Getting Started with Amazon EC2 and Amazon SQS

Building Scalable, Reliable Amazon EC2 Applications with Amazon SQS

Overview

Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud. With Amazon EC2 you can build applications that start small but can scale up rapidly as demand increases. Amazon EC2 offers many features that make it possible to build such applications, including:

a) Ability to increase or decrease capacity within minutes
b) Ability to commission one, hundreds, or even thousands of server instances simultaneously
c) A web service API to control the scaling of instances depending on your needs
d) A "pay only for what you use" pricing model

However, to take advantage of the full power of Amazon EC2, it is important to architect your application correctly—from the very beginning. Otherwise, it is hard to re-architect your application later in the lifecycle without disrupting the service, losing data, or rewriting substantial parts of it. To ensure that your application scales with increasing load, you must ensure (among other things) that it has no bottlenecks or single points of failure, and that it can continue to run even when one or more components fail.

Amazon Simple Queue Service (Amazon SQS) is a complementary web service that enables you to build highly scalable EC2 applications easily. Amazon SQS is a highly reliable, scalable message queuing service that enables asynchronous message-based communication between distributed components of an application. The components can be computers or EC2 instances or a combination of both. With Amazon SQS you can send any number messages to an Amazon SQS queue at any time from any component. The messages can be retrieved from the same component or a different one right away or at a later time[1]. No message is ever lost in the interim; each message is persistently stored in highly available, highly reliable queues. Multiple processes can read/write from/to an Amazon SQS queue at the same time without interfering with each other.

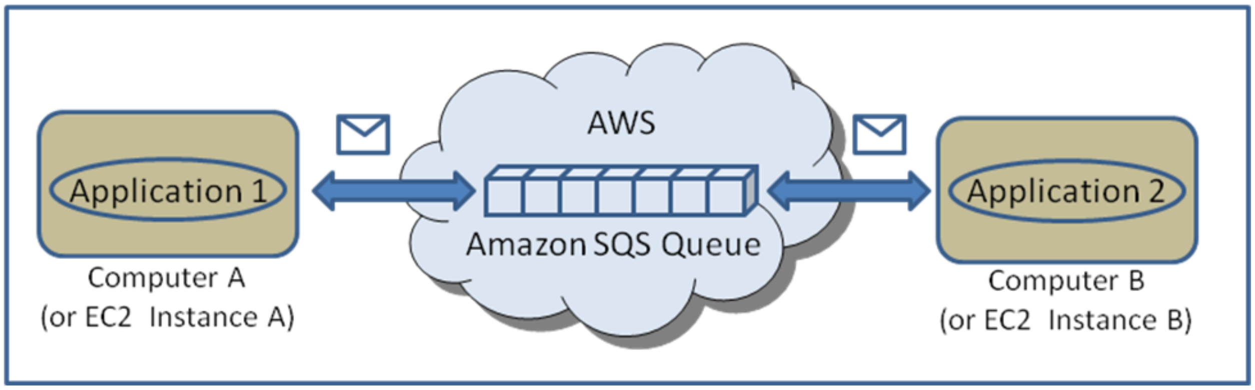

Figure 1: Two distributed applications communicating asynchronously by passing messages through an Amazon SQS queue.

[1] Within 4 days (with the newer Amazon SQS API version) and within 15 days (with the older API version)

Amazon SQS derives its reliability and scalability from Amazon's web-scale messaging infrastructure on which it is built. As with other Amazon Web Services, with Amazon SQS you pay for only what you use.

This paper provides a brief overview of the capabilities of SQS and how it can used to build scalable EC2 applications.

Why Amazon SQS for Scalable, Reliable EC2 Applications?

All systems that have to scale to meet increasing load while continuing to provide reliable and predictable performance must satisfy some unique requirements. Amazon SQS has many features specifically designed to enable you to build scalable, reliable and high-performing EC2 applications. Let's explore this in more detail with a simple but fairly common scenario.

Example: Online Photo Processing Service for Consumers

Assume you want to offer an online photo processing service for consumers. The services lets consumer specify the operations they want performed on their photos. Operations could include redeye reduction, cropping, thumb nail creation, customization, re-coloring, teeth whitening, etc. Users upload the photos to your web site and specify the tasks to be performed on the photos. Users can submit as few as one or as many as hundreds of photos in a single upload session. After submitting the photos, users can come back and check on the status of their photos. When the photos have been processed, they can download the photos from the web site.

Let's assume that different operations take different processing times, ranging from a few seconds to several minutes. Therefore, the time to complete the user's request depends on the number of photos, the size of the photos, and the processing operations to be performed.

For now let's start with these basic requirements and add more as we go along.

