subsequent GET request.
*/
   public String createPresignedGetUrl(String bucketName, String keyName) {
       try (S3Presigner presigner = S3Presigner.create()) {

           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .build();

           GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL will
expire in 10 minutes.
                   .getObjectRequest(objectRequest)
                   .build();

           PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
           logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

Download the object by using any one of the following three approaches.

Use JDK `HttpURLConnection` (since v1.1) class to do the download.

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
   public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
           connection.setRequestMethod("GET");
           // Download the result of executing the request.
           try (InputStream content = connection.getInputStream()) {
               IoUtils.copy(content, byteArrayOutputStream);
           }
           logger.info("HTTP response code is " + connection.getResponseCode());

       } catch (S3Exception | IOException e) {
           logger.error(e.getMessage(), e);
       }
       return byteArrayOutputStream.toByteArray();
   }
```

Use JDK `HttpClient` (since v11) class to do the download.

```
   /* Use the JDK HttpClient (since v11) class to do the download. */
   public byte[] useHttpClientToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpResponse<InputStream> response = httpClient.send(requestBuilder
                           .uri(presignedUrl.toURI())
                           .GET()
                           .build(),
                   HttpResponse.BodyHandlers.ofInputStream());
```

```
            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use the AWS SDK for Java `SdkHttpClient` class to do the download.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToPut(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));
```

```
                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Generate a pre-signed URL for an upload, then upload a file (PUT request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
```

```
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generate the URL.

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

Upload a file object by using any one of the following three approaches.

Use the JDK `HttpURLConnection` (since v1.1) class to do the upload.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File
 fileToPut, Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use the JDK `HttpClient` (since v11) class to do the upload.

```
   /* Use the JDK HttpClient (since v11) class to do the upload. */
   public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           final HttpResponse<Void> response = httpClient.send(requestBuilder
                           .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                           .build(),
                   HttpResponse.BodyHandlers.discarding());

           logger.info("HTTP response code is " + response.statusCode());

       } catch (URISyntaxException | InterruptedException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

Use the AWS for Java V2 `SdkHttpClient` class to do the upload.

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
           metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
           // Finish building the request.
           SdkHttpRequest request = requestBuilder.build();
```

```
            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a presigned URL to upload an object to a bucket.

```
import https from "https";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";
```

```
const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

function put(url, data) {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          resolve(responseBody);
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
}
```

```
export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.txt";

  // There are two ways to generate a presigned URL.
  // 1. Use createPresignedUrl without the S3 client.
  // 2. Use getSignedUrl in conjunction with the S3 client and GetObjectCommand.
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (err) {
    console.error(err);
  }
};
```

Create a presigned URL to download an object from a bucket.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
```

```
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.jpg";

  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
```

```
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (err) {
    console.error(err);
  }
};
```

- For more information, see AWS SDK for JavaScript Developer Guide.

Kotlin

**SDK for Kotlin**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a `GetObject` presigned request and use the URL to download an object.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()
```

```
    return objectContents
}
```

Create a `GetObject` presigned request with advanced options.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Create a `PutObject` presigned request and use it to upload an object.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
```

```
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- For more information, see AWS SDK for Kotlin developer guide.

PHP

**SDK for PHP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
```

```
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
```

```
$runner->run();
```

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Generate a presigned URL that can perform an S3 action for a limited time. Use the Requests package to make a request with the URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
```

```
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "r") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
```

```
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

Generate a presigned POST request to upload a file.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
```

```
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
```

```
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request("PUT", presigned_url.request_uri, object_content,
 "content_type" => "")
  end

  case response
  when Net::HTTPSuccess
    puts "Content uploaded!"
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create presigning requests to GET and PUT S3 objects.

```
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);

    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## A web page that lists Amazon S3 objects using an AWS SDK

The following code example shows how to list Amazon S3 objects in a web page.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The following code is the relevant React component that makes calls to the AWS SDK. A runnable version of the application containing this component can be found at the preceding GitHub link.

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
```

```
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- For API details, see ListObjects in *AWS SDK for JavaScript API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Delete incomplete multipart uploads to Amazon S3 using an AWS SDK

The following code example shows how to how to delete or stop incomplete Amazon S3 multipart uploads.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

To stop multipart uploads that are in-progress or incomplete for any reason, you can get a list uploads and then delete them as shown in the following example.

```
   public static void abortIncompleteMultipartUploadsFromList() {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(upload.key())
                   .expectedBucketOwner(accountId)
                   .uploadId(upload.uploadId())
                   .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
```

```
            }
        }
    }
```

To delete incomplete multipart uploads that were initiated before or after a date, you can selectively delete multipart uploads based on a point in time as shown in the following example.

```
    static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .expectedBucketOwner(accountId)
                        .uploadId(upload.uploadId())
                        .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

If you have access to the upload ID after you begin a multipart upload, you can delete the in-progress upload by using the ID.

```
    static void abortMultipartUploadUsingUploadId() {
        String uploadId = startUploadReturningUploadId();
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
                .uploadId(uploadId)
                .bucket(bucketName)
                .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

To consistently delete incomplete multipart uploads older that a certain number of days, set up a bucket lifecycle configuration for the bucket. The following example shows how to create a rule to delete incomplete uploads older than 7 days.

```
    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules =
 List.of(LifecycleRule.builder()
                .abortIncompleteMultipartUpload(b -> b.
                        daysAfterInitiation(7))
                .status("Enabled")
                .filter(SdkBuilder::build) // Filter element is required.
                .build());

        // If the action is successful, the service sends back an HTTP 200
 response with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
 s3Client.putBucketLifecycleConfiguration(b -> b
                .bucket(bucketName)
                .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
```

```
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory

The following code example shows how to download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to download all S3 objects in the same S3 bucket. View the complete file and test.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- For API details, see DownloadDirectory in *AWS SDK for Java 2.x API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Get an Amazon S3 object from a Multi-Region Access Point by using an AWS SDK

The following code example shows how to get an object from a Multi-Region Access Point.

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Configure the S3 client to use the Asymmetric Sigv4 (Sigv4a) signing algorithm.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a)
signing algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

Use the Multi-Region Access Point ARN instead of a bucket name to retrieve the object.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }
```

```
        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- For more information, see AWS SDK for Kotlin developer guide.
- For API details, see GetObject in *AWS SDK for Kotlin API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Get an object from an Amazon S3 bucket using an AWS SDK, specifying an If-Modified-Since header

The following code example shows how to read data from an object in an S3 bucket, but only if that bucket has not been modified since the last retrieval time.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
use aws_sdk_s3::{
    error::SdkError,
    operation::head_object::HeadObjectError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client, Error,
};
```

```
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
```

```
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.
```

```
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2): (Result<DateTime, SdkError<HeadObjectError>>,
String) =
        match head_object_output {
            Ok(head_object) => (
                Ok(head_object.last_modified().cloned().unwrap()),
                head_object.e_tag.unwrap(),
            ),
            Err(err) => match err {
                SdkError::ServiceError(err) => {
                    // Get the raw HTTP response. If its status is 304, the
                    // object has not changed. This is the expected code path.
                    let http = err.raw();
                    match http.status().as_u16() {
                        // If the HTTP status is 304: Not Modified, return a
                        // tuple containing the values of the HTTP
                        // `last-modified` and `etag` headers.
                        304 => (
                            Ok(DateTime::from_str(
                                http.headers().get("last-modified").unwrap(),
                                DateTimeFormat::HttpDate,
                            )
                            .unwrap()),
```

```
                            http.headers().get("etag").map(|t|
 t.into()).unwrap(),
                        ),
                        // Any other HTTP status code is returned as an
                        // `SdkError::ServiceError`.
                        _ => (Err(SdkError::ServiceError(err)), String::new()),
                    }
                }
                // Any other kind of error is returned in a tuple containing the
                // error and an empty string.
                _ => (Err(err), String::new()),
            },
        };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- For API details, see GetObject in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Get started with Amazon S3 buckets and objects using an AWS SDK

The following code examples show how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class S3_Basics
{
    public static async Task Main()
    {
        // Create an Amazon S3 client object. The constructor uses the
        // default user installed on the system. To work with Amazon S3
        // features in a different AWS Region, pass the AWS Region as a
        // parameter to the client constructor.
        IAmazonS3 client = new AmazonS3Client();
        string bucketName = string.Empty;
        string filePath = string.Empty;
        string keyName = string.Empty;

        var sepBar = new string('-', Console.WindowWidth);
```

```
            Console.WriteLine(sepBar);
            Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
            Console.WriteLine("procedures. This application will:");
            Console.WriteLine("\n\t1. Create a bucket");
            Console.WriteLine("\n\t2. Upload an object to the new bucket");
            Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
 bucket");
            Console.WriteLine("\n\t4. List the items in the new bucket");
            Console.WriteLine("\n\t5. Delete all the items in the bucket");
            Console.WriteLine("\n\t6. Delete the bucket");
            Console.WriteLine(sepBar);

            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();

            var success = await S3Bucket.CreateBucketAsync(client, bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
                Console.Write("Please enter the path and filename of the file to
 upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
```

```
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
                    filePath = string.Empty;
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set the file path to an empty string to avoid overwriting the
            // file we just uploaded to the bucket.
            filePath = string.Empty;

            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(filePath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (File.Exists($"{filePath}\\{keyName}"))
                {
                    Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                    filePath = string.Empty;
                }
            }

            // Download an object from a bucket.
```

```
            success = await S3Bucket.DownloadObjectFromBucketAsync(client,
bucketName, keyName, filePath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            while (string.IsNullOrEmpty(folderName))
            {
                Console.Write("Please enter the name of the folder to copy your
object to: ");
                folderName = Console.ReadLine();
            }

            while (string.IsNullOrEmpty(keyName))
            {
                // Get the name to give to the object once uploaded.
                Console.Write("Enter the name of the object to copy: ");
                keyName = Console.ReadLine();
            }

            await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await S3Bucket.ListBucketContentsAsync(client, bucketName);

            // Delete the contents of the bucket.
            await S3Bucket.DeleteBucketContentsAsync(client, bucketName);

            // Deleting the bucket too quickly after deleting its contents will
            // cause an error that the bucket isn't empty. So...
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();
```

```
            // Delete the bucket.
            await S3Bucket.DeleteBucketAsync(client, bucketName);
        }
    }
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Bash

**AWS CLI with Bash script**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
```

```
    cd bucket-lifecycle-operations || exit

    source ./bucket_operations.sh
    cd ..
  fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88

local bucket_name
bucket_name=$(generate_random_name "doc-example-bucket")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
  echo "Created demo bucket named $bucket_name"
else
  errecho "The bucket failed to create. This demo will exit."
  return 1
fi

local file_name
while [ -z "$file_name" ]; do
  echo -n "Enter a file you want to upload to your bucket: "
  get_input
  file_name=$get_input_result

  if [ ! -f "$file_name" ]; then
    echo "Could not find file $file_name. Are you sure it exists?"
    file_name=""
  fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
  echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
  result=1
```

```
 fi

 local destination_file
 destination_file="$file_name.download"
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi

 if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
   local to_key
   to_key="demo/$key"
   if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
     echo "Copied $key in the bucket $bucket_name to the  $to_key."
   else
     result=1
   fi
 fi

 local bucket_items
 bucket_items=$(list_items_in_bucket "$bucket_name")

 # shellcheck disable=SC2181
 if [[ $? -ne 0 ]]; then
   result=1
 fi

 echo "Your bucket contains the following items."
 echo -e "Name\t\tSize"
 echo "$bucket_items"

 if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
   bucket_items=$(echo "$bucket_items" | cut -f 1)

   if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
     echo "The following items were deleted from the bucket $bucket_name"
```

```
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

The Amazon S3 functions used in this scenario.

```
###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
```

```
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
 }

 # Retrieve the calling parameters.
 while getopts "b:r:h" option; do
   case "${option}" in
     b) bucket_name="${OPTARG}" ;;
     r) region_code="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done

 if [[ -z "$bucket_name" ]]; then
   errecho "ERROR: You must provide a bucket name with the -b parameter."
   usage
   return 1
 fi

 local bucket_config_arg
 # A location constraint for "us-east-1" returns an error.
 if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
   bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
 fi

 iecho "Parameters:\n"
 iecho "    Bucket name:   $bucket_name"
 iecho "    Region code:   $region_code"
 iecho ""

 # If the bucket already exists, we don't want to try to create it.
 if (bucket_exists "$bucket_name"); then
```

```
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
```

```
    return 1
  fi
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
```

```
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response
```

```
  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")
```

```
  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

################################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- For API details, see the following topics in *AWS CLI Command Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
```

```
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = "doc-example-bucket-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
```

```
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                            clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }
```

```
        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
```

```
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
```

```
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
 objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
 bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Define a struct that wraps bucket and object actions used by the scenario.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
```

```
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}



// UploadFile reads from a file and puts the data into an object in a bucket.
```

```
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
```

```
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
```

```
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(bucketName string, objectKey string,
 folderName string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(fmt.Sprintf("%v/%v", folderName, objectKey)),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v/%v. Here's why: %v\n",
   bucketName, objectKey, bucketName, folderName, objectKey, err)
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
```

```
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(bucketName string, objectKeys []string)
 error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(context.TODO(),
 &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds},
 })
 if err != nil {
  log.Printf("Couldn't delete objects from bucket %v. Here's why: %v\n",
 bucketName, err)
 } else {
  log.Printf("Deleted %v objects.\n", len(output.Deleted))
 }
 return err
```

```
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

Run an interactive scenario that shows you how work with S3 buckets and objects.

```
// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 4. Download an object to a local file.
// 5. Download a large object by using a download manager.
// 6. Copy an object to a different folder in the bucket.
// 7. List objects in the bucket.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
```

```
func RunGetStartedScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong with the demo.\n", r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
 log.Printf("Let's list up to %v buckets for your account:", count)
 buckets, err := bucketBasics.ListBuckets()
 if err != nil {
  panic(err)
 }
 if len(buckets) == 0 {
  log.Println("You don't have any buckets!")
 } else {
  if count > len(buckets) {
   count = len(buckets)
  }
  for _, bucket := range buckets[:count] {
   log.Printf("\t%v\n", *bucket.Name)
  }
 }

 bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
 bucket:",
  demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
  if err != nil {
   panic(err)
  } else {
```

```
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
rand.Seed(time.Now().Unix())
rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
```

```
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting objects.")
 err = bucketBasics.DeleteObjects(bucketName, objKeys)
 if err != nil {
  panic(err)
 }
 log.Println("Deleting bucket.")
 err = bucketBasics.DeleteBucket(bucketName)
 if err != nil {
  panic(err)
 }
 log.Printf("Deleting downloaded file %v.\n", downloadFileName)
```

```
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <bucketName> <key> <objectPath> <savePath> <toBucket>

                Where:
                    bucketName - The Amazon S3 bucket to create.
                    key - The key to use.
                    objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).
                    savePath - The path where the file is saved after it's
 downloaded (for example, C:/AWS/book2.pdf).
                    toBucket - An Amazon S3 bucket to where an object is copied
 to (for example, C:/AWS/book2.pdf).\s
                    """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes(s3, bucketName, key, savePath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3
bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
```

```
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully
performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteBucket(S3Client client, String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
               .bucket(bucket)
               .build();

       client.deleteBucket(deleteBucketRequest);
       System.out.println(bucket + " was deleted.");
   }

   /**
    * Upload an object in parts.
    */
   public static void multipartUpload(S3Client s3, String bucketName, String
key) {
       int mB = 1024 * 1024;
       // First create a multipart upload and get the upload id.
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
       String uploadId = response.uploadId();
       System.out.println(uploadId);

       // Upload all the different parts of the object.
       UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
               .bucket(bucketName)
               .key(key)
               .uploadId(uploadId)
               .partNumber(1).build();

       String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
               .eTag();
       CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

       UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
               .uploadId(uploadId)
```

```
                .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all
uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();
```

```
            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String
key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }
```

```
            listObjectsReqManual = listObjectsReqManual.toBuilder()
                    .continuationToken(listObjResponse.nextContinuationToken())
                    .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

        // Process response pages.
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }
```

```
    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
 StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .copySource(encodedUrl)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " +
 toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

First, import all the necessary modules.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "url";
import { readdirSync, readFileSync, writeFileSync } from "fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

The preceding imports reference some helper utilities. These utilities are local to the GitHub repository linked at the start of this section. For your reference, see the following implementations of those utilities.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox } from "@inquirer/prompts";
```

```
export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && prompt + " ";
    let ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };

  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
```

```
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

Objects in S3 are stored in 'buckets'. Let's define a function for creating a new bucket.

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

Buckets contain 'objects'. This function uploads the contents of a directory to your bucket as objects.

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (let file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
```

```
};
```

After uploading objects, check to confirm that they were uploaded correctly. You can use ListObjects for that. You'll be using the 'Key' property, but there are other useful properties in the response also.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(contentsList + "\n");
};
```

Sometimes you might want to copy an object from one bucket to another. Use the CopyObject command for that.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  } else {
    const copy = async () => {
      try {
        const sourceBucket = await prompter.input({
          message: "Enter source bucket name:",
        });
        const sourceKey = await prompter.input({
          message: "Enter source key:",
        });
        const destinationKey = await prompter.input({
          message: "Enter destination key:",
        });

        const command = new CopyObjectCommand({
          Bucket: destinationBucket,
          CopySource: `${sourceBucket}/${sourceKey}`,
```

```
          Key: destinationKey,
        });
        await s3Client.send(command);
        await copyFileFromBucket({ destinationBucket });
      } catch (err) {
        console.error(`Copy error.`);
        console.error(err);
        const retryAnswer = await prompter.confirm({ message: "Try again?" });
        if (retryAnswer) {
          await copy();
        }
      }
    };
    await copy();
  }
};
```

There's no SDK method for getting multiple objects from a bucket. Instead, you'll create a list of objects to download and iterate over them.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

  for (let content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

It's time to clean up your resources. A bucket must be empty before it can be deleted. These two functions empty and delete the bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

The 'main' function pulls everything together. If you run this file directly the main function will be called.

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
```

```
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Kotlin

### SDK for Kotlin

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)
```

```
    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
```

```
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}
```

```
suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }
```

```
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

PHP

**SDK for PHP**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        echo("\n");
```

```
        echo("-------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("-------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "doc-example-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
```

```
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
```

```
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)

    bucket_name = f"doc-example-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
```

```
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
```

```
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
```

```
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
  def upload_file(bucket)
    File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
    s3_object = bucket.object(File.basename("demo.txt"))
    s3_object.upload_file("demo.txt")
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      puts("Enter a name for the downloaded file: ")
      file_name = gets.chomp
```

```
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
 bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
```

```
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the Amazon S3 getting started demo!")
  puts("-" * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Code for the binary crate which runs the scenario.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_service::error::Error;
use uuid::Uuid;


#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
 initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        println!("{:?}", e);
    };
```

```
    Ok(())
}

async fn initialize_variables() -> (Region, Client, String, String, String,
 String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
    s3_service::list_objects(&client, &bucket_name).await?;
    s3_service::delete_objects(&client, &bucket_name).await?;
    s3_service::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

A library crate with common actions called by the binary.

```
use aws_sdk_s3::operation::{
    copy_object::{CopyObjectError, CopyObjectOutput},
    create_bucket::{CreateBucketError, CreateBucketOutput},
    get_object::{GetObjectError, GetObjectOutput},
    list_objects_v2::ListObjectsV2Output,
    put_object::{PutObjectError, PutObjectOutput},
};
use aws_sdk_s3::types::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete,
 ObjectIdentifier,
};
use aws_sdk_s3::{error::SdkError, primitives::ByteStream, Client};
use error::Error;
use std::path::Path;
use std::str;

pub mod error;

pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}

pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
```

```
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
```

```
        }
    }

    Ok(())
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}

pub async fn download_object(
    client: &Client,
    bucket_name: &str,
    key: &str,
) -> Result<GetObjectOutput, SdkError<GetObjectError>> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
```

```
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}

pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- For API details, see the following topics in *AWS SDK for Rust API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

SAP ABAP

### SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
  CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
  CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.


