GitHub Basics For Research Teams

Ava Braiter
ab2493b@american.edu

Table of Contents

To Do Before Starting	Creating a Free GitHub Account and Downloading Git
Introduction	What are Git and GitHub? How is it useful?
GitHub Website	Taking a look at github.com
Bash and Repo Basics	Learning simple Bash commands and navigating a repository
Core Git Workflow	Learning about Git's workflow: edit, add, stage, commit, push, pull, repeat
Collaborating with Others	How can we interact with others on the same repository?
Git and GitHub History	How can we see changes on Git and GitHub?
Git Cheat Sheet	The basic commands and core info all in one place

01

To Do Before Starting

Setting up Git and GitHub

Install RStudio

- R is a statistical software that we will use for this tutorial
- If you do not have R or RStudio installed, install them using these steps:
- Install R with this link
- Install RStudio with this link

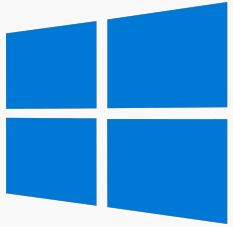


Installing Bash and Git

- Bash is a type of shell, which is a program that takes commands from your keyboard and sends them to your operating system
- Bash is a shell that allows you to run Git, which I will explain later
- Mac devices already have a shell (zsh or bash) that can run Git
- Windows users will need to install Git



Windows: Installing Git



- Go to this link and download the “Standalone Installer” version of Git that is applicable to your device (likely Windows/x64 Setup)
- Run the .exe file and go through the installation wizard steps

- Verify Git is installed by opening “Git Bash” on your computer
- Git Bash will be referred to as the terminal from now on
- Run this command:
  - git --version
- If something similar to the following is returned, Git is successfully installed
  - git version 2.x.x

Mac: Verifying Git is Installed



- Git should automatically be on your computer
- Verify that it is by opening the “Terminal” application and running this command:
  - git --version
- If something similar to the following is returned, Git is already installed
  - git version 2.x.x
- If git --version does not work, go to this link and install homebrew
- Once you have installed homebrew, run this command in terminal
  - brew install git
- After Git has been installed, try running “git --version” again

Creating a GitHub Account



- I will explain what GitHub is in the presentation
- Go to https://github.com/ and create a free account
- Connect your GitHub account to Git by opening your terminal and typing:
  - git config --global user.name “YOUR FULL NAME”
    - This should be your name and not your GitHub username
  - git config --global user.email “YOUR EMAIL ADDRESS”

Connecting Git and GitHub

- We will use HTML authentication to connect Git and GitHub
- We will use Rstudio to create a personal access token
- Open RStudio and run this command
  - usethis::create_github_token()
- If you do not have the usethis package installed, run install.packages("usethis")
- A GitHub browser will open
- Change the note to the device being used
  - Ex: "Personal Macbook"
- Change the expiration date to 6-12 months from today

Now, Git and GitHub are Set Up!

You may move onto the tutorial now!



02

Introduction

What are Git and GitHub and how can we use it?

Git vs. GitHub



- Tracks changes to files
- Saves versions of files with commits
- Runs on your computer



- Stores Git repositories online
- Lets people collaborate and share changes
- Can host public or private repos

Git tracks changes to projects, while GitHub lets people share and collaborate on it.

Why Git and GitHub are Useful for Research

- Track changes to analysis code
- Collaborate without overwriting other's work
- See who changed what, when, and why
- Restore earlier versions of files if something goes wrong
- Keep one shared version of the file
- Improve reproducibility with documentation

GitHub vs. OneDrive



- Tracks line-by-line changes
- R, Python, codebooks, interview guides, protocols, etc.
- Better at project documentation, especially across file versions



- Best for Word, Excel, PowerPoint, and PDFs
- Good at live co-editing
- Syncs files across teams
- Better for general document storage

GitHub is better for tracking changes and reproducible project workflow; OneDrive is better for shared documents and files

03

GitHub Website

The basics of github.com

What to Look for on GitHub

1. Repositories
2. Files and Folders
3. README
4. Code Button/HTTPS Cloning
5. Commit/File History
6. Collaborators

- Screenshots of each are in the appendix



04

Navigating a Repo on Local

Using Bash or zsh to navigate a repository

Working Directory

- Working directory: the folder your terminal is currently using
- Commands that you run on the terminal act on the current folder you are on
- Git commands should be run inside the repository you are using
- Think about the folders on your computer like this:

```
Documents/
└── github-training/   ← working directory
    ├── analysis/
    ├── data/
    └── README.md
```

- The github-training repository is located in the “Documents” folder on this computer
- Since we are working on the github-training repository, the working directory should be set there.

So, before using Git, **know what folder you are in**

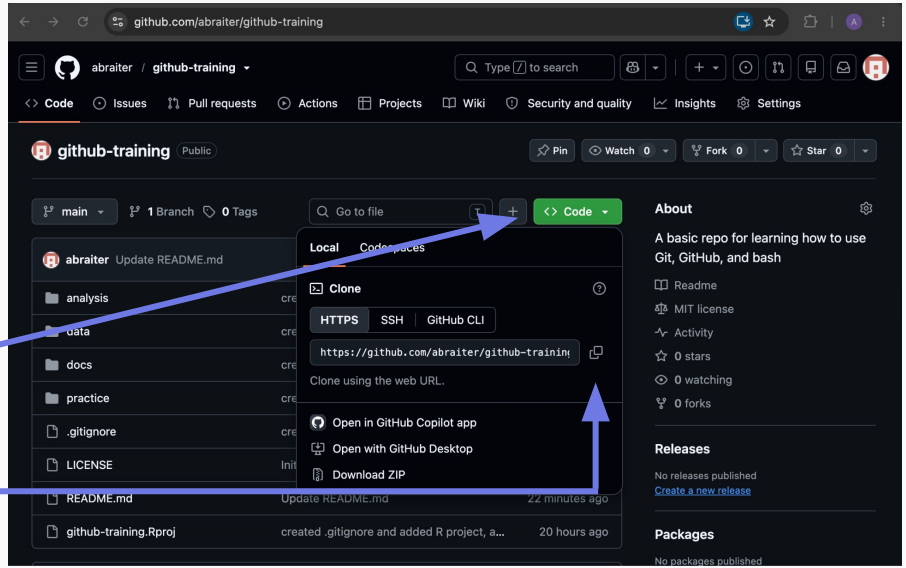
Useful Commands

Command	Meaning
pwd	Print working directory
cd <folder name>	Move into a folder
cd ..	Move up one folder
ls	List files and folders
mkdir	Create a folder
rmdir	Remove a folder

- These commands allow us to navigate folders using the terminal
- There are many other useful commands you can look up
  - here is one resource
- Let's practice!
  - Open the terminal and use these commands to navigate to a folder where you will store the github-training repository!
  - Screenshots will be in the appendix

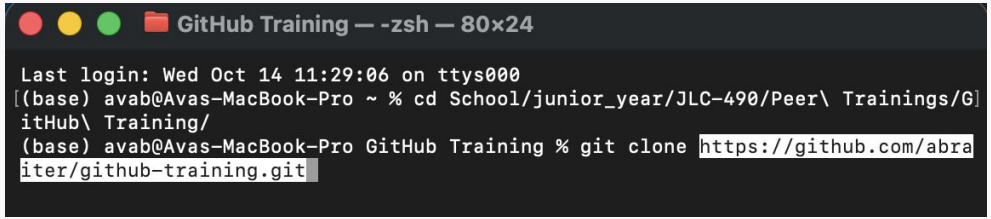
Cloning a Repository

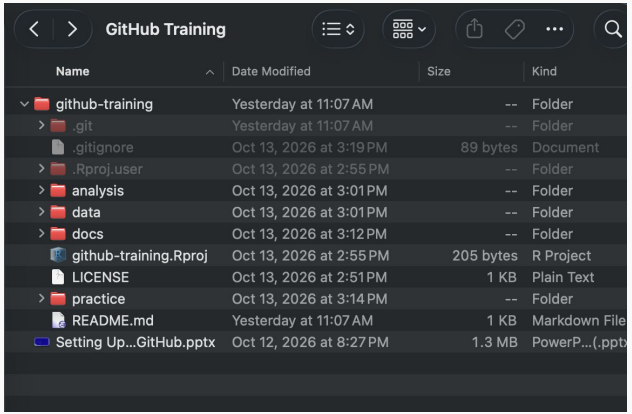
- For this workshop, we will use a repository that I have created
- Go to this link:
  - https://github.com/abraiter/github-training
- Click on “<> Code”
- Copy the HTTPS path



Cloning a Repository (2)

- In the terminal, navigate to the working directory where you want to place this repo
  - use the commands we just learned
- Go to your terminal and run:
  - git clone <HTTPS path>
- Paste the link you just copied for the HTTPS path
- The repo is now on your local device!