Getting Started with Amazon SQS

The figure below provides an overview of the architecture of a simple system that meets the requirements listed above. It includes Amazon SQS working in conjunction with Amazon EC2 and Amazon S3.

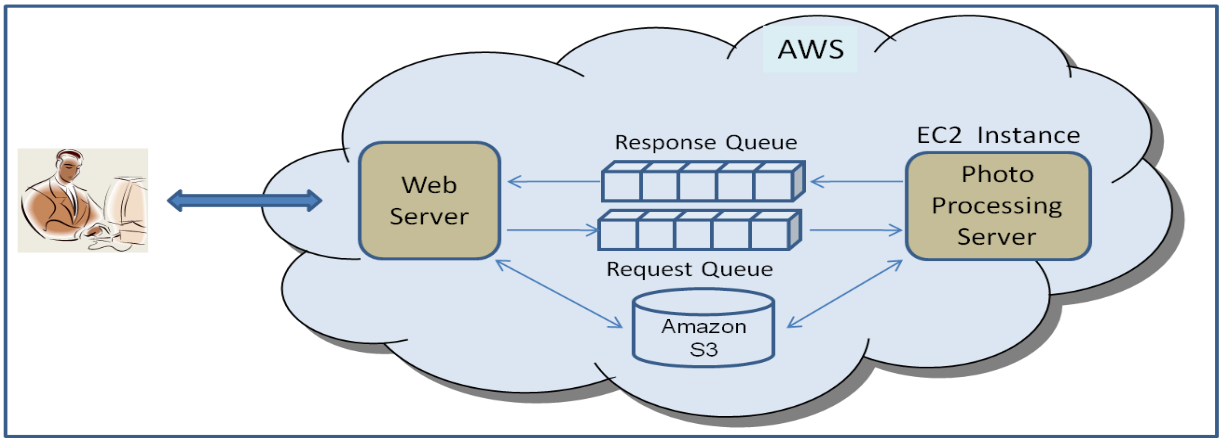

Figure 2: Architecture showing the use of Amazon SQS as a job dispatch mechanism for EC2

Here is the end-to-end flow of a typical request:

Every user request results in a message being queued into the Amazon SQS "Request" queue. At the same time, the application stores the photos in Amazon S3. The message in the queue contains (among other things) the photo processing operation to be performed and a pointer to the location of the photos in Amazon S3. The Photo Processing Server, running in an EC2 instance, reads messages from the queue, processes the request, and on completion, posts a status message to the "Response" queue.

What makes this a preferred architecture for building a scalable, reliable EC2 application? Consider the following real world possibilities:

1. What happens if, due to a bug or some other reason, the Photo Processing Server crashes or otherwise becomes temporarily unavailable, as shown in the figure below?

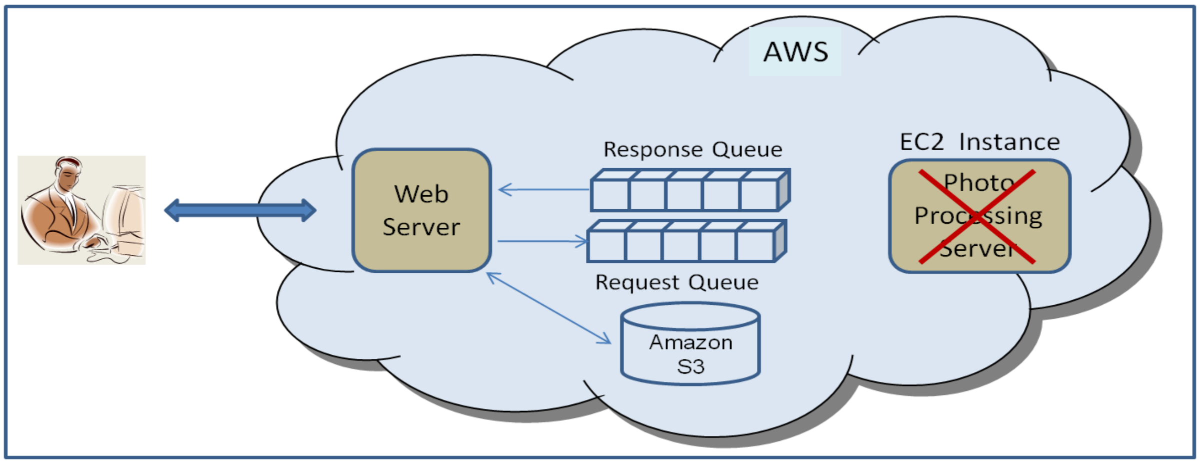

Figure 3: How Amazon SQS isolates failures of a component from the rest of the system.

