Revision Control

git

Afif Elghraoui
NIH HPC staff
staff@hpc.nih.gov

NIH

BIOWULF
HIGH PERFORMANCE COMPUTING AT THE NIH

Why use Git?

- Sanity
  - No need to manually maintain multiple copies of the same file.
- History
- Annotations
- Undoability
- Collaboration

What Goes in Git?

- Content that is manually generated and maintained.
- Primarily text files. Don’t commit large binary files because Git can’t compare and track them efficiently. If you need to track them, consider tools like git-lfs or git-annex.

Basic Concepts

- A **repository** is a directory whose contents you choose to mange with Git.
- A **commit** is a set of changes to one or more files in the repository.

Sanity
avoiding copies of the same file

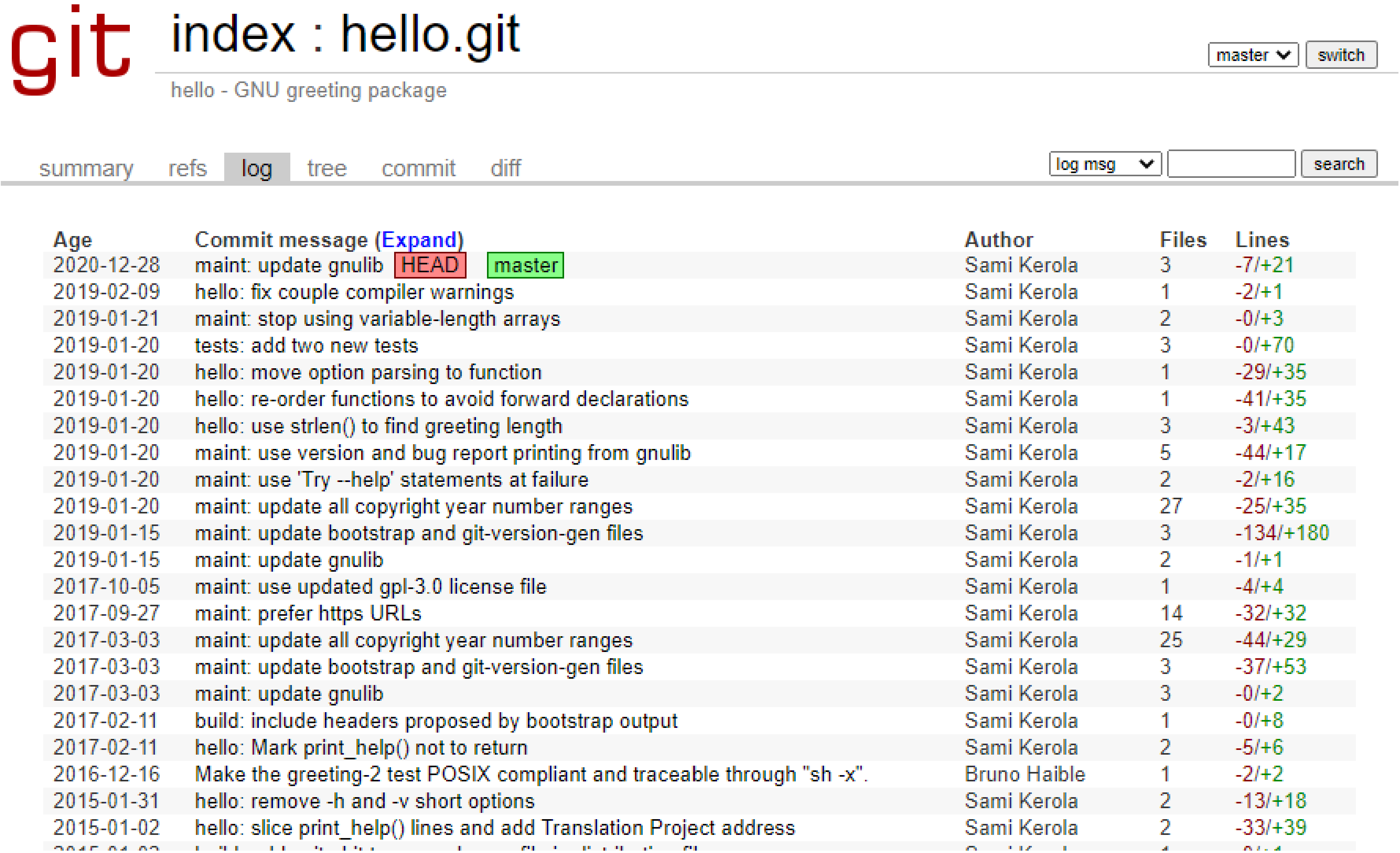
Without version control:

```
$ ls
   myscript
   myscript2
   myscript3
   myscript4
```