"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content
    ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                    iv_bucket = iv_bucket_name
                    iv_key = iv_key
                ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }|
        ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name
         ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
```

```
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
        ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Swift

### SDK for Swift

> **Note**
>
> This is prerelease documentation for an SDK in preview release. It is subject to change.

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

A Swift class that handles calls to the SDK for Swift.

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime
import Smithy

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let client: S3Client

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async {
        do {
            client = try S3Client(region: "us-east-2")
        } catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            exit(1)
        }
    }
```

```
    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

```
    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

```
    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }

    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

```
    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        } catch {
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
```

```
    }
}
```

A Swift command-line program to manage the SDK calls.

```
import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
```

```
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.
```

```
        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Get started with encryption for Amazon S3 objects using an AWS SDK

The following code example shows how to get started with encryption for Amazon S3 objects.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
```

```
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
        /// object will be uploaded.</param>
```

```
        /// <param name="keyName">The name of the object to upload to the Amazon
 S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
 DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
 client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
```

```
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }
```

```
        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
```

```
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- CopyObject
- GetObject
- GetObjectMetadata

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Get started with tags for Amazon S3 objects using an AWS SDK

The following code example shows how to get started with tags for Amazon S3 objects.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";
```

```
            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
```

```
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
```

```
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- For API details, see GetObjectTagging in *AWS SDK for .NET API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Get the legal hold configuration of an Amazon S3 object using an AWS SDK

The following code examples show how to get the legal hold configuration of an S3 bucket.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
```

```
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- For API details, see GetObjectLegalHold in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
```

```
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }
```

- For API details, see GetObjectLegalHold in *AWS SDK for Java 2.x API Reference*.

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import { fileURLToPath } from "url";
import { GetObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
```

```
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "DOC-EXAMPLE-BUCKET", "OBJECT_KEY");
}
```

- For API details, see GetObjectLegalHold in *AWS SDK for JavaScript API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Work with Amazon S3 object lock features using an AWS SDK

The following code examples show how to work with S3 object lock features.

.NET

### AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
 lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
 or overwrite objects.
        5. Clean up objects and buckets.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
```

```
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Workflow is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
```

```
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this workflow, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
 true);
        await
 _s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
 false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking
 with a default retention period.");
        await
 _s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
 true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
 buckets.");
```

```
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
```

```
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                    }
                }
            }
        }
        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    // <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
```

```
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
```

```
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
```

```
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
```

```
                var hasRetentionPeriod = retention?.Mode ==
 ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
 DateTime.UtcNow.Date;
                await
 _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
 hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
```

```
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

A wrapper class for S3 functions.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
```

```
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
```

```
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
```

```
   public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
   {
       var enabledString = enableObjectLock ? "Enabled" : "Disabled";
       var timeDifference = retainUntilDate.Subtract(DateTime.Now);
       try
       {
           // First, enable Versioning on the bucket.
           await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
           {
               BucketName = bucketName,
               VersioningConfig = new S3BucketVersioningConfig()
               {
                   EnableMfaDelete = false,
                   Status = VersionStatus.Enabled
               }
           });

           var request = new PutObjectLockConfigurationRequest()
           {
               BucketName = bucketName,
               ObjectLockConfiguration = new ObjectLockConfiguration()
               {
                   ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                   Rule = new ObjectLockRule()
                   {
                       DefaultRetention = new DefaultRetention()
                       {
                           Mode = retention,
                           Days = timeDifference.Days // Can be specified in
days or years but not both.
                       }
                   }
               }
           };

           var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
           return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
       }
       catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
```

```
            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
```

```
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
{bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
```

```
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
 {bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration

- PutObjectRetention

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}
```

```
var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}
```

```
// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
 bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
 context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
```

```
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")
```

```
buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
for _, bucket := range buckets {
 for index, objKey := range bucket.objectKeys {
  switch index {
  case 0:
   if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't set legal hold on %s.\n", objKey)
     default:
      panic(err)
     }
    } else {
     log.Printf("Legal hold set on %s.\n", objKey)
    }
   }
  case 1:
   q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
    "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
    "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
   if scenario.questioner.AskBool(q, "y") {
    err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
     default:
      panic(err)
     }
    } else {
     log.Printf("Retention period set to 1 for %s.", objKey)
     bucket.retentionEnabled = true
    }
   }
```

```
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
 for choice != Finish {
  objList := scenario.GetAllObjects(ctx)
  objChoices := scenario.makeObjectChoiceList(objList)
  choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
 interactiveChoices)
  switch choice {
  case ListAll:
   log.Println("The current objects in the example buckets are:")
   for _, objChoice := range objChoices {
    log.Println("\t", objChoice)
   }
  case DeleteObject, DeleteRetentionObject:
```

```
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
  obj := objList[objChoice]
  deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Println("Nothing to delete.")
   default:
    panic(err)
   }
  } else if deleted {
   log.Printf("Object %s deleted.\n", obj.key)
  }
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Println("Couldn't upload to nonexistent bucket.")
   default:
    panic(err)
   }
  } else {
   log.Printf("Uploaded new content to object %s.\n", obj.key)
  }
 case ViewRetention:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Can't get retention configuration for %s.\n", obj.key)
   default:
    panic(err)
   }
```

```
   } else if retention != nil {
    log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
 retention.Mode, retention.RetainUntilDate)
   } else {
    log.Printf("Object %s does not have object retention configured.\n", obj.key)
   }
  case ViewLegalHold:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 view:\n", objChoices)
   obj := objList[objChoice]
   legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
 obj.key, obj.versionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if legalHold != nil {
    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
   } else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
   }
  case Finish:
   log.Println("Let's clean up.")
  }
  log.Println(strings.Repeat("-", 88))
 }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
```

```
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock workflow scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
```

```
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Workflow Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Define a struct that wraps S3 actions used in this example.

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
```

```
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
```

```
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
```

```
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
```

```
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
```

```
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
```

```
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
```

```
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
```

```
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 }
```

```
 return err
}
```

Clean up resources.

```
// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 legalHold        bool
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
```

```
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
```

```
versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
if err != nil {
 switch err.(type) {
 case *types.NoSuchBucket:
  log.Printf("No objects to get from %s.\n", bucket.name)
 default:
  panic(err)
 }
}
delObjects := make([]types.ObjectIdentifier, len(versions))
for i, version := range versions {
 if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
  status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
   default:
    panic(err)
   }
  } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
   err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
    default:
     panic(err)
    }
   }
  }
 }
 delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
bucket.retentionEnabled)
if err != nil {
 switch err.(type) {
```

```
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter =
 DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
 Object Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
```

```
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
```

```
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
```

```
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                    }
                }
                // Check for a retention period.
```

```
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal
lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);

        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();
```

```
        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
```

```
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
```

```
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

A wrapper class for S3 functions.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
 zone.
        ZonedDateTime zonedDateTime =
 futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();
```

```
        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
```

```
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
```

```
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
```

```
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
```

```
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
```

```
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

```
    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

index.js - Entrypoint for the workflow. This orchestrates all of the steps. Visit GitHub to see the implementation details for Scenario, ScenarioInput, ScenarioOutput, and ScenarioAction.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
```

```
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
```

```
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios);
}
```

welcome.steps.js - Output welcome messages to the console.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
```

```
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    `Welcome to the Amazon Simple Storage Service (S3) Object Locking Workflow
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.`,
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

setup.steps.js - Deploy buckets, objects, and file settings.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */
```

```
/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const bucketPrefix = "js-object-locking";

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    `The following buckets will be created:
         ${bucketPrefix}-no-lock with object lock False.
         ${bucketPrefix}-lock-enabled with object lock True.
         ${bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${bucketPrefix}-retention-after-creation`;

    await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
    await client.send(
      new CreateBucketCommand({
        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
```

```
      }),
    );
    await client.send(new CreateBucketCommand({ Bucket: retentionBucketName }));

    state.noLockBucketName = noLockBucketName;
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    `The following test files will be created:
         file0.txt in ${bucketPrefix}-no-lock.
         file1.txt in ${bucketPrefix}-no-lock.
         file0.txt in ${bucketPrefix}-lock-enabled.
         file1.txt in ${bucketPrefix}-lock-enabled.
         file0.txt in ${bucketPrefix}-retention-after-creation.
         file1.txt in ${bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
```

```
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
```

```
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    `A bucket can be configured to use object locking with a default retention
 period.
   A default retention period will be configured for ${bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    await client.send(
      new PutObjectLockConfigurationCommand({
```

```
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    `Object lock policies can also be added to existing buckets.
   An object lock policy will be added to ${bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
```

```
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
```

```
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
```

```
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
```

```
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
```

```
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

repl.steps.js - View and delete files in the buckets.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
```

```
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    `Explore the S3 locking features by selecting one of the following choices`,
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
```

```
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
```

```
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
```

```
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
```

```
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
```

```
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

clean.steps.js - Destroy all created resources.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
```

```
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        } else {
          throw e;
        }
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
```

```
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );

        if (legalHold.LegalHold?.Status === "ON") {
          await client.send(
            new PutObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
              LegalHold: {
                Status: "OFF",
              },
            }),
          );
        }
      } catch (err) {
        console.log(
          `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
        );
      }

      try {
        const retention = await client.send(
          new GetObjectRetentionCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );

        if (retention.Retention?.Mode === "GOVERNANCE") {
          await client.send(
            new DeleteObjectCommand({
              Bucket: bucket,
              Key,
              VersionId,
              BypassGovernanceRetention: true,
            }),
          );
        }
```

```
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Manage access control lists (ACLs) for Amazon S3 buckets using an AWS SDK

The following code example shows how to manage access control lists (ACLs) for Amazon S3 buckets.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket1";
        string newBucketName = "doc-example-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
```

```
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
```

```
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
 get the
        /// ACL list.</param>
```

```
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;
```

```
            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK

The following code example shows how to manage versioned S3 objects in batches with a Lambda function.

Python

**SDK for Python (Boto3)**

Shows how to manipulate Amazon Simple Storage Service (Amazon S3) versioned objects in batches by creating jobs that call AWS Lambda functions to perform processing. This example creates a version-enabled bucket, uploads the stanzas from the poem *You Are Old, Father William* by Lewis Carroll, and uses Amazon S3 batch jobs to twist the poem in various ways.

**Learn how to:**

- Create Lambda functions that operate on versioned objects.
- Create a manifest of objects to update.
- Create batch jobs that invoke Lambda functions to update objects.
- Delete Lambda functions.
- Empty and delete a versioned bucket.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon S3

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Parse Amazon S3 URIs using an AWS SDK

The following code example shows how to parse Amazon S3 URIs to extract important components like the bucket name and object key.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Parse an Amazon S3 URI by using the S3Uri class.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
```

```
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                     capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
 method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
 returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
```

```
        // Console output: 'rawQueryParameters: {versionId=[abc123],
 partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
 s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
```

```
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Perform a multipart copy of an Amazon S3 object using an AWS SDK

The following code example shows how to perform a multipart copy of an Amazon S3 object.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "doc-example-bucket1";
    private const string TargetBucket = "doc-example-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";
```

```
        /// <summary>
        /// This method starts the multi-part upload.
        /// </summary>
        public static async Task Main()
        {
            var s3Client = new AmazonS3Client();
            Console.WriteLine("Copying object...");
            await MPUCopyObjectAsync(s3Client);
        }

        /// <summary>
        /// This method uses the passed client object to perform a multipart
        /// copy operation.
        /// </summary>
        /// <param name="client">An Amazon S3 client object that will be used
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };
```

```
                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Perform a multipart upload of an Amazon S3 object using an AWS SDK

The following code example shows how to perform a multipart upload to an Amazon S3 object.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The code examples use the following imports.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Use the S3 Transfer Manager on top of the AWS CRT-based S3 client to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

```
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
```

```
        transferManager.close();
    }
```

Use the S3Client API to perform a multipart upload.

```
    public void multipartUploadWithS3Client(String filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
```

```
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Use the S3AsyncClient API with multipart support enabled to perform a multipart upload.

```
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                        .bucket(bucketName)
                        .key(key),
                Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- CompleteMultipartUpload
- CreateMultipartUpload
- UploadPart

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Receive and process Amazon S3 event notifications by using an AWS SDK.

The following code example shows how to work with S3 event notifications in an object-oriented way.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

This example show how to process S3 notification event by using Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
```

```
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String
queueArn) {
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the
queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)
```

```
.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
```

```
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Send S3 event notifications to Amazon EventBridge using an AWS SDK

The following code example shows how to enable a bucket to send S3 event notifications to EventBridge and route notifications to an Amazon SNS topic and Amazon SQS queue.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
```

```
                                }
                              }
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Track an Amazon S3 object upload or download using an AWS SDK

The following code example shows how to track an Amazon S3 object upload or download.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Track the progress of a file upload.

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                               String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
```

```
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

Track the progress of a file download.

```
    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                              String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
```

```
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- GetObject
- PutObject

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Example approaches for unit and integration testing with an AWS SDK

The following code example shows how to examples for best-practice techniques when writing unit and integration tests using an AWS SDK.

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Cargo.toml for testing examples.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

# snippet-start:[testing.rust.Cargo.toml]
[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
```

```
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "~4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
# snippet-end:[testing.rust.Cargo.toml]

[[bin]]
name = "main"
path = "src/main.rs"
```

Unit testing example using automock and a service wrapper.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// snippet-start:[testing.rust.wrapper]
// snippet-start:[testing.rust.wrapper-uses]
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};
// snippet-end:[testing.rust.wrapper-uses]

// snippet-start:[testing.rust.wrapper-which-impl]
#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;
// snippet-end:[testing.rust.wrapper-which-impl]

// snippet-start:[testing.rust.wrapper-impl]
#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}
```

```
#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}
// snippet-end:[testing.rust.wrapper-impl]

// snippet-start:[testing.rust.wrapper-func]
#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
```

```
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}
// snippet-end:[testing.rust.wrapper-func]
// snippet-end:[testing.rust.wrapper]

// snippet-start:[testing.rust.wrapper-test-mod]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.wrapper-tests]
    use super::*;
    use mockall::predicate::eq;

    // snippet-start:[testing.rust.wrapper-test-single]
    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
```

```
}
// snippet-end:[testing.rust.wrapper-test-single]

// snippet-start:[testing.rust.wrapper-test-multiple]
#[tokio::test]
async fn test_multiple_pages() {
    // Create the Mock instance with two pages of objects now
    let mut mock = MockS3Impl::default();
    mock.expect_list_objects()
        .with(eq("test-bucket"), eq("test-prefix"), eq(None))
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(5).build(),
                    s3::types::Object::builder().size(2).build(),
                ]))
                .set_next_continuation_token(Some("next".to_string()))
                .build())
        });
    mock.expect_list_objects()
        .with(
            eq("test-bucket"),
            eq("test-prefix"),
            eq(Some("next".to_string())),
        )
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(3).build(),
                    s3::types::Object::builder().size(9).build(),
                ]))
                .build())
        });

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
        .await
        .unwrap();

    assert_eq!(19, size);
}
// snippet-end:[testing.rust.wrapper-test-multiple]
```

```
    // snippet-end:[testing.rust.wrapper-tests]
}
// snippet-end:[testing.rust.wrapper-test-mod]
```

Integration testing example using StaticReplayClient.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// snippet-start:[testing.rust.replay-uses]
use aws_sdk_s3 as s3;
// snippet-end:[testing.rust.replay-uses]

#[allow(dead_code)]
// snippet-start:[testing.rust.replay]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
```

```
        }
    }
    Ok(total_size_bytes)
}
// snippet-end:[testing.rust.replay]

#[allow(dead_code)]
// snippet-start:[testing.rust.replay-tests]
// snippet-start:[testing.rust.replay-make-credentials]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}
// snippet-end:[testing.rust.replay-make-credentials]

// snippet-start:[testing.rust.replay-test-module]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.replay-test-single]
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
```

```
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }
    // snippet-end:[testing.rust.replay-test-single]

    // snippet-start:[testing.rust.replay-test-multiple]
    #[tokio::test]
    async fn test_multiple_pages() {
        // snippet-start:[testing.rust.replay-create-replay]
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
```

```
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        // snippet-end:[testing.rust.replay-create-replay]
        // snippet-start:[testing.rust.replay-create-client]
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );
        // snippet-end:[testing.rust.replay-create-client]

        // Run the code we want to test with it
        // snippet-start:[testing.rust.replay-test-and-verify]
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
        // snippet-end:[testing.rust.replay-test-and-verify]
    }
    // snippet-end:[testing.rust.replay-test-multiple]
}
// snippet-end:[testing.rust.replay-tests]
// snippet-end:[testing.rust.replay-test-module]
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Recursively upload a local directory to an Amazon Simple Storage Service (Amazon S3) bucket

The following code example shows how to upload a local directory recursively to an Amazon Simple Storage Service (Amazon S3) bucket.

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to upload a local directory. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
```

```
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- For API details, see UploadDirectory in *AWS SDK for Java 2.x API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Upload or download large files to and from Amazon S3 using an AWS SDK

The following code examples show how to upload or download large files to and from Amazon S3.

For more information, see Uploading an object using multipart upload.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Call functions that transfer files to and from an S3 bucket using the Amazon S3 TransferUtility.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
```

```
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
{
```

```
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
```

```
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);
```

```
    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
```

```
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

Upload a single file.

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</
param>
```

```
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

Upload an entire local directory.

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
```

```
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
```

```
            }
        }
```

Download a single file.

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

Download contents of an S3 bucket.

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
```

```
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
```

```
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

Track the progress of an upload using the TransferUtility.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;

    /// <summary>
    /// This example shows how to track the progress of a multipart upload
    /// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
    /// upload to an Amazon S3 bucket.
    /// </summary>
    public class TrackMPUUsingHighLevelAPI
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "sample_pic.png";
            string path = "filepath/directory/";
            string filePath = $"{path}{keyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            await TrackMPUAsync(client, bucketName, filePath, keyName);
        }

        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
```

```
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
```

```
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

Upload an object with encryption.

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
    {
        public static async Task Main()
        {
            string existingBucketName = "doc-example-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();
```

```
            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
```

```
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }
```

```
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");

                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a large object by using an upload manager to break the data into parts and upload them concurrently.

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

Download a large object by using a download manager to get the data in parts and download them concurrently.

```
// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Call functions that transfer files to and from an S3 bucket using the S3TransferManager.

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

Upload an entire local directory.

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
       completedDirectoryUpload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryUpload.failedTransfers().size();
   }
```

Upload a single file.

```
   public String uploadFile(S3TransferManager transferManager, String
bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
```

```
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

JavaScript

**SDK for JavaScript (v3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a large file.

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
```

```
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );

    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
              PartNumber: i + 1,
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
```

```
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open
 the file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

Download a large file.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
```

```
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
```

```
    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that transfer files using several of the available transfer manager settings. Use a callback class to write callback progress during file transfer.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.
```

```
    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
```

```
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
```

```
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
```

```
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstrate the transfer manager functions and report results.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")
```

```
class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
```

```
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
```

```
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
```

```
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.