05

Core Git Workflow

Staging, Committing,
Pushing, Pulling

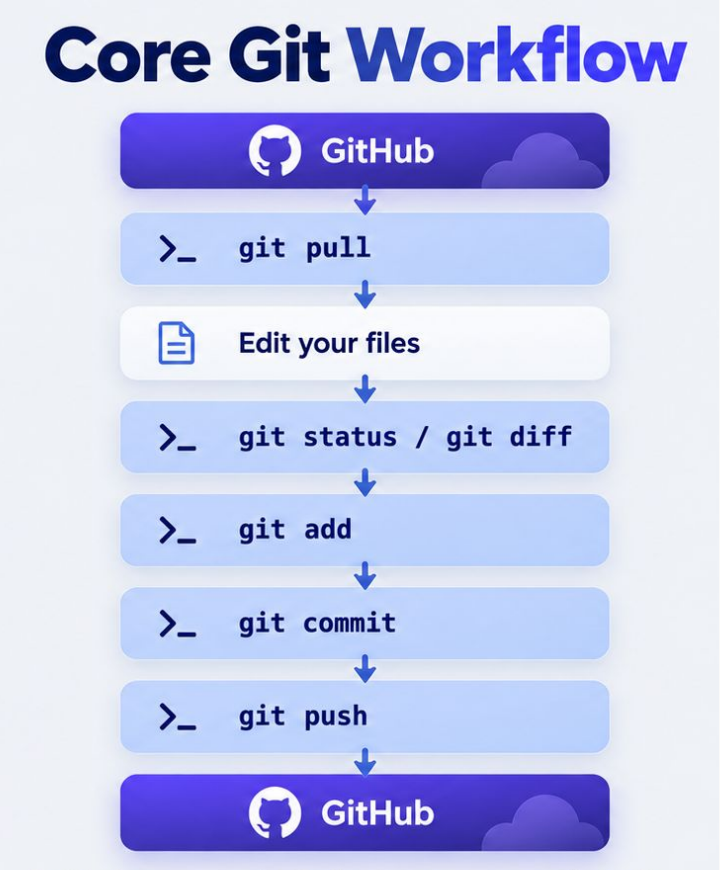
Git Workflow

- Six main steps allow you to pull changes from GitHub and push your changes to GitHub
  - Pull
  - Edit
  - Review
  - Add/Stage
  - Commit
  - Push

- Pull: gets the newest changes from GitHub onto your local
- Edit: make changes to files
- Review: see what you have changed
- Add/Stage: Choose what files will go into the next version of the project
- Commit: save a new version of the project locally
- Push: send your commits to GitHub so your team can see them

Git Workflow

- Here are the six steps again
  1. Pull
  2. Edit
  3. Review
  4. Add/Stage
  5. Commit
  6. Push
- Try them now by following screenshots in the appendix!



06

Collaborating on a Repository

NOTE: This will only work with the github-training repo during the live session; if you are viewing this after the session, you may need to create your own repository to practice. The repository just needs a .txt file

Collaborating on a Repo

- I will add you as a collaborator to the GitHub repository
- With this, you will be able to push to a GitHub repo without the owner's approval
    - (Look into forks if you cannot be a collaborator on someone's repo)

- Now, let's work on the skills we learned together!



Going Through Git Workflow

- Now, everyone should be a collaborator and have the repo on their local
- I will assign everyone a number
- Let's go through the Git Workflow together:

1. git pull
2. Go into practice.txt and add your name to the line that corresponds with your number. Save your changes.
3. Stage your edits with git add practice/practice.txt
4. Commit your edits with git commit -m "added name"
5. We will push the edits one person at a time

Merge Conflicts

- A merge conflict happens if two people change the same part of a file
- Git reports the conflict and has the user decide what to keep
- There are various merge conflicts and different ways to fix them
  - Since there are so many, I will not go through all of them

07

Project History

Looking at past commits on Git and GitHub

Commit History on Local

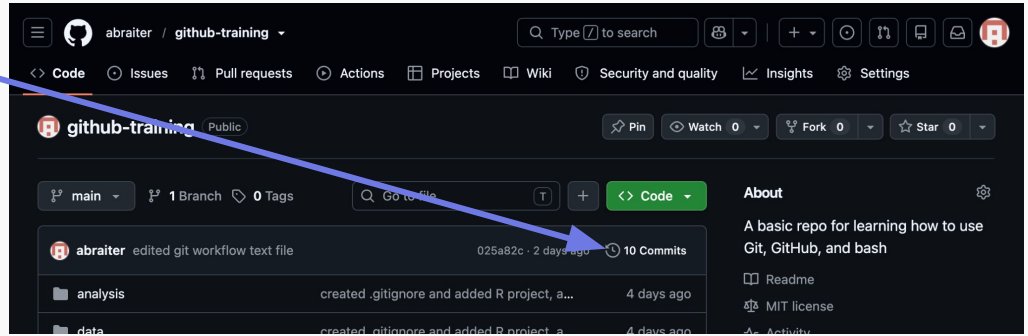
- Look at previous commits
  - git log --oneline
  - Each line is a commit in order
  - The commit ID is on the left
  - The commit message is on the right

```
(base) avab@Avas-MacBook-Pro github-training % git log --oneline
025a82c (HEAD -> main, origin/main, origin/HEAD) edited git workflow text file
b5752ca edited git_workflow text file git status f
d269600 update readme Merge branch 'main' of https://github.com/abraiter/github-
training
80280e2 edited readme
102a85e Create git_workflow.txt
8bf8d28 edited readme on local
166de99 Update README.md
2709414 edited README file
ce4ca34 created .gitignore and added R project, analysis, fake csv file, documen
tation, and practice.txt
eb7d7f8 Initial commit
(base) avab@Avas-MacBook-Pro github-training %
```

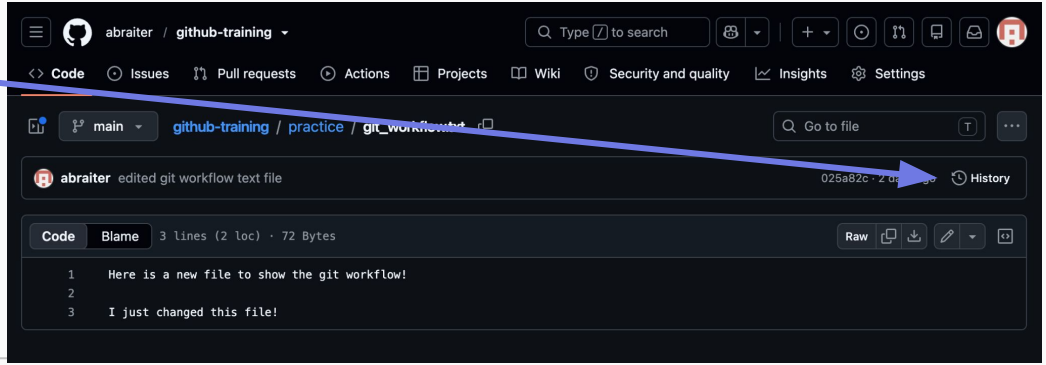
Repository History on GitHub

- Repository commit history
  - View all commits
  - You can click into each commit and view specific changes
- File-specific history
  - View line-by-line changes

Repo Commit History



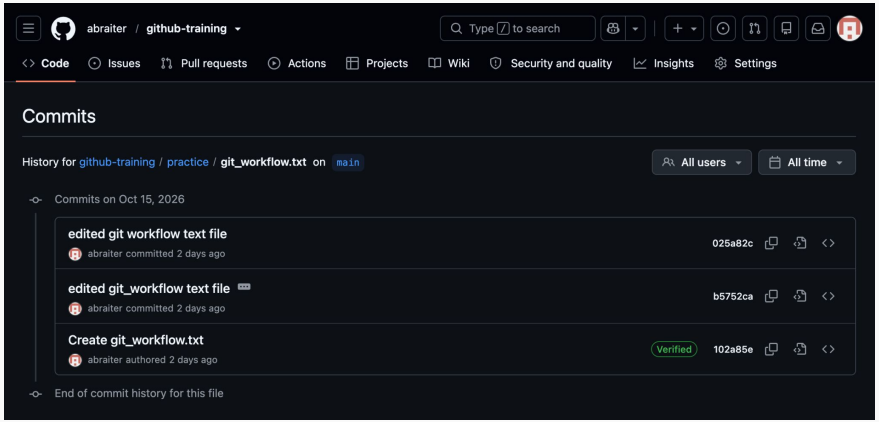
File-Specific Commit History

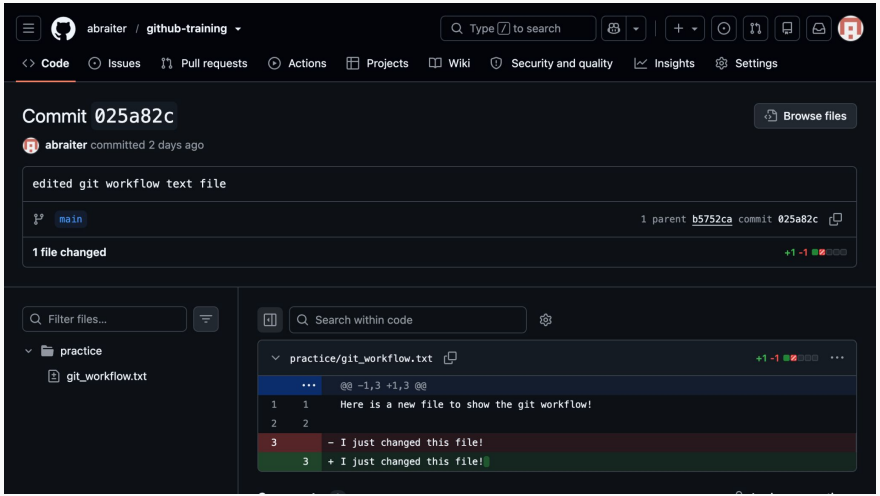


GitHub File Version History

- Opening an old version of a file

Comparing changes between commits





08 Wrap-Up & Cheat Sheet

Key Takeaways

- Git tracks changes on your computer
- GitHub helps team collaborate on projects
- Core Git Workflow:
  1. Pull
  2. Edit
  3. Review
  4. Add
  5. Commit
  6. Push
- Use git status to see what files have been staged
- Git history lets you see past changes and recover earlier versions
- For team projects, pull before starting and push when ready to share
- You don’t need to memorize everything, use this slide deck and the cheat sheet →

Cheat Sheet (1)

Terminal Commands

Command	Meaning
pwd	Print working directory
cd <folder name>	Move into a folder
cd ..	Move up one folder
ls	List files and folders
mkdir	Create a folder
rmdir	Remove a folder