With version control:

```
$ ls
   myscript
$ ./myscript --version
   4
$ git checkout v3
$ ./myscript --version
   3
```

History



index : hello.git

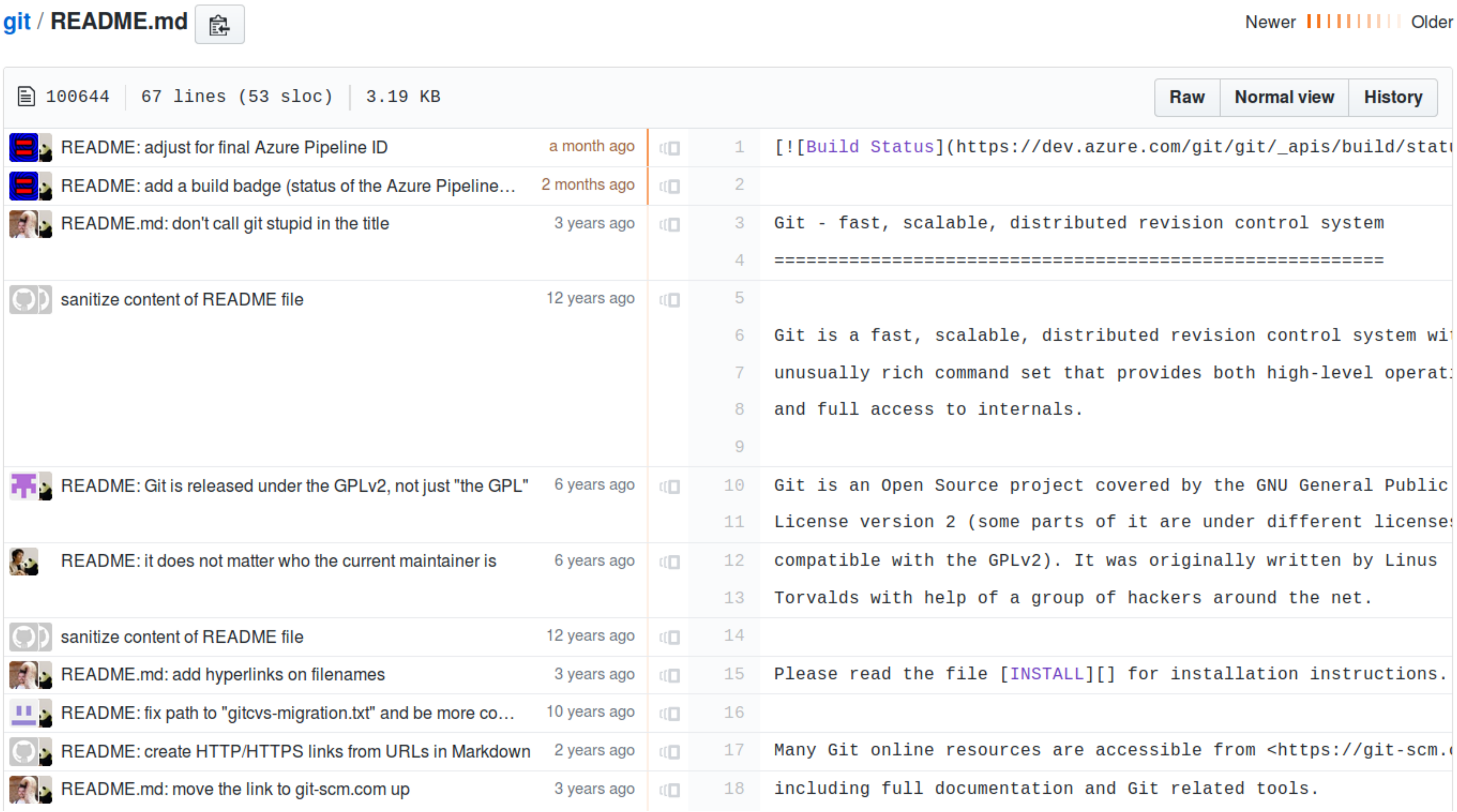
hello - GNU greeting package

Age	Commit message (Expand)	Author	Files	Lines
2020-12-28	maint: update gnulib HEAD master	Sami Kerola	3	-7/+21
2019-02-09	hello: fix couple compiler warnings	Sami Kerola	1	-2/+1
2019-01-21	maint: stop using variable-length arrays	Sami Kerola	2	-0/+3
2019-01-20	tests: add two new tests	Sami Kerola	3	-0/+70
2019-01-20	hello: move option parsing to function	Sami Kerola	1	-29/+35
2019-01-20	hello: re-order functions to avoid forward declarations	Sami Kerola	1	-41/+35
2019-01-20	hello: use strlen() to find greeting length	Sami Kerola	3	-3/+43
2019-01-20	maint: use version and bug report printing from gnulib	Sami Kerola	5	-44/+17
2019-01-20	maint: use 'Try --help' statements at failure	Sami Kerola	2	-2/+16
2019-01-20	maint: update all copyright year number ranges	Sami Kerola	27	-25/+35
2019-01-15	maint: update bootstrap and git-version-gen files	Sami Kerola	3	-134/+180
2019-01-15	maint: update gnulib	Sami Kerola	2	-1/+1
2017-10-05	maint: use updated gpl-3.0 license file	Sami Kerola	1	-4/+4
2017-09-27	maint: prefer https URLs	Sami Kerola	14	-32/+32
2017-03-03	maint: update all copyright year number ranges	Sami Kerola	25	-44/+29
2017-03-03	maint: update bootstrap and git-version-gen files	Sami Kerola	3	-37/+53
2017-03-03	maint: update gnulib	Sami Kerola	3	-0/+2
2017-02-11	build: include headers proposed by bootstrap output	Sami Kerola	1	-0/+8
2017-02-11	hello: Mark print_help() not to return	Sami Kerola	2	-5/+6
2016-12-16	Make the greeting-2 test POSIX compliant and traceable through "sh -x".	Bruno Haible	1	-2/+2
