Reminders

Upcoming due dates
Mon Oct 6th Quiz 1
Wed Oct 9th Pre-course survey
Fri Oct 10th Discussion Lab 1
#FinAid quiz on Canvas

Discussion section this week is a tour of Datahub and an ungraded Python review notebook

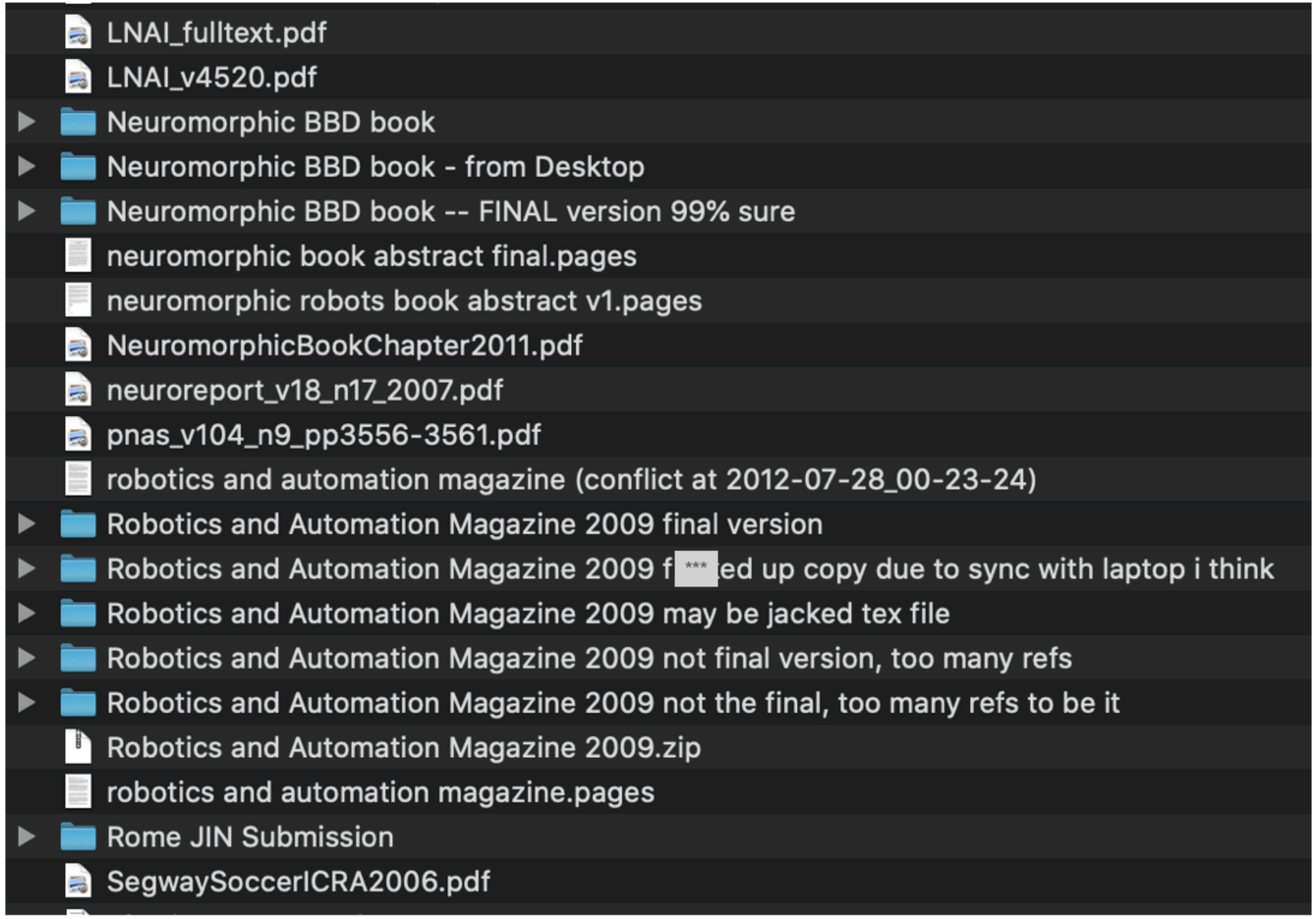
Version Control

Data Science in Practice

Jason G. Fleischer, PhD **Dept. of Cognitive Science** **UC San Diego** **https://jgfleischer.com**

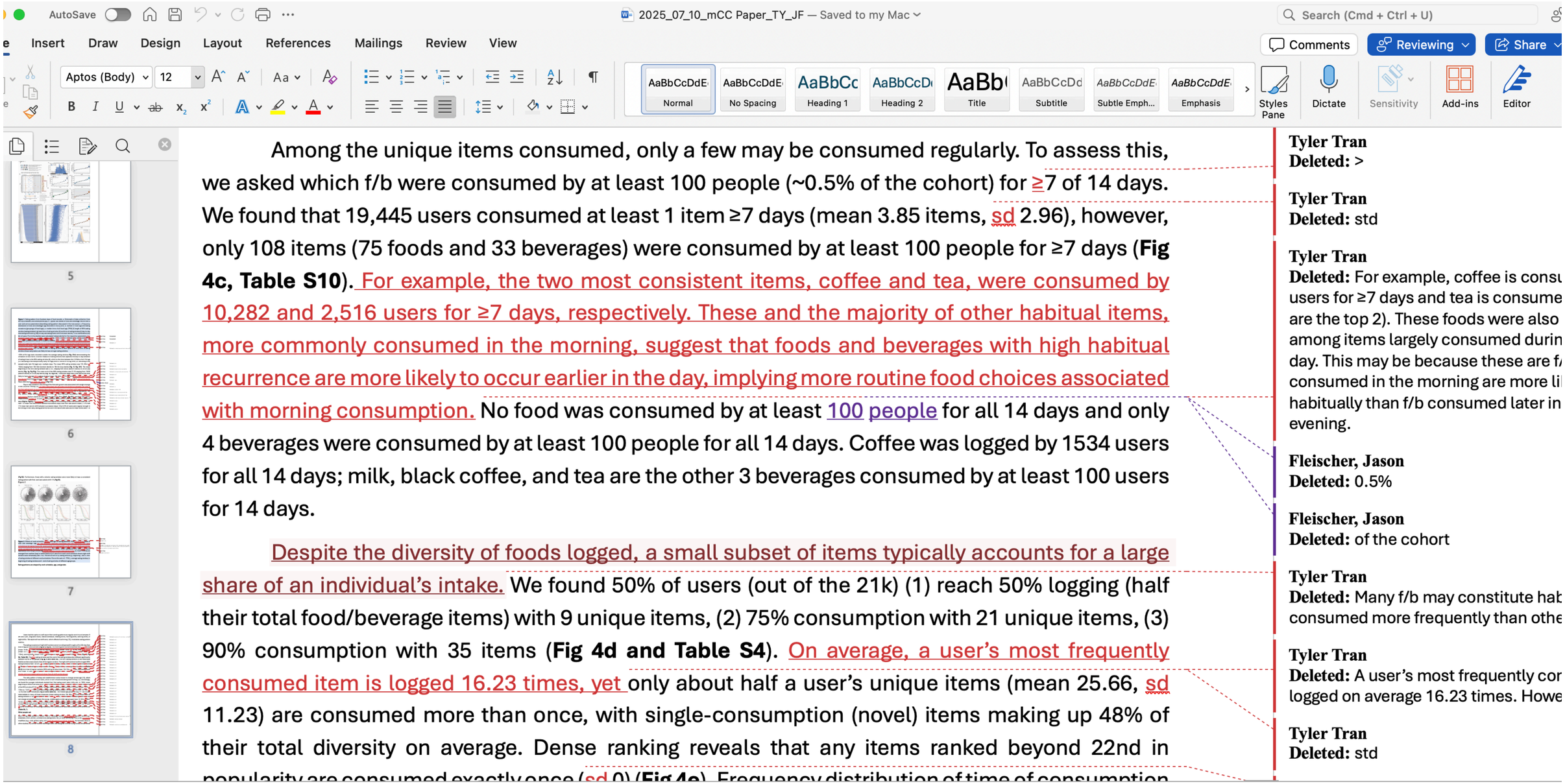
This sucks

archived version of my Documents folder from ~ 2012



Several months after finishing a writing project, I wanted to keep only the final version of the many different revisions… figuring out which one was the version actually sent to the publisher was hard!

A step in the right direction



AutoSave

2025_07_10_mCC Paper_TY_JF — Saved to my Mac

Search (Cmd + Ctrl + U)

Insert Draw Design Layout References Mailings Review View

Comments Reviewing Share

Aptos (Body) 12

Normal No Spacing Heading 1 Heading 2 Title Subtitle Subtle Emph... Emphasis

Styles Pane Dictate Sensitivity Add-ins Editor

5 6 7 8

Among the unique items consumed, only a few may be consumed regularly. To assess this, we asked which f/b were consumed by at least 100 people (~0.5% of the cohort) for ≥7 of 14 days. We found that 19,445 users consumed at least 1 item ≥7 days (mean 3.85 items, sd 2.96), however, only 108 items (75 foods and 33 beverages) were consumed by at least 100 people for ≥7 days (**Fig 4c, Table S10**). For example, the two most consistent items, coffee and tea, were consumed by 10,282 and 2,516 users for ≥7 days, respectively. These and the majority of other habitual items, more commonly consumed in the morning, suggest that foods and beverages with high habitual recurrence are more likely to occur earlier in the day, implying more routine food choices associated with morning consumption. No food was consumed by at least 100 people for all 14 days and only 4 beverages were consumed by at least 100 people for all 14 days. Coffee was logged by 1534 users for all 14 days; milk, black coffee, and tea are the other 3 beverages consumed by at least 100 users for 14 days.

Despite the diversity of foods logged, a small subset of items typically accounts for a large share of an individual's intake. We found 50% of users (out of the 21k) (1) reach 50% logging (half their total food/beverage items) with 9 unique items, (2) 75% consumption with 21 unique items, (3) 90% consumption with 35 items (**Fig 4d and Table S4**). On average, a user's most frequently consumed item is logged 16.23 times, yet only about half a user's unique items (mean 25.66, sd 11.23) are consumed more than once, with single-consumption (novel) items making up 48% of their total diversity on average. Dense ranking reveals that any items ranked beyond 22nd in popularity are consumed exactly once (sd 0) (**Fig 4e**). Frequency distribution of time of consumption

Tyler Tran
Deleted: >

Tyler Tran
Deleted: std

Tyler Tran
Deleted: For example, coffee is consu users for ≥7 days and tea is consume are the top 2). These foods were also among items largely consumed durin day. This may be because these are f/ consumed in the morning are more li habitually than f/b consumed later in evening.

Fleischer, Jason
Deleted: 0.5%

Fleischer, Jason
Deleted: of the cohort

Tyler Tran
Deleted: Many f/b may constitute hab consumed more frequently than othe

Tyler Tran
Deleted: A user's most frequently cor logged on average 16.23 times. Howe

Tyler Tran
Deleted: std

Version Control

- Enables multiple people to simultaneously work on a single project.

- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.

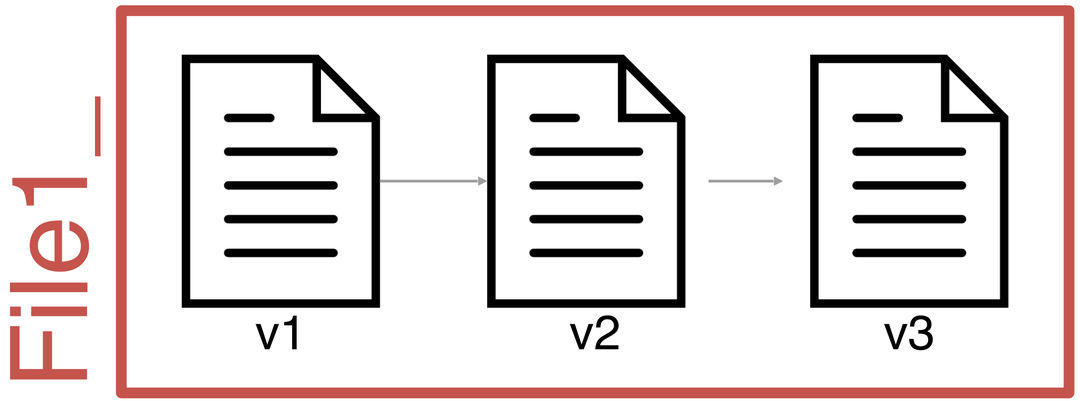
- Thus, temporary or partial edits by one person do not interfere with another person's work

What is version control?

A way to manage the evolution of a set of files

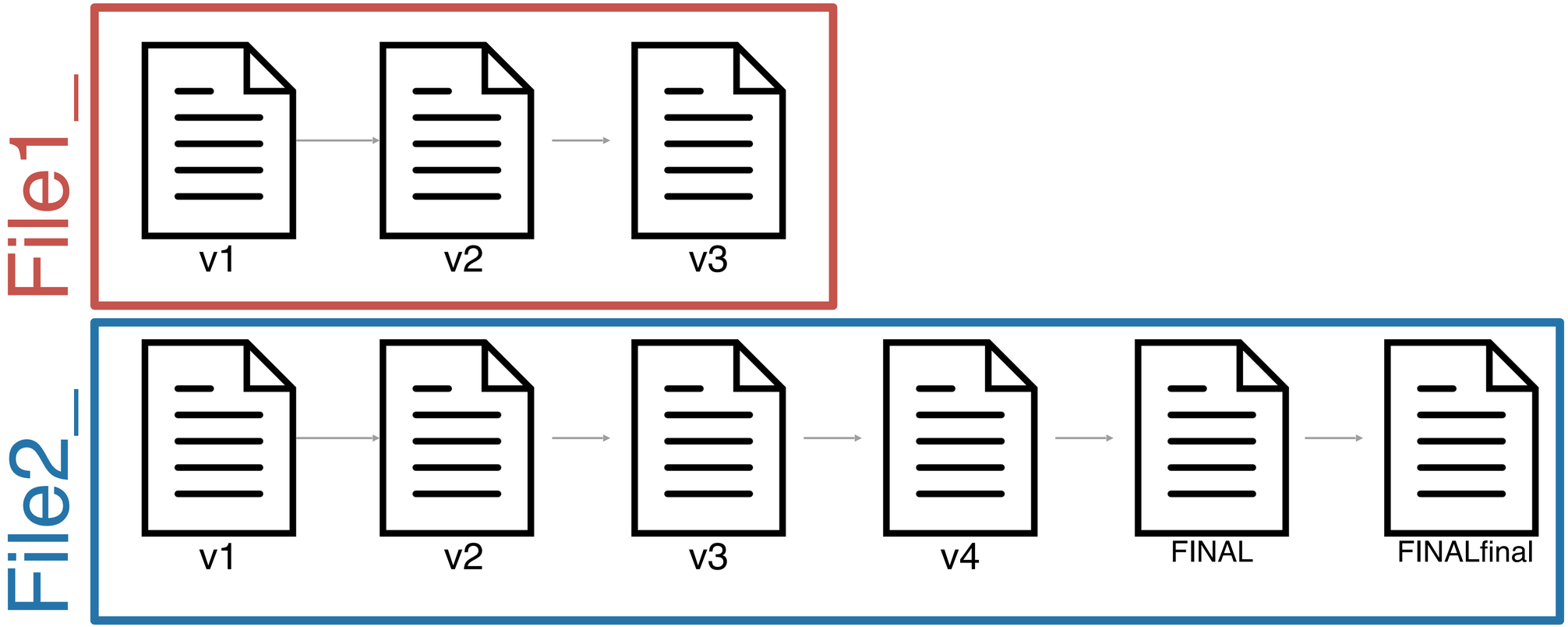
What is version control?

