Intro to Git and GitHub

sarah stanley
digital humanities librarian
scstanley@fsu.edu

Getting Started

key vocabulary

Version control - (also sometimes referred to as “versioning”) the management of changes to files, as well as management of the files themselves

Git - the *protocol* for versioning files

GitHub - the open source platform and social media site for sharing code and files (built on top of git). There are other platforms!

a scenario

You've written some code to scrape a bunch of data from HathiTrust. It's working ok, but you want to tweak a few things. Suddenly, your colleague/advisor/student walks into your office, so you save your work and close out of your code editor without testing out your changes. When you come back to test out your "optimized" code, you realize that—whatever you did during the last session—it's completely borked your code. You can't remember what changes you made

How can you get back your previous changes?

saving vs. versioning

Saving creates a snapshot of the document at the time you last saved it. It is very difficult to revert to previous versions of the files.

Using version control allows you to switch back to previous versions of the file, and document what types of changes were made during each stage. Creates revision metadata.

If you've ever created multiple versions of the same file with suffixes like "file_draft1.docx," "file_draft2.docx," "file_final.docx" but *far better*.

What We’re Doing Today

objectives

- Understanding git as a protocol for version control
  - Basic terms
  - Some command line practice
- Understanding GitHub
  - Interface for sharing code
  - GitHub for project management
  - GitHub as social media

how we'll get there

- DHBox: set up a trial account at http://dhbox.org/
- GitHub (we'll set up accounts in a bit)
- Some examples from Sarah

Creating Files and Directories

using the command line

- You can access this on your own computer too!
  - If you use a PC, the commands we're learning today are not quite the same as the ones you'd use
- Allows you to create and delete files and directories
  - But be careful, deletions are permanent
- There's a lot of other cool stuff you can do, but we won't get into that today

creating directories & files

- You use the `mkdir` command to create new directories (aka “folders”)
- The syntax for that is `mkdir directoryname`
- If you wanted to create a “test” directory, you would enter:
  - `mkdir test`
- To create a file, you use the `touch` command
- The syntax is similar to the `mkdir` command: `touch filename.ext`
  - `touch test.md`

changing directories

- You use the `cd` command to move from one directory to another
  - This is like going to your file manager and moving from “documents” to “downloads”
- `cd directoryname` moves you *down* to the child directory
- `cd ..` moves you *up* to the parent directory
- `cd ~` moves you to the root directory

some things to note

- Don’t put spaces in your directory or file names!
- When creating a file, try to append it with the proper extention
  - Today we’ll be using .txt (plaintext file) and .md (markdown file)
  - If you want to learn more about markdown, you should attend next week’s session, where we’re talking about how to build out your scholarly webpresence!

let’s try it out

- Create a “documents” directory
- Create a plaintext file called “test” within the documents directory
- Go back up a directory and create a “code-snippets” directory
- Create a “test.py” (Python) file within the “code-snippets” directory
- Create a “README.md” file in the “documents” directory
- Do the same in the “code-snippets” directory

Now Let’s Do Some Version Control

configuring git

In order to start using git, you will need to configure it with a user name and a user email (you only need to do this once).

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your@email.com"
```

initialize the repository

Navigate to the code-snippets repository and type `git status`

You will be returned a message that says “not a git repository” (or something similar). This means that the directory you’re working in does not have any .git information associated with it. To remedy this, type:

```
git init
```

adding files

Type `git status` again. This time you will get a message that you have 2 unstaged files. Any files that you want to commit, you will need to *add*. (This allows you to commit one file in a repository, but not the others, or to completely ignore files that you don't want in the git ecosystem). To add all files, type

```
git add .
```

To add only one file, type the name of the file you want to add.

```
git add filename.ext
```

committing

Committing files will allow you to keep a record of what the file looked like at the time of commit (and it will allow you to revert to that version). **It is important to commit with succinct, but detailed messages so you know what changes were made after that commit.**

committing, part ii

To commit files, enter the following into the terminal:

```
git commit -m "your message goes in quotes"
```

You should then get a message that says the *extent* of the changes to the files (i.e. number of files/lines added deleted and the total overall change).

some additional exercises

- Type “Hello World!” into your README.md file in code-snippets
- Create a new markdown file in the code-snippets directory called “test”
- Type a few sentences into the test file
- Now commit your changes to the README.md file without committing the new test file
- Now add “test.md” to the repository and commit

log

To look at the histories of the files you have committed you can use

```
git log
```

remember

If you are using git to commit files on the command line, you are *only* committing them locally on your machine. If you want to *share* your files on a platform like GitHub, you will need to take a few extra steps.

git push

We are not going to be going into the details of how to initialize a GitHub repository using the command line today, but I still want to show you what it looks like to commit files and then have them update on GitHub.

```
** What follows is a short demonstration on pushing
                    files to GitHub **
```

a note on branches

- Branches are a super useful way of maintaining code when multiple people need to work on separate issues within the same files.
- Branching allows you to create and edit a version of the files which you can then merge back in with other branches.
- If this confuses you, DON'T WORRY (it's next-level stuff!)
- You will often see instructions to specify branch when pushing to GitHub
- i.e. `git push origin master` (push to the "master" branch)

GitHub

get thee an account

Go on over to github.com and sign up for an account. Best if your username is just alphabetic characters (I have a number at the end of mine, which has never caused me any problems, personally, but I've met people who have said they had issues ¯_(ツ)_/¯)

create new repository

- Create a name for your repository
  - Again, no spaces! Try to use only alphabetic characters and hyphens/underscores
- Public repositories are free; private ones cost $$
  - There are other options for GitHub-like services that give your private repos for free (like Atlassian's BitBucket)
- **Initialize this repository with README!!!**

github in the browser

- Edit README.md and committing the changes
- Add a new markdown file called “about-me” and another one called “cv” and commit the changes
- Edit “cv.md” to include a header with your name and a section some educational information. Commit the changes.
    - Want to learn more about markdown? Come to one of next week’s sessions!

cloning repositories

- We've learned about two very different ways of using git: one with a browser, the other from the command line
  - How do we synch between the web and our own computers?
- To have a GitHub repository on our own machines, we need to use a process called "cloning"
  - This puts a copy of the directory somewhere in your file system (wherever you choose) with the associated git information

a few options

- How do we get information held locally to GitHub?
- Using git commands + push from the command line as shown earlier
- Using the GitHub Desktop client
  - ** Here’s a short demonstration of what that looks like **

some review

- *git* is a protocol for version control that allows you to *commit* changes (essentially create a snapshot of what the file looked like at a given time)
- You can provide *commit messages* that allow you to document how a file changes over time
- GitHub is a code-sharing platform that runs on git
- git is something that you can manage locally, and then *push* to GitHub to share with the world!

Questions?

Thank you!

if you have any questions, send me an email at scstanley {at} fsu {dot} edu