        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)
```

```
        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
```

```
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )
```

```
    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )

    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
```

```
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;
```

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_service::error::Error;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
```

```
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        panic!("Bad file size.");
    }
    if chunk_count > MAX_CHUNKS {
        panic!("Too many chunks! Try increasing your chunk size.")
    }

    let mut upload_parts: Vec<CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
```

```
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();
        //Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();
```

```
    let data: GetObjectOutput = s3_service::download_object(&client,
 &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_service::delete_objects(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_service::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Upload a stream of unknown size to an Amazon S3 object using an AWS SDK

The following code example shows how to upload a stream of unknown size to an Amazon S3 object.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use the AWS CRT-based S3 Client.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown
 size, use the AWS CRT-based S3 client. For more information, see
     *                          https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/crt-based-s3-client.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
 Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
```

```
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

Use the Amazon S3 Transfer Manager.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

    /**
     * @param transferManager - To upload content from a stream of unknown size,
 use the S3TransferManager based on the AWS CRT-based S3 client.
     *                        For more information, see https://
docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
```

```
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        return upload.completionFuture().join();
    }
}
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Use checksums to work with an Amazon S3 object using an AWS SDK

The following code example shows how to use checksums to work with an Amazon S3 object.

Java

### SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The code examples use a subset of the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
```

```
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Specify a checksum algorithm for the `putObject` method when you build the `PutObjectRequest`.

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

Verify the checksum for the `getObject` method when you build the GetObjectRequest.

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

Pre-calculate a checksum for the `putObject` method when you build the `PutObjectRequest`.

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

Use the S3 Transfer Manager on top of the AWS CRT-based S3 client to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

You can specify a checksum algorithm for the SDK to use. By default, the SDK uses the CRC32 algorithm.

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Use the S3Client API or (S3AsyncClient API) to perform a multipart upload. If you specify an additional checksum, you must specify the algorithm to use on the initiation of the upload. You must also specify the algorithm for each part request and provide the checksum calculated for each part after it is uploaded.

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(algorithm)); // Checksum specified on
initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
```

```
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on
each part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide
the calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
```

```
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
   }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Work with Amazon S3 object integrity features using an AWS SDK

The following code example shows how to work with S3 object integrity features.

C++

**SDK for C++**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario demonstrating Amazon S3 object integrity features.

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
```

```
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
              << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
                  << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                          SHA256);


    gUseCalculatedChecksum = !askYesNoQuestion(
            "Let the SDK calculate the checksum for you? (y/n) ");
```

```
    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
              << std::endl;
    std::cout << "Object integrity will be verified using the "
              << stringForHashMethod(chosenHashMethod) << " algorithm."
              << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
                << "A checksum computed by this workflow will be used for object
integrity verification,"
                << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
                << "A checksum computed by the SDK will be used for object
integrity verification."
                << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
```

```
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;
```

```
   introductoryTransferManagerUploadExplanations(key);

   HASH_METHOD transferManagerHashMethod = chosenHashMethod;
   if (transferManagerHashMethod == DEFAULT) {
       transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.

       std::cout << "The default checksum algorithm for TransferManager is "
                 << stringForHashMethod(transferManagerHashMethod)
                 << std::endl;
   }

   // Upload the large file using the transfer manager.
   if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                client)) {
       std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   std::vector<Aws::String> retrievedTransferManagerPartHashes;
   Aws::String retrievedTransferManagerFinalHash;

   // Retrieve all the hashes for the TransferManager upload.
   if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
       std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   AwsDoc::S3::Hasher locallyCalculatedFinalHash;
   std::vector<Aws::String> locallyCalculatedPartHashes;

   // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
   if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
```

```
                                    UPLOAD_BUFFER_SIZE,
                                    locallyCalculatedFinalHash,
                                    locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                         locallyCalculatedFinalHash, "TransferManager upload",
                         transferManagerHashMethod,
                         retrievedTransferManagerPartHashes,
                         locallyCalculatedPartHashes);

    printAsterisksLine();

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    multiPartUploadExplanations(key, chosenHashMethod);

    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          MULTI_PART_TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5;  // The default hash method for
multipart uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
```

```
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
              stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
```

```
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
 integrity hashing methods.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded
 object.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

 request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
```

```
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
```

```
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
```

```
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
```

```
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual
 parts retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                      const Hasher &localHash,
                                      const Aws::String &uploadtype,
                                      HASH_METHOD hashMethod,
                                      const std::vector<Aws::String>
 &retrievedPartHashes,
                                      const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
 << std::endl;
    if (!retrievedPartHashes.empty()) {
```

```
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
"'"
                      << std::endl;
            std::cerr << "Remote number of hashes - '"
                      << retrievedPartHashes.size()
                      << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                          << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
```

```
                          << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                          << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
                                         const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
 std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                          const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
 Transfer Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
```

```
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     bool useDefaultHashMethod,
                                     const std::shared_ptr<Aws::S3::S3Client>
 &client) {
   std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
           "executor", 25);
   Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
   transfer_config.s3Client = client;
   transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
   if (!useDefaultHashMethod) {
       if (hashMethod == MD5) {
           transfer_config.computeContentMD5 = true;
       } else {
           transfer_config.checksumAlgorithm =
 getChecksumAlgorithmForHashMethod(
                   hashMethod);
       }
   }
   transfer_config.errorCallback = transferManagerErrorCallback;
   transfer_config.transferStatusUpdatedCallback =
 transferManagerStatusCallback;

   std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
           transfer_config);

   std::cout << "Uploading the file..." << std::endl;
   std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
 transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

             bucket, key,

             "text/plain",

             Aws::Map<Aws::String, Aws::String>());
   uploadHandle->WaitUntilFinished();
   bool success =
           uploadHandle->GetStatus() ==
 Aws::Transfer::TransferStatus::COMPLETED;
```

```
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
 uploaded to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash
 values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
   \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes)
 {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
 uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
```

```
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
```

```
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
 &calculatedHash,
```

```
                                                                const
 std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
```

```
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
```

```
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
 hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
 concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
 &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {
    // Get object size.
```

```
   ioStream->seekg(0, ioStream->end);
   size_t objectSize = ioStream->tellg();
   ioStream->seekg(0, ioStream->beg);

   Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
   if (!useDefaultHashMethod) {
       if (hashMethod != MD5) {
           checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
       }
   }
   Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
   if (uploadID.empty()) {
       return false;
   }

   std::vector<unsigned char> totalHashBuffer;
   bool uploadSucceeded = true;
   std::streamsize uploadedBytes = 0;
   int partNumber = 1;
   Aws::Vector<Aws::S3::Model::CompletedPart> parts;
   while (uploadedBytes < objectSize) {
       std::cout << "Uploading part " << partNumber << "." << std::endl;

       std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
       std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
       ioStream->read((char *) buffer.data(), bytesToRead);
       Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
       std::shared_ptr<Aws::IOStream> body =
               Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                              &preallocatedStreamBuf);

       Hasher hasher;
       if (!hasher.calculateObjectHash(*body, hashMethod)) {
           std::cerr << "Error calculating hash." << std::endl;
           uploadSucceeded = false;
           break;
```

```
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        Aws::String calculatedHash;
        if (gUseCalculatedChecksum) {
            calculatedHash = base64HashString;
        }
        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,
                                                                        client);
        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:
```

```
completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }

            parts.push_back(completedPart);
        } else {
            std::cerr << "Error uploading part. " <<
                      uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
{
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
```

```
        } else {
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
 {
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
```

```
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
 valid." << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration
 &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
```

```
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
            << std::endl;
    printAsterisksLine();
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an
 object.\n"
            << "The default checksum algorithm for PutObject and MultiPart upload
 is an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
 the SDK calculate the value."
              << std::endl;
    std::cout
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;
```

```
    std::cout
            << "This workflow will locally compute checksums for files uploaded
 to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums
 are calculated."
            << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have
 failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using
 the GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
            << std::endl;
    std::cout
```

```
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
 uploading large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally
 and then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part.
 The final checksum is a concatenation of"
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
```

```
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
            << " in the section \"Using part-level checksums for multipart
 uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
 file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                    | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();
```

```
    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Work with Amazon S3 versioned objects using an AWS SDK

The following code example shows how to:

- Create a versioned S3 bucket.
- Get all versions of an object.
- Roll an object back to a previous version.
- Delete and restore a versioned object.
- Permanently delete all versions of an object.

Python

### SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that wrap S3 actions.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
```

```
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
```

```
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
```

```
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Upload the stanza of a poem to a versioned object and perform a series of actions on it.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
```

```
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )
```

```
    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - CreateBucket
  - DeleteObject

- ListObjectVersions
- PutBucketLifecycleConfiguration

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Serverless examples for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon S3 with AWS SDKs.

**Examples**

- Invoke a Lambda function from an Amazon S3 trigger

## Invoke a Lambda function from an Amazon S3 trigger

The following code examples show how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

.NET

**AWS SDK for .NET**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
```

```
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
```

```
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

**SDK for Go V2**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
```

```
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

**SDK for Java 2.x**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;
```

```
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
```

```
}
```

JavaScript

### SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

exports.handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
```

```
    }
};
```

Consuming an S3 event with Lambda using TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

### SDK for PHP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());
```

```
            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

**SDK for Python (Boto3)**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')
```

```
def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

**SDK for Ruby**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
```

```
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

**SDK for Rust**

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Serverless examples repository.

Consuming an S3 event with Lambda using Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
```

```
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }
```

```
    Ok(())
}
```

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Cross-service examples for Amazon S3 using AWS SDKs

The following sample applications use AWS SDKs to combine Amazon S3 with other AWS services. Each example includes a link to GitHub, where you can find instructions on how to set up and run the application.

**Examples**

- Build an Amazon Transcribe app
- Convert text to speech and back to text using an AWS SDK
- Create a photo asset management application that lets users manage photos using labels
- Create an Amazon Textract explorer application
- Detect PPE in images with Amazon Rekognition using an AWS SDK
- Detect entities in text extracted from an image using an AWS SDK
- Detect faces in an image using an AWS SDK
- Detect objects in images with Amazon Rekognition using an AWS SDK
- Detect people and objects in a video with Amazon Rekognition using an AWS SDK
- Save EXIF and other image information using an AWS SDK
- Transform data for your application with S3 Object Lambda

## Build an Amazon Transcribe app

The following code example shows how to use Amazon Transcribe to transcribe and display voice recordings in the browser.

JavaScript

**SDK for JavaScript (v3)**

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

This example is also available in the AWS SDK for JavaScript v3 developer guide.

**Services used in this example**

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Convert text to speech and back to text using an AWS SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon S3 bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

**SDK for Rust**

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon S3 bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Polly
- Amazon S3
- Amazon Transcribe

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Create a photo asset management application that lets users manage photos using labels

The following code examples show how to create a serverless application that lets users manage photos using labels.

.NET

**AWS SDK for .NET**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

**SDK for C++**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

**SDK for Java 2.x**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

**SDK for JavaScript (v3)**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

**SDK for Kotlin**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

**SDK for PHP**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

**SDK for Rust**

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For a deep dive into the origin of this example see the post on AWS Community.

**Services used in this example**

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

**SDK for JavaScript (v3)**

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

**SDK for Python (Boto3)**

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Detect PPE in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

**SDK for Java 2.x**

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

**SDK for JavaScript (v3)**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Detect entities in text extracted from an image using an AWS SDK

The following code example shows how to use Amazon Comprehend to detect entities in text extracted by Amazon Textract from an image that is stored in Amazon S3.

Python

**SDK for Python (Boto3)**

Shows how to use the AWS SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker, see the directions in TextractAndComprehendNotebook.ipynb.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Comprehend
- Amazon S3
- Amazon Textract

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Detect faces in an image using an AWS SDK

The following code example shows how to:

- Save an image in an Amazon S3 bucket.
- Use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (such as smiling).
- Display those details.

Rust

**SDK for Rust**

Save the image in an Amazon S3 bucket with an **uploads** prefix, use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (smiling, etc.), and display those details.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Detect objects in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect objects by category in images.

.NET

**AWS SDK for .NET**

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

**SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

**SDK for JavaScript (v3)**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

**SDK for Kotlin**

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3

- Amazon SES

Python

**SDK for Python (Boto3)**

Shows you how to use the AWS SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to AWS to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.
- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Java

**SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

**SDK for JavaScript (v3)**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.

- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Save EXIF and other image information using an AWS SDK

The following code example shows how to:

- Get EXIF information from a a JPG, JPEG, or PNG file.
- Upload the image file to an Amazon S3 bucket.
- Use Amazon Rekognition to identify the three top attributes (labels) in the file.
- Add the EXIF and label information to an Amazon DynamoDB table in the Region.

Rust

**SDK for Rust**

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon S3 bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

## Transform data for your application with S3 Object Lambda

The following code example shows how to transform data for your application with S3 Object Lambda.

.NET

**AWS SDK for .NET**

Shows how to add custom code to standard S3 GET requests to modify the requested object retrieved from S3 so that the object suit the needs of the requesting client or application.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

**Services used in this example**

- Lambda
- Amazon S3

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

# Troubleshooting

This section describes how to troubleshoot Amazon S3 features and explains how to get request IDs that you'll need when you contact AWS Support.

**Topics**



## Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3

**Important**

On May 13, 2024, we started deploying a change to eliminate charges for unauthorized requests that aren't initiated by the bucket owner. After the deployment of this change is completed, bucket owners will never incur request or bandwidth charges for requests that return `AccessDenied` (HTTP `403 Forbidden`) errors when these requests are initiated from outside of their individual AWS account or AWS organization. For more information on a full list of HTTP 3XX and 4XX status codes that won't be billed, see Billing for Amazon S3 error responses. This billing change requires no updates to your applications and applies to all S3 buckets. When deployment of this change is completed in all AWS Regions, we'll update our documentation.

The following topics cover the most common causes of Access Denied (403 Forbidden) errors in Amazon S3.

> **Note**
>
> For `Access Denied` (HTTP `403 Forbidden`), S3 doesn't charge the bucket owner when the request is initiated outside of the bucket owner's individual AWS account or the bucket owner's AWS organization.

**Topics**



> **Note**
>
> If you're trying to troubleshoot a permissions issue, start with the Bucket policies and IAM policies section, and be sure to follow the guidance in Tips for checking permissions.

# Bucket policies and IAM policies

## Bucket-level operations

If there is no bucket policy in place, then the bucket implicitly allows requests from any AWS Identity and Access Management (IAM) identity in the bucket-owning account. The bucket also implicitly denies requests from any other IAM identities from any other accounts, and anonymous (unsigned) requests. However, if there is no IAM user policy in place, the requester (unless they are the root user) is implicitly denied from making any requests. For more information about this evaluation logic, see Determining whether a request is denied or allowed within an account in the *IAM User Guide*.

## Object-level operations

If the object is owned by the bucket-owning account, the bucket policy and IAM user policy will function in the same way for object-level operations as they do for bucket-level operations. For example, if there is no bucket policy in place, then the bucket implicitly allows object requests from any IAM identity in the bucket-owning account. The bucket also implicitly denies object requests from any other IAM identities from any other accounts, and anonymous (unsigned) requests. However, if there is no IAM user policy in place, the requester (unless they are the root user) is implicitly denied from making any object requests.

If the object is owned by an external account, then access to the object can be granted only through object access control lists (ACLs). The bucket policy and IAM user policy can still be used to deny object requests.

Therefore, to ensure that your bucket policy or IAM user policy is not causing an Access Denied (403 Forbidden) error, make sure that the following requirements are met:

- For same-account access, there must not be an explicit `Deny` statement against the requester you are trying to grant permissions to, in either the bucket policy or the IAM user policy. If you want to grant permissions by using only the bucket policy and the IAM user policy, there must be at least one explicit `Allow` statement in one of these policies.
- For cross-account access, there must not be an explicit `Deny` statement against the requester you are trying to grant permissions to, in either the bucket policy or the IAM user policy. If you want to grant cross-account permissions by using only the bucket policy and IAM user policy, then both the bucket policy and the IAM user policy of the requester must include an explicit `Allow` statement.

> **Note**
>
> `Allow` statements in a bucket policy apply only to objects that are owned by the same bucket-owning account. However, `Deny` statements in a bucket policy apply to all objects regardless of object ownership.

**To review or edit your bucket policy**

> **Note**
>
> To view or edit a bucket policy, you must have the `s3:GetBucketPolicy` permission.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the name of the bucket that you want to view or edit a bucket policy for.
4. Choose the **Permissions** tab.
5. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.

To review or edit your bucket policy by using the AWS Command Line Interface (AWS CLI), use the get-bucket-policy command.

> **Note**
>
> If you get locked out of a bucket because of an incorrect bucket policy, sign in to the AWS Management Console by using your root user credentials. To regain access to your bucket, make sure to delete the bucket policy by using your root user credentials.

## Tips for checking permissions

To check whether the requester has proper permissions to perform an Amazon S3 operation, try the following:

- Identify the requester. If it's an unsigned request, then it's an anonymous request without an IAM user policy. If it's a request using a presigned URL, then the user policy will be the same as the one for the IAM user or role that signed the request.
- Verify that you're using the correct IAM user or role. You can verify your IAM user or role by checking the upper-right corner of the AWS Management Console or by using the aws sts get-caller-identity command.

- Check the IAM policies that are related to the IAM user or role. You can use one of the following methods:
  - Test IAM policies with the IAM policy simulator.
  - Review the different IAM policy types.
- If needed, edit your IAM user policy.
- Review the following examples of policies that explicitly deny or allow access:
  - Explicit allow IAM user policy: IAM: Allows and denies access to multiple services programmatically and in the console
  - Explicit allow bucket policy: Granting permissions to multiple accounts to upload objects or set object ACLs for public access
  - Explicit deny IAM user policy: AWS: Denies access to AWS based on the requested AWS Region
  - Explicit deny bucket policy: Require SSE-KMS for all objects written to a bucket

## Amazon S3 ACL settings

When checking your ACL settings, first review your Object Ownership setting to check whether ACLs are enabled on the bucket. Be aware that ACL permissions can be used only to grant permissions and cannot be used to reject requests. ACLs also cannot be used to grant access to requesters that are rejected by explicit denials in bucket policies or IAM user policies.

### The Object Ownership setting is set to bucket owner enforced

If the bucket owner enforced setting is enabled, then ACL settings are unlikely to cause an Access Denied (403 Forbidden) error because this setting disables all ACLs that apply to bucket and objects. Bucket owner enforced is the default (and recommended) setting for Amazon S3 buckets.

### The Object Ownership setting is set to bucket owner preferred or object writer

ACL permissions are still valid with the bucket owner preferred setting or the object writer setting. There are two kinds of ACLs: bucket ACLs and object ACLs. For the differences between these two types of ACLs, see Mapping of ACL permissions and access policy permissions.

Depending on the action of the rejected request, check the ACL permissions for your bucket or the object:

- If Amazon S3 rejected a `LIST`, PUT object, `GetBucketAcl`, or `PutBucketAcl` request, then review the ACL permissions for your bucket.

> **Note**
>
> You cannot grant GET object permissions with bucket ACL settings.

- If Amazon S3 rejected a GET request on an S3 object, or a PutObjectAcl request, then review the ACL permissions for the object.

> **Important**
>
> If the account that owns the object is different from the account that owns the bucket, then access to the object isn't controlled by the bucket policy.

### Troubleshooting an Access Denied (403 Forbidden) error from a GET object request during cross-account object ownership

Review the bucket's Object Ownership settings to determine the object owner. If you have access to the object ACLs, then you can also check the object owner's account. (To view the object owner's account, review the object ACL setting in the Amazon S3 console.) Alternatively, you can also make a `GetObjectAcl` request to find the object owner's canonical ID to verify the object owner account. By default, ACLs grant explicit allow permissions for GET requests to the object owner's account.

After you've confirmed that the object owner is different from the bucket owner, then depending on your use case and access level, choose one of the following methods to help address the Access Denied (403 Forbidden) error:

- **Disable ACLs (recommended)** – This method will apply to all objects and can be performed by the bucket owner. This method automatically gives the bucket owner ownership and full control over every object in the bucket. Before you implement this method, check the prerequisites for disabling ACLs. For information about how to set your bucket to bucket owner enforced (recommended) mode, see Setting Object Ownership on an existing bucket.

> **Important**
>
> To prevent an Access Denied (403 Forbidden) error, be sure to migrate the ACL permissions to a bucket policy before you disable ACLs. For more information, see Bucket policy examples for migrating from ACL permissions.