- Clone a repository in the terminal using
  - git clone <repo-url>

Cheat Sheet (2)

Core Git Workflow

Step	Command
Pull	git pull
Edit	Edit your files
Review	git diff
Stage	git add <file name>
Commit	git commit -m “describe edits”
Push	git push

- Useful hints:
  - Use git status to check what files have been staged for commit
- Project History:
  - git log --oneline
    - View commit history
  - git show <commit-id>
    - View changes from a commit
- You can also use GitHub to view project history and who made what edits

Resources

Resources/References

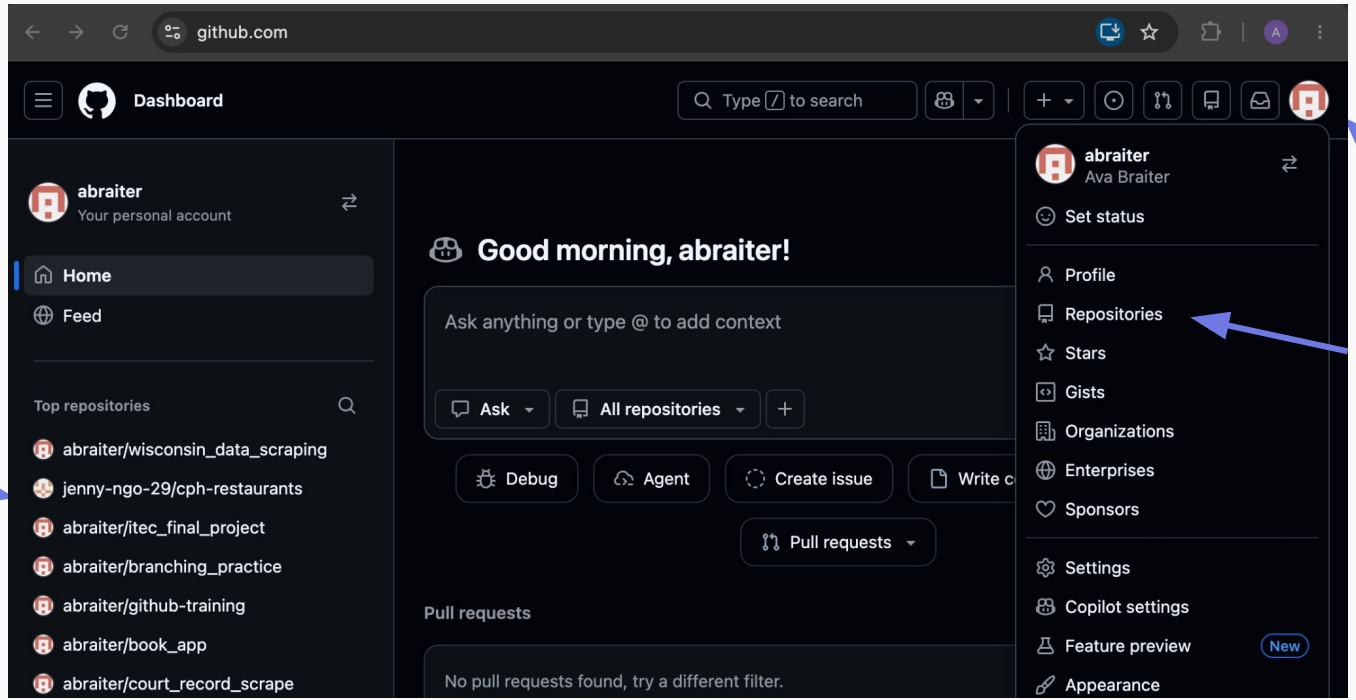
- https://usethis.r-lib.org/articles/git-credentials.html
- https://cran.r-project.org/
- https://docs.posit.co/ide/user/#rstudio-ide-oss-downloads
- https://git-scm.com/install/
- https://github.com/
- https://docs.github.com/en/get-started/using-git
- https://docs.github.com/en/get-started/using-github/hello-world

Appendix

Appendix: GitHub Website

GitHub: Repositories

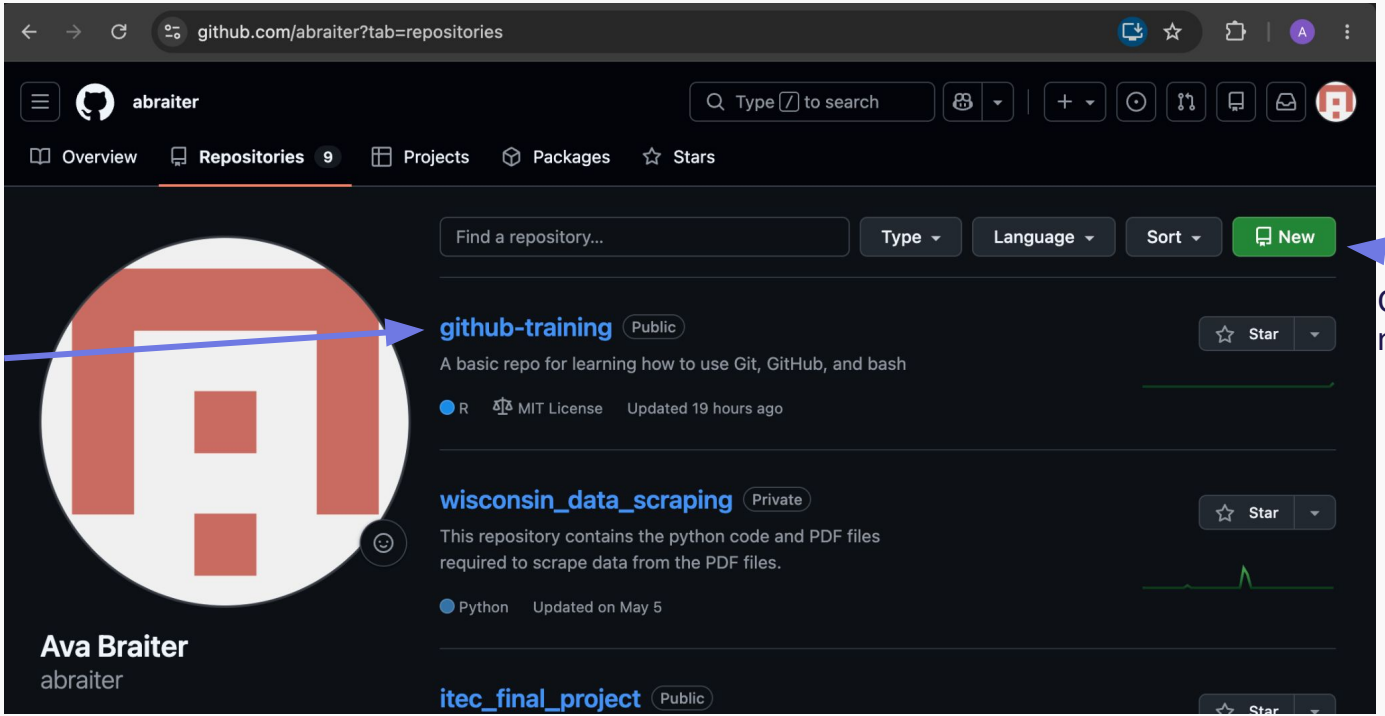
On the homepage, top repositories are located on the left side



Alternatively, click on your profile in the top right, and click on “Repositories”

GitHub: Repositories

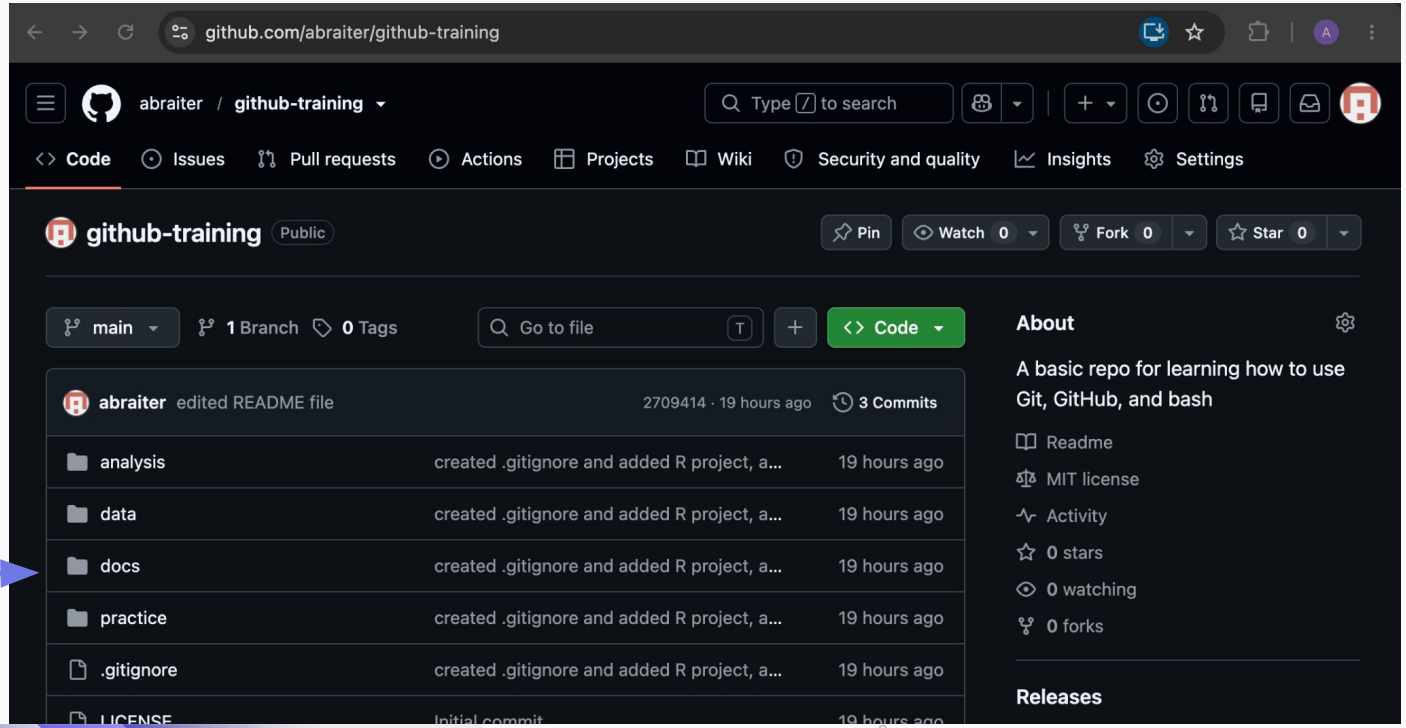
Here, all your repositories are listed. Click on a repository you want to look at