2015-01-31	hello: remove -h and -v short options	Sami Kerola	2	-13/+18
2015-01-02	hello: slice print_help() lines and add Translation Project address	Sami Kerola	2	-33/+39

https://git.savannah.gnu.org/cgit/hello.git/log/

Annotations

git blame



100644 | 67 lines (53 sloc) | 3.19 KB

README: adjust for final Azure Pipeline ID	a month ago	[![Build Status](https://dev.azure.com/git/git/_apis/build/stat
README: add a build badge (status of the Azure Pipeline…	2 months ago	
README.md: don't call git stupid in the title	3 years ago	Git - fast, scalable, distributed revision control system
		===
sanitize content of README file	12 years ago	
		Git is a fast, scalable, distributed revision control system wi
		unusually rich command set that provides both high-level operat
		and full access to internals.
README: Git is released under the GPLv2, not just "the GPL"	6 years ago	Git is an Open Source project covered by the GNU General Public
		License version 2 (some parts of it are under different license
README: it does not matter who the current maintainer is	6 years ago	compatible with the GPLv2). It was originally written by Linus
		Torvalds with help of a group of hackers around the net.
sanitize content of README file	12 years ago	
README.md: add hyperlinks on filenames	3 years ago	Please read the file [INSTALL][] for installation instructions.
README: fix path to "gitcvs-migration.txt" and be more co…	10 years ago	
README: create HTTP/HTTPS links from URLs in Markdown	2 years ago	Many Git online resources are accessible from <https://git-scm.
README.md: move the link to git-scm.com up	3 years ago	including full documentation and Git related tools.