- **Change the object owner to the bucket owner** – This method can be applied to individual objects, but only the object owner (or a user with the appropriate permissions) can change an object's ownership. Additional PUT costs might apply. (For more information, see Amazon S3 pricing.) This method grants the bucket owner full ownership of the object, allowing the bucket owner to control access to the object through a bucket policy.

  To change the object's ownership, do one of the following:

  - You (the bucket owner) can copy the object back to the bucket.
  - You can change the Object Ownership setting of the bucket to bucket owner preferred. If versioning is disabled, the objects in the bucket are overwritten. If versioning is enabled, duplicate versions of the same object will appear in the bucket, which the bucket owner can set a lifecycle rule to expire. For instructions on how to change your Object Ownership setting, see Setting Object Ownership on an existing bucket.

    > **Note**
    >
    > When you update your Object Ownership setting to bucket owner preferred, the setting is only applied to new objects that are uploaded to the bucket.

  - You can have the object owner upload the object again with the `bucket-owner-full-control` canned object ACL.

  > **Note**
  >
  > For cross-account uploads, you can also require the `bucket-owner-full-control` canned object ACL in your bucket policy. For an example bucket policy, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- **Keep the object writer as the object owner** – This method doesn't change the object owner, but it does allow you to grant access to objects individually. To grant access to an object, you

must have the `PutObjectAcl` permission for the object. Then, to fix the Access Denied (403 Forbidden) error, add the requester as a grantee to access the object in the object's ACLs. For more information, see Configuring ACLs.

## S3 Block Public Access settings

If the failed request involves public access or public policies, then check the S3 Block Public Access settings on your account, bucket, or S3 access point. Starting in April 2023, all Block Public Access settings are enabled by default for new buckets. For more information about how Amazon S3 defines "public," see The meaning of "public".

When set to `TRUE`, Block Public Access settings act as explicit deny policies that override permissions allowed by ACLs, bucket policies, and IAM user policies. To determine whether your Block Public Access settings are rejecting your request, review the following scenarios:

- If the specified access control list (ACL) is public, then the `BlockPublicAcls` setting rejects your `PutBucketAcl` and `PutObjectACL` calls.
- If the request includes a public ACL, then the `BlockPublicAcls` setting rejects your `PutObject` calls.
- If the `BlockPublicAcls` setting is applied to an account and the request includes a public ACL, then any `CreateBucket` calls that include public ACLs will fail.
- If your request's permission is granted only by a public ACL, then the `IgnorePublicAcls` setting rejects the request.
- If the specified bucket policy allows public access, then the `BlockPublicPolicy` setting rejects your `PutBucketPolicy` calls.
- If the `BlockPublicPolicy` setting is applied to an access point, then all `PutAccessPointPolicy` and `PutBucketPolicy` calls that specify a public policy and are made through the access point will fail.
- If the access point or bucket has a public policy, then the `RestrictPublicBuckets` setting rejects all cross-account calls except for AWS service principals. This setting also rejects all anonymous (or unsigned) calls.

To review and update your Block Public Access setting configurations, see Configuring block public access settings for your S3 buckets.

# Amazon S3 encryption settings

Amazon S3 supports server-side encryption on your bucket. Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in AWS data centers and decrypts it for you when you access it.

By default, Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Amazon S3 also allows you to specify the server-side encryption method when uploading objects.

**To review your bucket's server-side encryption status and encryption settings**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the bucket that you want to check the encryption settings for.
4. Choose the **Properties** tab.
5. Scroll down to the **Default encryption** section and view the **Encryption type** settings.

To check your encryption settings by using the AWS CLI, use the get-bucket-encryption command.

**To check the encryption status of an object**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the name of the bucket that contains the object.
4. From the **Objects** list, choose the name of the object that you want to add or change encryption for.

   The object's details page appears.
5. Scroll down to the **Server-side encryption settings** section to view the object's server-side encryption settings.

To check your object encryption status by using the AWS CLI, use the head-object command.

### Encryption and permissions requirements

Amazon S3 supports three types of server-side encryption:

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)

Based on your encryption settings, make sure that the following permissions requirements are met:

- **SSE-S3** – No extra permissions are required.
- **SSE-KMS (with a customer managed key)** – To upload objects, the `kms:GenerateDataKey` permission on the AWS KMS key is required. To download objects and perform multipart uploads of objects, the `kms:Decrypt` permission on the KMS key is required.
- **SSE-KMS (with an AWS managed key)** – The requester must be from the same account that owns the `aws/s3` KMS key. The requester must also have the correct Amazon S3 permissions to access the object.
- **SSE-C (with a customer provided key)** – No additional permissions are required. You can configure the bucket policy to require and restrict server-side encryption with customer-provided encryption keys for objects in your bucket.

If the object is encrypted with a customer managed key, make sure that the KMS key policy allows you to perform the `kms:GenerateDataKey` or `kms:Decrypt` actions. For instructions on checking your KMS key policy, see Viewing a key policy in the *AWS Key Management Service Developer Guide*.

## S3 Object Lock settings

If your bucket has S3 Object Lock enabled and the object is protected by a retention period or legal hold, Amazon S3 returns an Access Denied (403 Forbidden) error when you try to delete the object.

**To check whether the bucket has Object Lock enabled**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. From the **Buckets** list, choose the name of the bucket that you want to review.
4. Choose the **Properties** tab.
5. Scroll down to the **Object Lock** section. Verify whether the **Object Lock** setting is **Enabled** or **Disabled**.

To determine whether the object is protected by a retention period or legal hold, view the lock information for your object.

If the object is protected by a retention period or legal hold, check the following:

- If the object version is protected by the compliance retention mode, there is no way to permanently delete it. A permanent `DELETE` request from any requester, including the root user, will result in an Access Denied (403 Forbidden) error. Also, be aware that when you submit a `DELETE` request for an object that's protected by the compliance retention mode, Amazon S3 creates a delete marker for the object.
- If the object version is protected with governance retention mode and you have the `s3:BypassGovernanceRetention` permission, you can bypass the protection and permanently delete the version. For more information, see Bypassing governance mode.
- If the object version is protected by a legal hold, then a permanent `DELETE` request can result in an Access Denied (403 Forbidden) error. To permanently delete the object version, you must remove the legal hold on the object version. To remove a legal hold, you must have the `s3:PutObjectLegalHold` permission. For more information about removing a legal hold, see Configuring S3 Object Lock.

## VPC endpoint policy

If you're accessing Amazon S3 by using a virtual private cloud (VPC) endpoint, make sure that the VPC endpoint policy is not blocking you from accessing your Amazon S3 resources. By default, the VPC endpoint policy allows all requests to Amazon S3. You can also configure the VPC endpoint policy to restrict certain requests. For information about how to check your VPC endpoint policy, see Control access to VPC endpoints by using endpoint policies in the *AWS PrivateLink Guide*.

## AWS Organizations policies

If your AWS account belongs to an organization, AWS Organizations policies can block you from accessing Amazon S3 resources. By default, AWS Organizations policies do not block any requests

to Amazon S3. However, make sure that your AWS Organizations policies haven't been configured to block access to S3 buckets. For instructions on how to check your AWS Organizations policies, see Listing all policies in the *AWS Organizations User Guide*.

## Access point settings

If you receive an Access Denied (403 Forbidden) error while making requests through Amazon S3 access points, you might need to check the following:

- The configurations for your access points
- The IAM user policy that's used for your access points
- The bucket policy that's used to manage or configure your cross-account access points

#### Access point configurations and policies

- When you create an access point, you can choose to designate **Internet** or **VPC** as the network origin. If the network origin is set to VPC only, Amazon S3 will reject any requests made to the access point that don't originate from the specified VPC. To check the network origin of your access point, see Creating access points restricted to a virtual private cloud.
- With access points, you can also configure custom Block Public Access settings, which work similarly to the Block Public Access settings at the bucket or account level. To check your custom Block Public Access settings, see Managing public access to access points.
- To make successful requests to Amazon S3 by using access points, make sure that the requester has the necessary IAM permissions. For more information, see Configuring IAM policies for using access points.
- If the request involves cross-account access points, make sure that the bucket owner has updated the bucket policy to authorize requests from the access point. For more information, see Granting permissions for cross-account access points.

If the Access Denied (403 Forbidden) error still persists after checking all the items in this topic, retrieve your Amazon S3 request ID and contact AWS Support for additional guidance.

# Troubleshoot Batch Operations

The following topics list common errors to help you troubleshoot issues that you might encounter during Batch Operations.

**Common Errors**

- Job report isn't delivered when there is a permissions issue or an S3 Object Lock retention mode is enabled
- S3 Batch Replication failure with error: Manifest generation found no keys matching the filter criteria
- Batch Operations failures occur after adding a new replication rule to an existing replication configuration
- Batch Operations failing objects with the error 400 InvalidRequest: Task failed due to missing VersionId
- Create job failure with job tag option enabled
- Access Denied to read the manifest

## Job report isn't delivered when there is a permissions issue or an S3 Object Lock retention mode is enabled

The following error occurs if required permissions are missing or Object Lock retention mode (either governance mode or compliance mode) is enabled on the destination bucket.

Error: Reasons for failure. The job report could not be written to your report bucket. Please check your permissions.

The IAM role and trust policy must be configured to allow S3 Batch Operations access to PUT objects in the bucket where the report will be delivered. If these required permissions are missing a job report delivery failure occurs.

When a retention mode is enabled, the bucket is write-once-read-many (WORM) protected. Object Lock with retention mode enabled on the destination bucket is not supported so job completion report delivery attempts fail. To fix this problem, choose a destination bucket for your job completion reports that doesn't have an Object Lock retention mode enabled.

## S3 Batch Replication failure with error: Manifest generation found no keys matching the filter criteria

Error: Manifest generation found no keys matching the filter criteria.

This error occurs for one of the following reasions:

- When objects in the source bucket are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.

  To use Batch Replication on these objects, first restore them to the S3 Standard storage class by using an S3 Initiate Restore Object operation in a Batch Operations job. For more information, see Restoring an archived object and Restore objects (Batch Operations). After you've restored the objects, you can replicate them by using a Batch Replication job.

- When the provided filter criteria doesn't match any valid objects in the source bucket.

  Verify and correct the filter criteria. For example, in the Batch Replication rule, the filter criteria is looking for all objects in the *amzn-s3-demo-bucket* with the prefix `Tax/`. If the prefix name was entered inaccurately, with a slash in the beginning and the end `/Tax/` instead of only at the end, then no S3 objects were found. To resolve the error, correct the prefix, in this case, from `/Tax/` to `Tax/` in the replication rule.

## Batch Operations failures occur after adding a new replication rule to an existing replication configuration

Batch Operations attempts to perform existing object replication for every rule in the source bucket's replication configuration. If there are problems with any of the existing replication rules, failures might occur.

The Batch Operations job's completion report explains the job failure reasons. For a list of common errors, see Amazon S3 replication failure reasons.

## Batch Operations failing objects with the error 400 InvalidRequest: Task failed due to missing VersionId

The following example error occurs if a Batch Operations job is performing actions on objects in a versioned bucket and encounters an object in the manifest with an empty version ID field.

Error: *BUCKET_NAME,prefix/file_name*,failed,400,InvalidRequest,Task failed due to missing VersionId

This error occurs because the version ID field in the manifest is an empty string, instead of the literal `null` string.

Batch Operations will fail for that particular object or objects, but not the entire job. This problem occurs if the manifest format is configured to use version IDs during the operation. Non-versioned jobs don't encounter this issue because they operate only on the most recent version of each object and ignore the version IDs in the manifest.

To fix this problem, convert the empty version IDs to `null` strings. For more information, see the section called “Converting empty version ID strings to null strings”.

## Create job failure with job tag option enabled

Without the `s3:PutJobTagging` permission, creating Batch Operations jobs with the job tag option enabled causes `403 access denied` errors.

To create Batch Operations jobs with the job tag option enabled, the AWS Identity and Access Management (IAM) user that's creating the Batch Operations job must have the `s3:PutJobTagging` permission in addition to the `s3:CreateJob` permission.

For more information about the permissions required for Batch Operations, see the section called “Granting permissions”.

## Access Denied to read the manifest

If Batch Operations can't read the manifest file when you attempt to create a Batch Operations job, the following errors can occur.

**AWS CLI**

Reason for failure Reading the manifest is forbidden: AccessDenied

**Amazon S3 console**

Warning: Unable to get the manifest object's ETag. Specify a different object to continue.

To solve this problem, do the following:

- Verify that the IAM role for the AWS account that you used to create the Batch Operations job has `s3:GetObject` permissions. The account's IAM role must have `s3:GetObject` permissions to allow Batch Operations to read the manifest file.

  For more information about the permissions required for Batch Operations, see the section called “Granting permissions”.

- Check the manifest objects' metadata for any access mismatches with S3 Object Ownership. For more information about S3 Object Ownership, see the section called "Controlling object ownership".
- Check whether AWS Key Management Service (AWS KMS) keys are used to encrypt the manifest file.

  Batch Operations support *CSV inventory reports* that are AWS KMS-encrypted. However, Batch Operations don't support *CSV manifest files* that are AWS KMS-encrypted. For more information, see Configuring Amazon S3 Inventory and Specifying a manifest.

# Troubleshoot Amazon S3 Lifecycle issues

The following information can help you troubleshoot common issues with Amazon S3 Lifecycle rules.

**Topics**



## I ran a list operation on my bucket and saw objects that I thought were expired or transitioned by a lifecycle rule.

S3 Lifecycle object transitions and object expirations are asynchronous operations. Therefore, there might be a delay between the time that the objects are eligible for expiration or transition and

the time that they are actually transitioned or expired. Changes in billing are applied as soon as the lifecycle rule is satisfied, even if the action isn't complete. The exception to this behavior is if you have a lifecycle rule set to transition to the S3 Intelligent-Tiering storage class. In that case, billing changes don't occur until the object has transitioned to S3 Intelligent-Tiering. For more information about changes in billing, see Setting lifecycle configuration on a bucket.

> **Note**
>
> Amazon S3 doesn't transition objects that are smaller than 128 KB from the S3 Standard or S3 Standard-IA storage class to the S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage class.

## How do I monitor the actions taken by my lifecycle rules?

To monitor actions taken by lifecycle rules, you can use the following features:

- **S3 Event Notifications** – You can set up S3 Event Notifications so that you're notified of any S3 Lifecycle expiration or transition events.
- **S3 server access logs** – You can enable server access logs for your S3 buckets to capture S3 Lifecycle actions, such as object transitions to another storage class or object expirations. For more information, see Lifecycle and logging.

To view the changes in your storage caused by lifecycle actions on a daily basis, we recommend using S3 Storage Lens dashboards instead of using Amazon CloudWatch metrics. In your Storage Lens dashboard, you can view the following metrics, which monitor the object count or size:

- **Current version bytes**
- **Current version object count**
- **Noncurrent version bytes**
- **Noncurrent version object count**
- **Delete marker object count**
- **Delete marker storage bytes**
- **Incomplete multipart upload bytes**
- **Incomplete multipart upload object count**

# My S3 object count still increases, even after setting up lifecycle rules on a versioning-enabled bucket.

In a versioning-enabled bucket, when an object is expired, the object isn't completely deleted from the bucket. Instead, a delete marker is created as the newest version of the object. Delete markers are still counted as objects. Therefore, if a lifecycle rule is created to expire only the current versions, then the object count in the S3 bucket actually increases instead of going down.

For example, let's say an S3 bucket is versioning-enabled with 100 objects, and a lifecycle rule is set to expire current versions of the object after 7 days. After the seventh day, the object count increases to 200 because 100 delete markers are created in addition to the 100 original objects, which are now the noncurrent versions. For more information about S3 Lifecycle configuration rule actions for versioning-enabled buckets, see Setting lifecycle configuration on a bucket.

To permanently remove objects, add an additional lifecycle configuration to delete the previous versions of the objects, expired delete markers, and incomplete multipart uploads. For instructions on how to create new lifecycle rules, see Setting lifecycle configuration on a bucket.

**Note**

- Amazon S3 rounds the transition or expiration date of an object to midnight UTC the next day.

  When evaluating objects for lifecycle actions, Amazon S3 uses the object creation time in UTC. For example, consider a nonversioned bucket with a lifecycle rule that's configured to expire objects after one day. Suppose that an object was created on January 1 at 17:05 Pacific Daylight Time (PDT), which corresponds to January 2 at 00:05 UTC. The object becomes one day old at 00:05 UTC on January 3, which makes it eligible for expiration when S3 Lifecycle evaluates objects at 00:00 UTC on January 4.

  Because Amazon S3 lifecycle actions occur asynchronously, there might be some delay between the date specified in the lifecycle rule and the actual physical transition of the object. For more information, see Transition or expiration delay.

  For more information, see Lifecycle rules: Based on an object's age.

- For S3 objects that are protected by Object Lock, current versions are not permanently deleted. Instead, a delete marker is added to the objects, making them noncurrent. Noncurrent versions are then preserved and are not permanently expired.

## How do I empty my S3 bucket by using lifecycle rules?

S3 Lifecycle rules are an effective tool to empty an S3 bucket with millions of objects. To delete a large number of objects from your S3 bucket, make sure to use these two pairs of lifecycle rules:

- **Expire current versions of objects** and **Permanently delete previous versions of objects**
- **Delete expired delete markers** and **Delete incomplete multipart uploads**

For steps on how to create a lifecycle configuration rule, see Setting lifecycle configuration on a bucket.

**Note**

For S3 objects that are protected by Object Lock, current versions are not permanently deleted. Instead, a delete marker is added to the objects, making them noncurrent. Noncurrent versions are then preserved and are not permanently expired.

## My Amazon S3 bill increased after transitioning objects to a lower-cost storage class.

There are several reasons that your bill might increase after transitioning objects to a lower-cost storage class:

- S3 Glacier overhead charges for small objects

  For each object that is transitioned to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, a total overhead of 40 KB is associated with this storage update. As part of the 40 KB overhead, 8 KB is used to store metadata and the name of the object. This 8 KB is charged according to S3 Standard rates. The remaining 32 KB is used for indexing and related metadata. This 32 KB is charged according to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive pricing.

Therefore, if you're storing many smaller sized objects, we don't recommend using lifecycle transitions. Instead, to reduce any overhead charges, consider aggregating many smaller objects into a smaller number of large objects before storing them in Amazon S3. For more information about cost considerations, see Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival).

- Minimum storage charges

  Some S3 storage classes have minimum storage-duration requirements. Objects that are deleted, overwritten, or transitioned from those classes before the minimum duration is satisfied are charged a prorated early transition or deletion fee. These minimum storage-duration requirements are as follows:

  - S3 Standard-IA and S3 One Zone-IA – 30 days
  - S3 Glacier Flexible Retrieval and S3 Glacier Instant Retrieval – 90 days
  - S3 Glacier Deep Archive – 180 days

  For more information about these requirements, see the *Constraints* section of Transitioning objects using S3 Lifecycle. For general S3 pricing information, see Amazon S3 pricing and the AWS Pricing Calculator.

- Lifecycle transition costs

  Each time an object is transitioned to a different storage class by a lifecycle rule, Amazon S3 counts that transition as one transition request. The costs for these transition requests are in addition to the costs of these storage classes. If you plan to transition a large number of objects, consider the request costs when transitioning to a lower tier. For more information, see Amazon S3 pricing.

## I've updated my bucket policy, but my S3 objects are still being deleted by expired lifecycle rules.

`Deny` statements in a bucket policy don't prevent the expiration of the objects defined in a lifecycle rule. Lifecycle actions (such as transitions or expirations) don't use the S3 `DeleteObject` operation. Instead, S3 Lifecycle actions are performed by using internal S3 endpoints. (For more information, see Lifecycle and logging.)

To prevent your lifecycle rule from taking any action, you must edit, delete, or disable the rule.