A way to manage the evolution of a set of files



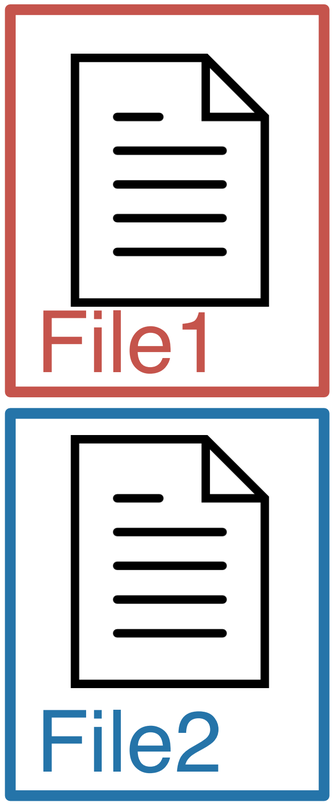
What is version control?

A way to manage the evolution of a set of files



What is version control?

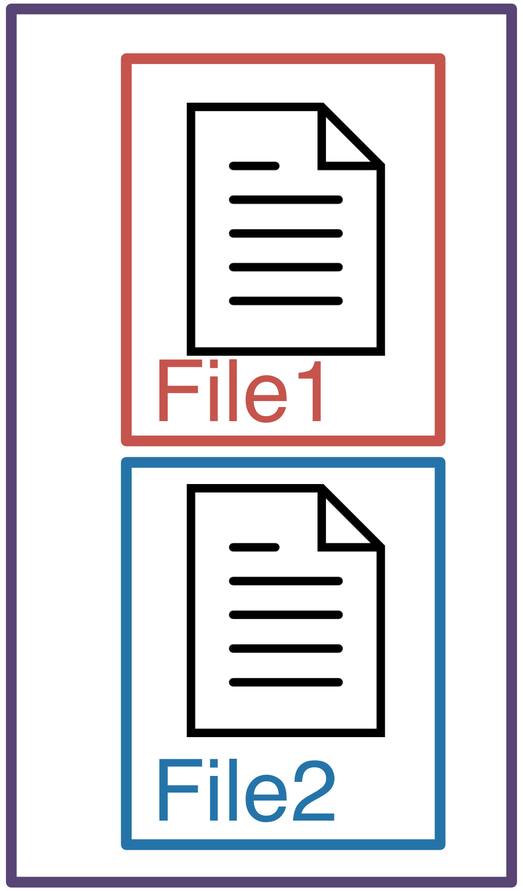
A way to manage the evolution of a set of files



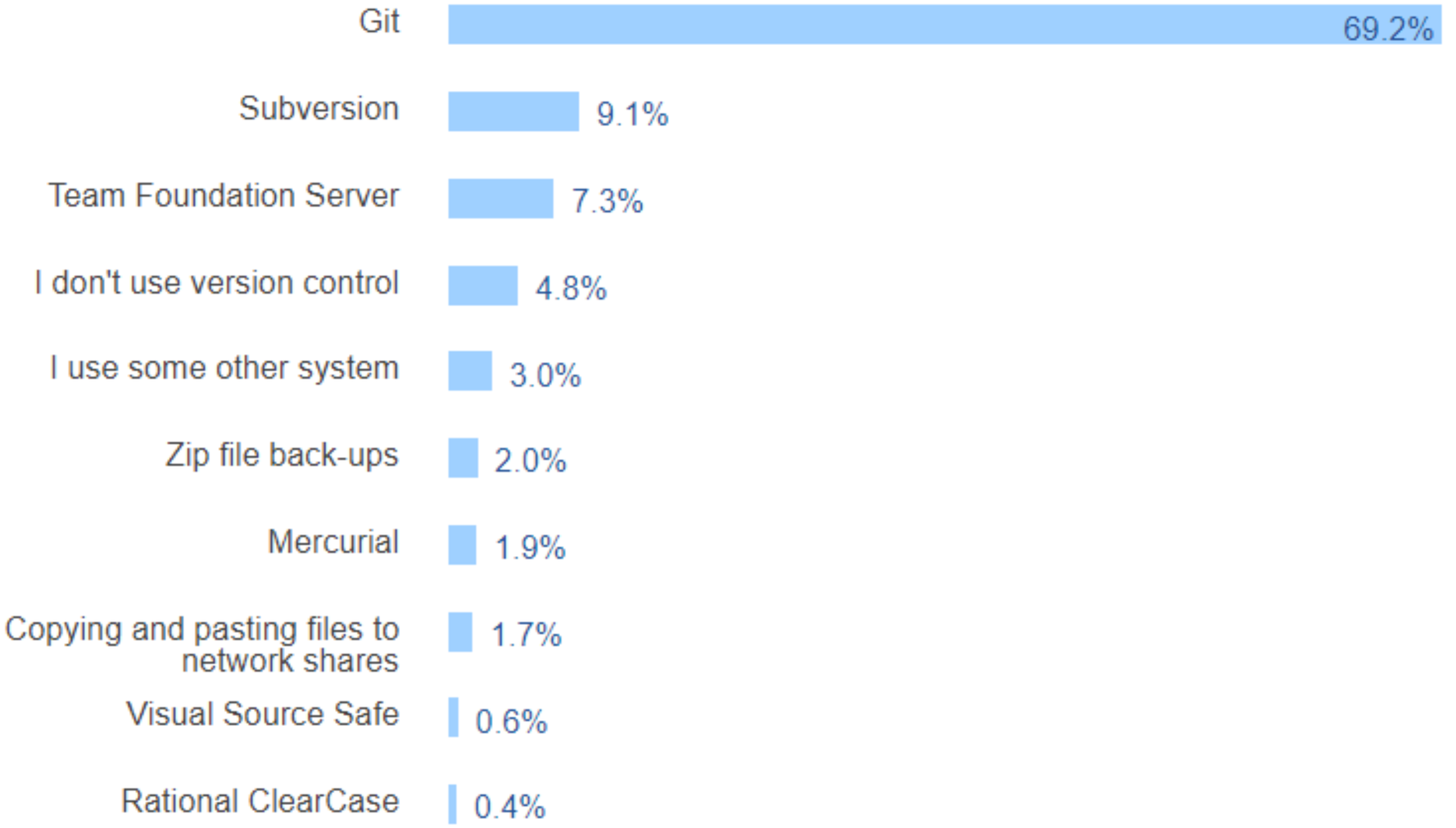
When using a version control system, you have **one copy of each file** and the *version control system tracks the changes* that have occurred over time

What is version control?

A way to manage the evolution of a set of files



The set of files is referred to as a **repository (repo)**


Git
69.2%
Subversion
9.1%
Team Foundation Server
7.3%
I don't use version control
4.8%
I use some other system
3.0%
Zip file back-ups
2.0%
Mercurial
1.9%
Copying and pasting files to network shares
1.7%
Visual Source Safe
0.6%
Rational ClearCase
0.4%

git & GitHub

git

the version control system

~ Track Changes from Microsoft Word….on steroids

GitHub (or Bitbucket or GitLab) is the home **where your git-based projects live** on the Internet.

~ Dropbox + social media for programmers

What version control looks like

```
$ git clone https://www.github.com/username/repo.git
$ git pull

[… edit some files, make changes …]

$ git add -A
$ git commit -m “informative commit message”
$ git push
```

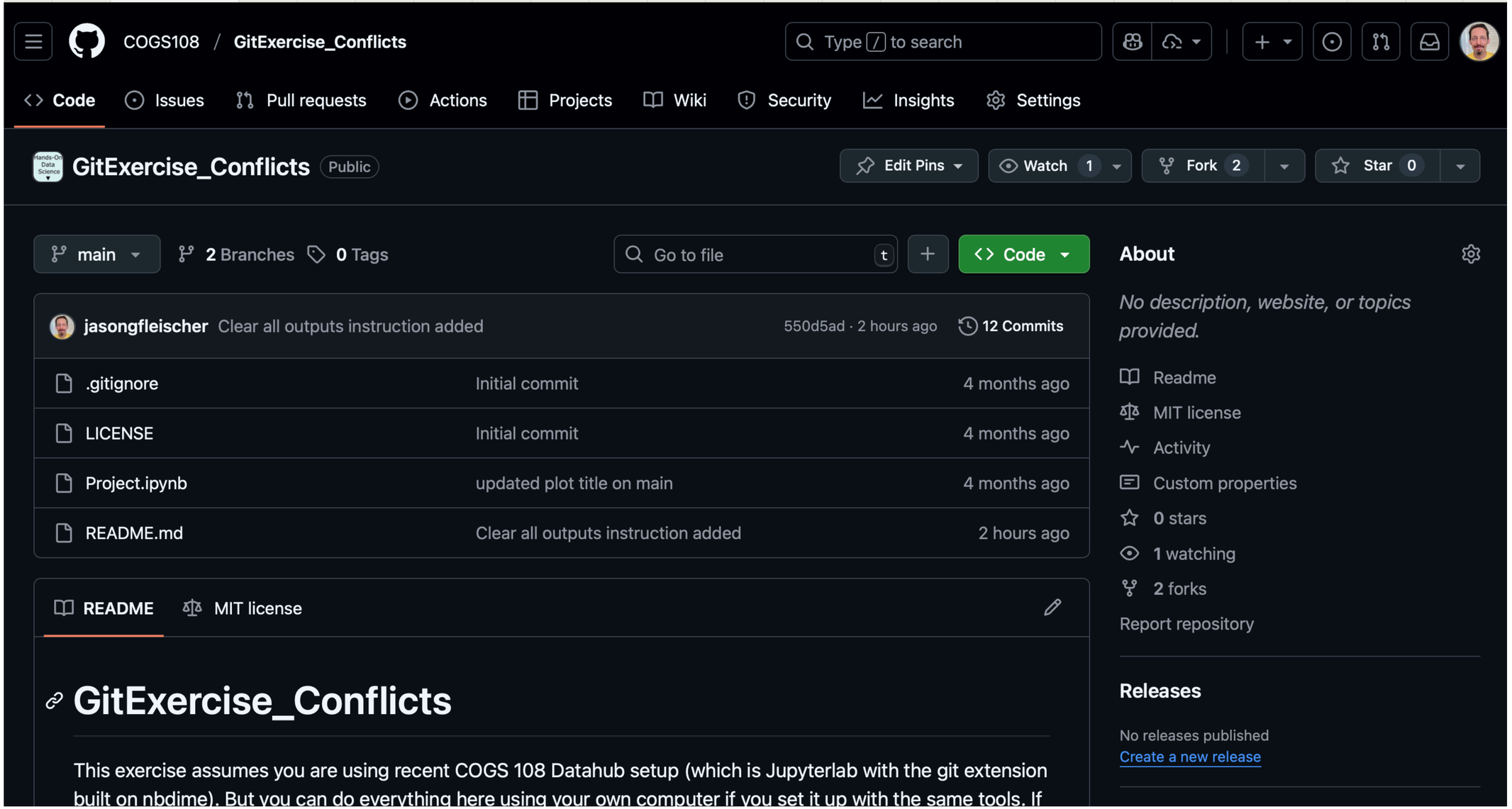
Git on command line

What version control looks like

```
jfleischer@dsmlp-jupyter-jfleischer:~/private/GitExercise_Con
flicts$ git diff
diff --git a/Project.ipynb b/Project.ipynb
index d938ddb..3337e25 100644
--- a/Project.ipynb
+++ b/Project.ipynb
@@ -13,10 +13,21 @@
    },
    {
     "cell_type": "code",
-    "execution_count": null,
+    "execution_count": 4,
     "id": "8d446506",
     "metadata": {},
-    "outputs": [],
+    "outputs": [
+     {
+      "data": {
+       "image/png": "iVBORw0KGgoAAAANSUhEUgAAAiEAAAGHCAYAAABm
```