https://github.com/git/git/blame/master/README.md

Undoability
Keeping your scripts tidier

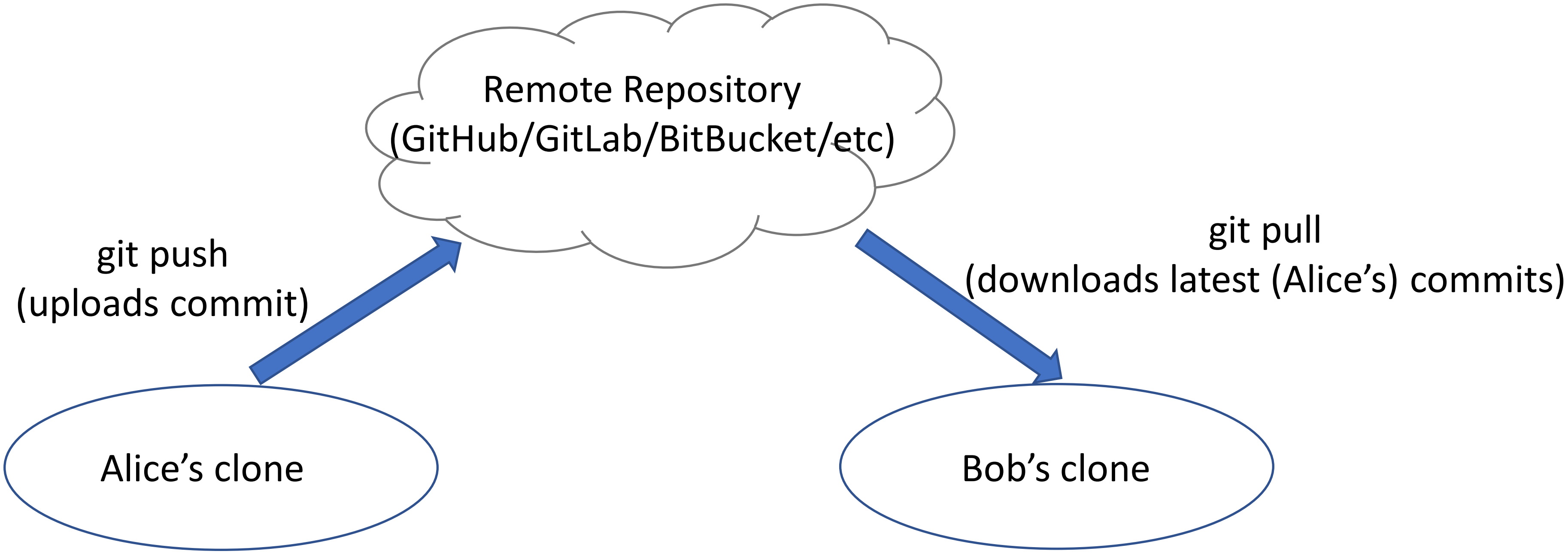
Without version control:

```
##myscript

#### Not sure if we’ll need this code again later
# x = [f.lower() for f in x]

def foo():
    ...
```

With version control, you can safely delete code and undo the deletion later.

```
$ nano myscript # delete commented section above
$ git add myscript && git commit -m “Don’t lower-case elements of x”
$ git revert <commitID> # oops. Turns out we needed it after all
```

Collaboration



Remote Repository (GitHub/GitLab/BitBucket/etc)

git push (uploads commit)

git pull (downloads latest (Alice's) commits)