## Can I recover S3 objects that are expired by S3 Lifecycle rules?

The only way to recover objects that are expired by S3 Lifecycle is through versioning, which must be in place before the objects become eligible for expiration. You cannot undo the expiration operations that are performed by lifecycle rules. If objects are permanently deleted by the S3 Lifecycle rules that are in place, you cannot recover these objects. To enable versioning on a bucket, see the section called "Using S3 Versioning".

If you have applied versioning to the bucket and the noncurrent versions of the objects are still intact, you can restore previous versions of the expired objects. For more information about the behavior of S3 Lifecycle rule actions and versioning states, see the *Lifecycle actions and bucket versioning state* table in Elements to describe lifecycle actions.

> **Note**
>
> If the S3 bucket is protected by AWS Backup or S3 Replication, you might also be able to use these features to recover your expired objects.

## How can I exclude a prefix from my lifecycle rule?

S3 Lifecycle doesn't support excluding prefixes in your rules. Instead, use tags to tag all of the objects that you want to include in the rule. For more information about using tags in your lifecycle rules, see the section called "Example 1: Specifying a filter".

## How can I include multiple prefixes in my lifecycle rule?

S3 Lifecycle doesn't support including multiple prefixes in your rules. Instead, use tags to tag all of the objects that you want to include in the rule. For more information about using tags in your lifecycle rules, see the section called "Example 1: Specifying a filter".

However, if you have one or more prefixes that start with the same characters, you can include all of those prefixes in your rule by specifying a partial prefix with no trailing slash (/) in the filter. For example, suppose that you have these prefixes:

```
sales1999/
sales2000/
sales2001/
```

To include all three prefixes in your rule, specify `<Prefix>sales</Prefix>` in your lifecycle rule.

# Troubleshooting replication

This section lists troubleshooting tips for Amazon S3 Replication and information about S3 Batch Replication errors.

**Topics**



## Troubleshooting tips for S3 Replication

If object replicas don't appear in the destination bucket after you configure replication, use these troubleshooting tips to identify and fix issues.

- The majority of objects replicate within 15 minutes. The time that it takes Amazon S3 to replicate an object depends on several factors, including the source and destination Region pair, and the size of the object. For large objects, replication can take up to several hours. For visibility into replication times, you can use S3 Replication Time Control (S3 RTC).

  If the object that is being replicated is large, wait a while before checking to see whether it appears in the destination. You can also check the replication status of the source object. If the object replication status is `PENDING`, Amazon S3 has not completed the replication. If the object replication status is `FAILED`, check the replication configuration that's set on the source bucket. Additionally, to receive information about failures during replication, you can set up Amazon S3 Event Notifications replication to receive failure events. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.
- You can call the `HeadObject` API operation to check the replication status of an object. The `HeadObject` API operation returns the `PENDING`,`COMPLETED`, or `FAILED` replication status of an object. In a response to a `HeadObject` API call, the replication status is returned in the `x-amz-replication-status` element.

  **Note**

  To run `HeadObject`, you must have read access to the object that you're requesting. A `HEAD` request has the same options as a `GET` request, without performing a `GET`

operation. For example, to run a `HeadObject` request by using the AWS Command Line Interface (AWS CLI), you can run the following command. Replace the *`user input placeholders`* with your own information.

```
aws s3api head-object --bucket my-bucket --key index.html
```

- After `HeadObject` returns the objects with a `FAILED` replication status, you can use S3 Batch Replication to replicate those failed objects. Alternatively, you can re-upload the failed objects to the source bucket, which will initiate replication for the new objects.
- In the replication configuration on the source bucket, verify the following:
  - The Amazon Resource Name (ARN) of the destination bucket is correct.
  - The key name prefix is correct. For example, if you set the configuration to replicate objects with the prefix `Tax`, then only objects with key names such as `Tax/document1` or `Tax/document2` are replicated. An object with the key name `document3` is not replicated.
  - The status of the replication rule is `Enabled`.
- Verify that versioning has not been suspended on any bucket in the replication configuration. Both the source and destination buckets must have versioning enabled.
- If a replication rule is set to **Change object ownership to the destination bucket owner**, then the AWS Identity and Access Management (IAM) role that's used for replication must have the `s3:ObjectOwnerOverrideToBucketOwner` permission. This permission is granted on the resource (in this case, the destination bucket). For example, the following `Resource` statement shows how to grant this permission on the destination bucket:

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::DestinationBucket/*"
}
```

- If the destination bucket is owned by another account, the owner of the destination bucket must also grant the `s3:ObjectOwnerOverrideToBucketOwner` permission to the source bucket owner through the destination bucket policy. To use the following example bucket policy, replace the *`user input placeholders`* with your own information:

```
{
```

```
  "Version": "2012-10-17",
  "Id": "Policy1644945280205",
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::DestinationBucket/*"
    }
  ]
}
```

> **Note**
>
> If the destination bucket's object ownership settings include **Bucket owner enforced**, then you don't need to update the setting to **Change object ownership to the destination bucket owner** in the replication rule. The object ownership change will occur by default. For more information about changing replica ownership, see Changing the replica owner.

- If you're setting the replication configuration in a cross-account scenario, where the source and destination buckets are owned by different AWS accounts,the destination buckets can't be configured as Requester Pays buckets. For more information, see Using Requester Pays buckets for storage transfers and usage.
- If a bucket's source objects are encrypted with an AWS Key Management Service (AWS KMS) key, then the replication rule must be configured to include AWS KMS-encrypted objects. Make sure to select **Replicate objects encrypted with AWS KMS** under your **Encryption** settings in the Amazon S3 console. Then, select an AWS KMS key for encrypting destination objects.

> **Note**
>
> If the destination bucket is in a different account, specify an AWS KMS customer managed key that is owned by the destination account. Don't use the default Amazon

S3 managed key (`aws/s3`). Using the default key encrypts the objects with the Amazon S3 managed key that is owned by the source account, preventing the object from being shared with another account. As a result, the destination account won't be able to access the objects in the destination bucket.

To use an AWS KMS key that belongs to the destination account to encrypt the destination objects, the destination account must grant the `kms:GenerateDataKey` and `kms:Encrypt` permissions to the replication role in the KMS key policy. To use the following example statement in your KMS key policy, replace the *`user input placeholders`* with your own information:

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
}
```

If you use an asterisk (*) for the `Resource` statement in the AWS KMS key policy, the policy grants permission to use the KMS key to only the replication role. The policy doesn't allow the replication role to elevate its permissions.

By default, the KMS key policy grants the root user full permissions to the key. These permissions can be delegated to other users in the same account. Unless there are `Deny` statements in the source KMS key policy, using an IAM policy to grant the replication role permissions to the source KMS key is sufficient.

**Note**

KMS key policies that restrict access to specific CIDR ranges, VPC endpoints, or S3 access points can cause replication to fail.

If either the source or destination KMS keys grant permissions based on the encryption context, confirm that Amazon S3 Bucket Keys are turned on for the buckets. If the buckets have S3 Bucket Keys turned on, the encryption context must be the bucket-level resource, like this:

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::SOURCE_BUCKET_NAME"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::DESTINATION_BUCKET_NAME"
     ]
```

In addition to the permissions granted by the KMS key policy, the source account must add the following minimum permissions to the replication role's IAM policy:

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": [
        "SourceKmsKeyArn"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
        "DestinationKmsKeyArn"
    ]
}
```

For more information about how to replicate objects that are encrypted with AWS KMS, see Replicating encrypted objects.

- If the destination bucket is owned by another AWS account, verify that the bucket owner has a bucket policy on the destination bucket that allows the source bucket owner to replicate objects.

For an example, see Configuring replication when source and destination buckets are owned by different accounts.

- If your objects still aren't replicating after you've validated the permissions, check for any explicit Deny statements in the following locations:
  - Deny statements in the source or destination bucket policies. Replication fails if the bucket policy denies access to the replication role for any of the following actions:

    Source bucket:

    ```
    "s3:GetReplicationConfiguration",
    "s3:ListBucket",
    "s3:GetObjectVersionForReplication",
    "s3:GetObjectVersionAcl",
    "s3:GetObjectVersionTagging"
    ```

    Destination buckets:

    ```
    "s3:ReplicateObject",
    "s3:ReplicateDelete",
    "s3:ReplicateTags"
    ```

  - Deny statements or permissions boundaries attached to the IAM role can cause replication to fail.
  - Deny statements in AWS Organizations service control policies attached to either the source or destination accounts can cause replication to fail.
- If an object replica doesn't appear in the destination bucket, the following issues might have prevented replication:
  - Amazon S3 doesn't replicate an object in a source bucket that is a replica created by another replication configuration. For example, if you set a replication configuration from bucket A to bucket B to bucket C, Amazon S3 doesn't replicate object replicas in bucket B to bucket C.
  - A source bucket owner can grant other AWS accounts permission to upload objects. By default, the source bucket owner doesn't have permissions for the objects created by other accounts. The replication configuration replicates only the objects for which the source bucket owner has access permissions. The source bucket owner can grant other AWS accounts permissions

to create objects conditionally, requiring explicit access permissions on those objects. For an example policy, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- Suppose that in the replication configuration, you add a rule to replicate a subset of objects that have a specific tag. In this case, you must assign the specific tag key and value at the time the object is created in order for Amazon S3 to replicate the object. If you first create an object and then add the tag to the existing object, Amazon S3 doesn't replicate the object.
- Use Amazon S3 Event Notifications to notify you of instances when objects do not replicate to their destination AWS Region. Amazon S3 event notifications are available through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.

  You can also view replication failure reasons by using Amazon S3 Event Notifications. To review the list of failure reasons, see Amazon S3 replication failure reasons.

## Batch Replication errors

To troubleshoot objects that aren't replicating to the destination bucket, check the different types of permissions for the your bucket, replication role, and IAM role that's used to create the Batch Replication job. Also, make sure to check the public access settings and bucket ownership settings.

While using Batch Replication, you might encounter one of these errors:

- Batch operation status is failed with reason: The job report could not be written to your report bucket.

  This error occurs if the IAM role that's used for the Batch Operations job is unable to put the completion report into the location that was specified when you created the job. To resolve this error, check that the IAM role has `PutObject` permissions for the bucket where you want to save the Batch Operations completion report. It's a best practice to deliver the report to a bucket different from the source bucket.
- Batch operation is completed with failures and Total failed is not 0.

  This error occurs if there are insufficient object permissions issues with the Batch Replication job that is running. If you're using a replication rule for your Batch Replication job, make sure that the IAM role used for replication has the proper permissions to access objects from either

the source or destination bucket. You can also check the Batch Replication completion report to review the specific Amazon S3 replication failure reason.

- Batch job ran successfully but the number of objects expected in destination bucket is not the same.

  This error occurs when there's a mismatch between the objects listed in the manifest that's supplied in the Batch Replication job and the filters that you selected when you created the job. You might also receive this message when the objects in your source bucket don't match any replication rules and aren't included in the generated manifest.

# Troubleshoot server access logging

The following topics can help you troubleshoot issues that you might encounter when setting up logging with Amazon S3.

**Topics**

- Common error messages when setting up logging
- Troubleshooting delivery failures

## Common error messages when setting up logging

The following common error messages can appear when you're enabling logging through the AWS Command Line Interface (AWS CLI) and AWS SDKs:

Error: Cross S3 location logging not allowed

If the destination bucket (also known as a *target bucket*) is in a different Region than the source bucket, a Cross S3 location logging not allowed error occurs. To resolve this error, make sure that the destination bucket configured to receive the access logs is in the same AWS Region and AWS account as the source bucket.

Error: The owner for the bucket to be logged and the target bucket must be the same

When you're enabling server access logging, this error occurs if the specified destination bucket belongs to a different account. To resolve this error, make sure that the destination bucket is in the same AWS account as the source bucket.

> **Note**
>
> We recommend that you choose a destination bucket that's different from the source bucket. When the source bucket and destination bucket are the same, additional logs are created for the logs that are written to the bucket, which can increase your storage bill. These extra logs about logs can also make it difficult to find the particular logs that you're looking for. For simpler log management, we recommend saving access logs in a different bucket. For more information, see the section called “How do I enable log delivery?”.

Error: The target bucket for logging does not exist

The destination bucket must exist prior to setting the configuration. This error indicates that the destination bucket doesn't exist or can't be found. Make sure that the bucket name is spelled correctly, and then try again.

Error: Target grants not allowed for bucket owner enforced buckets

This error indicates that the destination bucket uses the Bucket owner enforced setting for S3 Object Ownership. The Bucket owner enforced setting doesn't support destination (target) grants. For more information, see Permissions for log delivery.

## Troubleshooting delivery failures

To avoid server access logging issues, make sure that you're following these best practices:

- **The S3 log delivery group has write access to the destination bucket** – The S3 log delivery group delivers server access logs to the destination bucket. A bucket policy or bucket access control list (ACL) can be used to grant write access to the destination bucket. However, we recommend that you use a bucket policy instead of an ACL. For more information about how to grant write access to your destination bucket, see Permissions for log delivery.

  > **Note**
  >
  > If the destination bucket uses the Bucket owner enforced setting for Object Ownership, be aware of the following:
  >
  > - ACLs are disabled and no longer affect permissions. This means that you can't update your bucket ACL to grant access to the S3 log delivery group. Instead, to grant access

to the logging service principal, you must update the bucket policy for the destination bucket.

- You can't include destination grants in your `PutBucketLogging` configuration.

- **The bucket policy for the destination bucket allows access to the logs** – Check the bucket policy of the destination bucket. Search the bucket policy for any statements that contain `"Effect": "Deny"`. Then, verify that the `Deny` statement isn't preventing access logs from being written to the bucket.
- **S3 Object Lock isn't enabled on the destination bucket** – Check if the destination bucket has Object Lock enabled. Object Lock blocks server access log delivery. You must choose a destination bucket that doesn't have Object Lock enabled.
- **Amazon S3 managed keys (SSE-S3) is selected if default encryption is enabled on the destination bucket** – You can use default bucket encryption on the destination bucket only if you use server-side encryption with Amazon S3 managed keys (SSE-S3). Default server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) is not supported for server access logging destination buckets. For more information about how to enable default encryption, see Configuring default encryption.
- **The destination bucket does not have Requester Pays enabled** – Using a Requester Pays bucket as the destination bucket for server access logging is not supported. To allow delivery of server access logs, disable the Requester Pays option on the destination bucket.
- **Review your AWS Organizations service control policy** – When you're using AWS Organizations, check the service control policies to make sure that Amazon S3 access is allowed. Service control policies specify the maximum permissions for the affected accounts. Search the service control policy for any statements that contain `"Effect": "Deny"` and verify that `Deny` statements aren't preventing any access logs from being written to the bucket. For more information, see Service control policies (SCPs) in the *AWS Organizations User Guide*.
- **Allow some time for recent logging configuration changes to take effect** – Enabling server access logging for the first time, or changing the destination bucket for logs, requires time to fully take effect. It might take longer than an hour for all requests to be properly logged and delivered.

  To check for log delivery failures, enable request metrics in Amazon CloudWatch. If the logs are not delivered within a few hours, look for the `4xxErrors` metric, which can indicate log delivery failures. For more information about enabling request metrics, see the section called “Creating a metrics configuration for all objects”.

# Troubleshoot versioning

The following topics can help you troubleshoot some common Amazon S3 versioning issues.

**Topics**



## I want to recover objects that were accidentally deleted in a versioning-enabled bucket

In general, when object versions are deleted from S3 buckets, there is no way for Amazon S3 to recover them. However, if you have enabled S3 Versioning on your S3 bucket, a DELETE request that doesn't specify a version ID cannot permanently delete an object. Instead, a delete marker is added as a placeholder. This delete marker becomes the current version of the object.

To verify whether your deleted objects are permanently deleted or temporarily deleted (with a delete marker in their place), do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object.
4. In the **Objects** list, Turn on the **Show versions** toggle to the right of the search bar, and then search for the deleted object in the search bar. This toggle is available only if versioning was previously enabled on the bucket.

   You can also use S3 Inventory to search for deleted objects.
5. If you can't find the object after toggling **Show versions** or creating an inventory report, and you also cannot find a delete marker of the object, the deletion is permanent and the object cannot be recovered.

You can also verify a deleted object's status by using the `HeadObject` API operation from the AWS Command Line Interface (AWS CLI). To do so, use the following `head-object` command and replace the *`user input placeholders`* with your own information:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

If you run the `head-object` command on a versioned object whose current version is a delete marker, you will receive a 404 Not Found error. For example:

An error occurred (404) when calling the HeadObject operation: Not Found

If you run the `head-object` command on a versioned object and provide the object's version ID, Amazon S3 retrieves the object's metadata, confirming that the object still exists and is not permanently deleted.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

If the object is found and the newest version is a delete marker, the previous version of the object still exists. Because the delete marker is the current version of the object, you can recover the object by deleting the delete marker.

After you permanently remove the delete marker, the second newest version of the object becomes the current version of the object, making your object available once again. For a visual depiction of how objects are recovered, see Removing delete markers.

To remove a specific version of an object, you must be the bucket owner. To delete a delete marker permanently, you must include its version ID in a `DeleteObject` request. To delete the delete marker, use the following command, and replace the *`user input placeholders`* with your own information:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

For more information about the `delete-object` command, see delete-object in the *AWS CLI Command Reference*. For more information about permanently deleting delete markers, see Managing delete markers.

## I want to permanently delete versioned objects

In a versioning-enabled bucket, a `DELETE` request without a version ID cannot permanently delete an object. Instead, such a request inserts a delete marker.

To permanently delete versioned objects, you can choose from the following methods:

- Create an S3 Lifecycle rule to permanently delete noncurrent versions. To permanently delete noncurrent versions, select **Permanently delete noncurrent versions of objects**, and then enter a number under **Days after objects become noncurrent**. You can optionally specify the number of newer versions to retain by entering a value under **Number of newer versions to retain**. For more information about creating this rule, see Setting an S3 Lifecycle configuration.
- Delete a specified version by including the version ID in the `DELETE` request. For more information, see How to delete versioned objects permanently.
- Create a lifecycle rule to expire current versions. To expire current versions of objects, select **Expire current versions of objects**, and then add a number under **Days after object creation**. For more information about creating this lifecycle rule, see Setting an S3 Lifecycle configuration.
- To permanently delete all versioned objects and delete markers, create two lifecycle rules: one to expire current versions and permanently delete noncurrent versions of objects, and the other to delete expired object delete markers.

In a versioning-enabled bucket, a `DELETE` request that doesn't specify a version ID can remove only objects with a `NULL` version ID. If the object was uploaded when versioning was enabled, a `DELETE` request that doesn't specify a version ID creates a delete marker of that object.

> **Note**
>
> For S3 Object Lock-enabled buckets, a `DELETE` object request with a protected object version ID causes a 403 Access Denied error. A `DELETE` object request without a version ID adds a delete marker as the newest version of the object with a 200 OK response. Objects protected by Object Lock cannot be permanently deleted until their retention periods and

legal holds are removed. For more information, see the section called "How S3 Object Lock works".

## I'm experiencing performance degradation after enabling bucket versioning

Performance degradation can occur on versioning-enabled buckets if there are too many delete markers or versioned objects, and if best practices aren't followed.

**Too many delete markers**

After you enable versioning on a bucket, a `DELETE` request without a version ID made to an object creates a delete marker with a unique version ID. Lifecycle configurations with an **Expire current versions of objects** rule add a delete marker with a unique version ID to every object. Excessive delete markers can reduce performance in the bucket.

When versioning is suspended on a bucket, Amazon S3 marks the version ID as `NULL` on newly created objects. An expiration action in a versioning-suspended bucket causes Amazon S3 to create a delete marker with `NULL` as the version ID. In a versioning-suspended bucket, a `NULL` delete marker is created for any delete request. These `NULL` delete markers are also called expired object delete markers when all object versions are deleted and only a single delete marker remains. If too many `NULL` delete markers accumulate, performance degradation in the bucket occurs.