   The failure will be transparent to the end user. Users can continue to upload photos to the web site, and in turn the web server can continue to send messages to the Amazon SQS Request queue. The messages will remain in the queue until the Photo Processing Server is back online. The key point to note is that the Photo Processing Server does not have to do anything extra to remember the last message it processed before the crash. Amazon SQS will ensure that the server will resume processing the messages from where it left off. Furthermore, there is no special code required to deal with the message that was being processed when the server crashed. That's because even though the message was read from the Amazon SQS queue, the message remains in the queue until the server explicitly deletes it. So if the server fails while processing a message and therefore before deleting the message, it will find the message again when it comes back online.

2. Next, let's consider the case where the Photo Processing Server or the EC2 instance in which it is running cannot be restarted for an extended period of time. While users will be able to post their photos to the site, they will not be able to get back their processed photos. The solution is to start another, identical, Photo Processing Server, in its own EC2 instance, to replace the failed server. Amazon SQS enables this to be easily accomplished as shown in the figure below.

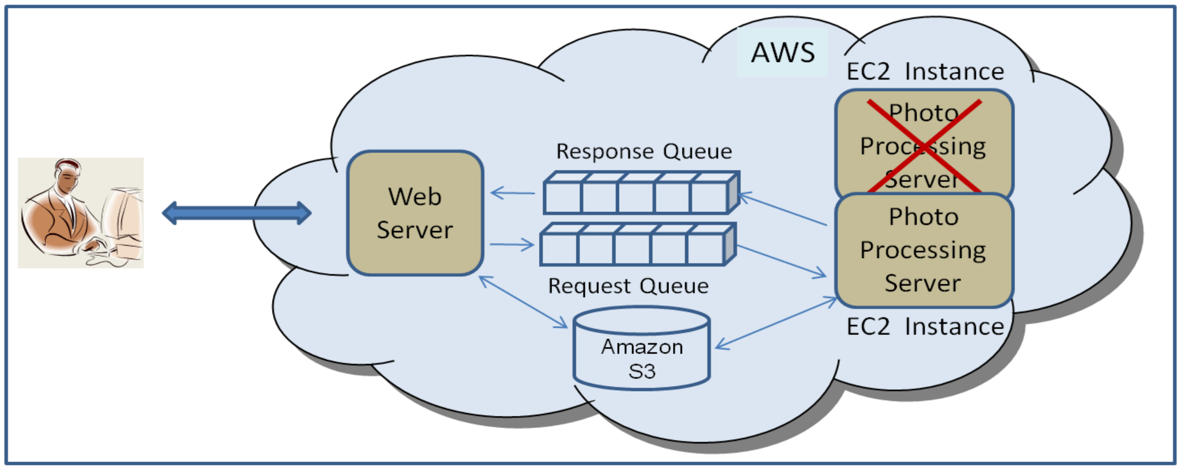

Figure 4: Implementing a high-availability solution with Amazon SQS

Amazon SQS makes it possible to just drop in a replacement server without impacting the rest of the system. All that is needed to implement this solution is to make sure that the replacement server is pointing to the same Amazon SQS Request/Response queue pair.

3. For the next scenario, consider the case where a single Photo Processing Server is not sufficient to meet user demand. With Amazon SQS, it is possible to introduce additional instances of the Photo Processing Server to meet increasing demand, as shown in the figure below.

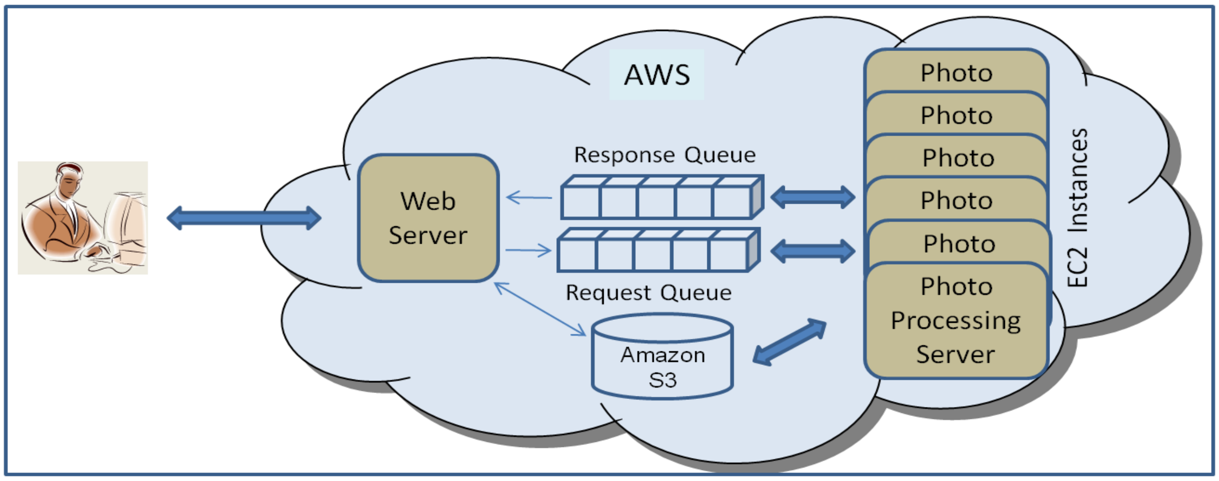

Figure 4: How to run multiple EC2 instances simultaneously with Amazon SQS to handle increased load