Alice's clone

Bob's clone

Creates commit

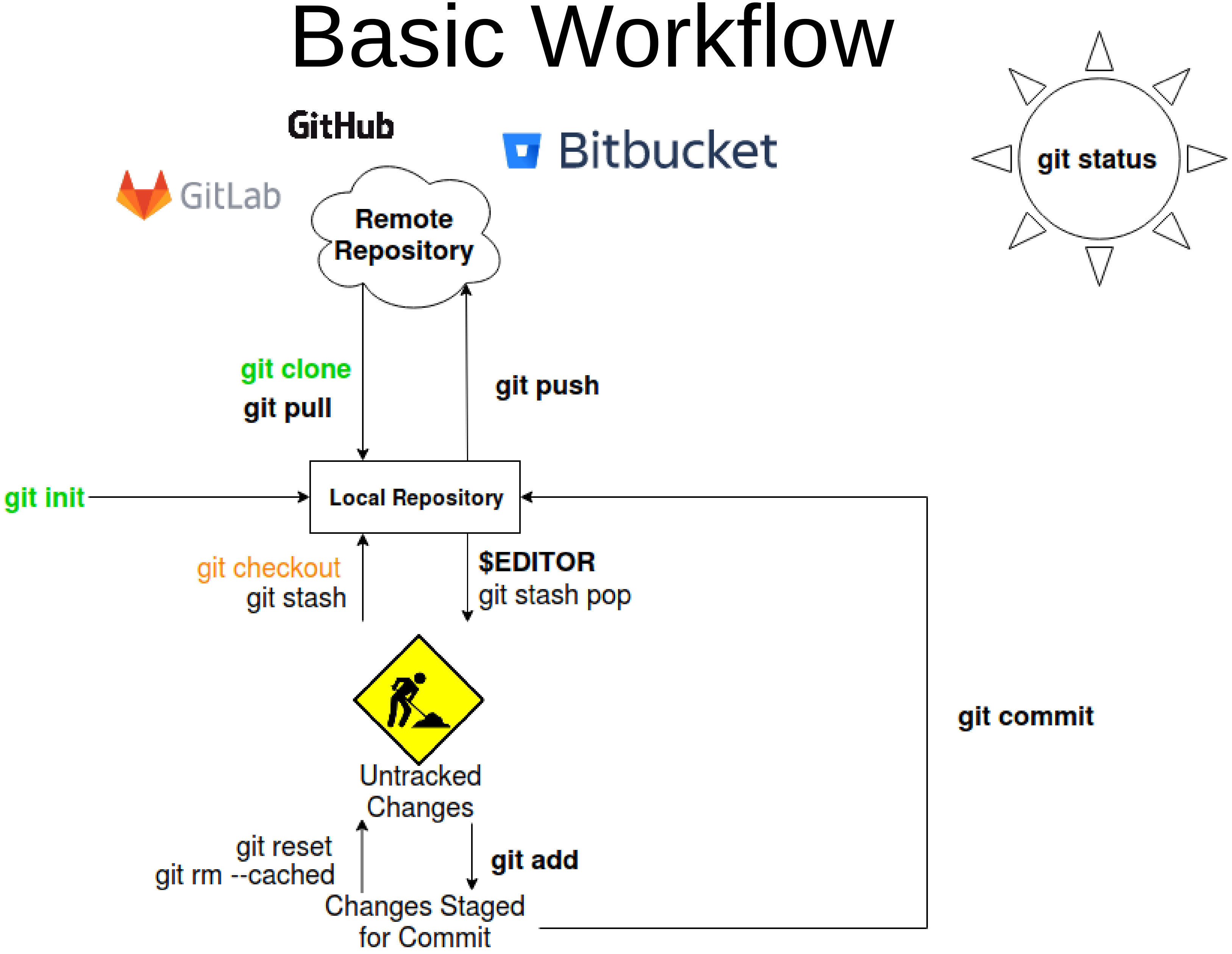
*Works in progress can be developed in **branches** to allow you to incrementally make commits until the full set of changes are ready, while leaving the main branch with a stable working version.

Getting started

```
git config --global user.name “Fulan bin Fulan”

git config --global user.email fulan@example.com

export EDITOR=nano

echo “!!” >> ~/.bashrc
```

- Global configurations go into `~/.gitconfig` by default. Local configurations are per-repository.