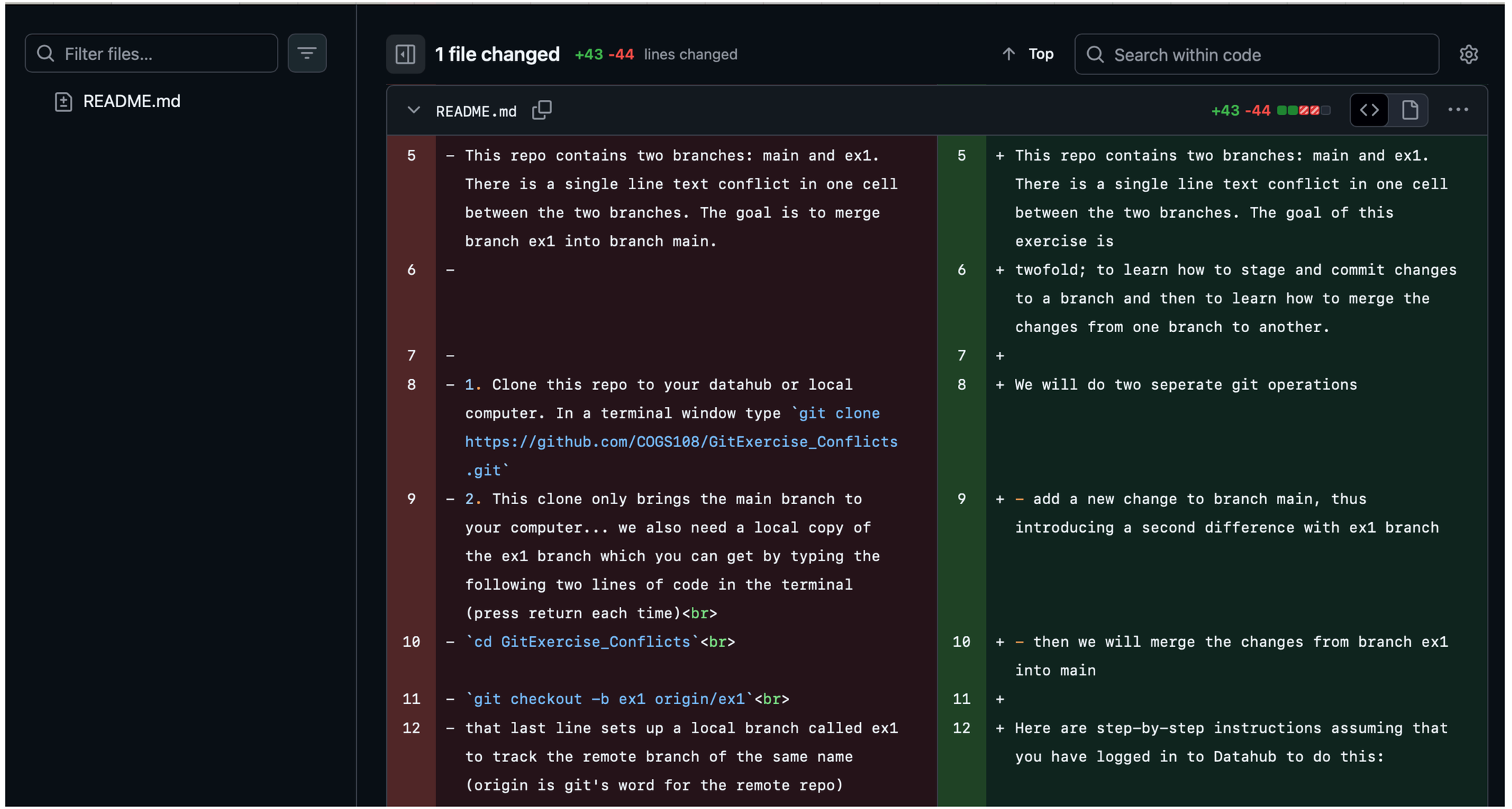
Git on command line

What version control looks like



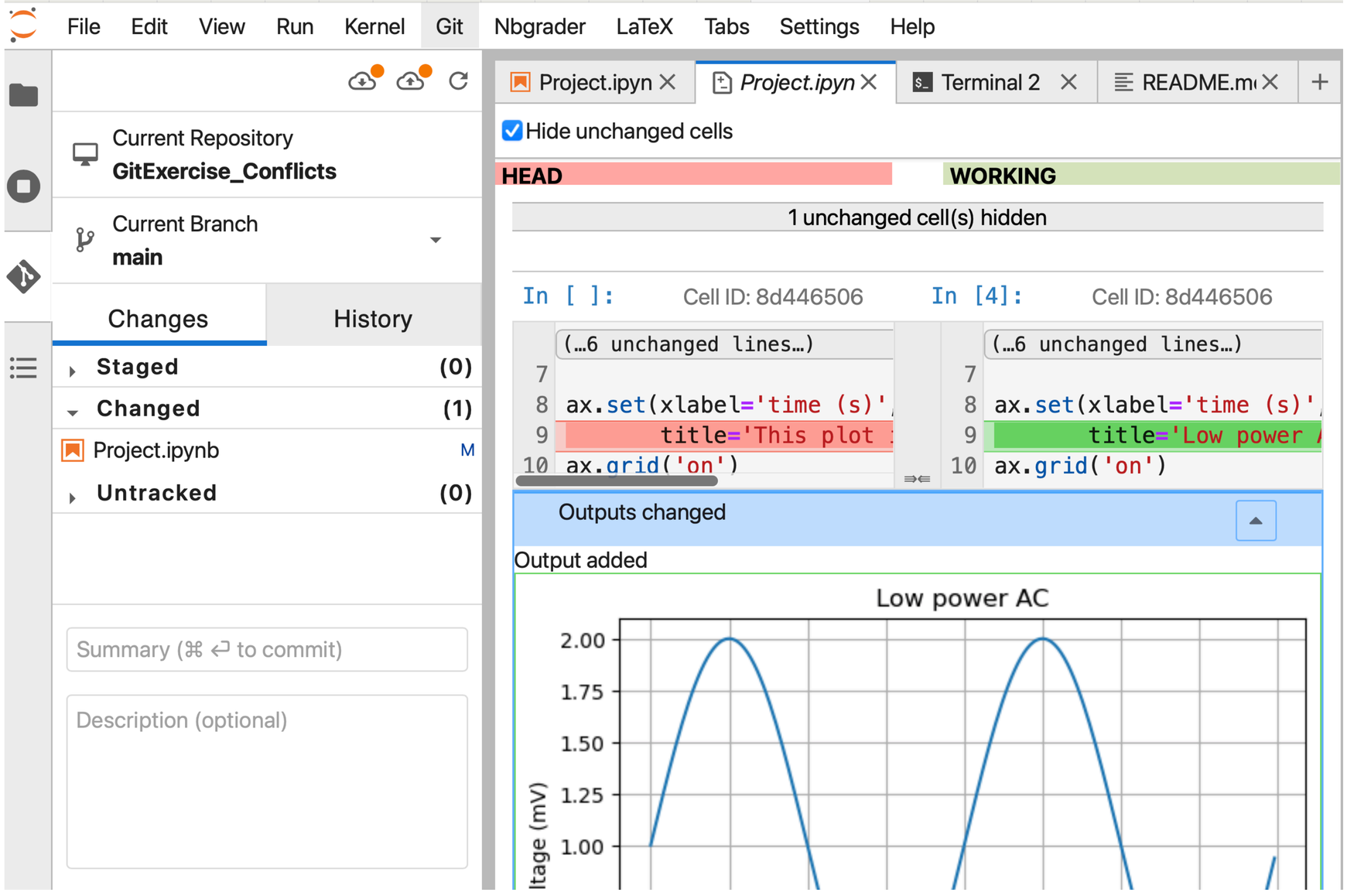
GitHub

What version control looks like



GitHub

What version control looks like



Datahub
@ UCSD

https://forms.gle/8UeUL2Ux4YtG2CVr8

Version Controller

How do you typically interact with git?

A. I don’t

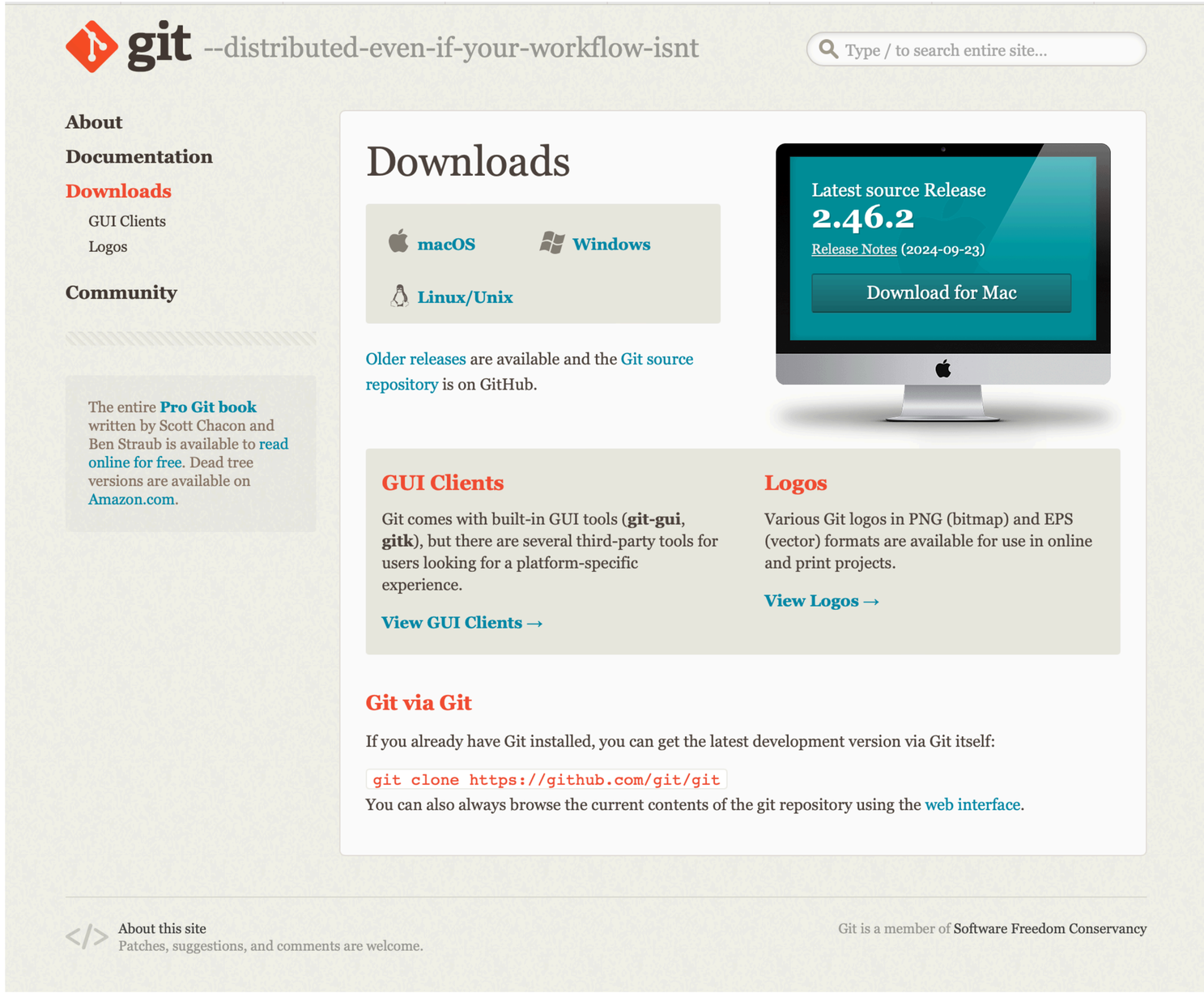
B. command line

C. GUI GitHub Desktop

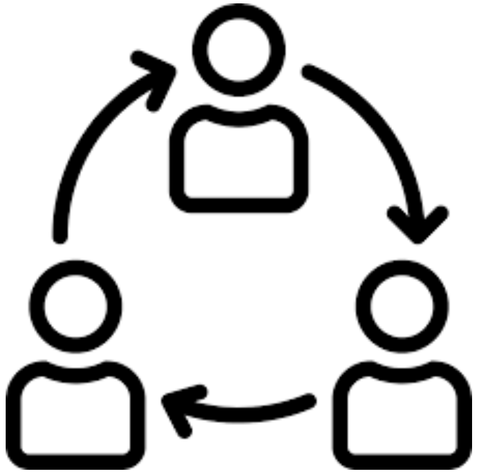
D. GUI: SourceTree

E. GUI: other

https://git-scm.com/downloads



Why version control with git and GitHub?