Or, create a new repository

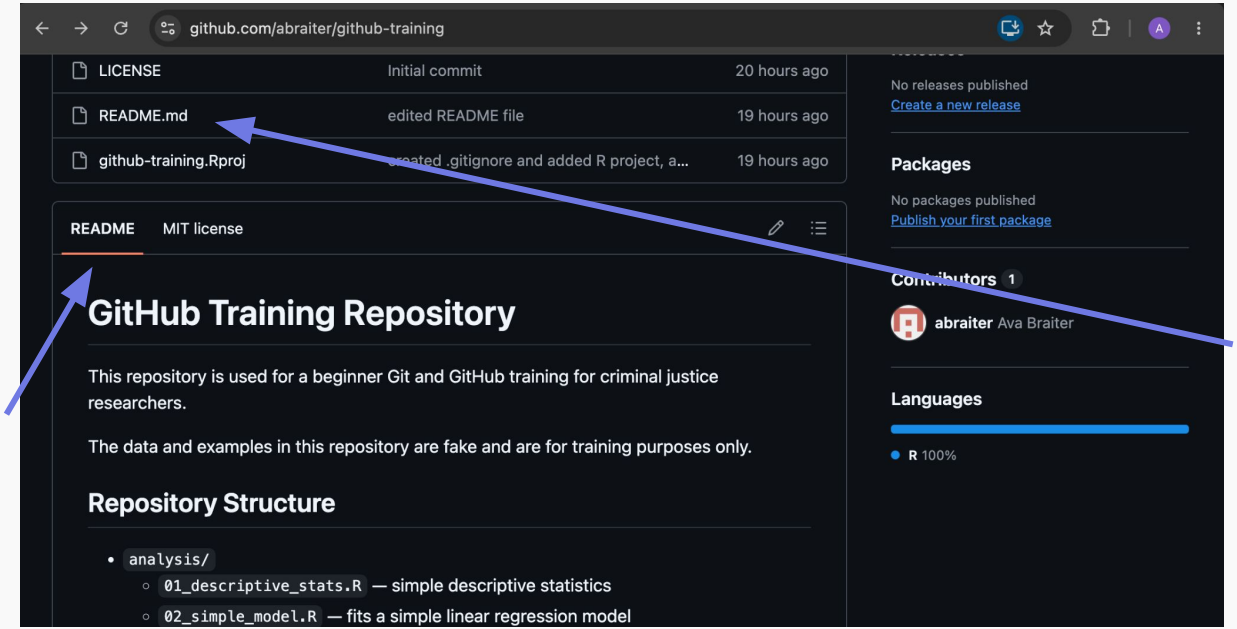
GitHub: Files and Folders

This is the main page of a repository. Here, you can view different files and folders



GitHub: README

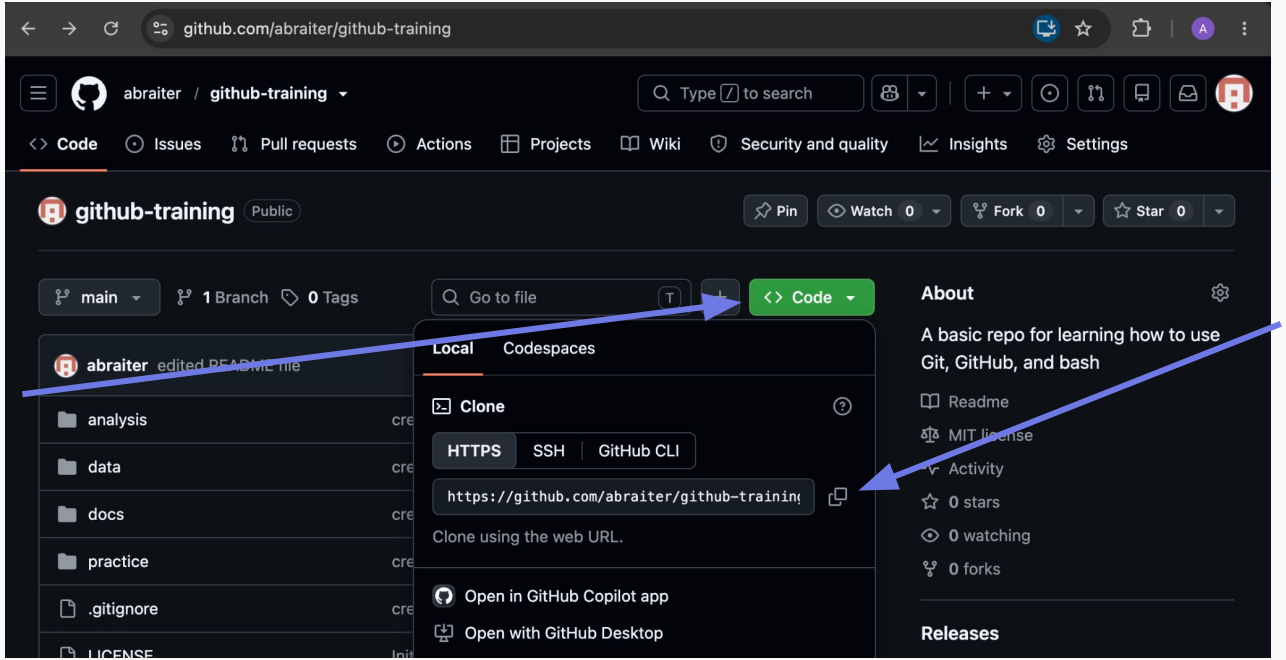
Most Repositories have a README file. This provides users with information about the project.



It can be found by scrolling to the bottom of the repo or by clicking on the README.md file

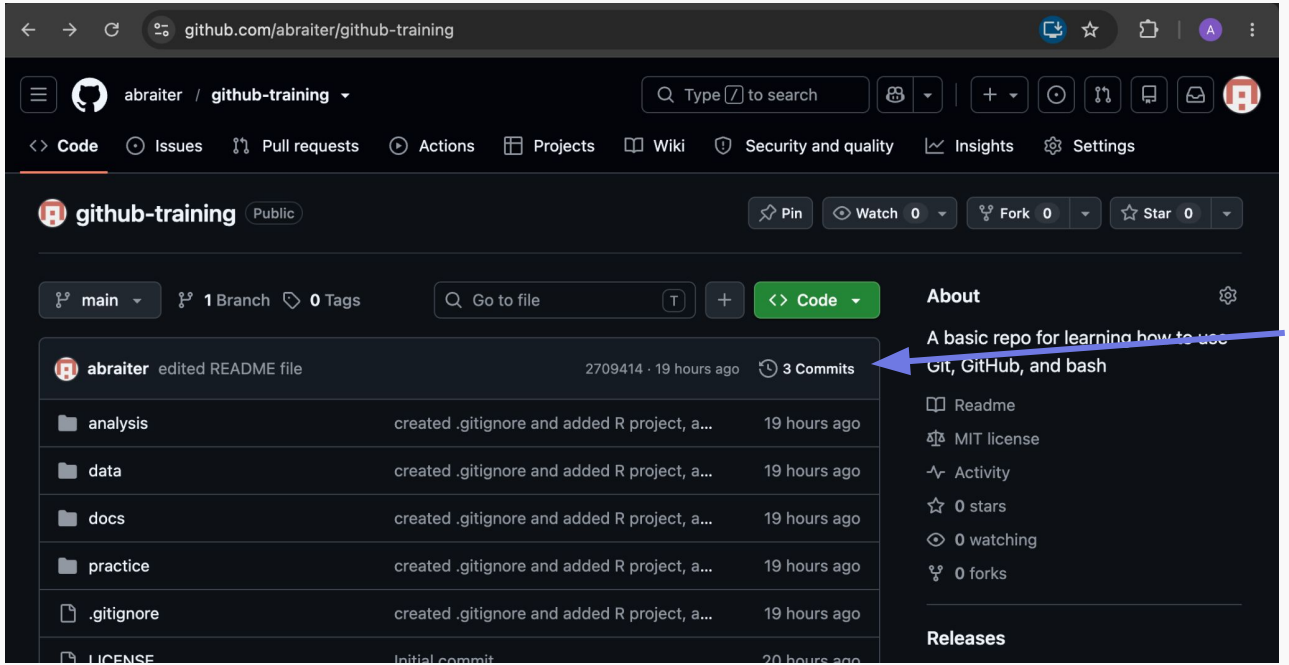
GitHub: Code Button and Cloning

Scroll back to the top of the repo. Here, you can find the code button, which allows you to clone the repo to your local device



Click the code button and copy the HTTPS link for the repo. You can run “git clone <REPO LINK> to clone the repo to your local device