Basic Workflow



GitHub

Bitbucket

GitLab

Remote Repository

git status

git clone

git pull

git push

git init

Local Repository

git checkout

git stash

$EDITOR

git stash pop

git commit

Untracked Changes

git reset

git rm --cached

git add

Changes Staged for Commit

🟩 initialization

🟧 destructive operation

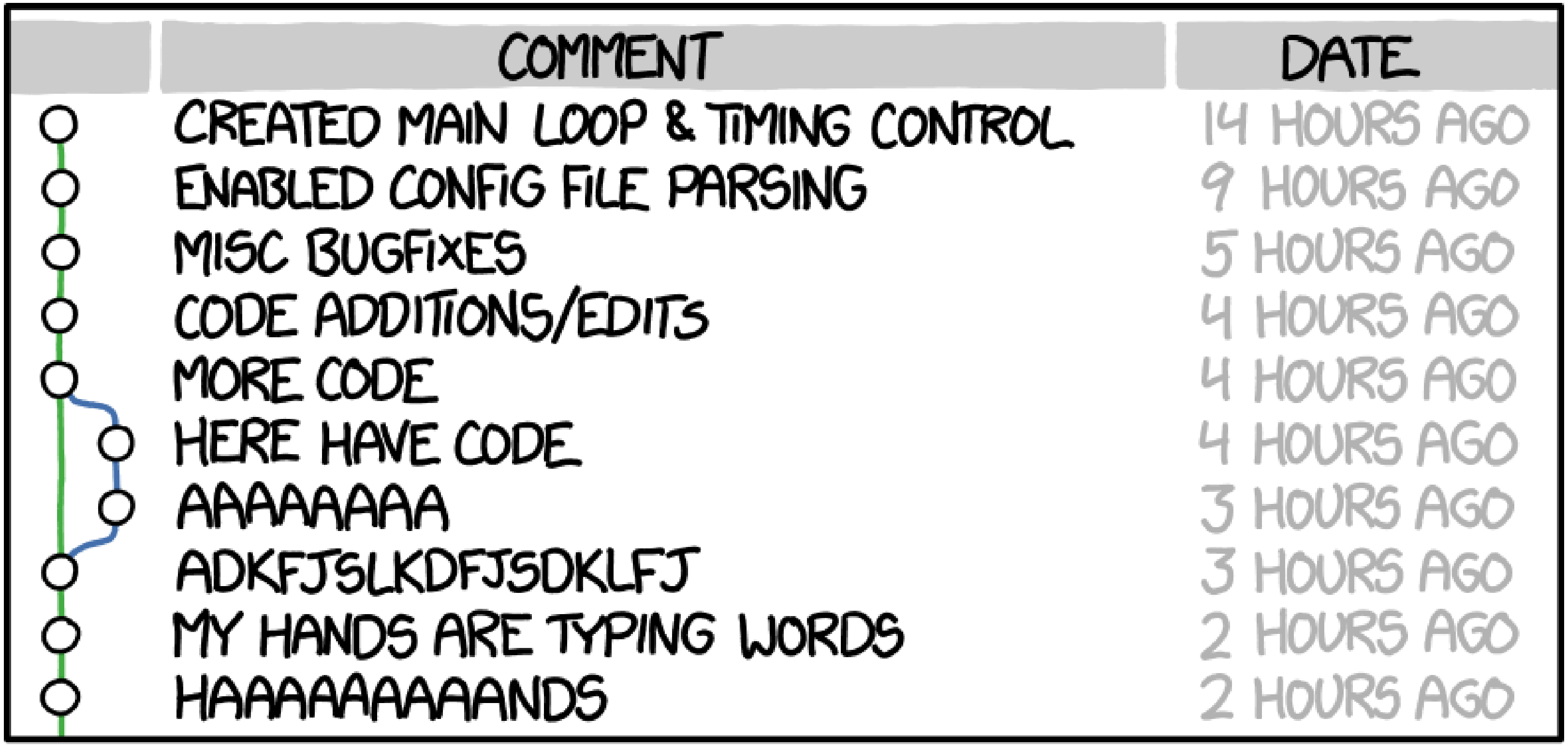
BIOWULF

HIGH PERFORMANCE COMPUTING AT THE NIH

Ignoring files: .gitignore

- Specify file names or patterns in a file named .gitignore in your repository to avoid accidentally committing unwanted files like
  - Editor backups
  - Program outputs
  - Sensitive information

Commit Messages



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

https://xkcd.com/1296/

Commit Messages

- Bad:

```
Updated README
```

- Good:

```
README: update installation steps due to OS upgrade


The new version of the OS no longer comes with support for foo, which we were relying on.
```

Inspecting the Repository

- `git log`
  - See the change history of a file: `git log -p filename`
- `git blame`
- `git diff`
- A graphical utility: `gitk(1)`
- Use `git checkout` to move to different points in history