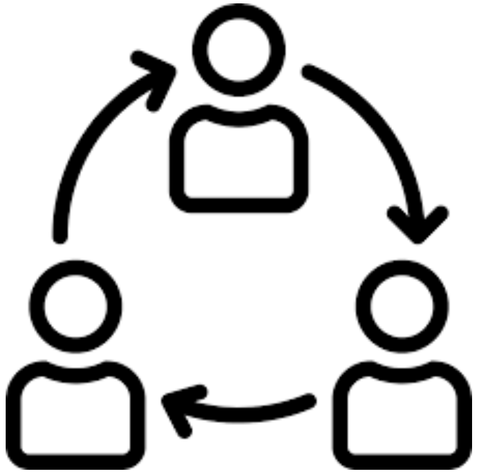
Collaboration

Returning to a safe state

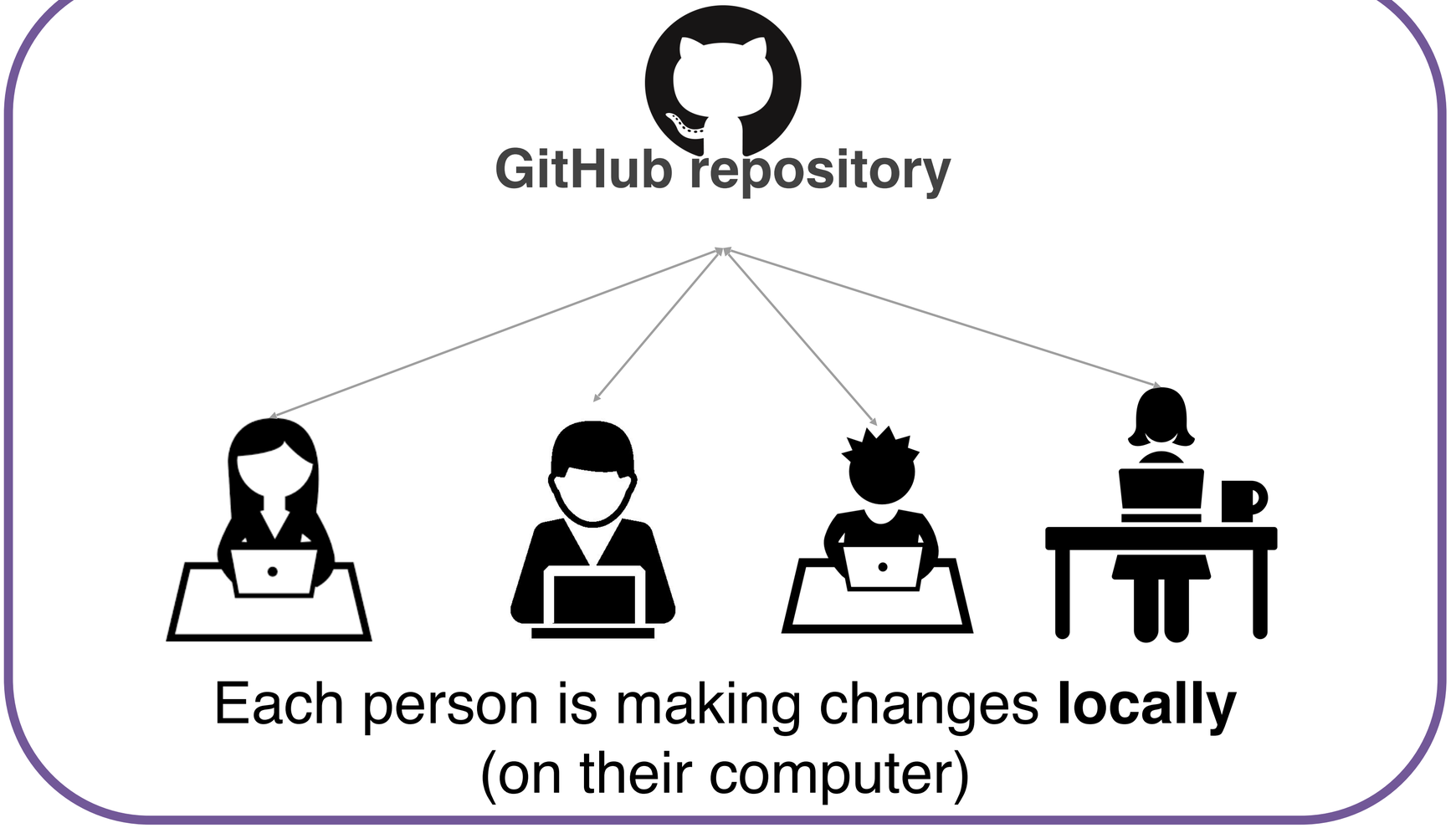
Exposure for your work

Tracking others' work

Collaborate like you do with Google Docs



Collaboration



Make changes locally, while knowing a stable copy exists

Returning to a safe state

You're free and safe to **try things out locally**. You'll only send changes to the repo when you're at a stable point

Your repositories will be visible to others!

Exposure for your work

Your public GitHub repos are your coding social media

And vice versa, you can search for the code you need

For instance, this might come in handy when thinking about class projects

https://github.com/topics/datascience-projects

april-fools

DuckMasterAI / rickroll-bot

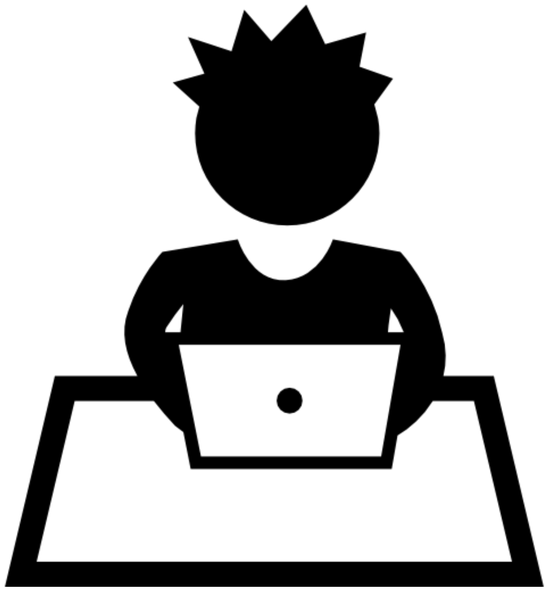
A simple bot to rickroll your friends on Discord!

discord-bot discord-py april-fools rickroll never-gonna-give-you-up never-gonna-let-you-down rick-astley discord-py-bot

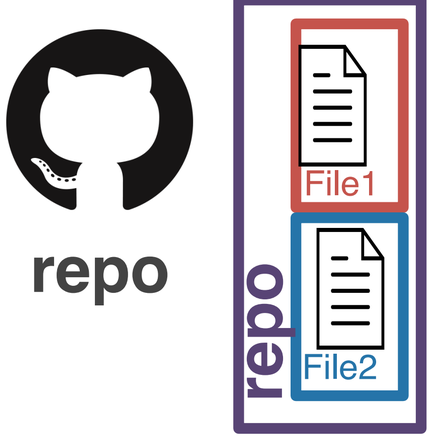
Updated 2 hours ago Python

Keep up with others' work easily

Tracking others' work

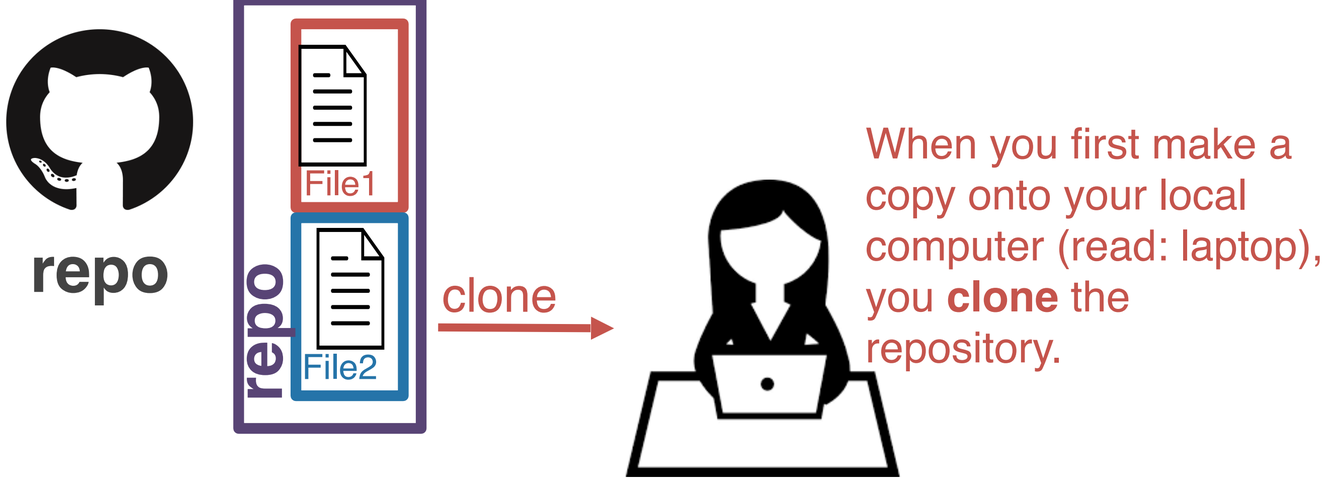


As a social platform, you can see others' work too!

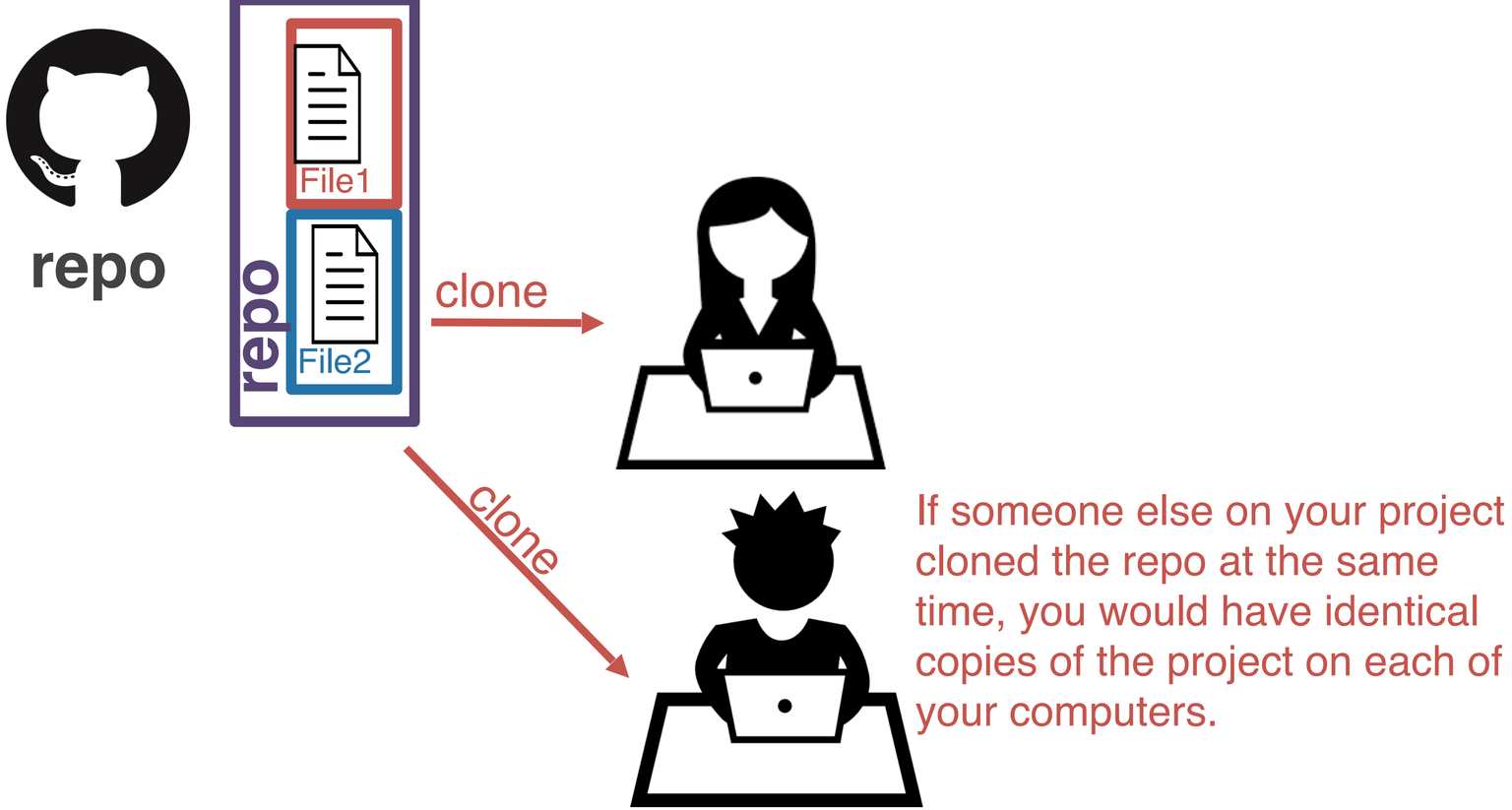


A **GitHub repo** contains all the files and folders for your project.

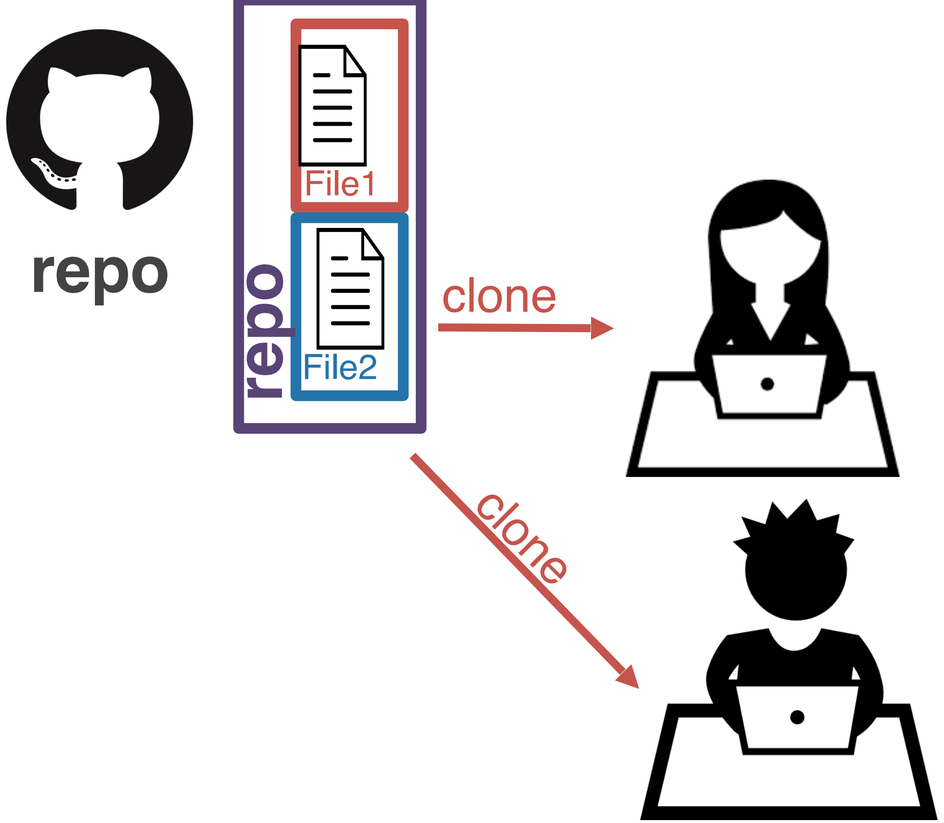
GitHub is a **remote host**. The files are geographically distant from any files on your computer.



When you first make a copy onto your local computer (read: laptop), you **clone** the repository.