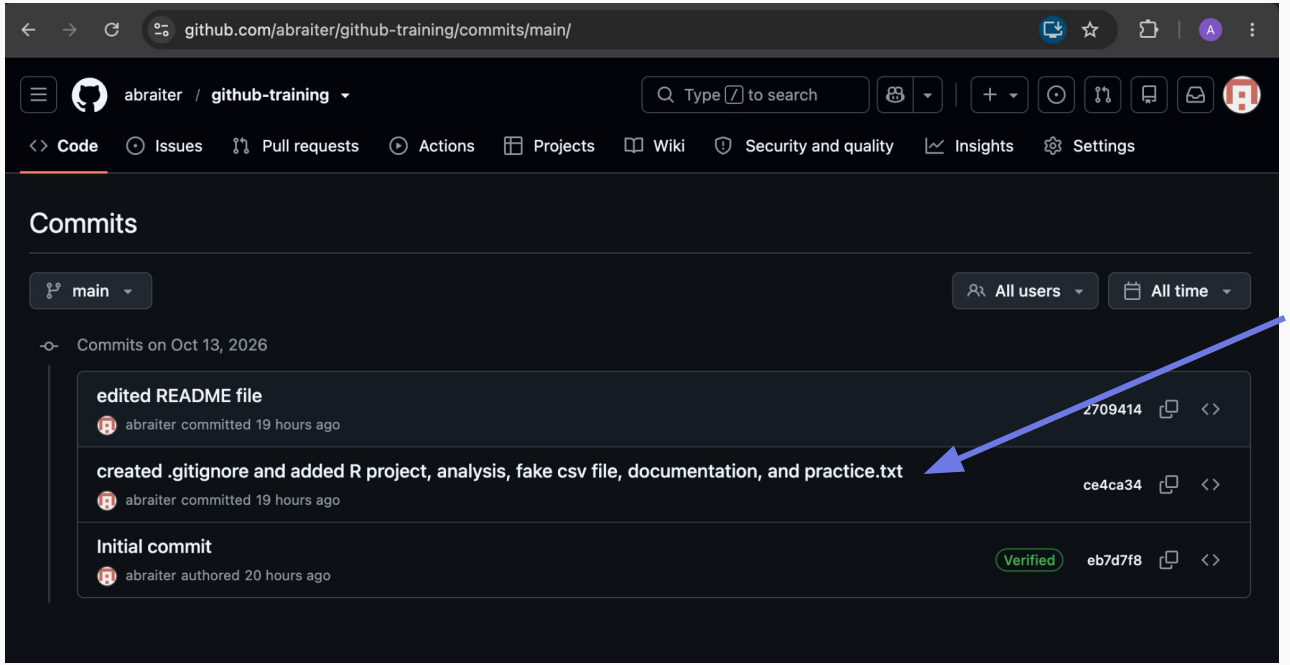
GitHub: Commit and File History



Click on "Commits" to view the commits or version history of the repo

GitHub: Commit and File History

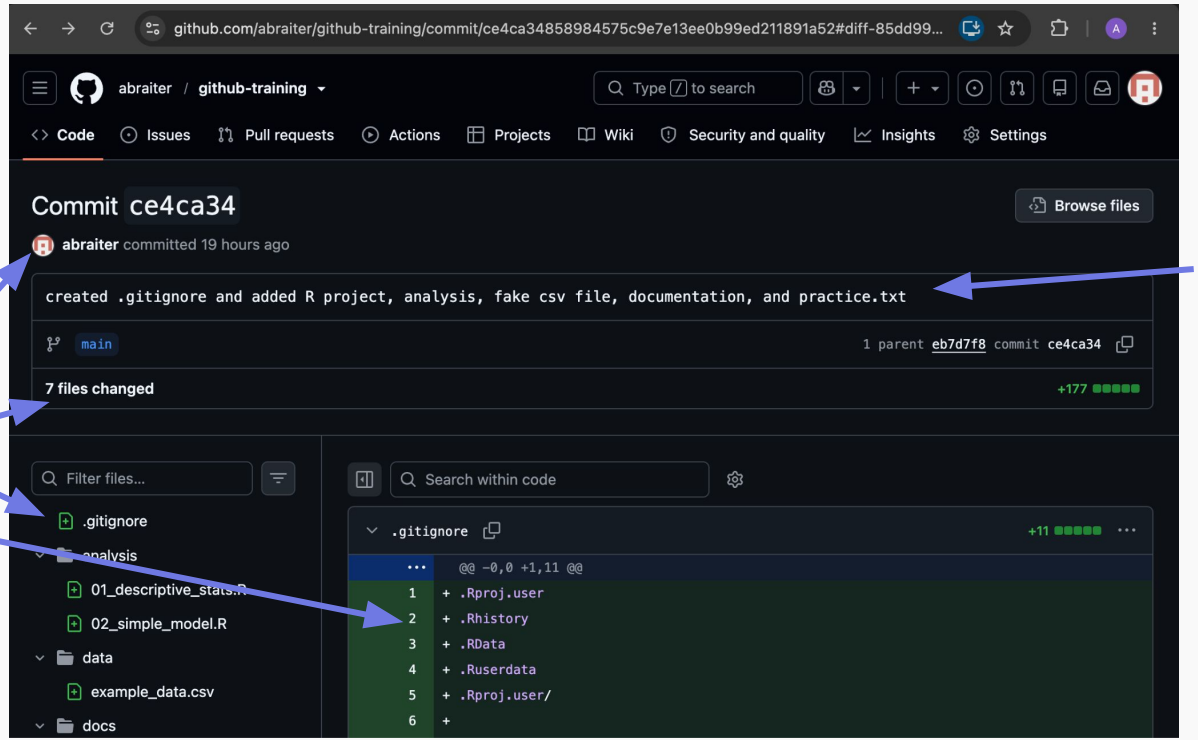
Here, you can see a history of all of the commits or versions of the repository.



Click on a specific commit to see what changed and who changed it

GitHub: Commit and File History

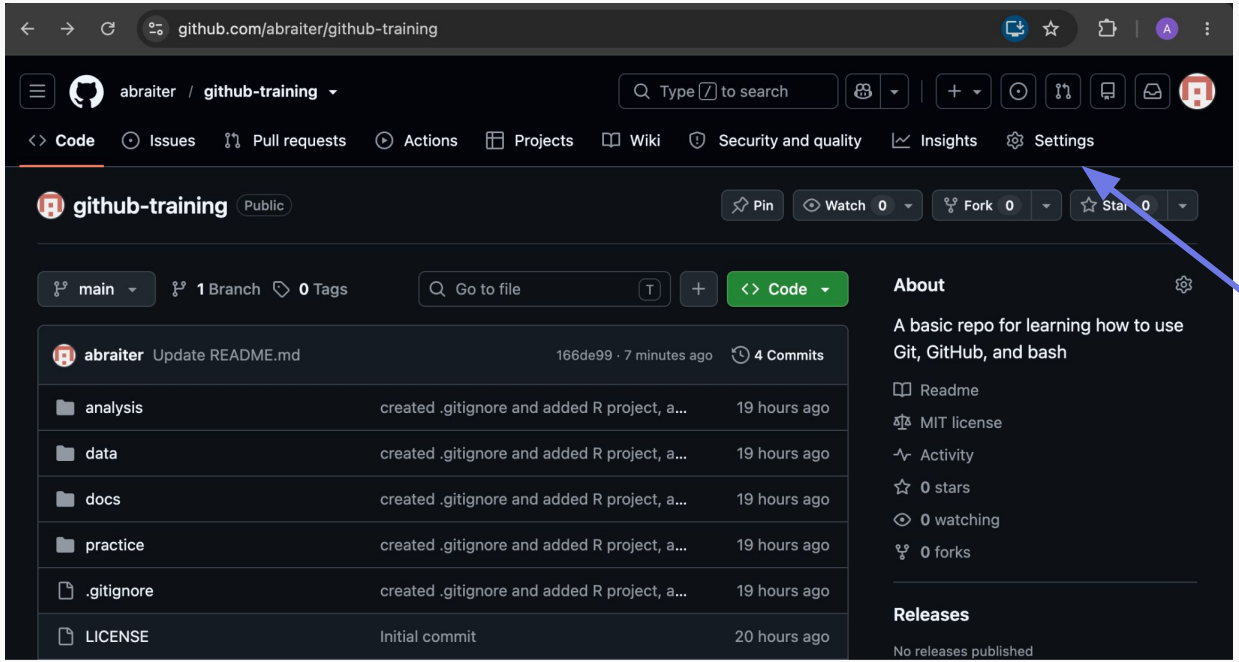
Here is a specific commit. You can see whose commit it is, what files they changed, and what lines in each file were changed.