**Too many versioned objects**

If a versioning-enabled bucket contains objects with millions of versions, an increase in 503 Service Unavailable errors can occur. If you notice a significant increase in the number of HTTP 503 Service Unavailable responses received for `PUT` or `DELETE` object requests to a versioning-enabled bucket, you might have one or more objects in the bucket with millions of versions. When you have objects with millions of versions, Amazon S3 automatically throttles requests to the bucket. Throttling requests protects your bucket from an excessive amount of request traffic, which could potentially impede other requests made to the same bucket.

To determine which objects have millions of versions, use S3 Inventory. S3 Inventory generates a report that provides a flat file list of the objects in a bucket. For more information, see Amazon S3 Inventory.

To verify if there are high number of versioned objects in the bucket, use S3 Storage Lens metrics to view the **Current version object count**, **Noncurrent version object count**, and **Delete marker**

**object count**. For more information about Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

The Amazon S3 team encourages customers to investigate applications that repeatedly overwrite the same object, potentially creating millions of versions for that object, to determine whether the application is working as intended. For instance, an application overwriting the same object every minute for a week can create over ten thousand versions. We recommend storing less than one hundred thousand versions for each object. If you have a use case that requires millions of versions for one or more objects, contact the AWS Support team for assistance with determining a better solution.

**Best practices**

To prevent versioning-related performance degradation issues, we recommend that you employ the following best practices:

- Enable a lifecycle rule to expire the previous versions of objects. For example, you can create a lifecycle rule to expire noncurrent versions after 30 days of the object being noncurrent. You can also retain multiple noncurrent versions if you don't want to delete all of them. For more information, see Setting an S3 Lifecycle configuration.
- Enable a lifecycle rule to delete expired object delete markers that don't have associated data objects in the bucket. For more information, see Removing expired object delete markers.

For additional Amazon S3 performance-optimization best practices, see Best practices design patterns.

# Getting Amazon S3 request IDs for AWS Support

Whenever you contact AWS Support because you've encountered errors or unexpected behavior in Amazon S3, you must provide the request IDs associated with the failed action. AWS Support uses these request IDs to help resolve the problems that you're experiencing.

Request IDs come in pairs, are returned in every response that Amazon S3 processes (even the erroneous ones), and can be accessed through verbose logs. There are a number of common methods for getting your request IDs, including S3 access logs and AWS CloudTrail events or data events.

After you've recovered these logs, copy and retain those two values, because you'll need them when you contact AWS Support. For information about contacting AWS Support, see Contact AWS or the AWS Support Documentation.

## Using HTTP to obtain request IDs

You can obtain your request IDs, `x-amz-request-id` and `x-amz-id-2` by logging the bits of an HTTP request before it reaches the target application. There are a variety of third-party tools that can be used to recover verbose logs for HTTP requests. Choose one that you trust, and then run the tool to listen on the port that your Amazon S3 traffic travels on, as you send out another Amazon S3 HTTP request.

For HTTP requests, the pair of request IDs will look like the following:

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

**Note**

HTTPS requests are encrypted and hidden in most packet captures.

## Using a web browser to obtain request IDs

Most web browsers have developer tools that you can use to view request headers.

For web browser-based requests that return an error, the pair of requests IDs will look like the following examples.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

To obtain the request ID pair from successful requests, use your browser's developer tools to look at the HTTP response headers. For information about developer tools for specific browsers, see *Amazon S3 Troubleshooting - How to recover your S3 request IDs* in AWS re:Post.

# Using the AWS SDKs to obtain request IDs

The following sections include information for configuring logging by using an AWS SDK. Although you can enable verbose logging on every request and response, we don't recommend enabling logging in production systems, because large requests or responses can significantly slow down an application.

For AWS SDK requests, the pair of request IDs will look like the following examples.

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied  AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

## Using the SDK for Go to obtain request IDs

You can configure logging by using SDK for Go. For more information, see Response metadata in the *SDK for Go V2 Developer Guide*.

## Using the SDK for PHP to obtain request IDs

You can configure logging by using PHP. For more information, see How can I see what data is sent over the wire? in the *AWS SDK for PHP Developer Guide*.

## Using the SDK for Java to obtain request IDs

You can enable logging for specific requests or responses to catch and return only the relevant headers. To do this, import the `com.amazonaws.services.s3.S3ResponseMetadata` class. Afterward, you can store the request in a variable before performing the actual request. To get the logged request or response, call `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`.

### Example

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " + md.getRequestId());
```

Alternatively, you can use verbose logging of every Java request and response. For more information, see Verbose Wire Logging in the *AWS SDK for Java Developer Guide*.

## Using the AWS SDK for .NET to obtain request IDs

You can configure logging with the AWS SDK for .NET by using the built-in `System.Diagnostics` logging tool. For more information, see the Logging with the AWS SDK for .NET *AWS Developer Blog* post.

> **Note**
>
> By default, the returned log contains only error information. To get the request IDs, the config file must have `AWSLogMetrics` (and optionally, `AWSResponseLogging`) added.

## Using the SDK for Python (Boto3) to obtain request IDs

With the AWS SDK for Python (Boto3), you can log specific responses. You can use this feature to capture only the relevant headers. The following code shows how to log parts of the response to a file:

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

You can also catch exceptions and log relevant information when an exception is raised. For more information, see Discerning useful information from error responses in the *AWS SDK for Python (Boto) API Reference*.

Additionally, you can configure Boto3 to output verbose debugging logs by using the following code:

```
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

For more information, see set_stream_logger in the *AWS SDK for Python (Boto) API Reference*.

### Using the SDK for Ruby to obtain request IDs

You can get your request IDs using the SDK for Ruby Versions 1, 2, or 3.

- **Using the SDK for Ruby - Version 1**– You can enable HTTP wire logging globally with the following line of code.

```
s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

- **Using the SDK for Ruby - Version 2 or Version 3**– You can enable HTTP wire logging globally with the following line of code.

```
s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

For tips on getting wire information from an AWS client, see Debugging tip: Getting wire trace information from a client.

## Using the AWS CLI to obtain request IDs

To get your request IDs when using the AWS Command Line Interface (AWS CLI), add `--debug` to your command.

## Using Windows PowerShell to obtain request IDs

For information on recovering logs with Windows PowerShell, see the Response Logging in AWS Tools for Windows PowerShell .NET Development blog post.

## Using AWS CloudTrail data events to obtain request IDs

An Amazon S3 bucket that is configured with CloudTrail data events to log S3 object-level API operations provides detailed information about actions that are taken by a user, role, or an AWS service in Amazon S3. You can identify S3 request IDs by querying CloudTrail events with Athena.

## Using S3 server access logging to obtain request IDs

An Amazon S3 bucket configured for S3 server access logging provides detailed records for each request that is made to the bucket. You can identify S3 request IDs by querying the server access logs using Athena.

# Document history

- **Current API version:** 2006-03-01

The following table describes the important changes in each release of the *Amazon Simple Storage Service API Reference* and the *Amazon S3 User Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

| Change | Description | Date |
| --- | --- | --- |
| Amazon S3 Select is no longer available to new customers. | Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more | July 25, 2024 |
| Amazon S3 Inventory supports the s3:InventoryAccessibleOptionalFields condition key | Amazon S3 Inventory supports the s3:InventoryAccessibleOptionalFields condition key to control whether users can include optional metadata fields in their reports. For more information, see Control S3 Inventory report configuration creation. | February 20, 2024 |
| IPv6 support for S3 on Outposts | You can now access S3 on Outposts buckets using IPv6 via S3 on Outposts dual-stack endpoints. IPv6 support for S3 on Outposts allows you to manage your S3 on Outposts buckets and control plane resources over IPv6 networks. | January 16, 2024 |

| | | |
|---|---|---|
| New high-performance, single-zone Amazon S3 storage class – S3 Express One Zone | Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. For more information, see S3 Express One Zone. | November 28, 2023 |
| Mountpoint for Amazon S3 adds support for S3 Express One Zone | You can now mount S3 Express One Zone directory buckets with Mountpoint. | November 28, 2023 |
| Lambda invocation schema version | Amazon S3 Batch Operations introduces a new Lambda invocation schema version for use with Batch Operations jobs that act on directory buckets. For more information, see Using Lambda and Amazon S3 batch operations with directory buckets. | November 28, 2023 |
| Import action for directory buckets | Amazon S3 introduces the import action. Import is a streamlined method for creating Amazon S3 Batch Operations jobs to copy objects from general purpose buckets to directory buckets. For more information, see Importing objects into a directory bucket. | November 28, 2023 |

| | | |
|---|---|---|
| Manage S3 access with S3 Access Grants | Amazon S3 Access Grants enables you to manage data permissions at scale for AWS Identity and Access Management (IAM) principals in addition to directory identities from corporate directories such as Azure AD. You can now enforce least-privilege S3 permissions and easily scale those permissions based on your business needs. For more information, see Managing access with S3 Access Grants. | November 26, 2023 |
| Mountpoint for Amazon S3 adds caching feature | With Mountpoint, you can now configure caching for repeatedly accessed data. | November 22, 2023 |
| Enhanced Amazon S3 Batch Operations manifest generation | You can now direct Amazon S3 Batch Operations to generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for batch replication jobs that you create in the Amazon S3 console, or for any job type that you create by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information, see Creating an Amazon S3 Batch Operations job. | November 22, 2023 |

| | | |
|---|---|---|
| Existing Amazon S3 buckets can now add Object Lock configurations | You can now enable Object Lock on existing Amazon S3 bucket. You may set legal holds and retention periods for new or existing buckets. For more information, see Using Object Lock. | November 20, 2023 |
| S3 Storage Lens request metrics for prefixes | S3 Storage Lens introduces request metrics for prefixes within an Amazon S3 bucket. For more information, see Metrics categories. | November 17, 2023 |
| Amazon S3 Storage Lens groups | S3 Storage Lens introduces Storage Lens groups, a custom defined filter for objects based on object metadata. For more information, see Working with Amazon S3 Storage Lens groups. | November 15, 2023 |
| New IAM policy | S3 on Outposts introduces `AWSServiceRoleForS3OnOutposts`, a service-linked role to help manage network resources for you. For more information, see Using service-linked roles for S3 on Outposts. | October 3, 2023 |

| | | |
|---|---|---|
| Amazon S3 provides the `Last-Modified` time for delete markers | Amazon S3 provides the `Last-Modified` time of delete markers in the response headers of S3 `Head` and `Get` API operations. For more information, see Working with delete markers. | September 27, 2023 |
| Amazon S3 update to AWS managed policy | Amazon S3 added `s3:Describe*` permissions to `AmazonS3ReadOnlyAccess`. For more information, see AWS managed policies for Amazon S3. | August 11, 2023 |
| Improved start times for Standard restore requests made through S3 Batch Operations | Standard retrievals for restore requests that are made through S3 Batch Operations now can start within minutes. For more information, see Archive Retrieval Options. | August 9, 2023 |
| Added Mountpoint, a high-throughput client for mounting an Amazon S3 bucket as a local file system. | With Mountpoint, your applications can access objects stored in Amazon S3 through file operations, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface. | August 9, 2023 |

| | | |
|---|---|---|
| Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS) | Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) applies two layers of encryption to objects when they are uploaded to Amazon S3. For more information, see Using dual-layer server-side encryption with AWS KMS keys. | June 13, 2023 |
| Amazon S3 enables S3 Block Public Access and disables S3 access control lists (ACLs) for all new buckets. | Amazon S3 now automatically enables S3 Block Public Access and disables S3 access control lists (ACLs) for all new S3 buckets in all AWS Regions. For more information, see Blocking public access to your Amazon S3 storage and Controlling ownership of objects and disabling ACLs for your bucket. | April 27, 2023 |
| S3 Replication Operations Failed metric | Amazon S3 adds new Amazon CloudWatch metric to monitor S3 Replication failures. For more information, see Monitoring progress with replication metrics. | April 5, 2023 |
| Private DNS | AWS PrivateLink for Amazon S3 now supports Private DNS. For more information, see Private DNS. | March 14, 2023 |

| | | |
|---|---|---|
| Cross-account access points support in the Amazon S3 console | Amazon S3 now supports creating cross-account access points with the Amazon S3 console. For more information, see Creating access points. | March 14, 2023 |
| Amazon S3 on Outposts supports S3 Replication on Outposts | With local S3 Replication, you can automatically replicate objects to a single Outposts destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Outposts or within the same Outposts as the source bucket. For more information, see Replicating objects for S3 on Outposts. | March 14, 2023 |
| Amazon S3 Object Lambda Access Point alias | When you create an Object Lambda Access Point, Amazon S3 automatically generates a unique alias for your Object Lambda Access Point. You can use this alias instead of an Amazon S3 bucket name or the Object Lambda Access Point Amazon Resource Name (ARN) in a request for access point data plane operations. For more information, see How to use a bucket-style alias for your Object Lambda Access Point. | March 14, 2023 |

| | | |
|---|---|---|
| Amazon S3 Multi-Region Access Points cross-account support | Amazon S3 now supports creating cross-account Multi-Region Access Points with the Amazon S3 console. For more information, see Creating Multi-Region Access Points. | March 14, 2023 |
| Cross-account access points | Amazon S3 supports creating cross-account access points. You can create a cross-account access point by using the AWS Command Line Interface (AWS CLI) or the REST API `CreateAccessPoint` operation. For more information, see Creating access points. | November 30, 2022 |
| Amazon S3 supports failover controls for Amazon S3 Multi-Region Access Points | Amazon S3 introduces failover control for Multi-Region Access Points. These controls let you shift S3 data access request traffic routed through an Amazon S3 Multi-Region Access Point to an alternate AWS Region within minutes to test and build highly available applications. For more information, see Amazon S3 Multi-Region Access Point failover controls. | November 28, 2022 |

| | | |
|---|---|---|
| Amazon S3 Storage Lens increases organization-wide visibility with 34 new metrics | S3 Storage Lens introduces 34 additional metrics to uncover deeper cost-optimization opportunities, identify data-protection best practices, and improve the performan ce of application workflows. For more information, see S3 Storage Lens metrics. | November 17, 2022 |
| Amazon S3 supports higher restore request rates for S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive | Amazon S3 supports restore requests at a rate of up to 1,000 transactions per second, per AWS account for the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes. | November 15, 2022 |
| Amazon S3 on Outposts supports additional S3 Lifecycle actions and filters | S3 on Outposts supports additional S3 Lifecycle rules to optimize capacity mana gement. You can expire objects as they age or are replaced with newer version s. You can create a lifecycle rule for a whole bucket or a subset of objects in a bucket by filtering with prefixes, object tags, or object size. For more information, see Creating and managing a lifecycle configuration. | November 2, 2022 |

| | | |
|---|---|---|
| S3 Replication support for SSE-C objects | You can replicate objects that are created using server-side encryption with customer -provided keys. For more information about replicating encrypted objects, see Replicating objects created with server-side encryption (SSE-C, SSE-S3, SSE-KMS). | October 24, 2022 |
| Amazon S3 on Outposts supports access point aliases | With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Every time you create an access point for a bucket, S3 on Outposts automatically generates an access point alias. You can use this access point alias instead of an access point ARN for any data plane operation. For more information, see Using a bucket-style alias for your S3 on Outposts bucket access point. | October 21, 2022 |
| S3 Object Lambda supports the `HeadObject`, `ListObjects`, and `ListObjectsV2` operations | You can use custom code to modify the data returned by standard S3 `GET`, `LIST`, or `HEAD` requests to filter rows, dynamically resize images, redact confidential data, and more. For more information, see Transforming objects with S3 Object Lambda. | October 4, 2022 |

| | | |
|---|---|---|
| Amazon S3 on Outposts supports S3 Versioning | When enabled, S3 Versioning saves multiple distinct copies of an object in the same bucket. You can use S3 Versioning to preserve, retrieve, and restore every version of every object stored in your Outposts buckets. S3 Versioning helps you recover from unintended user actions and application failures. For more information, see Managing S3 Versioning for your S3 on Outposts bucket. | September 21, 2022 |
| AWS Backup for Amazon S3 | AWS Backup is a fully managed, policy-based service that you can use to define a central backup policy to protect your Amazon S3 data. For more information, see Using AWS Backup for Amazon S3. | February 18, 2022 |

| | | |
|---|---|---|
| Use S3 Batch Replication to replicate existing objects | With S3 Batch Replication, you can replicate objects that existed before a replication configuration was in place. Replicating existing objects is done through the use of a Batch Operations job. S3 Batch Replication differs from live replication, which continuously and automatic ally copies new objects across Amazon S3 buckets. For more information, see Replicating existing objects with S3 Batch Replication. | February 8, 2022 |
| Rename of S3 Glacier Flexible Retrieval | The Glacier storage class has been renamed to S3 Glacier Flexible Retrieval. This change does not impact the API. | November 30, 2021 |

| | | |
|---|---|---|
| New S3 Object Ownership setting to disable ACLs | You can apply the bucket owner enforced setting for Object Ownership to disable ACLs for your bucket and the objects in it and take ownership of every object in your bucket. The bucket owner enforced setting simplifies access management for data stored in Amazon S3. For more information, see Controlling ownership of objects and disabling ACLs for your bucket. | November 30, 2021 |
| New S3 Intelligent-Tiering storage class | S3 Intelligent-Tiering Archive Instant Access is an additional storage class under S3 Intelligent-Tiering. For more information, see How S3 Intelligent-Tiering works. | November 30, 2021 |
| New S3 Glacier Instant Retrieval storage class | You can now place objects in the S3 Glacier Instant Retrieval storage class. For more information about this storage class, see Using Amazon S3 storage classes. | November 30, 2021 |

| | | |
|---|---|---|
| AWS Backup for Amazon S3 Preview | AWS Backup is a fully managed, policy-based service that you can use to define a central backup policy to protect your Amazon S3 data. For more information see, Using AWS Backup for Amazon S3. | November 30, 2021 |
| AWS Identity and Access Management Access Analyzer for Amazon S3 | IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. To learn more about validating policies using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. | November 30, 2021 |
| New event types | New event types added to Amazon S3 Event Notifications, see Amazon S3 Event Notifications. | November 29, 2021 |
| Enable Amazon EventBridge on buckets | You can enable EventBridge on Amazon S3 buckets to send events to Amazon EventBridge, see Using EventBridge. | November 29, 2021 |
| New S3 Lifecycle filters | You can create lifecycle rules based on object size or specify how many noncurrent object versions to keep. For more information, see Examples of S3 Lifecycle configuration. | November 23, 2021 |

| | | |
|---|---|---|
| Publish Amazon S3 Storage Lens metrics to Amazon CloudWatch | You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, like alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, the CloudWatch APIs enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information, see the Monitor S3 Storage Lens metrics in CloudWatch. | November 22, 2021 |
| Multi-Region Access Points | You can use Multi-Region Access Points to create a global endpoint that applications can use to fulfill requests from Amazon S3 buckets located in multiple AWS Regions. You can use this Multi-Region Access Point to route data to a bucket with the lowest latency. For more information about Multi-Region Access Points and how to use them, see Multi-Region Access Point in Amazon S3. | September 2, 2021 |

| | | |
|---|---|---|
| Amazon S3 on Outposts adds direct local access for applications | Run your applications outside the AWS Outposts virtual private cloud (VPC) and access your S3 on Outposts data. You can also access S3 on Outposts objects directly from your on-premises network. For more information about configuring S3 on Outposts endpoints using customer-owned IP (CoIP) addresses and accessing your objects by creating a local gateway from your on-premises network, see Accessing Amazon S3 on Outposts using VPC-only access points. | July 29, 2021 |
| Amazon S3 access point alias | When you create an access point, Amazon S3 automatically generates an alias that you can use instead of a bucket name for data access. You can use this access point alias instead of an Amazon Resource Name (ARN) for any access point data plane operation. For more information, see Using a bucket-style alias for your access point. | July 26, 2021 |