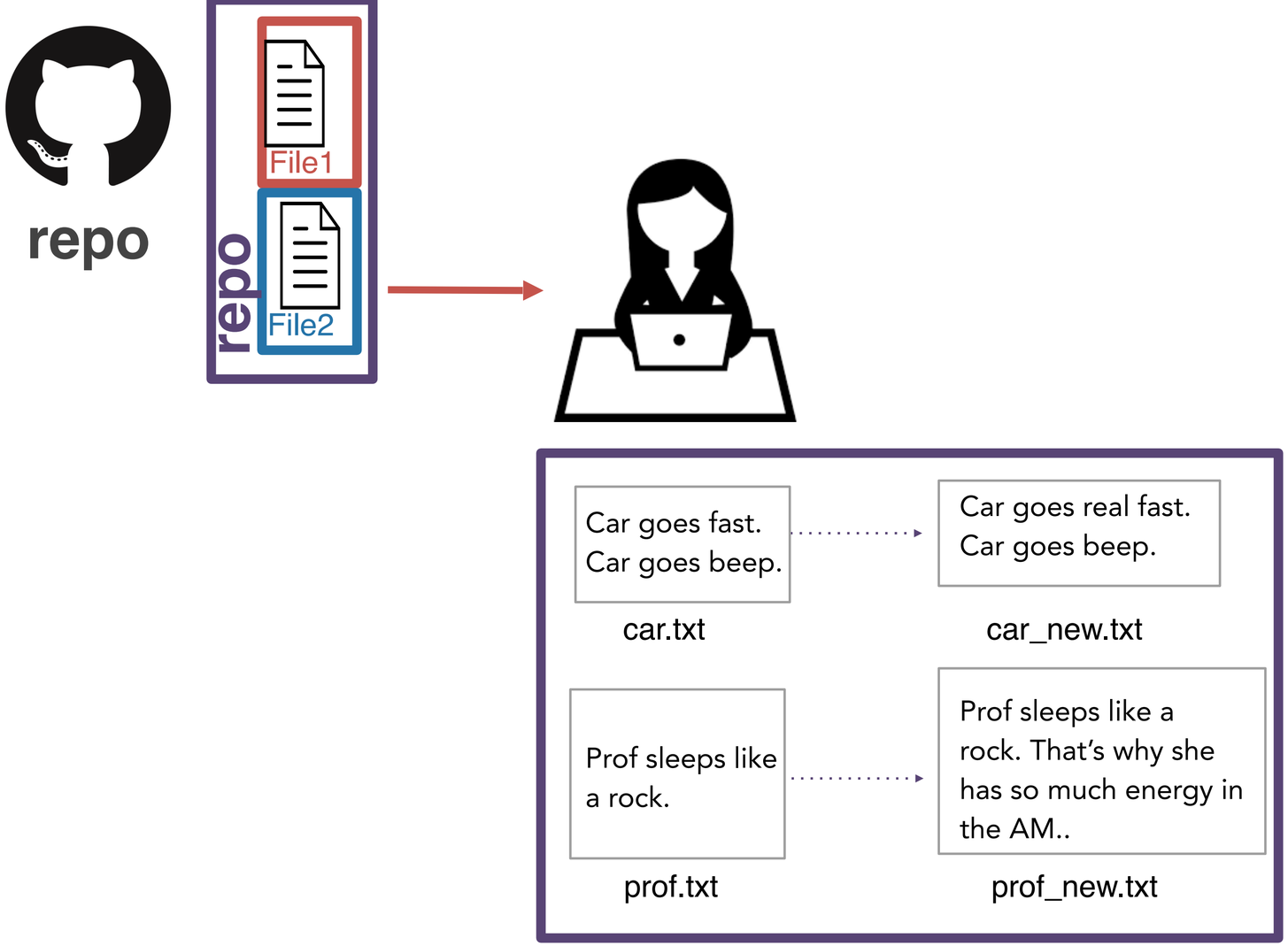
If someone else on your project cloned the repo at the same time, you would have identical copies of the project on each of your computers.



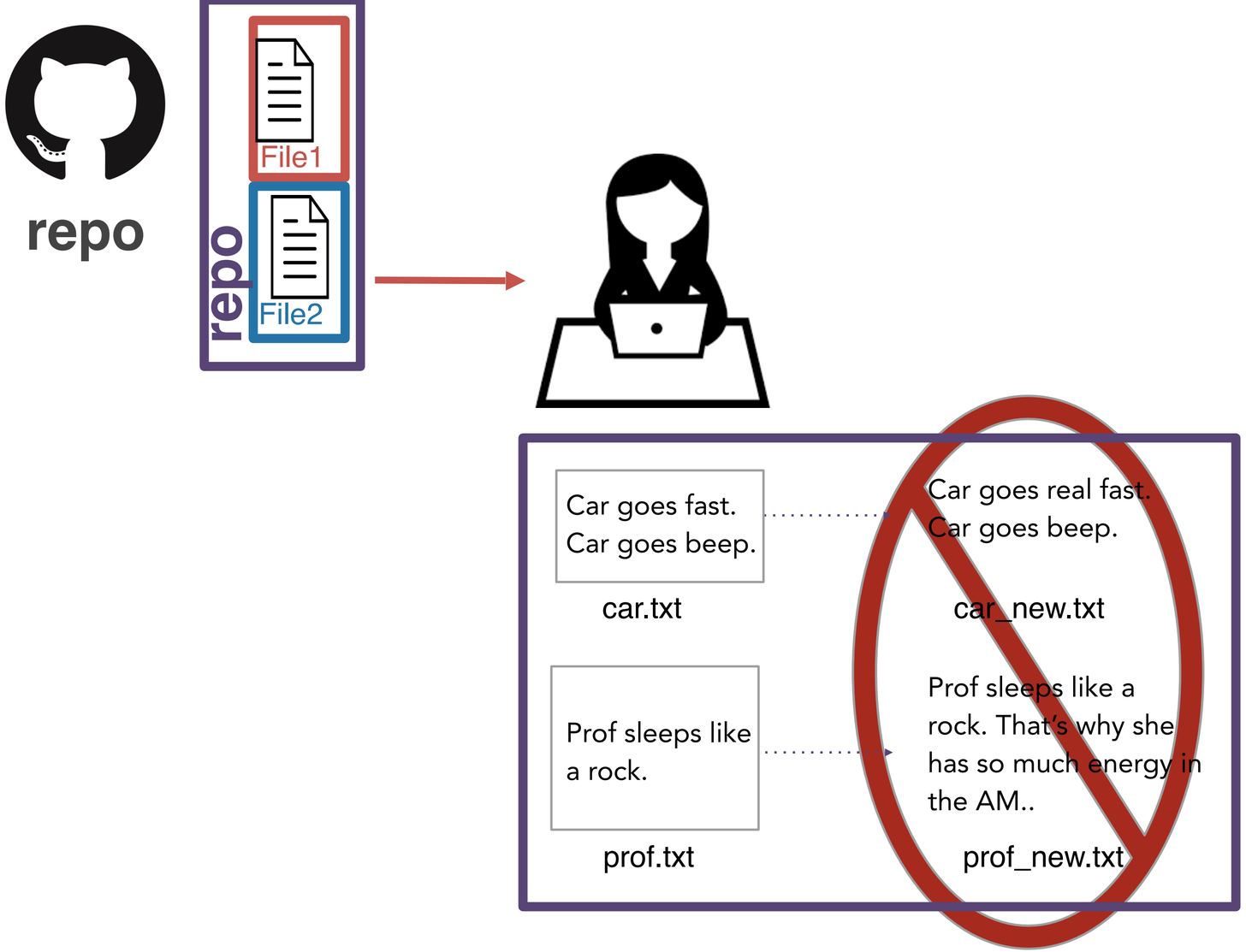
Yay! Everyone can work on the project!

You decide you want to change some of the text in the project.

You decide you want to change some of the text in the project.

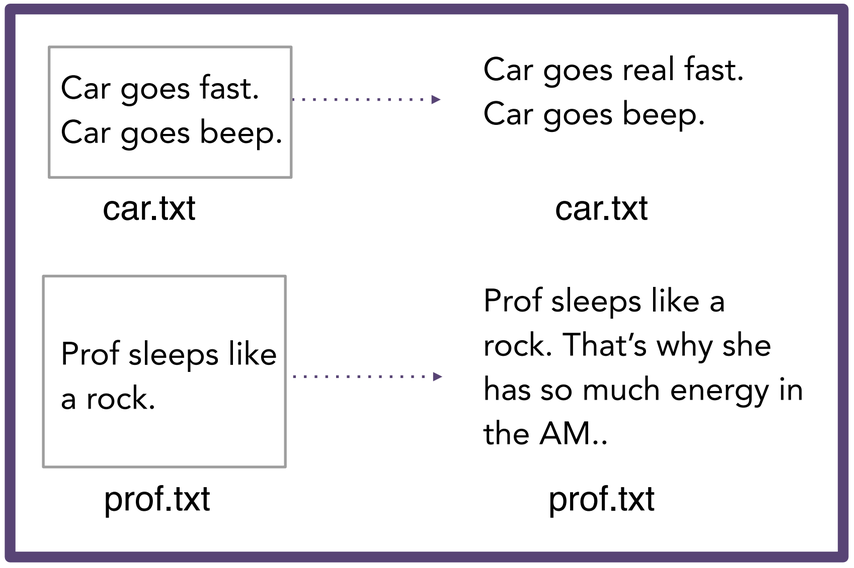


without git...you'd likely rename these files….

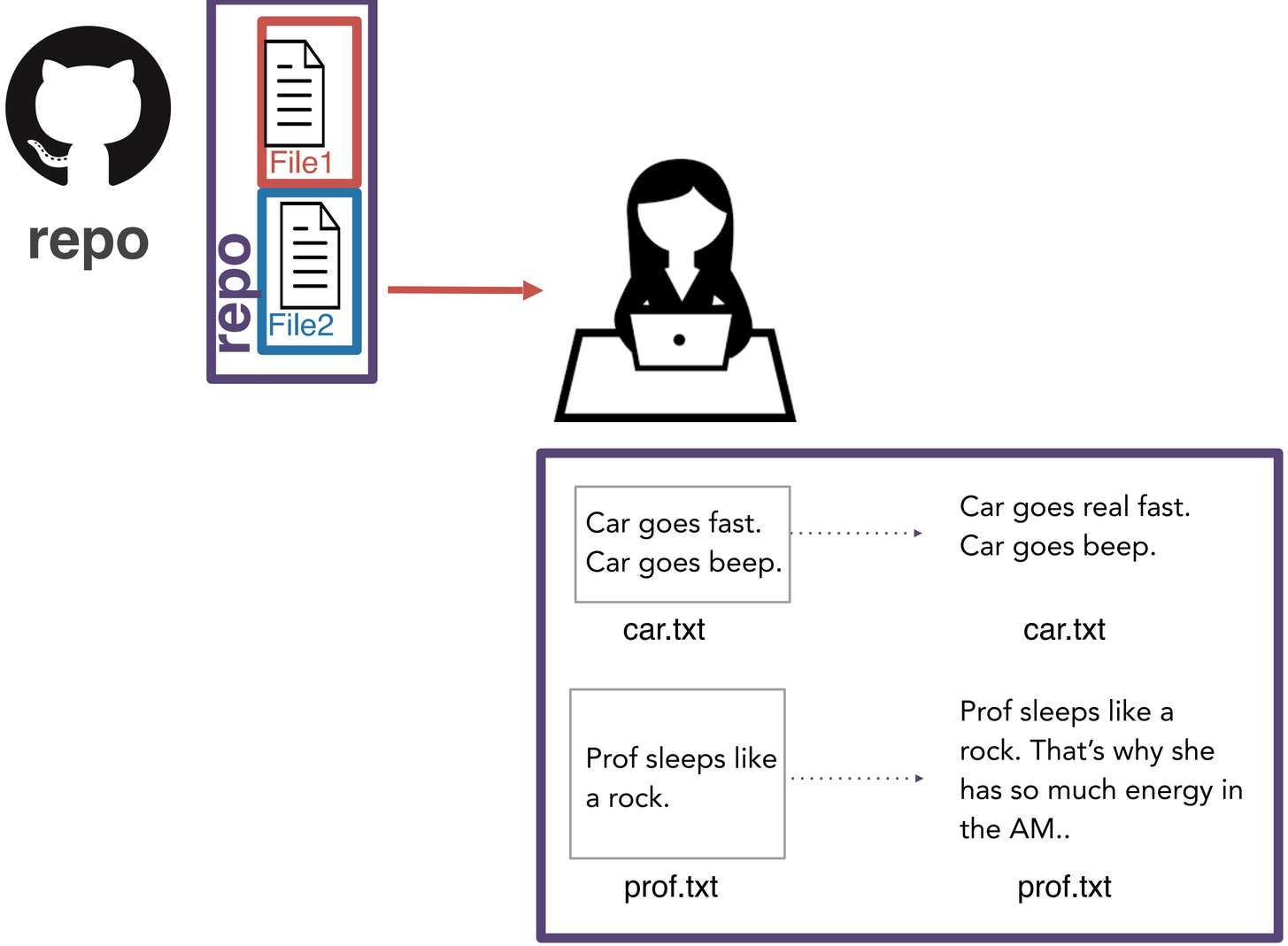


Thank goodness those days are over!