Documentation for each commit is at the top

GitHub: Collaborators

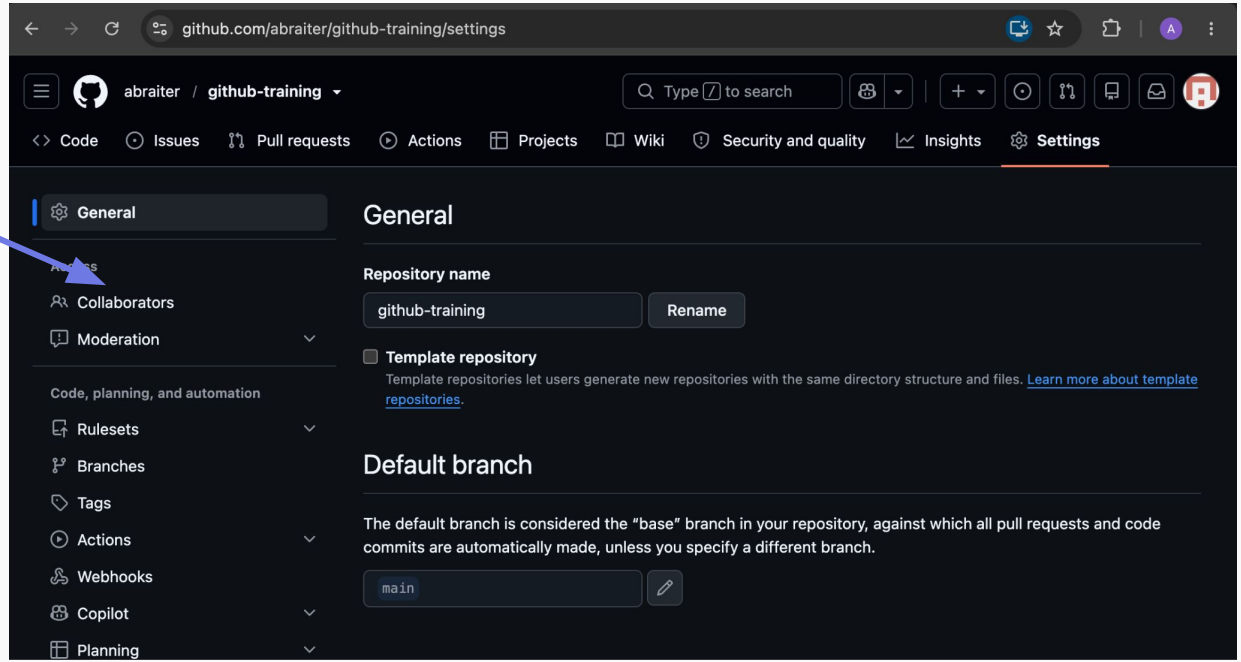
Collaborators are able to work directly on a shared repository and push changes back to GitHub



If you are the owner of a repo, you may add collaborators by going to the “Settings” tab.

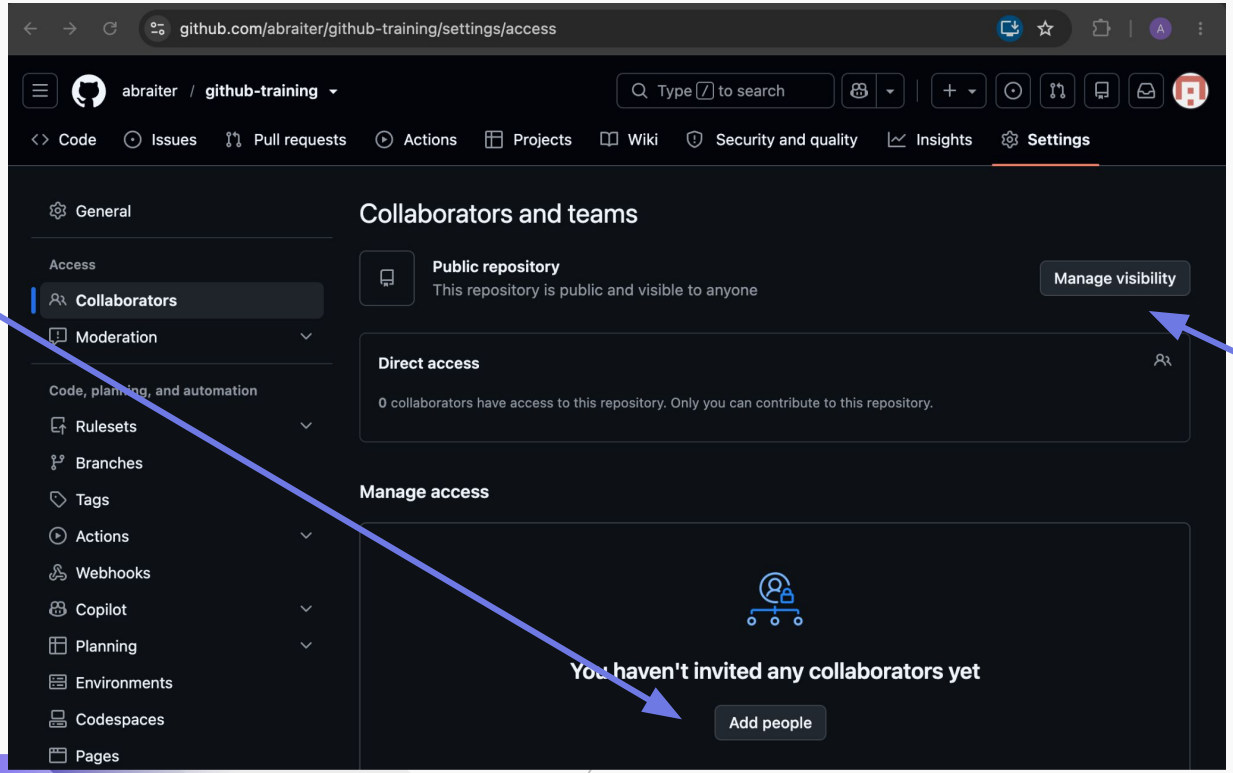
GitHub: Collaborators

Then, click on “Collaborators”



GitHub: Collaborators

Here, you may invite collaborators

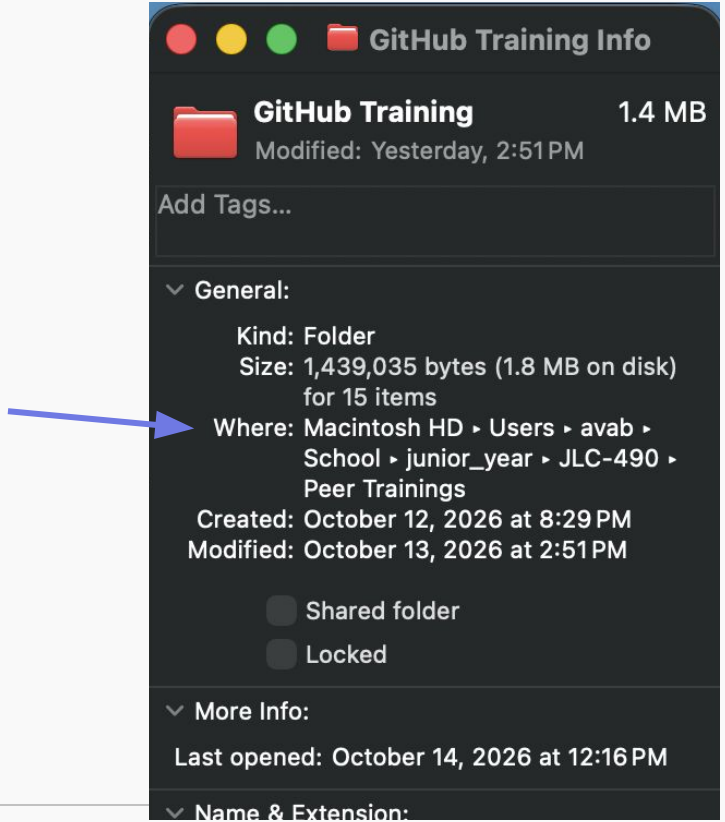


You can also manage the visibility of your repository (public vs. private)

Appendix: Navigating the Terminal

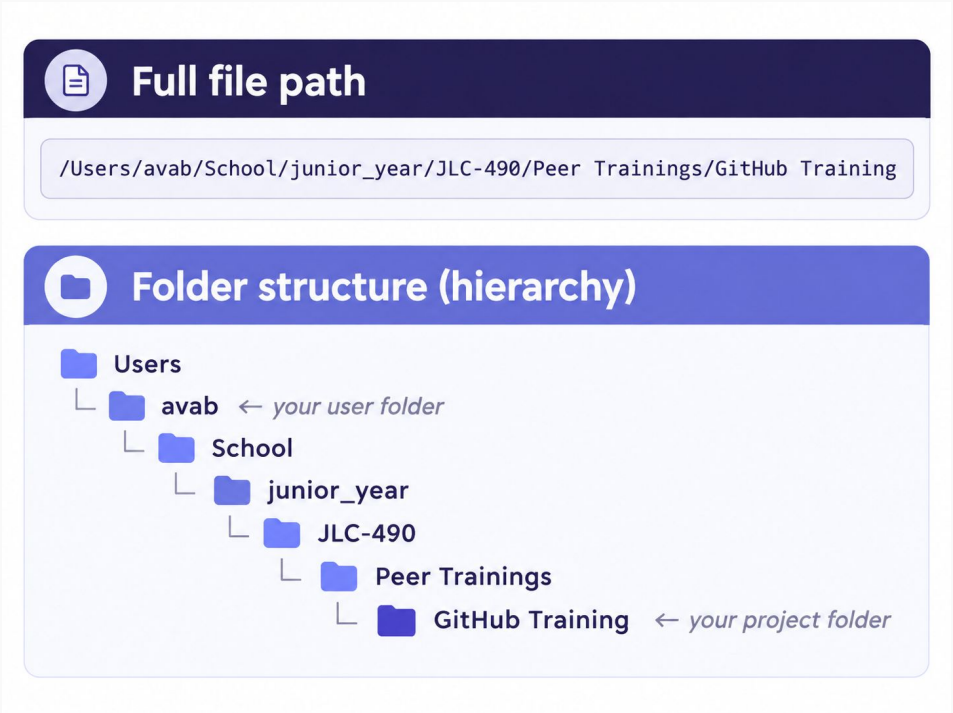
Choose a Location

- On your Finder/File Explorer, locate the folder where you want to store the github-training repo.
- Find the file path for that folder. To do that:
  - Right click on the folder
  - Select “Get Info”/”Properties”
  - Find where it says “Where”/”Location”
  - That is the file path