| | | |
|---|---|---|
| Amazon S3 Inventory and S3 Batch Operations support S3 Bucket Key status | Amazon S3 Inventory and Batch Operations support identifying and copying existing objects with S3 Bucket Keys. S3 Bucket Keys accelerate the reduction of server-side encryption costs for existing objects. For more information, see Amazon S3 Inventory and Batch Operations Copy object. | June 3, 2021 |
| Amazon S3 Storage Lens metrics account snapshot | The S3 Storage Lens account snapshot displays your total storage, object count, and average object size on the S3 console home (**Buckets**) page by summarizing metrics from your default dashboard. For more information, see S3 Storage Lens metrics account snapshot. | May 5, 2021 |
| Increased Amazon S3 on Outposts endpoint support | S3 on Outposts now supports up to 100 endpoints per Outpost. For more information, see S3 on Outposts network restrictions. | April 29, 2021 |

| | | |
|---|---|---|
| Amazon S3 on Outposts event notifications in Amazon CloudWatch Events | You can use CloudWatch Events to create a rule to capture any S3 on Outposts API event and get notified through all supported CloudWatch targets. For more information, see Receiving S3 on Outposts event notifications using CloudWatch Events. | April 19, 2021 |
| S3 Object Lambda | With S3 Object Lambda, you can add your own code to Amazon S3 GET requests to modify and process data as it is returned to an application. You can use custom code to modify the data returned by standard S3 GET requests to filter rows, dynamically resize images, redact confidential data, and more. For more information, see Transforming objects. | March 18, 2021 |

| | | |
|---|---|---|
| AWS PrivateLink | With AWS PrivateLink for Amazon S3, you can connect directly to S3 by using an interface endpoint in your virtual private cloud (VPC) instead of connecting over the internet. Interface endpoints are directly accessible from applications that are on premises or in a different AWS Region. For more information, see AWS PrivateLink for Amazon S3. | February 2, 2021 |
| Managing Amazon S3 on Outposts capacity with AWS CloudTrail | S3 on Outposts management events are available through CloudTrail logs. For more information, see Managing S3 on Outposts capacity with CloudTrail. | December 21, 2020 |
| Strong consistency | Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your S3 bucket in all AWS Regions. In addition, read operations on Amazon S3 Select, Amazon S3 access control lists, Amazon S3 Object Tags, and object metadata (for example, HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model. | December 1, 2020 |

| | | |
|---|---|---|
| Amazon S3 replica modification sync | Amazon S3 replica modification sync keeps object metadata, such as tags, ACLs, and Object Lock settings, in sync between source objects and replicas. When this feature is enabled, Amazon S3 replicates metadata changes made to either the source object or the replica copies. For more information, see Replicating metadata changes with replica modification sync. | December 1, 2020 |
| Amazon S3 Bucket Keys | Amazon S3 Bucket Keys reduce the cost of Amazon S3 server-side encryption with AWS Key Management Service (SSE-KMS). This new bucket-level key for server-side encryption can reduce AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS by using S3 Bucket Keys. | December 1, 2020 |

| | | |
|---|---|---|
| Amazon S3 Storage Lens | S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics export in CSV or Parquet format to an S3 bucket. For more information, see Assessing your storage activity and usage with S3 Storage Lens. | November 18, 2020 |
| Tracing S3 requests using AWS X-Ray | Amazon S3 integrates with X-Ray to propagate the trace context and give you one request chain with upstream and downstream nodes. For more information, see Tracing requests using X-Ray. | November 16, 2020 |

| | | |
|---|---|---|
| S3 Replication metrics | S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. For more information, see Replication metrics and Amazon S3 event notifications. | November 9, 2020 |
| S3 Intelligent-Tiering Archive Access and Deep Archive Access | S3 Intelligent-Tiering Archive Access and Deep Archive Access are additional storage tiers under S3 Intelligent-Tiering. For more information, see Storage class for automatically optimizing frequently and infrequently accessed objects . | November 9, 2020 |
| Delete marker replication | With delete marker replication, you can ensure that delete markers are copied to your destination buckets for your replication rules. For more information, see Using delete marker replication. | November 9, 2020 |
| S3 Object Ownership | Object Ownership is an S3 bucket setting that you can use to control ownership of new objects that are uploaded to your buckets. For more information, see Using S3 Object Ownership. | October 2, 2020 |

| | | |
|---|---|---|
| Amazon S3 on Outposts | With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and easily store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use S3 on Outposts through the AWS Management Console, AWS CLI, AWS SDKs, or REST API. For more information, see Using Amazon S3 on Outposts. | September 30, 2020 |
| Bucket owner condition | You can use the Amazon S3 bucket owner condition to ensure that the buckets you use in your S3 operations belong to the AWS accounts that you expect. For more information, see Bucket owner condition. | September 11, 2020 |
| S3 Batch Operations support for Object Lock Retention | You can now use Batch Operations with S3 Object Lock to apply retention settings to many Amazon S3 objects at once. For more information, see Setting S3 Object Lock Retention dates with S3 Batch Operations. | May 4, 2020 |

| | | |
|---|---|---|
| S3 Batch Operations support for Object Lock Legal Hold | You can now use Batch Operations with S3 Object Lock to add a legal hold to many Amazon S3 objects at once. For more information, see Using S3 Batch Operations for setting S3 Object Lock Legal Hold. | May 4, 2020 |
| Job tags for S3 Batch Operations | You can add tags to your S3 Batch Operations jobs to control and label those jobs. For more information, see Tags for S3 Batch Operations jobs. | March 16, 2020 |
| Amazon S3 access points | Amazon S3 access points simplify managing data access at scale for shared datasets in S3. Access points are named network endpoints that are attached to buckets that you can use to perform S3 object operations. For more information, see Managing data access with Amazon S3 access points. | December 2, 2019 |

| | | |
|---|---|---|
| Access Analyzer for Amazon S3 | Access Analyzer for Amazon S3 alerts you to S3 buckets that are configured to allow access to anyone on the internet or other AWS accounts, including accounts outside of your organization. For more information, see Using Access Analyzer for Amazon S3. | December 2, 2019 |
| S3 Replication Time Control (S3 RTC) | S3 Replication Time Control (S3 RTC) replicates most objects that you upload to Amazon S3 in seconds, and 99.99 percent of those objects within 15 minutes. For more information, see Replicating objects using S3 Replication Time Control (S3 RTC). | November 20, 2019 |
| Same-Region Replication | You can use Same-Region Replication (SRR) to copy objects across Amazon S3 buckets in the same AWS Region. For information about both Cross-Region Replication (CRR) and Same-Region Replication, see Replication. | September 18, 2019 |
| Cross-Region Replication support for S3 Object Lock | Cross-Region Replication now supports Object Lock. For more information, see What does Amazon S3 Replicate?. | May 28, 2019 |

| | | |
|---|---|---|
| S3 Batch Operations | By using S3 Batch Operations, you can perform large-scale Batch Operations on Amazon S3 objects. S3 Batch Operations can run a single operation on lists of objects that you specify. A single job can perform the specified operation on billions of objects containing exabytes of data. For more information, see Performing S3 Batch Operations. | April 30, 2019 |
| Asia Pacific (Hong Kong) Region | Amazon S3 is now available in the Asia Pacific (Hong Kong) Region. For more information about Amazon S3 Regions and endpoints, see Regions and endpoints in the *AWS General Reference*. | April 24, 2019 |
| Added a new field to the server access logs | Amazon S3 added the following new field to the server access logs: Transport Layer Security (TLS) version. For more information, see Server access log format. | March 28, 2019 |
| New archive storage class | Amazon S3 now offers a new archive storage class, S3 Glacier Deep Archive (`DEEP_ARCHIVE`), for storing rarely accessed objects. For more information, see Storage Classes. | March 27, 2019 |

| | | |
|---|---|---|
| Added new fields to the server access logs | Amazon S3 added the following new fields to the server access logs: Host Id, Signature Version, Cipher Suite, Authentication Type, and Host Header. For more information, see Server access log format. | March 5, 2019 |
| Support for Parquet-formatted Amazon S3 Inventory files | Amazon S3 now supports the Apache Parquet (Parquet) format in addition to the Apache optimized row columnar (ORC) and comma-separated values (CSV) file formats for inventory output files. For more information, see Inventory. | December 4, 2018 |
| S3 Object Lock | Amazon S3 now offers Object Lock functionality that provides Write Once Read Many (WORM) protections for Amazon S3 objects. For more information, see Locking Objects. | November 26, 2018 |
| Restore speed upgrade | Using Amazon S3 restore speed upgrade, you can change the speed of a restoration from the S3 Glacier Flexible Retrieval storage class to a faster speed while the restoration is in progress. For more information, see Restoring Archived Objects. | November 26, 2018 |

| | | |
|---|---|---|
| Restore Event Notifications | Amazon S3 Event Notifications now support initiation and completion events when restoring objects from the S3 Glacier Flexible Retrieval storage class. For more information, see Event Notifications. | November 26, 2018 |

| | | |
|---|---|---|
| PUT directly to the S3 Glacier Flexible Retrieval storage class | The Amazon S3 PUT operation now supports specifying S3 Glacier Flexible Retrieval as the storage class when creating objects. Previously, you had to transition objects to the S3 Glacier Flexible Retrieval storage class from another Amazon S3 storage class. Also, when using S3 Cross-Region Replication (CRR), you can now specify S3 Glacier Flexible Retrieval as the storage class for replicated objects. For more information about the S3 Glacier Flexible Retrieval storage class, see Storage Classes. For more information about specifying the storage class for replicated objects, Replication Configuration Overview. For more informati on about the direct PUT to S3 Glacier Flexible Retrieval REST API changes, see Document History: PUT directly to S3 Glacier Flexible Retrieval . | November 26, 2018 |

| | | |
|---|---|---|
| New storage class | Amazon S3 now offers a new storage class named S3 Intelligent-Tiering ( `INTELLIGENT_TIERING` ) that is designed for long-lived data with changing or unknown access patterns. For more information, see Storage Classes. | November 26, 2018 |
| Amazon S3 Block Public Access | Amazon S3 now includes the ability to block public access to buckets and objects on a per-bucket or account-wide basis. For more information, see Using Amazon S3 Block Public Access. | November 15, 2018 |
| Filtering enhancements in Cross-Region Replication (CRR) rules | In a CRR rule configuration, you can specify an object filter to choose a subset of objects to apply the rule to. Previously, you could filter only on an object key prefix. In this release, you can filter on an object key prefix, one or more object tags, or both. For more information, see CRR Setup: Replication Configuration Overview. | September 19, 2018 |

| | | |
|---|---|---|
| New Amazon S3 Select features | Amazon S3 Select now supports Apache Parquet input, queries on nested JSON objects, and two new Amazon CloudWatch monitoring metrics (`SelectScannedBytes` and `SelectReturnedBytes`). | September 5, 2018 |
| Updates now available over RSS | You can now subscribe to an RSS feed to receive notifications about updates to the *Amazon S3 User Guide*. | June 19, 2018 |

# Earlier updates

The following table describes the important changes in each release of the *Amazon S3 User Guide* before June 19, 2018.

| Change | Description | Date |
|---|---|---|
| Code examples update | Code examples updated:<br>• C#—Updated all of the examples to use the task-based asynchronous pattern. For more information, see Amazon Web Services Asynchronous APIs for .NET in the *AWS SDK for .NET Developer Guide*. Code examples are now compliant with version 3 of the AWS SDK for .NET.<br>• Java—Updated all of the examples to use the client builder model. For more information about the client builder model, see Creating Service Clients.<br>• | April 30, 2018 |

| Change | Description | Date |
| --- | --- | --- |
| | PHP—Updated all of the examples to use the AWS SDK for PHP 3.0. For more information about the AWS SDK for PHP 3.0, see AWS SDK for PHP.<br>• Ruby—Updated example code so that the examples work with the AWS SDK for Ruby version 3. | |
| Amazon S3 now reports S3 Glacier Flexible Retrieval and ONEZONE_IA storage classes to Amazon CloudWatch Logs storage metrics | In addition to reporting actual bytes, these storage metrics include per-object overhead bytes for applicable storage classes (ONEZONE_IA , STANDARD_IA , and S3 Glacier Flexible Retrieval ):<br>• For ONEZONE_IA and STANDARD_IA storage class objects, Amazon S3 reports objects smaller than 128 KB as 128 KB. For more information, see Using Amazon S3 storage classes.<br>• For S3 Glacier Flexible Retrieval storage class objects, the storage metrics report the following overheads:<br>  • A 32 KB per-object overhead, charged at S3 Glacier Flexible Retrieval storage class pricing<br>  • An 8 KB per-object overhead, charged at STANDARD storage class pricing<br>  For more information, see Transitioning objects using Amazon S3 Lifecycle.<br>For more information about storage metrics, see Monitoring metrics with Amazon CloudWatch. | April 30, 2018 |

| Change | Description | Date |
| --- | --- | --- |
| New storage class | Amazon S3 now offers a new storage class, `STANDARD_IA` (IA, for infrequent access) for storing objects. This storage class is optimized for long-lived and less frequently accessed data. For more information, see Using Amazon S3 storage classes. | April 4, 2018 |
| Amazon S3 Select | Amazon S3 now supports retrieving object content based on an SQL expression. For more information, see Filtering and retrieving data using Amazon S3 Select. | April 4, 2018 |
| Asia Pacific (Osaka-Local) Region | Amazon S3 is now available in the Asia Pacific (Osaka-Local) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.<br><br>**Important**<br>You can use the Asia Pacific (Osaka-Local) Region only in conjunction with the Asia Pacific (Tokyo) Region. To request access to Asia Pacific (Osaka-Local) Region, contact your sales representative. | February 12, 2018 |
| Amazon S3 Inventory creation timestamp | Amazon S3 Inventory now includes a timestamp of the date and start time of the creation of the Amazon S3 Inventory report. You can use the timestamp to determine changes in your Amazon S3 storage from the start time of when the inventory report was generated. | January 16, 2018 |
| Europe (Paris) Region | Amazon S3 is now available in the Europe (Paris) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | December 18, 2017 |

| Change | Description | Date |
| --- | --- | --- |
| China (Ningxia) Region | Amazon S3 is now available in the China (Ningxia) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | November 29, 2017 |
| Support for ORC-formatted Amazon S3 Inventory files | Amazon S3 now supports the Apache optimized row columnar (ORC) format in addition to comma-separated values (CSV) file format for inventory output files. Also, you can now query Amazon S3 inventory using standard SQL by using Amazon Athena, Amazon Redshift Spectrum, and other tools such as Presto, Apache Hive, and Apache Spark. For more information, see Amazon S3 Inventory. | November 17, 2017 |
| Default encryption for S3 buckets | Amazon S3 default encryption provides a way to set the default encryption behavior for an S3 bucket. You can set default encryption on a bucket so that all objects are encrypted when they are stored in the bucket. The objects are encrypted using server-side encryption with either Amazon S3 managed keys (SSE-S3) or AWS managed keys (SSE-KMS). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets. | November 06, 2017 |
| Encryption status in Amazon S3 Inventory | Amazon S3 now supports including encryption status in Amazon S3 Inventory so you can see how your objects are encrypted at rest for compliance auditing or other purposes. You can also configure to encrypt Amazon S3 Inventory with server-side encryption (SSE) or SSE-KMS so that all inventory files are encrypted accordingly. For more information, see Amazon S3 Inventory. | November 06, 2017 |

| Change | Description | Date |
| --- | --- | --- |
| Cross-Region Replication (CRR) enhancements | Cross-Region Replication now supports the following:<br>• In a cross-account scenario, you can add a CRR configuration to change replica ownership to the AWS account that owns the destination bucket. For more information, see Changing the replica owner.<br>• By default, Amazon S3 does not replicate objects in your source bucket that are created using server-side encryption using keys stored in AWS KMS In your CRR configuration, you can now direct Amazon S3 to replicate these objects. For more information, see Replicating encrypted objects (SSE-C, SSE-S3, SSE-KMS, DSSE-KMS). | November 06, 2017 |
| Europe (London) Region | Amazon S3 is now available in the Europe (London) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | December 13, 2016 |
| Canada (Central) Region | Amazon S3 is now available in the Canada (Central) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | December 8, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| Object tagging | Amazon S3 now supports object tagging. Object tagging enables you to categorize storage. Object key name prefixes also enable you to categorize storage, object tagging adds another dimension to it.<br><br>There are added benefits tagging offers. These include:<br><br>• Object tags enable fine-grained access control of permissions (for example, you could grant an IAM user permissions to read-only objects with specific tags).<br>• Fine-grained control in specifying lifecycle configuration. You can specify tags to select a subset of objects to which lifecycle rule applies.<br>• If you have Cross-Region Replication (CRR) configured, Amazon S3 can replicate the tags. You must grant necessary permission to the IAM role created for Amazon S3 to assume to replicate objects on your behalf.<br>• You can also customize CloudWatch metrics and CloudTrail events to display information by specific tag filters.<br><br>For more information, see Categorizing your storage using tags. | November 29, 2016 |
| Amazon S3 Lifecycle now supports tag-based filters | Amazon S3 now supports tag-based filtering in lifecycle configuration. You can now specify lifecycle rules in which you can specify a key prefix, one or more object tags, or a combination of both to select a subset of objects to which the lifecycle rule applies. For more information, see Managing your storage lifecycle. | November 29, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| CloudWatch request metrics for buckets | Amazon S3 now supports CloudWatch metrics for requests made on buckets. When you enable these metrics for a bucket, the metrics report at 1-minut e intervals. You can also configure which objects in a bucket will report these request metrics. For more information, see Monitoring metrics with Amazon CloudWatch. | November 29, 2016 |
| Amazon S3 Inventory | Amazon S3 now supports storage inventory. Amazon S3 Inventory provides a flat-file output of your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or a shared prefix (that is, objects that have names that begin with a common string).<br><br>For more information, see Amazon S3 Inventory. | November 29, 2016 |
| Amazon S3 Analytics – Storage Class Analysis | The new Amazon S3 analytics – storage class analysis feature observes data access patterns to help you determine when to transition less frequently accessed `STANDARD` storage to the `STANDARD_IA` (IA, for infrequent access) storage class. After storage class analysis observes the infrequent access patterns of a filtered set of data over a period of time, you can use the analysis results to help you improve your lifecycle configurations. This feature also includes a detailed daily analysis of your storage usage at the specified bucket, pr efix, or tag level that you can export to an S3 bucket. | November 29, 2016 |
| New Expedited and Bulk data retrieval s when restoring archived objects from S3 Glacier | Amazon S3 now supports Expedited and Bulk data retrievals in addition to Standard retrievals when restoring objects archived to S3 Glacier. For more information, see Restoring an archived object. | November 21, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| CloudTrail object logging | CloudTrail supports logging Amazon S3 object level API operations such as `GetObject`, `PutObject`, and `DeleteObject`. You can configure your event selectors to log object level API operations. For more information, see Logging Amazon S3 API calls using AWS CloudTrail. | November 21, 2016 |
| US East (Ohio) Region | Amazon S3 is now available in the US East (Ohio) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | October 17, 2016 |
| IPv6 support for Amazon S3 Transfer Acceleration | Amazon S3 now supports Internet Protocol version 6 (IPv6) for Amazon S3 Transfer Acceleration. You can connect to Amazon S3 over IPv6 by using the new dual-stack for Transfer Acceleration endpoint. For more information, see Getting started with Amazon S3 Transfer Acceleration. | October 6, 2016 |
| IPv6 support | Amazon S3 now supports Internet Protocol version 6 (IPv6). You can access Amazon S3 over IPv6 by using dual-stack endpoints. For more information, see Making requests to Amazon S3 over IPv6. | August 11, 2016 |
| Asia Pacific (Mumbai) Region | Amazon S3 is now available in the Asia Pacific (Mumbai) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | June 27, 2016 |
| Amazon S3 Transfer Acceleration | Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of Amazon CloudFront globally distributed edge locations.<br><br>For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration. | April 19, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| Lifecycle support to remove expired object delete markers | Lifecycle configuration `Expiration` action now allows you to direct Amazon S3 to remove expired object delete markers in a/ versioned bucket. For more information, see Elements to describe lifecycle actions. | March 16, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| Bucket lifecycle configuration now supports action to stop incomplete multipart uploads | Bucket lifecycle configuration now supports the `AbortIncompleteMultipartUpload` action that you can use to direct Amazon S3 to stop multipart uploads that don't complete within a specified number of days after being initiated. When a multipart upload becomes eligible for a stop operation, Amazon S3 deletes any uploaded parts and stops the multipart upload.<br><br>For conceptual information, see the following topics in the *Amazon S3 User Guide*:<br><br>• Aborting a multipart upload<br>• Elements to describe lifecycle actions<br><br>The following API operations have been updated to support the new action:<br><br>• PUT Bucket lifecycle – The XML configuration now allows you to specify the `AbortIncompleteMultipartUpload` action in a lifecycle configuration rule.<br>• List Parts and Initiate Multipart Upload – Both of these API operations now return two additional response headers (`x-amz-abort-date` , and `x-amz-abort-rule-id` ) if the bucket has a lifecycle rule that specifies the `AbortIncompleteMultipartUpload` action. These headers in the response indicate when the initiated multipart upload becomes eligible for a stop operation and which lifecycle rule is applicable. | March 16, 2016 |