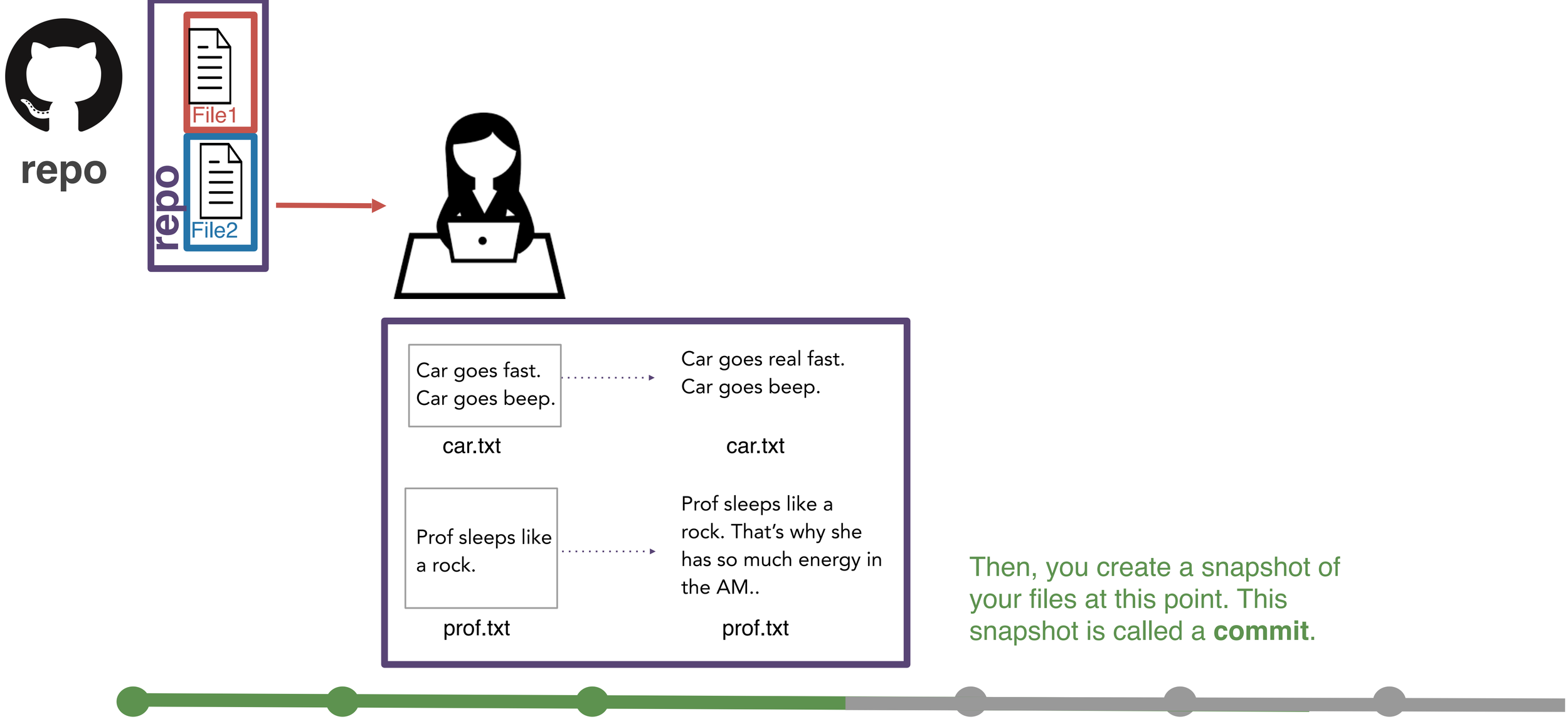
`git add file`	stages specified file (or folder)
`git add .`	stages new and modified files
`git add -u`	stages modified and deleted files
`git add -A`	stages new, modified, and deleted files
`git add *.csv`	stages any files with .csv extension
`git add *`	use with caution: stages everything

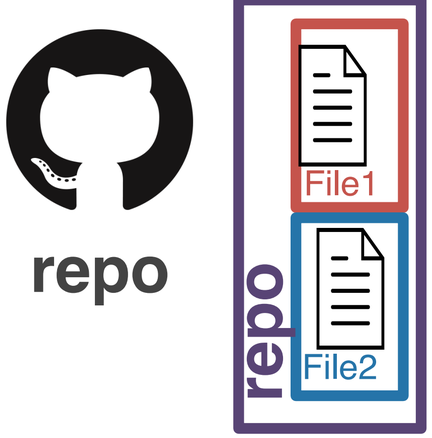


Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.



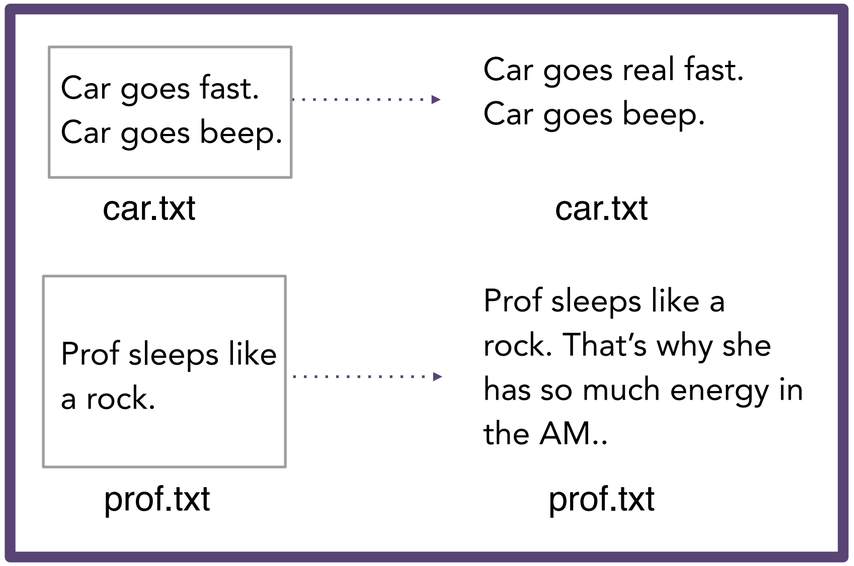
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.


repo
repo
File1
File2
Car goes fast.
Car goes beep.
car.txt
Car goes real fast.
Car goes beep.
car.txt
Prof sleeps like a rock.
prof.txt
Prof sleeps like a rock. That's why she has so much energy in the AM..
prof.txt
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.
A **commit** tracks who, what, and when



You can make commits more informative by adding a **commit message**.

Example: `git commit -m 'fix typos in car and prof'`



Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

A **commit** tracks who, what, and when

Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof

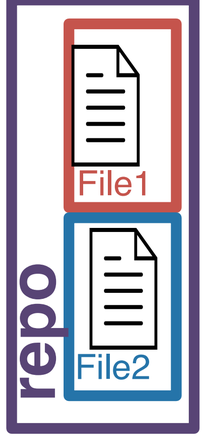
push

Remember, you're not the only one working on this project though! You want your teammates to have access to these changes! You **push** these changes back to the remote.

Shannon Ellis
3/28/21 3:28pm

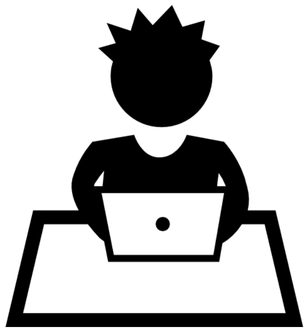
fix typos in car and prof

repo



Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof

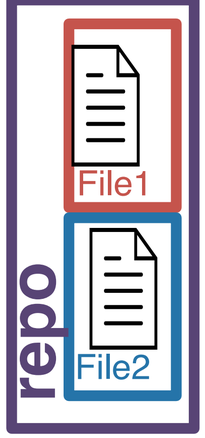


Your teammate is still working with the (out-of-date) copy he cloned earlier!

Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof

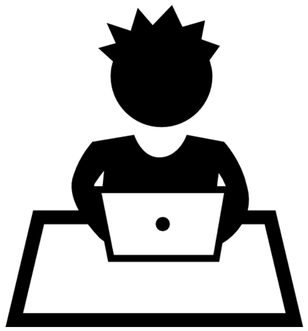
repo



To catch up, your teammate will have to **pull** the changes from GitHub (remote)

Shannon Ellis
3/28/21 3:28pm

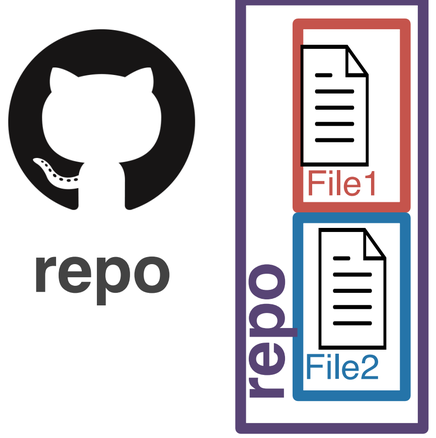
fix typos in car and prof

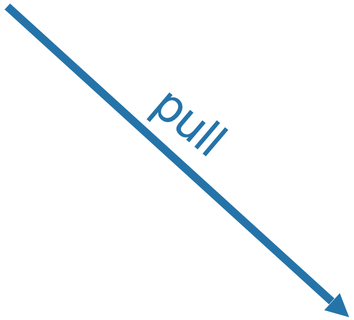


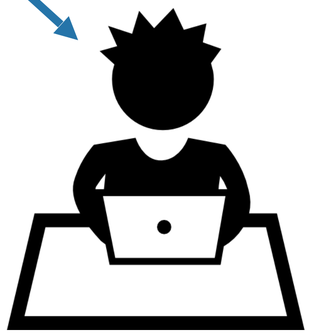
Your teammate is still working with the (out-of-date) copy he cloned earlier!

Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof



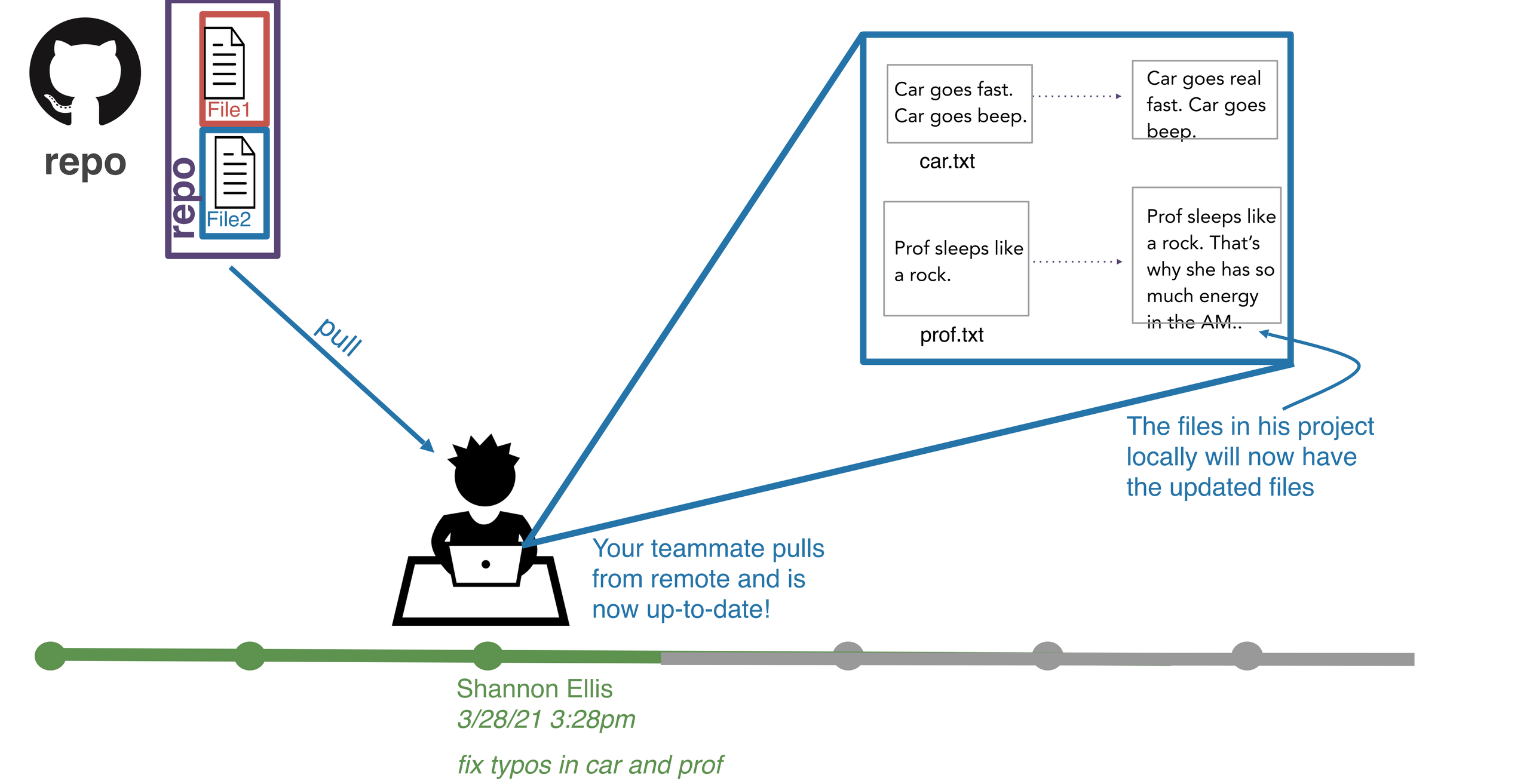


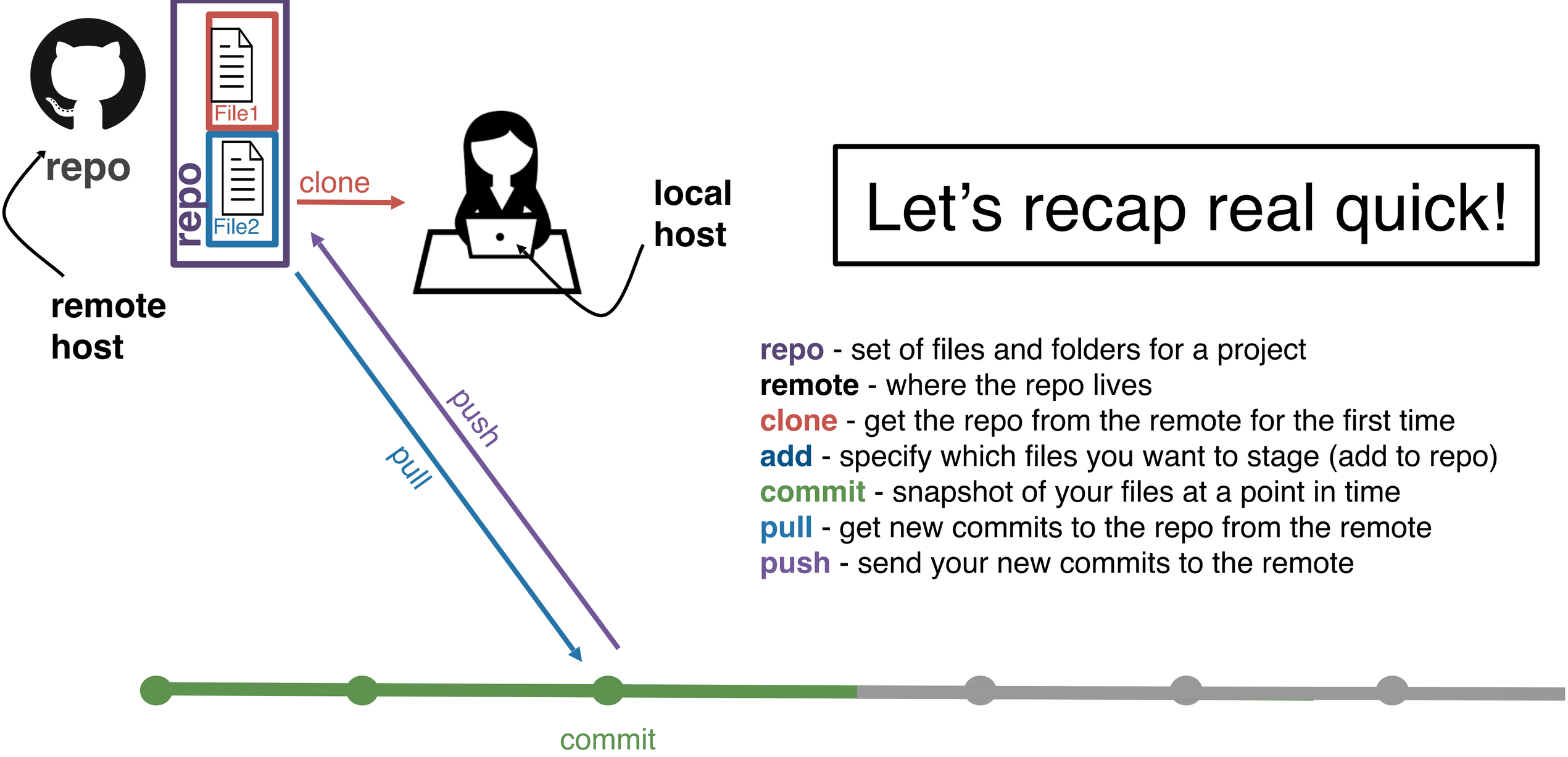


Your teammate pulls from remote and is now up-to-date!

Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof


repo
repo
File1
File2
pull
Car goes fast.
Car goes beep.
car.txt
Car goes real fast. Car goes beep.
Prof sleeps like a rock.
prof.txt
Prof sleeps like a rock. That's why she has so much energy in the AM..
The files in his project locally will now have the updated files
Your teammate pulls from remote and is now up-to-date!
Shannon Ellis
3/28/21 3:28pm
fix typos in car and prof



Let’s recap real quick!

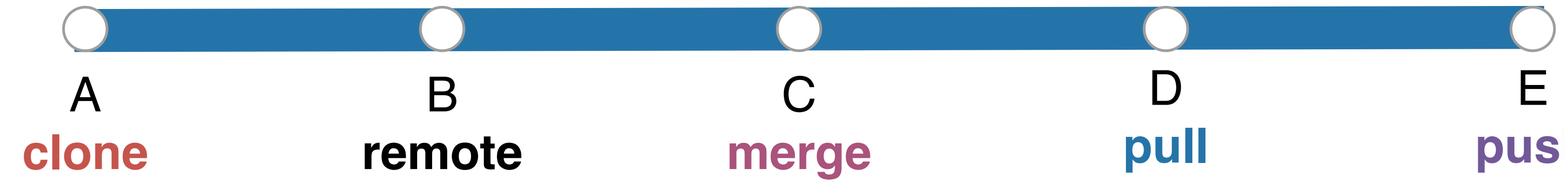
repo - set of files and folders for a project
remote - where the repo lives
clone - get the repo from the remote for the first time
add - specify which files you want to stage (add to repo)
commit - snapshot of your files at a point in time
pull - get new commits to the repo from the remote
push - send your new commits to the remote

Version Controller I

https://forms.gle/wHA2GSyuycFre5qr6

You've been working with a team on a project in a repo. You've made changes locally and you want to see them on the remote.

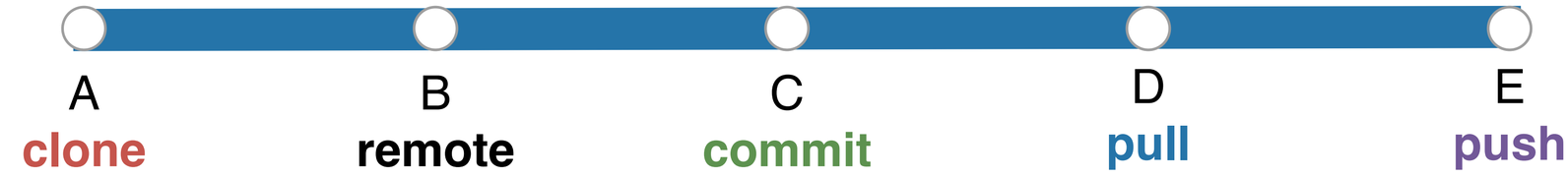
What do you do to get them on the remote?



Version Controller II

Your teammate has given you access to a GitHub repository to work on a project together. You want to get them for the first time on your computer locally.

What do you do to get the repo on your computer?



Git exercise #1

1. Login to datahub.ucsd.edu
2. Open a terminal
3. In the terminal type:

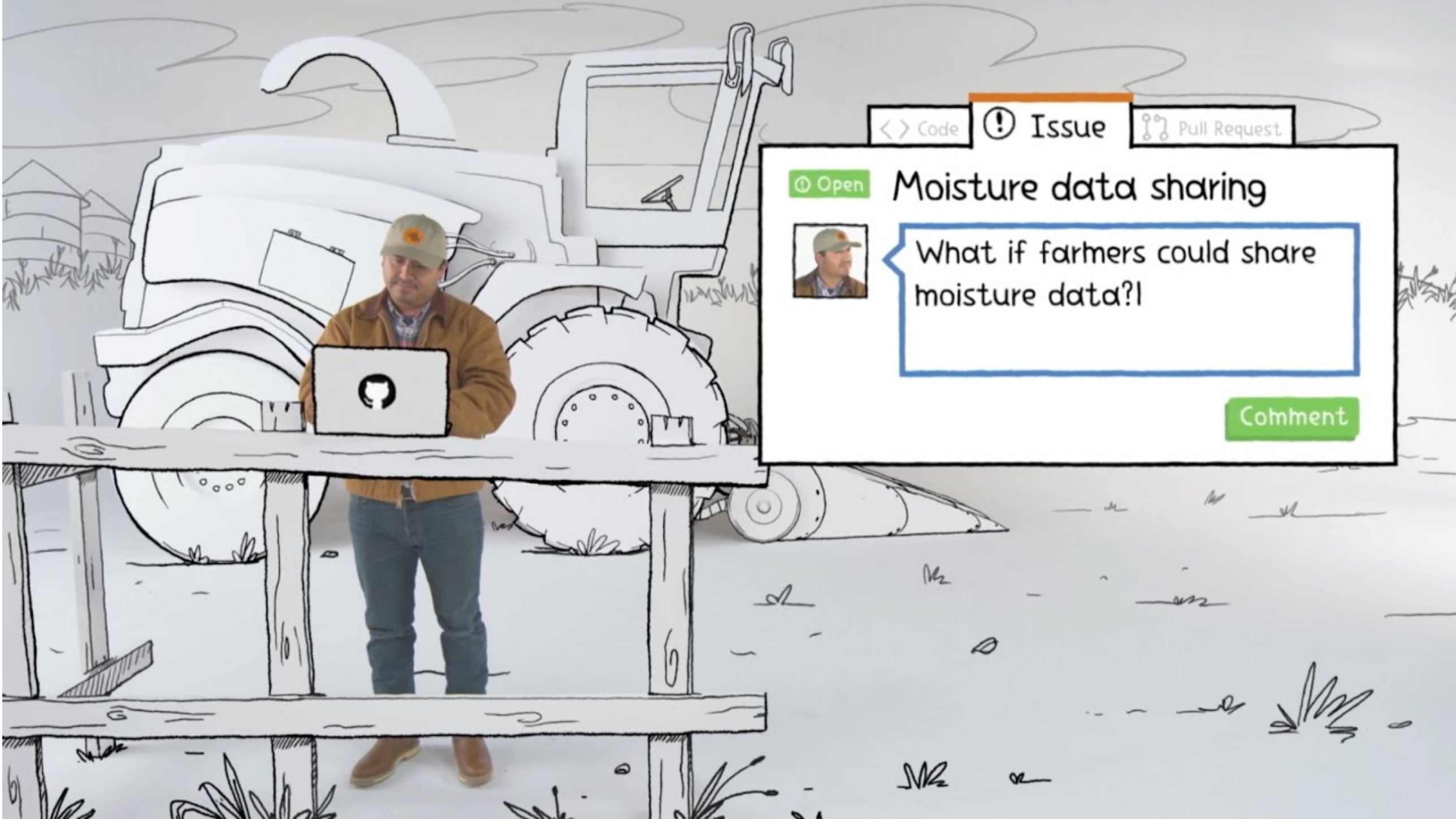
```
git config --global user.email "you@ucsd.edu"
git config --global user.name "Your Name"

git clone https://github.com/COGS108/Lectures-Fa25
```

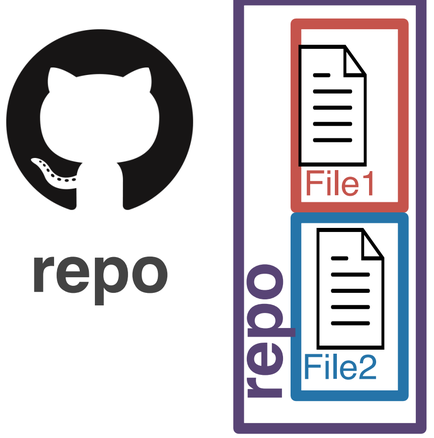
4. Ask yourself, how will the command look different when I want to update my lecture repo with the new slides next week. Try that command out!

Git exercise #2 - updating a repo

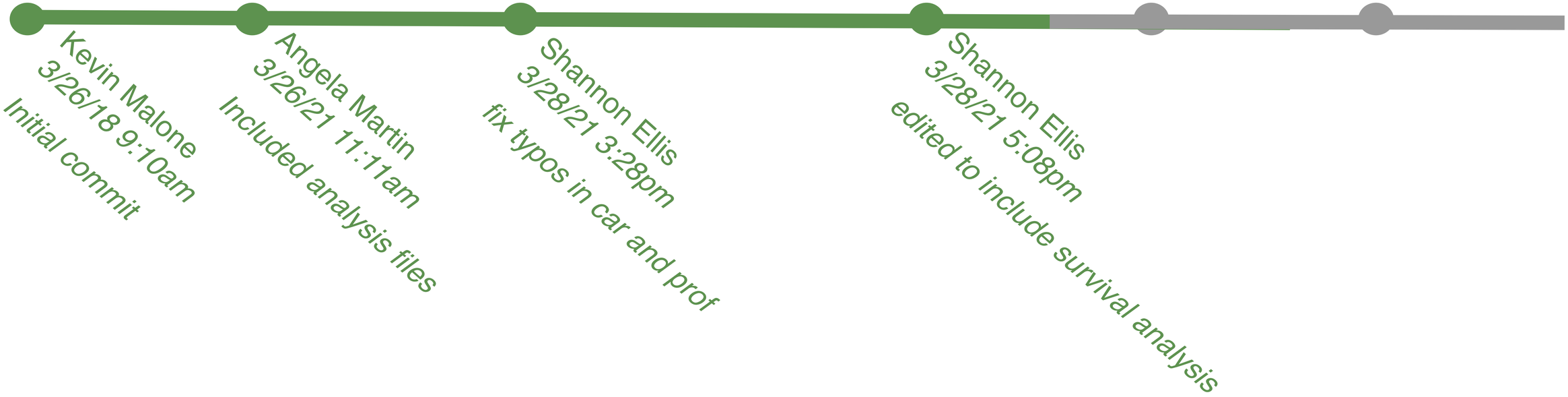
1. Change directory to the new repo Lectures-Sp25
2. Figure out if what (if anything) is different from the last commit, type:
   ```
   git status
   ```
3. Create a brand new file
   ```
   touch 007-new-movie.mov
   ```
4. Figure out if what (if anything) is different from the last commit, type:
   ```
   git status
   ```
5. Stage your changes
   ```
   git add 007-new-movie.mov
   ```
6. Commit your changes
   ```
   git commit -m “unreleased James Bond movie”
   ```
7. Question: What happens if you push to Github? Why?
8. Undo these unwanted changes to prevent problems in the future:
   ```
   git reset —hard HEAD~1
   ```

Link for the video on the next slide so you can watch it on your own:
https://youtube.com/watch?v=w3jLJU7DT5E

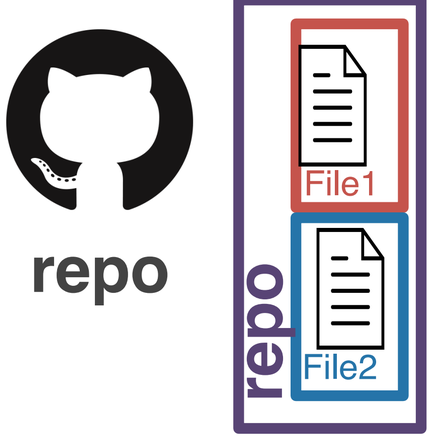

Code
Issue
Pull Request
Open
Moisture data sharing
What if farmers could share moisture data?!
Comment



Each time you create a commit, git tracks the changes made automatically.



By committing each time you make changes, git allows you to time travel!



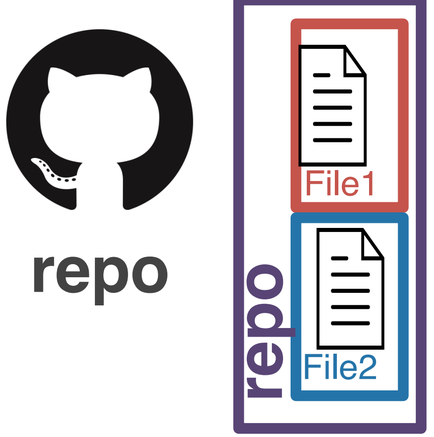
By committing each time you make changes, git allows you to time travel!