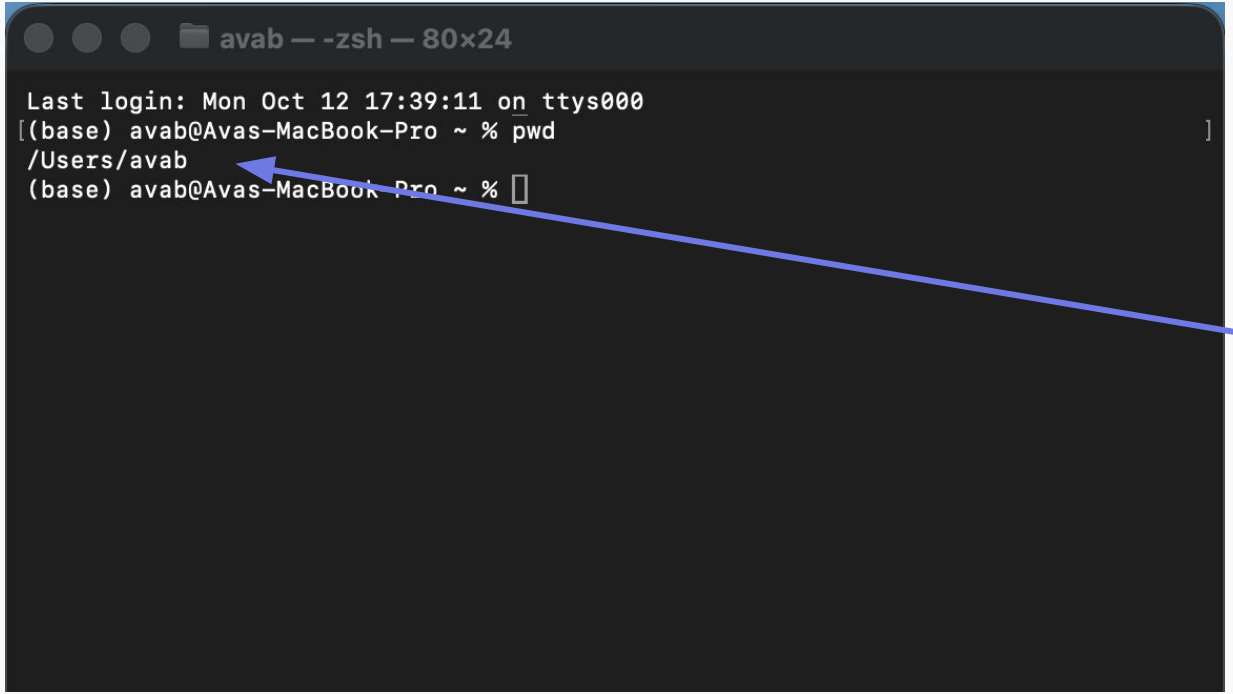
Choose a Location (2)

- Now, we will work our way through different directories to get to our destination. This is the path that I will take based on my file structure
  - So, the first folder I will navigate to is “School”
  - Then, I will navigate to junior_year



Print Working Directory: pwd

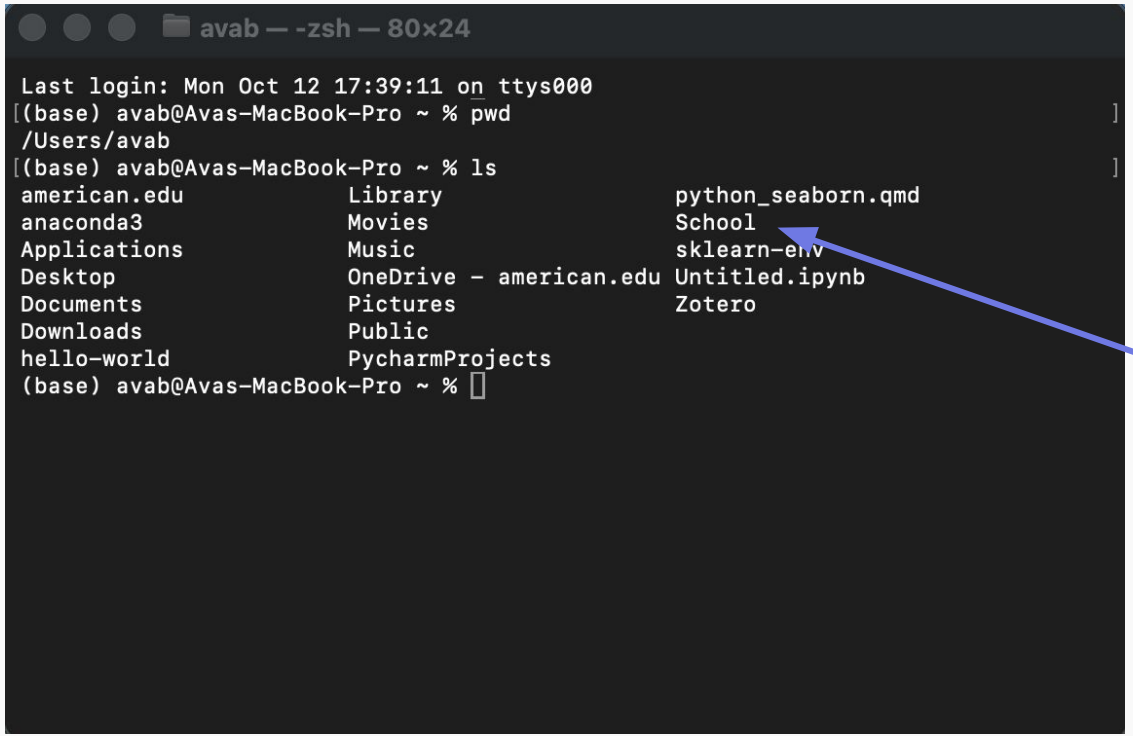
Open the terminal and type “pwd” and press enter



I am currently in the Users/avab folder on my computer

List Folders: ls

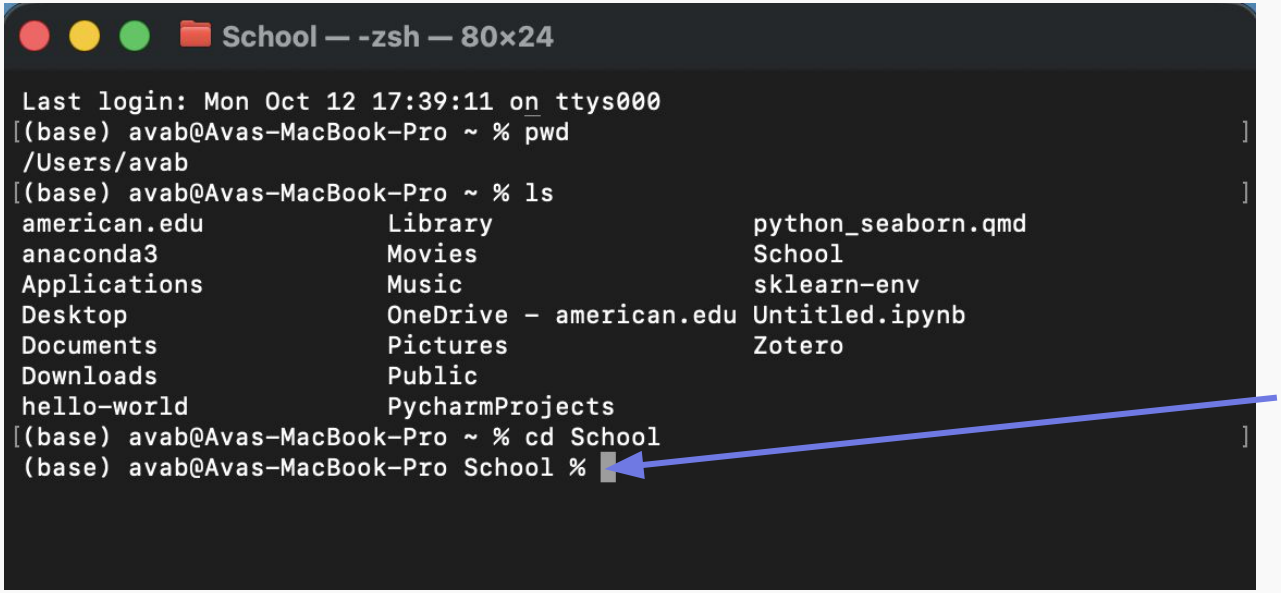
Run the command “ls” in your terminal. All of the folders and files under/below your current working directory will be listed



Remember the file path from earlier? Find the next folder down that we want to go into. For me, it is the “School” folder. You will use this in the next step

Change Directory: cd

To navigate to the next folder down, type “cd <name of next folder> and press Return/Enter



We can now see that I am in the “School” folder

Change Directory: cd (2)

- You can either repeat the process on the previous slide, but there a few shortcuts to improve the process:
- Shortcut 1:
  - Type “cd” and then press tab to get a list of all folders that are one level down from the working directory
  - Press tab to cycle through the folders

```
(base) avab@Avas-MacBook-Pro ~ % cd School
(base) avab@Avas-MacBook-Pro School % cd
APO/              OTAC_Desktop/     sophomore_year/
freshman_year/    personal/         study_abroad/
junior_year/      senior_year/      summer_courses/
```

- Shortcut 2:
  - If you know the file path, you can type it out, separating the folders with “/”
  - ex: School/junior_year/
  - This goes down two levels at a time
- Shortcut 3:
  - you can combine shortcuts 1 and 2 to quickly navigate to folders

```
(base) avab@Avas-MacBook-Pro ~ % cd School
(base) avab@Avas-MacBook-Pro School % cd junior_year/JLC-490/Peer\ Trainings/GitHub\ Training
(base) avab@Avas-MacBook-Pro GitHub Training %
```

Change Directory: cd .. (3)