Two specific features of Amazon SQS make this possible:

- A single Amazon SQS queue can be shared by multiple servers simultaneously.
- A server that is processing a message can prevent other servers from processing the same message at the same time by temporarily "locking" a message. The server can specify the amount of time the message is locked. When the server is done processing the message, it should delete the message. If the server fails while processing the message, another server can get the message after the lockout period.

These two features ensure that the number of processing servers can be easily changed dynamically to handle varying load. The entire process can be automated to ensure that at any given time the optimal number of EC2 instances is running. This practice is commonly referred to as "auto-scaling." For more details on auto-scaling, read "Auto-Scaling Amazon EC2 with Amazon SQS".

4. Finally consider this scenario: Because some of the photo processing operations take significantly longer time than the rest, you want to implement these longer-running operations in a separate, dedicated server. This can be implemented with minimal disruption to the rest of the system using Amazon SQS, as shown in the figure below.

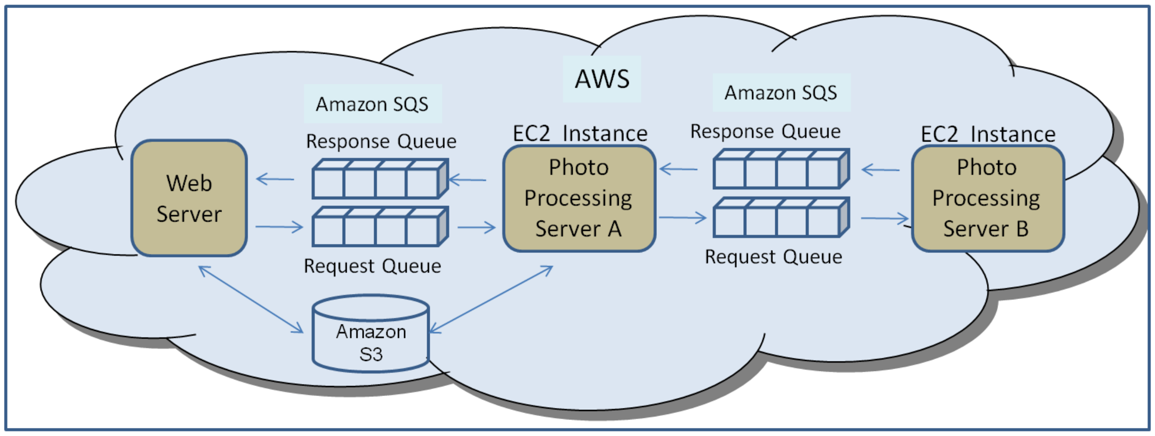

Figure 5: Pipeline processing with Amazon SQS

In the figure, the longer-running operations are implemented in Server B. Server A and Server B communicate through Amazon SQS Request/Response queues[2]. This setup commonly referred to as "pipeline processing," is often used to build large applications with multiple sub-components. Pipeline processing architectures implemented with Amazon SQS have the following advantages:

- Flexibility: Components can be easily rewired differently to change the workflow. A large monolithic server can be divided into multiple smaller servers without impacting the rest of the system.
- Piecemeal upgrades: Individual sub-components can be taken offline and upgraded without bringing the entire system to a halt. In a large-scale system with multiple sub-components, a forklift style "all at once" approach to upgrading is not practical or feasible.
- Tolerance to failures: Amazon SQS isolates sub-components from each other so the failure of one component does not impact the rest.

Getting Started

For practical, hands-on experience in using Amazon SQS with Amazon EC2, try the Sample application to get started with Amazon SQS and Amazon EC2 or build your own SQS-enables AMI using the SQS-EC2 Job Processor Sample.

Summary

Amazon SQS complements Amazon EC2. Using Amazon SQS you can build your first, simple EC2 application quickly and easily, with the assurance that you have the right architectural foundation to make it scale reliably if the need arises. SQS is easy to use, proven technology used by a majority of the large EC2 applications.

[2] For clarity, multiple instances of the servers are not shown, but it is possible to have multiple instances of both Server A and Server B.