| Change | Description | Date |
| --- | --- | --- |
| Asia Pacific (Seoul) Region | Amazon S3 is now available in the Asia Pacific (Seoul) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. | January 6, 2016 |
| New condition key and a multipart upload change | IAM policies now support an Amazon S3 `s3:x-amz-storage-class` condition key. For more information, see Bucket policy examples using condition keys.<br><br>You no longer need to be the initiator of a multipart upload to upload parts and complete the upload. For more information, see Multipart upload API and permissions. | December 14, 2015 |
| Renamed the US Standard Region | Changed the Region name string from "US Standard" to "US East (N. Virginia)." This is only a Region name update, there is no change in the functionality. | December 11, 2015 |
| New storage class | Amazon S3 now offers a new storage class, STANDARD_IA (IA, for infrequent access) for storing objects. This storage class is optimized for long-lived and less frequently accessed data. For more information, see Using Amazon S3 storage classes.<br><br>Lifecycle configuration feature updates now allow you to transition objects to the STANDARD_IA storage class. For more information, see Managing your storage lifecycle.<br><br>Previously, the Cross-Region Replication feature used the storage class of the source object for object replicas. Now, when you configure Cross-Region Replication, you can specify a storage class for the object replica created in the destination bucket. For more information, see Replicating objects overview. | September 16, 2015 |

| Change | Description | Date |
| --- | --- | --- |
| AWS CloudTrail integration | New AWS CloudTrail integration allows you to record Amazon S3 API activity in your S3 bucket. You can use CloudTrail to track S3 bucket creations or deletions, access control modifications, or lifecycle configuration changes. For more information, see Logging Amazon S3 API calls using AWS CloudTrail. | September 1, 2015 |
| Bucket limit increase | Amazon S3 now supports bucket limit increases. By default, customers can create up to 100 buckets in their AWS account. Customers who need additional buckets can increase that limit by submitting a service limit increase. For information about how to increase your bucket limit, go to AWS service quotas in the *AWS General Reference*. For more information, see Using the AWS SDKs and Bucket quotas, restrictions, and limitations. | August 4, 2015 |
| Consistency model update | Amazon S3 now supports read-after-write consistency for new objects added to Amazon S3 in the US East (N. Virginia) Region. Prior to this update, all Regions except US East (N. Virginia) Region supported read-after-write consistency for new objects uploaded to Amazon S3. With this enhancement, Amazon S3 now supports read-after-write consistency in all Regions for new objects added to Amazon S3. Read-after-write consistency allows you to retrieve objects immediately after creation in Amazon S3. For more information, see Regions. | August 4, 2015 |
| Event notifications | Amazon S3 Event Notifications have been updated to add notifications when objects are deleted and to add filtering on object names with prefix and suffix matching. For more information, see Amazon S3 Event Notifications. | July 28, 2015 |

| Change | Description | Date |
| --- | --- | --- |
| Amazon CloudWatch integration | New Amazon CloudWatch integration allows you to monitor and set alarms on your Amazon S3 usage through CloudWatch metrics for Amazon S3. Supported metrics include total bytes for Standard storage, total bytes for Reduced-Redundancy Storage, and total number of objects for a given S3 bucket. For more information, see Monitoring metrics with Amazon CloudWatch. | July 28, 2015 |
| Support for deleting and emptying non-empty buckets | Amazon S3 now supports deleting and emptying non-empty buckets. For more information, see Emptying a bucket. | July 16, 2015 |
| Bucket policies for Amazon VPC endpoints | Amazon S3 has added support for bucket policies for virtual private cloud (VPC) (VPC) endpoints. You can use S3 bucket policies to control access to buckets from specific VPC endpoints, or specific VPCs. VPC endpoints are easy to configure, are highly reliable, and provide a secure connection to Amazon S3 without requiring a gateway or a NAT instance. For more information, see Controlling access from VPC endpoints with bucket policies. | April 29, 2015 |
| Event notifications | Amazon S3 Event Notifications have been updated to support the switch to resource-based permissions for AWS Lambda functions. For more information, see Amazon S3 Event Notifications. | April 9, 2015 |
| Cross-Region Replication | Amazon S3 now supports Cross-Region Replication. Cross-Region Replication is the automatic, asynchronous copying of objects across buckets in different AWS Regions. For more information, see Replicating objects overview. | March 24, 2015 |

| Change | Description | Date |
| --- | --- | --- |
| Event notifications | Amazon S3 now supports new event types and destinations in a bucket notification configuration. Prior to this release, Amazon S3 supported only the *s3:ReducedRedundancyLostObject* event type and an Amazon SNS topic as the destination. For more information about the new event types, see Amazon S3 Event Notifications. | November 13, 2014 |
| Server-side encryption with customer-provided encryption keys | Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)<br><br>Amazon S3 now supports server-side encryption using AWS KMS. This feature allows you to manage the envelope key through AWS KMS, and Amazon S3 calls AWS KMS to access the envelope key within the permissions you set.<br><br>For more information about server-side encryption with AWS KMS, see Protecting Data Using Server-Side Encryption with AWS Key Management Service. | November 12, 2014 |
| Europe (Frankfurt) Region | Amazon S3 is now available in the Europe (Frankfurt) Region. | October 23, 2014 |
| Server-side encryption with customer-provided encryption keys | Amazon S3 now supports server-side encryption using customer-provided encryption keys (SSE-C). Server-side encryption enables you to request Amazon S3 to encrypt your data at rest. When using SSE-C, Amazon S3 encrypts your objects with the custom encryption keys that you provide. Since Amazon S3 performs the encryption for you, you get the benefits of using your own encryption keys without the cost of writing or executing your own encryption code.<br><br>For more information about SSE-C, see Server-Side Encryption (Using Customer-Provided Encryption Keys). | June 12, 2014 |

| Change | Description | Date |
| --- | --- | --- |
| Lifecycle support for versioning | Prior to this release, lifecycle configuration was supported only on nonversioned buckets. Now you can configure lifecycle on both nonversioned and versioning-enabled buckets. For more information, see Managing your storage lifecycle. | May 20, 2014 |
| Access control topics revised | Revised Amazon S3 access control documentation. For more information, see Identity and Access Management for Amazon S3. | April 15, 2014 |
| Server access logging topic revised | Revised server access logging documentation. For more information, see Logging requests with server access logging. | November 26, 2013 |
| .NET SDK samples updated to version 2.0 | .NET SDK samples in this guide are now compliant to version 2.0. | November 26, 2013 |
| SOAP Support Over HTTP deprecated | SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs. | September 20, 2013 |
| IAM policy variable support | IAM policy language now supports variables. When a policy is evaluated, any policy variables are replaced with values that are supplied by context-based information from the authenticated user's session. You can use policy variables to define general purpose policies without explicitly listing all the components of the policy. For more information about policy variables, see IAM Policy Variables Overview in the *IAM User Guide*.<br><br>For examples of policy variables in Amazon S3, see Identity-based policy examples for Amazon S3. | April 3, 2013 |

| Change | Description | Date |
| --- | --- | --- |
| Console support for Requester Pays | You can now configure your bucket for Requester Pays by using the Amazon S3 console. For more information, see Using Requester Pays buckets for storage transfers and usage. | December 31, 2012 |
| Root domain support for website hosting | Amazon S3 now supports hosting static websites at the root domain. Visitors to your website can access your site from their browser without specifying **www** in the web address (for example, they can use **example.com** instead of **www.example.com**). Many customers already host static websites on Amazon S3 that are accessible from a **www** subdomain (for example, **www.example.com**). Previously, to support root domain access, you needed to run your own web server to proxy root domain requests from browsers to your website on Amazon S3. Running a web server to proxy requests introduces additional costs, operational burden, and another potential point of failure. Now, you can take advantage of the high availability and durability of Amazon S3 for both **www** and root domain addresses. For more information, see Hosting a static website using Amazon S3. | December 27, 2012 |
| Console revision | Amazon S3 console has been updated. The documentation topics that refer to the console have been revised accordingly. | December 14, 2012 |

| Change | Description | Date |
| --- | --- | --- |
| Support for Archiving Data to S3 Glacier | Amazon S3 now supports a storage option that enables you to utilize S3 Glacier's low-cost storage service for data archival. To archive objects, you define archival rules identifying objects and a time frame when you want Amazon S3 to archive these objects to S3 Glacier. You can easily set the rules on a bucket using the Amazon S3 console or programmatically using the Amazon S3 API or AWS SDKs.<br><br>For more information, see Managing your storage lifecycle. | November 13, 2012 |
| Support for Website Page Redirects | For a bucket that is configured as a website, Amazon S3 now supports redirecting a request for an object to another object in the same bucket or to an external URL. For more information, see (Optional) Configuring a webpage redirect.<br><br>For information about hosting websites, see Hosting a static website using Amazon S3. | October 4, 2012 |
| Support for Cross-Origin Resource Sharing (CORS) | Amazon S3 now supports Cross-Origin Resource Sharing (CORS). CORS defines a way in which client web applications that are loaded in one domain can interact with or access resources in a different domain. With CORS support in Amazon S3, you can build rich client-side web applications on top of Amazon S3 and selectively allow cross-domain access to your Amazon S3 resources. For more information, see Using cross-origin resource sharing (CORS). | August 31, 2012 |
| Support for Cost Allocation Tags | Amazon S3 now supports cost allocation tagging, which allows you to label S3 buckets so you can more easily track their cost against projects or other criteria. For more information about using tagging for buckets, see Using cost allocation S3 bucket tags. | August 21, 2012 |

| Change | Description | Date |
| --- | --- | --- |
| Support for MFA-protected API access in bucket policies | Amazon S3 now supports MFA-protected API access, a feature that can enforce AWS Multi-Factor Authentication for an extra level of security when accessing your Amazon S3 resources. It is a security feature that requires users to prove physical possession of an MFA device by providing a valid MFA code. For more information, go to AWS Multi-Factor Authentication. You can now require MFA authentication for any requests to access your Amazon S3 resources.<br><br>To enforce MFA authentication, Amazon S3 now supports the `aws:MultiFactorAuthAge` key in a bucket policy. For an example bucket policy, see Requiring MFA. | July 10, 2012 |
| Object Expiration support | You can use Object Expiration to schedule automatic removal of data after a configured time period. You set object expiration by adding lifecycle configuration to a bucket. | 27 December 2011 |
| New Region supported | Amazon S3 now supports the South America (São Paulo) Region. For more information, see Accessing and listing an Amazon S3 bucket. | December 14, 2011 |
| Multi-Object Delete | Amazon S3 now supports Multi-Object Delete API that enables you to delete multiple objects in a single request. With this feature, you can remove large numbers of objects from Amazon S3 more quickly than using multiple individual DELETE requests. For more information, see Deleting Amazon S3 objects. | December 7, 2011 |
| New Region supported | Amazon S3 now supports the US West (Oregon) Region. For more information, see Buckets and Regions. | November 8, 2011 |
| Documentation Update | Documentation bug fixes. | November 8, 2011 |

| Change | Description | Date |
| --- | --- | --- |
| Documentation Update | In addition to documentation bug fixes, this release includes the following enhancements:<br>• New server-side encryption sections using the AWS SDK for PHP and the AWS SDK for Ruby (see Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)). | October 17, 2011 |
| Server-side encryption support | Amazon S3 now supports server-side encryption. It enables you to request Amazon S3 to encrypt your data at rest, that is, encrypt your object data when Amazon S3 writes your data to disks in its data centers. In addition to REST API updates, the AWS SDK for Java and .NET provide necessary functionality to request server-side encryption. You can also request server-side encryption when uploading objects using the AWS Management Console. To learn more about data encryption, go to Using Data Encryption. | October 4, 2011 |
| Documentation Update | In addition to documentation bug fixes, this release includes the following enhancements:<br>• Added Ruby and PHP samples to the Making requests section.<br>• Added sections describing how to generate and use presigned URLs. For more information, see Sharing objects with presigned URLs and Sharing objects with presigned URLs.<br>• Updated an existing section to introduce AWS Explorers for Eclipse and Visual Studio. For more information, see Developing with Amazon S3 using the AWS SDKs. | September 22, 2011 |

| Change | Description | Date |
| --- | --- | --- |
| Support for sending requests using temporary security credentials | In addition to using your AWS account and IAM user security credentials to send authenticated requests to Amazon S3, you can now send requests using temporary security credentials you obtain from AWS Identity and Access Management (IAM). You can use the AWS Security Token Service API or the AWS SDK wrapper libraries to request these temporary credentials from IAM. You can request these temporary security credentials for your own use or hand them out to federated users and applications. This feature enables you to manage your users outside AWS and provide them with temporary security credentials to access your AWS resources.<br><br>For more information, see Making requests.<br><br>For more information about IAM support for temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*. | August 3, 2011 |
| Multipart Upload API extended to enable copying objects up to 5 TB | Prior to this release, Amazon S3 API supported copying objects of up to 5 GB in size. To enable copying objects larger than 5 GB, Amazon S3 now extends the multipart upload API with a new operation, `Upload Part (Copy)`. You can use this multipart upload operation to copy objects up to 5 TB in size. For more information, see Copying, moving, and renaming objects.<br><br>For conceptual information about multipart upload API, see Uploading and copying objects using multipart upload. | June 21, 2011 |
| SOAP API calls over HTTP disabled | To increase security, SOAP API calls over HTTP are disabled. Authenticated and anonymous SOAP requests must be sent to Amazon S3 using SSL. | June 6, 2011 |

| Change | Description | Date |
| --- | --- | --- |
| IAM enables cross-account delegation | Previously, to access an Amazon S3 resource, an IAM user needed permissions from both the parent AWS account and the Amazon S3 resource owner. With cross-account access, the IAM user now only needs permission from the owner account. That is, If a resource owner grants access to an AWS account, the AWS account can now grant its IAM users access to these resources.<br><br>For more information, see Creating a role to delegate permissions to an IAM user in the *IAM User Guide*.<br><br>For more information on specifying principals in a bucket policy, see Principals for bucket policies. | June 6, 2011 |
| New link | This service's endpoint information is now located in the *AWS General Reference*. For more information, go to Regions and Endpoints in the AWS General Reference. | March 1, 2011 |
| Support for hosting static websites in Amazon S3 | Amazon S3 introduces enhanced support for hosting static websites. This includes support for index documents and custom error documents. When using these features, requests to the root of your bucket or a subfolder (for example, `http://mywebsite.com/subfolder` ) returns your index document instead of the list of objects in your bucket. If an error is encountered, Amazon S3 returns your custom error message instead of an Amazon S3 error message. For more information, see Hosting a static website using Amazon S3. | June 6, 2011 |

| Change | Description | Date |
| --- | --- | --- |
| This service's endpoint information is now located in the *AWS General Reference*. For more information, go to Regions and Endpoints in the AWS General Reference. | March 1, 2011 | |
| Support for hosting static websites in Amazon S3 | Amazon S3 introduces enhanced support for hosting static websites. This includes support for index documents and custom error documents. When using these features, requests to the root of your bucket or a subfolder (for example, `http://mywebsite.com/subfolder`) returns your index document instead of the list of objects in your bucket. If an error is encountered, Amazon S3 returns your custom error message instead of an Amazon S3 error message. For more information, see Hosting a static website using Amazon S3. | February 17, 2011 |
| Response Header API Support | The GET Object REST API now allows you to change the response headers of the REST GET Object request for each request. That is, you can alter object metadata in the response, without altering the object itself. For more information, see Downloading objects. | January 14, 2011 |
| Large object support | Amazon S3 has increased the maximum size of an object you can store in an S3 bucket from 5 GB to 5 TB. If you are using the REST API, you can upload objects of up to 5 GB in a single PUT operation. For larger objects, you must use the Multipart Upload REST API to upload objects in parts. For more information, see Uploading and copying objects using multipart upload. | December 9, 2010 |

| Change | Description | Date |
| --- | --- | --- |
| Multipart upload | Multipart upload enables faster, more flexible uploads into Amazon S3. It allows you to upload a single object as a set of parts. For more information, see Uploading and copying objects using multipart upload. | November 10, 2010 |
| Canonical ID support in bucket policies | You can now specify canonical IDs in bucket policies. For more information, see Principals for bucket policies | September 17, 2010 |
| Amazon S3 works with IAM | This service now integrates with AWS Identity and Access Management (IAM). For more information, go to AWS services that work with IAM in the *IAM User Guide*. | September 2, 2010 |
| Notifications | The Amazon S3 notifications feature enables you to configure a bucket so that Amazon S3 publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic when Amazon S3 detects a key event on a bucket. For more information, see Setting Up Notification of Bucket Events. | July 14, 2010 |
| Bucket policies | Bucket policies are an access management system that you use to set access permissions across buckets, objects, and sets of objects. This functionality supplements and in many cases replaces access control lists. For more information, see Bucket policies for Amazon S3. | July 6, 2010 |
| Path-style syntax available in all Regions | Amazon S3 now supports the path-style syntax for any bucket in the US Classic Region, or if the bucket is in the same Region as the endpoint of the request. For more information, see Virtual Hosting. | June 9, 2010 |
| New endpoint for Europe (Ireland) | Amazon S3 now provides an endpoint for Europe (Ireland) : `http://s3-eu-west-1.amazonaws.com` . | June 9, 2010 |

| Change | Description | Date |
| --- | --- | --- |
| Console | You can now use Amazon S3 through the AWS Management Console. You can read about all of the Amazon S3 functionality in the console in the Amazon Simple Storage Service User Guide. | June 9, 2010 |
| Reduced Redundancy | Amazon S3 now enables you to reduce your storage costs by storing objects in Amazon S3 with reduced redundancy. For more information, see Reduced Redundancy Storage. | May 12, 2010 |
| New Region supported | Amazon S3 now supports the Asia Pacific (Singapore) Region. For more information, see Buckets and Regions. | April 28, 2010 |
| Object Versioning | This release introduces object versioning. All objects now can have a key and a version. If you enable versioning for a bucket, Amazon S3 gives all objects added to a bucket a unique version ID. This feature enables you to recover from unintended overwrites and deletions. For more information, see Versioning and Using Versioning. | February 8, 2010 |
| New Region supported | Amazon S3 now supports the US West (N. California) Region. The new endpoint for requests to this Region is `s3-us-west-1.amazonaws.com` . For more information, see Buckets and Regions. | December 2, 2009 |
| AWS SDK for .NET | AWS now provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using .NET language-specific API operations instead of REST or SOAP. These libraries provide basic functions (not included in the REST or SOAP APIs), such as request authentication, request retries, and error handling so that it's easier to get started. For more information about language-specific libraries and resources, see Developing with Amazon S3 using the AWS SDKs. | November 11, 2009 |

# AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.