Marking Important Snapshots: versioning and `git tag`

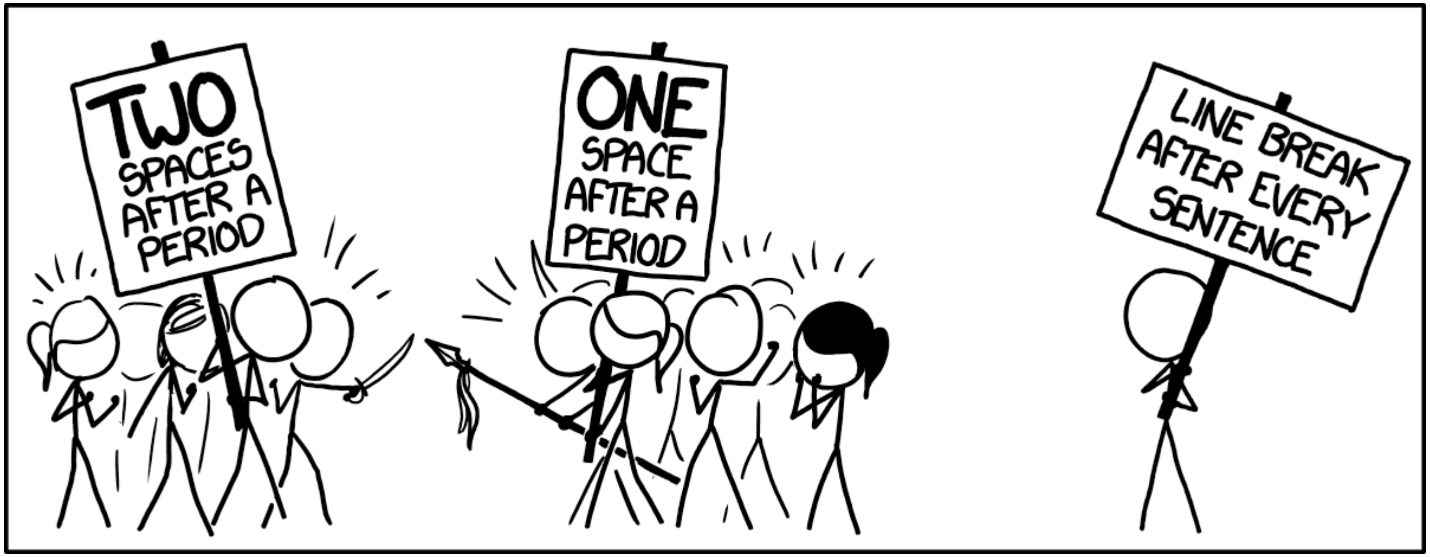
- Example: create an annotated tag marking the current snapshot as version 0.1.0:

```
git tag -a 0.1.0
```

- a commit id can be specified at the end if you don’t want the tag pointing to the current HEAD.
- Semantic versioning - https://semver.org/

Helpful Conventions

- Put meaningful components on their own line to be able to get more useful comparisons
  - When editing prose in a markup language, put a line break after every sentence to get this effect.



https://xkcd.com/1285/

Remote Repositories

- Common Git hosting providers are GitHub, GitLab, and Bitbucket
- Use `git remote` to manage remote repositories from your working directory.

Branching and Collaboration

- Create a new branch:

```
git branch branchname
```

- Switch to branch:

```
git checkout branchname
```

- ...or create the branch and switch to it in one step:

```
git checkout -b branchname
```

- Make your changes (edit, git add, git commit)
- Push your branch to the remote repository

```
git push origin branchname --set-upstream
```

- Switch back to the default branch

```
git checkout master
```

git merge

- Git can automatically merge branches if there are no conflicting changes.
- Merge conflicts are presented to you to resolve:
  - Both versions of the file are available for comparing.
  - Use a merge tool to make your life easier. See
  - `git mergetool --tool-help` for a list of options.
- Fun fact: `git pull` is actually a shorthand for `git fetch` followed by `git merge`

