Introduction to Git and GitHub

Scott Goldstein and Kristen Howard

August 8, 2022

Welcome!

- This presentation starts with some background and then uses a simple interactive example.
- It emphasizes core git operations. As a new user to git, you do not need to focus on everything that is possible with git.
- We will instead highlight library-specific examples of projects and advice not easily discoverable in other tutorials. (But we will still provide many resources at the end!)

What is version control?

- Any system that manages changes to files over time.
- Benefits of a version control system include:
    - Collaboration
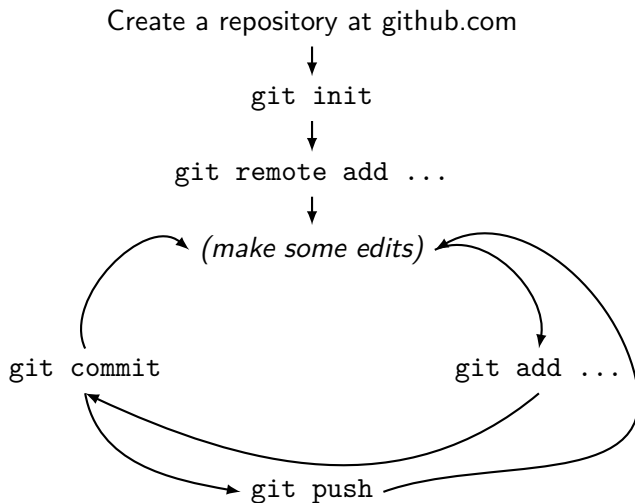   - Versioning

A short history of version control

- SCCS (Source Code Control System) (1972)
- RCS (Revision Control System) (1982)
- CVS (Concurrent Versions System) (1990)
- Subversion (2000)
- Git, Mercurial, Bazaar (2005)

What do librarians use git for?

- GitHub Pages – a free web server for simple static websites
  - Professional/portfolio websites
  - Miscellaneous websites for libraries (e.g., conferences, projects)
  - Prototypes/mockups
- Sharing documents or other assets (e.g., fonts, images)
- Sharing code
  - Python, R, etc. scripts
  - Drupal modules
  - Many other examples...

A very basic workflow

Create a repository at github.com

↓

`git init`

↓

`git remote add ...`

↓

(make some edits)

`git commit`   `git add ...`

`git push`

Some terminology

A **repository (repo)** refers to the set of files that make up your project.

- Local vs. remote

A **commit** is best thought of as a snapshot of your project at a point in time. As you work on your project, you will "snap" more commits. The ordered sequence of all these commits is called a **branch**. (A repository can have more than one branch, but today we will only be dealing with one branch, conventionally called *main*.)

Some terminology

In many ways, Git is like the version history of a Microsoft Word
file or Google Doc. But what makes it a little more complicated is
that projects typically consist of multiple files, and not all files need
to be put together in the same commits. It is *your* job to tell Git
which files are changing and when.

Working tree: What you are editing; the files you see in front of
you.
Staging area: The files you mark as needing to be updated in the
next commit. Files not on the stage are "carried over" from the
previous commit.

A graphical representation

WORKING TREE STAGE REPOSITORY

TERMINAL STATUS

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

3 untracked files

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

`git add 1 2 3`   3 untracked files

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

git add 1 2 3 **3 untracked files; 3 staged files**

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

`git commit`   3 untracked files; 3 staged files

A graphical representation

WORKING TREE   STAGE   REPOSITORY

1 2 3   1 2 3

TERMINAL   STATUS

git commit
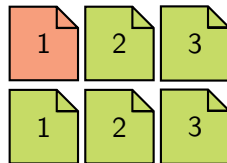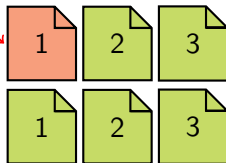
A graphical representation

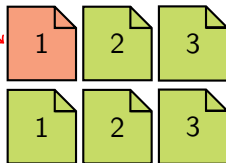WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

1 modified file

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

`git add 1`   1 modified file
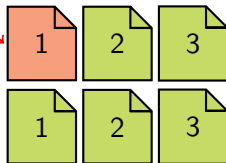
A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

`git add 1`   1 modified file; 1 staged file

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

`git commit`   1 modified file; 1 staged file

A graphical representation

WORKING TREE   STAGE   REPOSITORY

TERMINAL   STATUS

git commit

A graphical representation

WORKING TREE   STAGE   REPOSITORY

HEAD

TERMINAL   STATUS

git commit

A graphical representation

WORKING TREE   STAGE   REPOSITORY

A graphical representation

WORKING TREE   STAGE   REPOSITORY

HEAD

TERMINAL   STATUS

`git add 3`   1 modified file
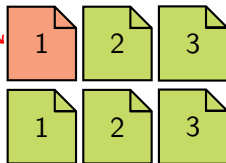
A graphical representation

WORKING TREE   STAGE   REPOSITORY

HEAD

TERMINAL   STATUS

`git add 3`   1 modified file; 1 staged file
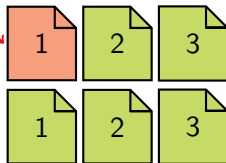
A graphical representation

WORKING TREE   STAGE   REPOSITORY

HEAD

TERMINAL   STATUS

`git commit`   1 modified file; 1 staged file
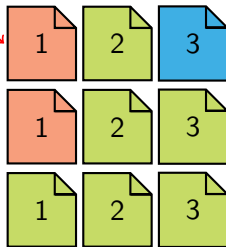
A graphical representation

WORKING TREE   STAGE   REPOSITORY

HEAD

TERMINAL   STATUS

git commit

Practice

Let's run through the example of a GitHub profile README.

Personal access token

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

SG

What's this token for?

Expiration *

Custom... ⇕   2022-12-31 📅 ⇕

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☐ repo:status	Access commit status
☐ repo_deployment	Access deployment status
☐ public_repo	Access public repositories
☐ repo:invite	Access repository invitations
☐ security_events	Read and write security events

Tips I learned the hard way

- Don't forget to configure your email address on your local machine to match your github.com email address
- Don't add too much
  - `git add --all --dry-run` is very useful!
  - Consider a .gitignore file
- Don't commit too much
  - Get in the habit of using **atomic commits**
- Frequently use `git status`, read the output and carefully think about it. Does it make sense? Doing this can prevent mistakes that you will later need to reverse.
- It is a good idea to follow best practices for the kind of project you are working on (i.e., follow Python best practices, follow Jekyll best practices, follow Drupal best practices, etc.)

Resources

Library Carpentry: Introduction to Git

git - the simple guide

git/github guide

Oh Shit, Git!?!

Appendix: All the commands in the terminal

```
mkdir MY_USERNAME
cd MY_USERNAME
git init -b main
git remote add origin \
    https://github.com/MY_USERNAME/MY_USERNAME.git
(create README.md in the directory MY_USERNAME)
git add README.md
git commit -m "First commit"
git push -u origin main (you will need your personal access token)
```