- Made a mistake? Move up a level by using the command: “cd ..”
- Other Notes
  - Folders with a space in it use “\”
  - For these folders, use the tab method to navigate through them

```
(base) avab@Avas-MacBook-Pro ~ % cd School
(base) avab@Avas-MacBook-Pro School % cd junior_year/JLC-490/Peer\ Trainings/Git
Hub\ Training
(base) avab@Avas-MacBook-Pro GitHub Training % cd ..
(base) avab@Avas-MacBook-Pro Peer Trainings %
```

Make and remove a Folder: mkdir & rmdir

We can make a folder by using the command “mkdir <FOLDER NAME>”

If you look at your Finder/Files Explorer, you will see a new folder has appeared

```
(base) avab@Avas-MacBook-Pro GitHub Training % ls
github-training                 Setting Up Git and GitHub.pptx
(base) avab@Avas-MacBook-Pro GitHub Training % mkdir fake_folder
(base) avab@Avas-MacBook-Pro GitHub Training % ls
fake_folder                     Setting Up Git and GitHub.pptx
github-training
(base) avab@Avas-MacBook-Pro GitHub Training % rmdir fake_folder
(base) avab@Avas-MacBook-Pro GitHub Training % ls
github-training                 Setting Up Git and GitHub.pptx
(base) avab@Avas-MacBook-Pro GitHub Training %
```

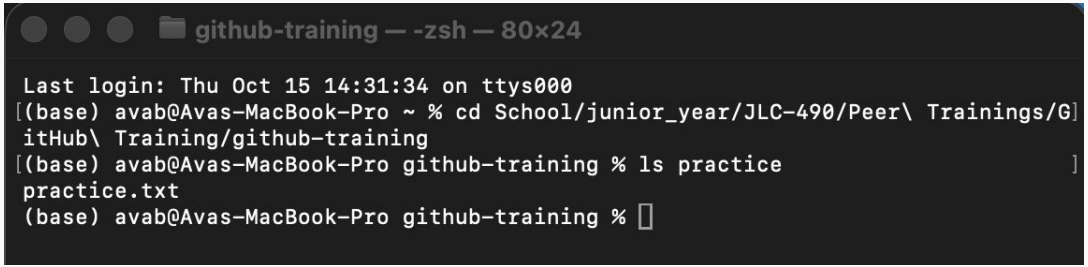
We don’t want this folder, so remove it with “rmdir <FOLDER NAME>”

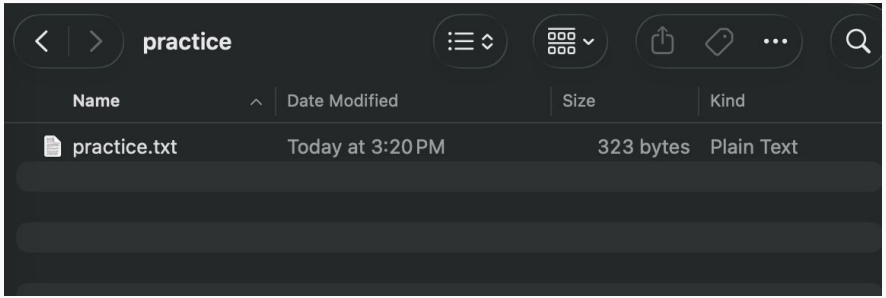
Be careful, this will delete a folder and all files inside of it, and can **not** be reversed

Appendix: Core Git Workflow

Pulling Changes from GitHub

Here, we see that I am in the github-training repo on my local.

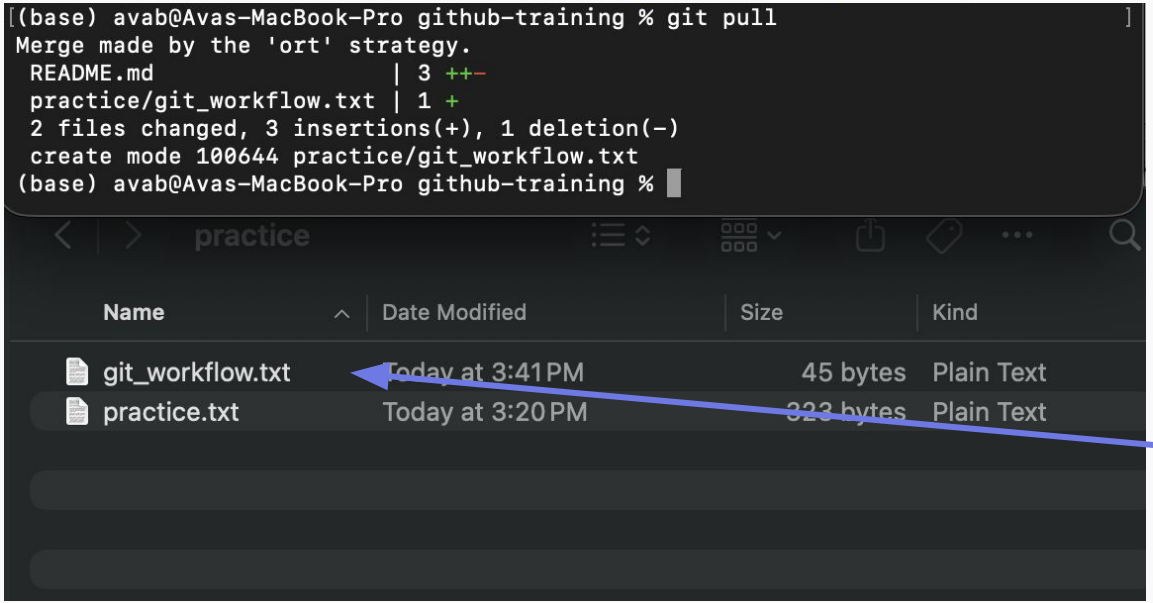




We also see that there is only one file in the “practice” folder.

Pulling Changes from GitHub (2)

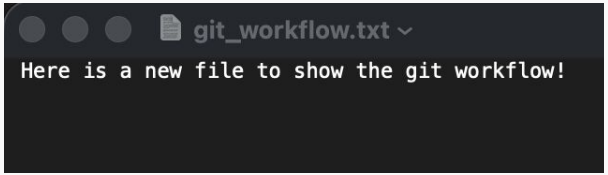
A new file was added to the GitHub, so I pulled the updated remote repo to my local



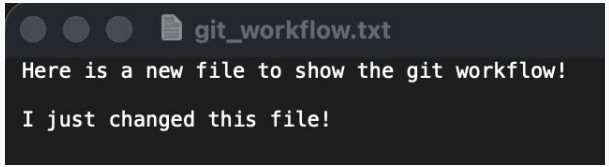
We can now see the new file on my computer

Editing and Reviewing Edits

This is the unchanged file from GitHub



I changed one line in the file and saved it



I can see what files I have edited with git status

```
(base) avab@Avas-MacBook-Pro github-training % git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   practice/git_workflow.txt

no changes added to commit (use "git add" and/or "git commit -a")
(base) avab@Avas-MacBook-Pro github-training %
```

We can use git diff to see line by line differences in files

```
(base) avab@Avas-MacBook-Pro github-training % git diff main
diff --git a/practice/git_workflow.txt b/practice/git_workflow.txt
index 167e238..8f739cd 100644
--- a/practice/git_workflow.txt
+++ b/practice/git_workflow.txt
@@ -1 +1,3 @@
 Here is a new file to show the git workflow!
+
+I just changed this file!
\ No newline at end of file
(base) avab@Avas-MacBook-Pro github-training %
```

Adding/Staging Changes

Staged files will be included in the new version of the repo when we run git commit in the next step.

To stage a file, we will do git add <FILENAME>

Alternatively, to stage all edited files, run git add --a

```
(base) avab@Avas-MacBook-Pro github-training % git add practice/git_workflow.txt

(base) avab@Avas-MacBook-Pro github-training % git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   practice/git_workflow.txt

(base) avab@Avas-MacBook-Pro github-training %
```

We can see git_workflow.txt has been staged because it is now green when we run git status, and it says that these changes are ready to be committed

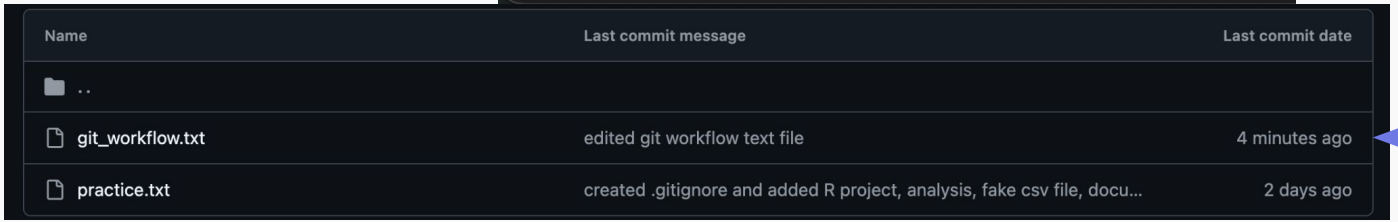
Committing and Pushing Changes

To commit changes, we use git commit -m “<DOCUMENTATION MESSAGE>”

```
(base) avab@Avas-MacBook-Pro github-training % git commit -m "edited git workflo
w text file"
[main 025a82c] edited git workflow text file
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) avab@Avas-MacBook-Pro github-training %
```

Commits are saved versions of your project stored on your local.
Only files that have been added and committed will be pushed to GitHub when you run git push

```
(base) avab@Avas-MacBook-Pro github-training % git push
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 1.38 KiB | 1.38 MiB/s, done.
Total 13 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.
To https://github.com/abraiter/github-training.git
   102a85e..025a82c  main -> main
(base) avab@Avas-MacBook-Pro github-training %
```



We can now see that the changes have been pushed to GitHub