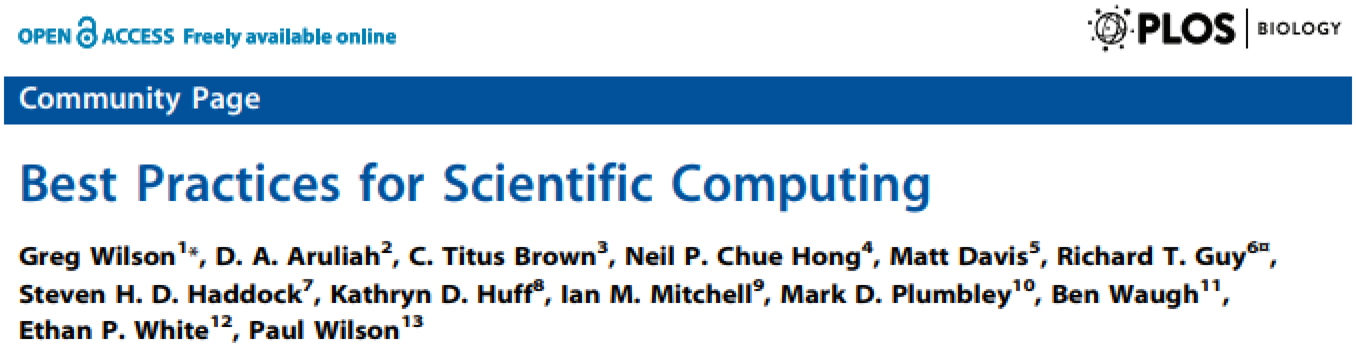
making sure code is easy to run

- Documentation
  - README files in the program’s repository
  - Usage information printed by the program
  - Informative comments in the code itself
- Using a workflow manager such as Snakemake
  - Allows for specifying how a set of programs should be run together to produce a final result.
  - https://github.com/NIH-HPC/snakemake-class

Git Resources

- reference book: https://git-scm.com/book/en/v2
- man pages:
  - git(1)
  - **man git-*subcommand*** or **git help *subcommand***
- more man pages:
  - giteveryday(7)
  - gittutorial(7)

Additional Resources



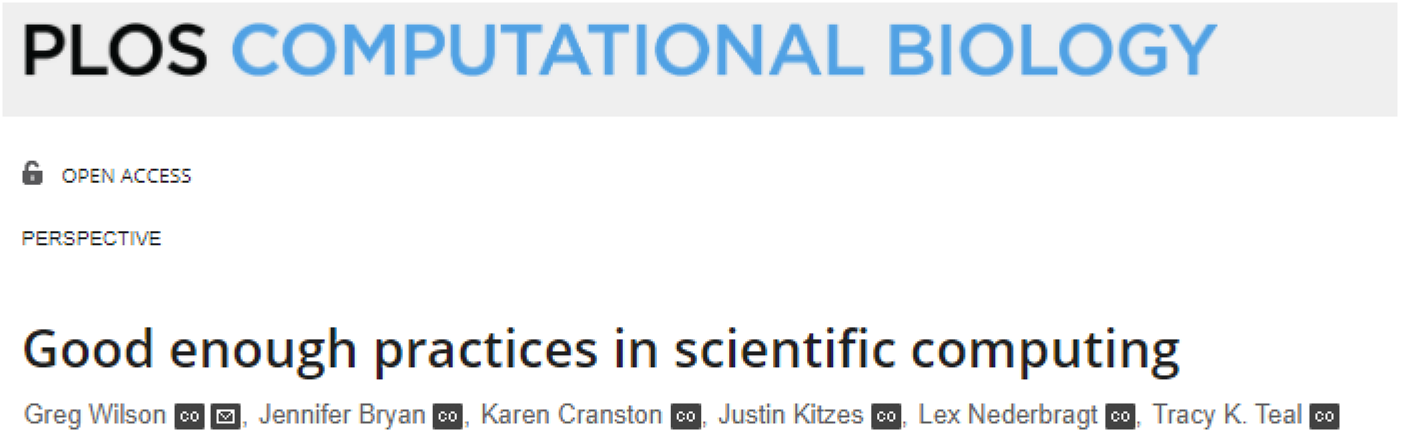
OPEN ACCESS Freely available online

PLOS | BIOLOGY

Community Page

Best Practices for Scientific Computing

Greg Wilson[1]*, D. A. Aruliah[2], C. Titus Brown[3], Neil P. Chue Hong[4], Matt Davis[5], Richard T. Guy[6¤], Steven H. D. Haddock[7], Kathryn D. Huff[8], Ian M. Mitchell[9], Mark D. Plumbley[10], Ben Waugh[11], Ethan P. White[12], Paul Wilson[13]



PLOS COMPUTATIONAL BIOLOGY

OPEN ACCESS

PERSPECTIVE

Good enough practices in scientific computing

Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, Tracy K. Teal