377dfcd00dd057542b112cf13be6cf1380b292ad

439301fe69e8f875c049ad0718386516b4878e22

456722223e9f9e0ee0a92917ba80163028d89251

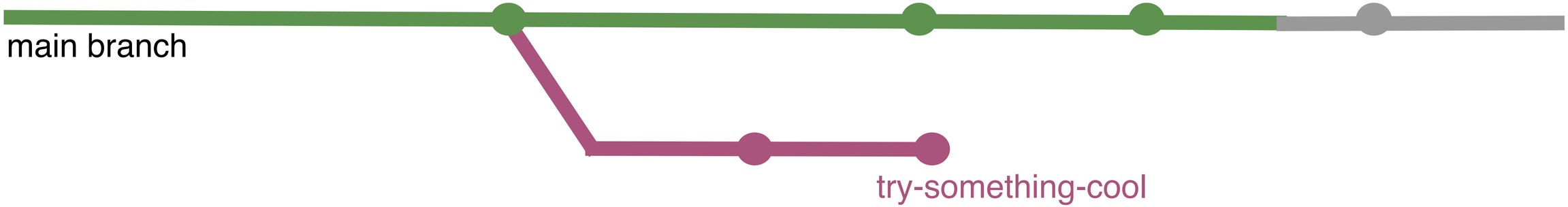
There's a unique id, known as a **hash**, associated with each commit.

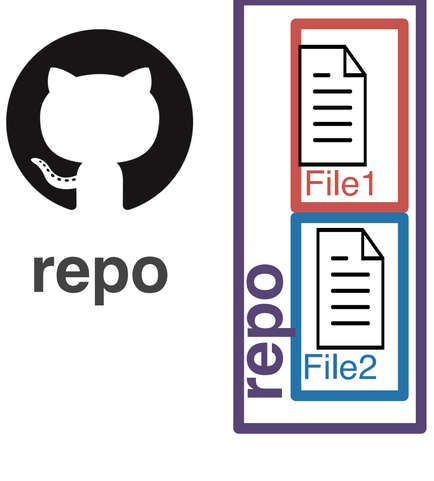


You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.

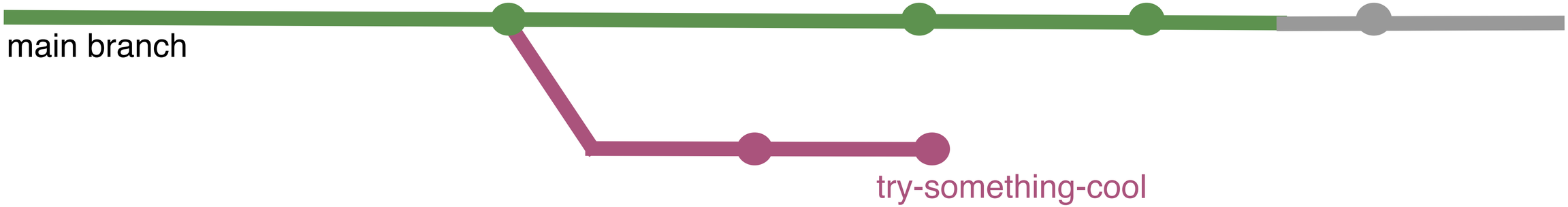
377dfcd00dd057542b112cf13be6cf1380b292ad

But...not everything is always linear.
Sometimes you want to try something out
and you're not sure it's going to work.
This is where you'll want to use a **branch**.

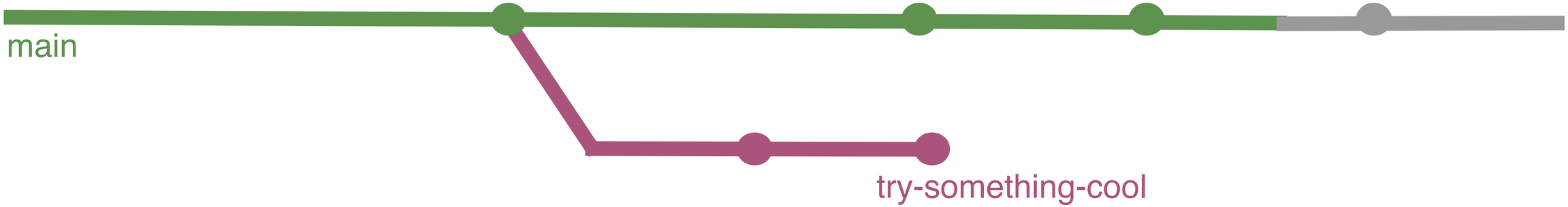


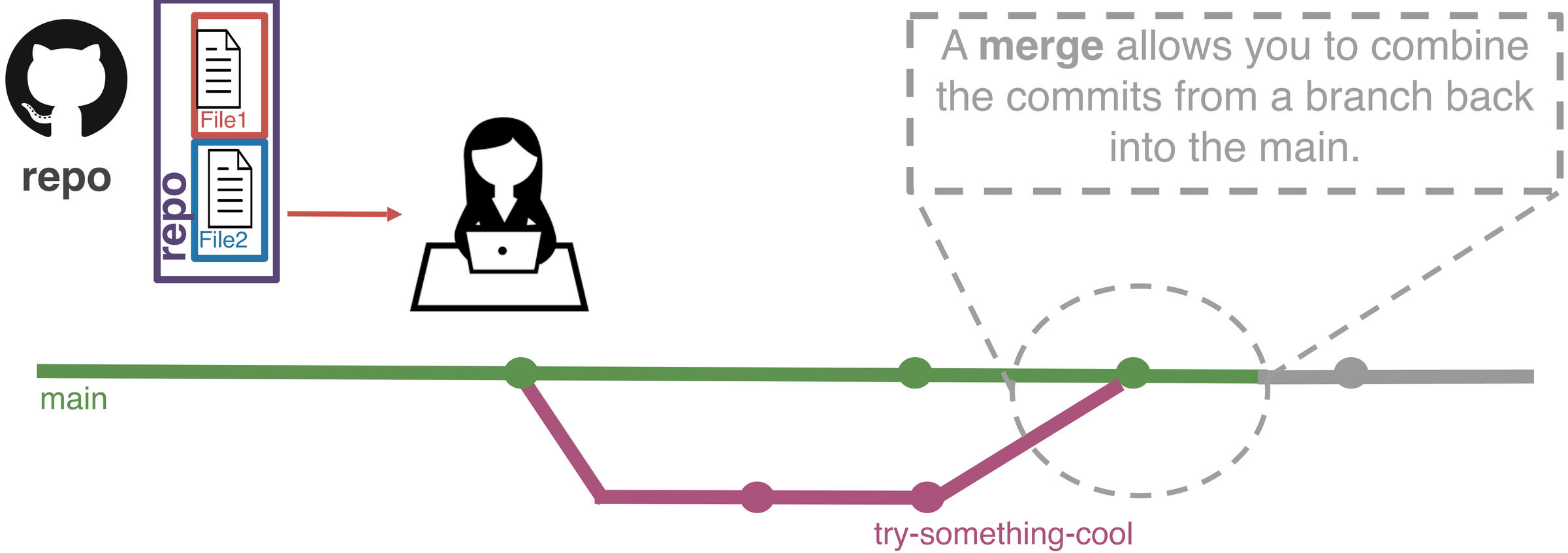


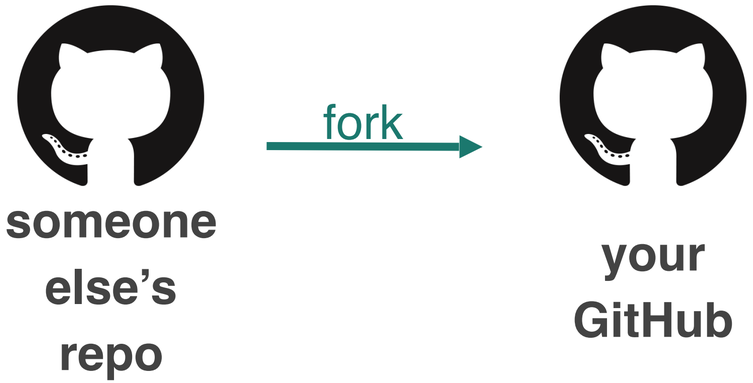
It’s a good way to experiment. It’s pretty easy to get rid of a branch later on should you not want to include the commits on that branch.



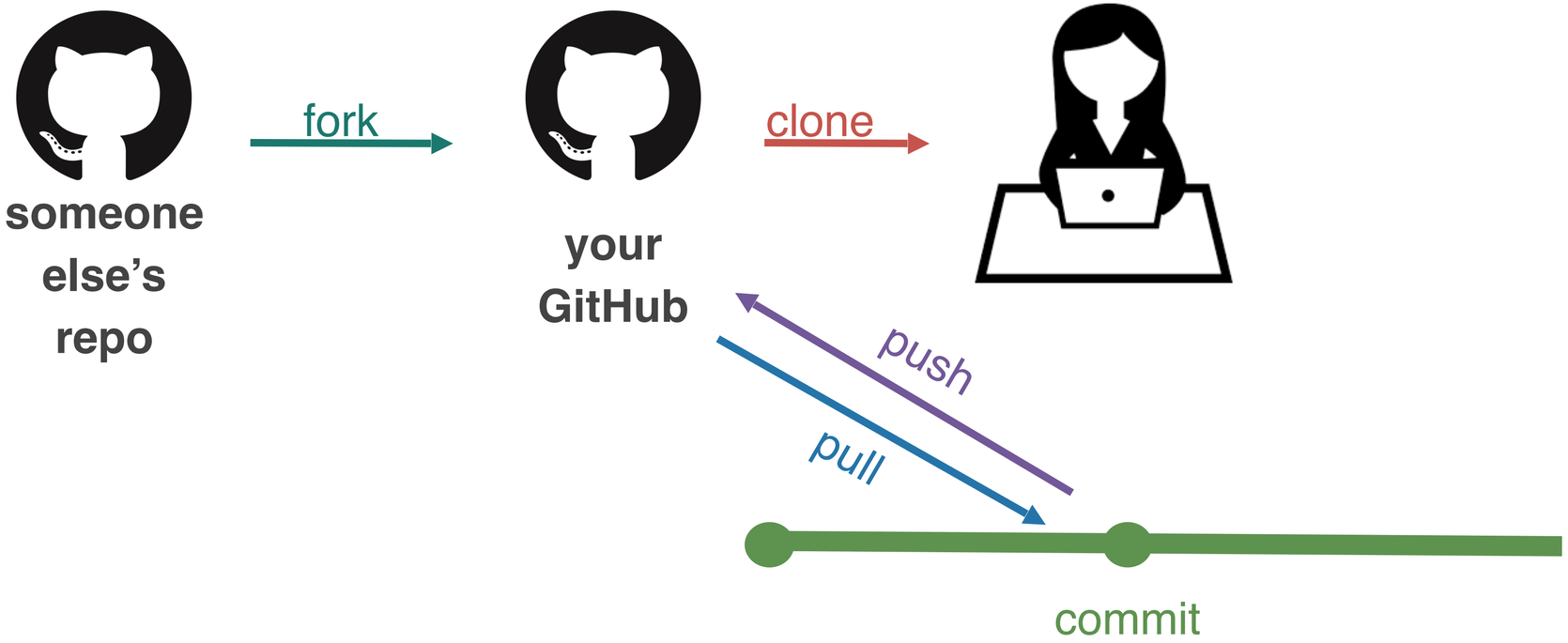
But...what if you DO want to include the changes you've made on your try-something-cool branch into the main branch?



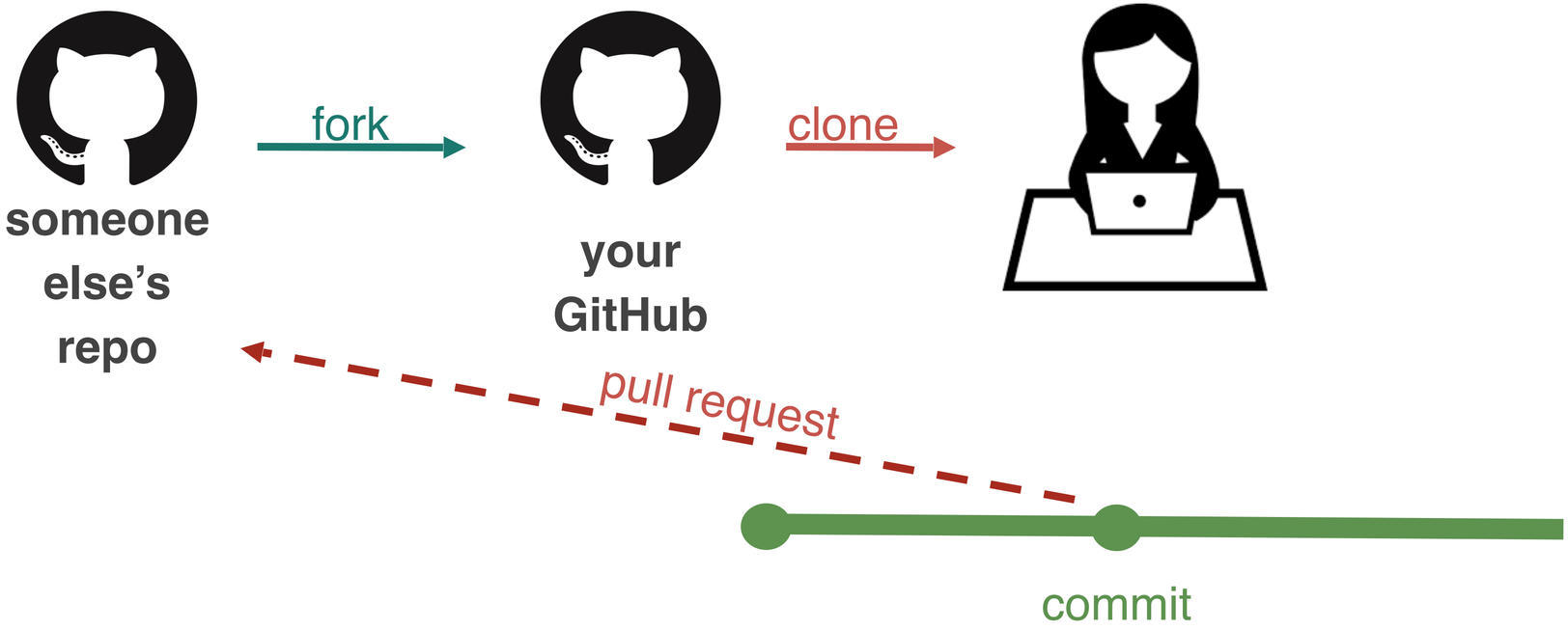

repo
repo
File1
File2
A **merge** allows you to combine the commits from a branch back into the main.
main
try-something-cool



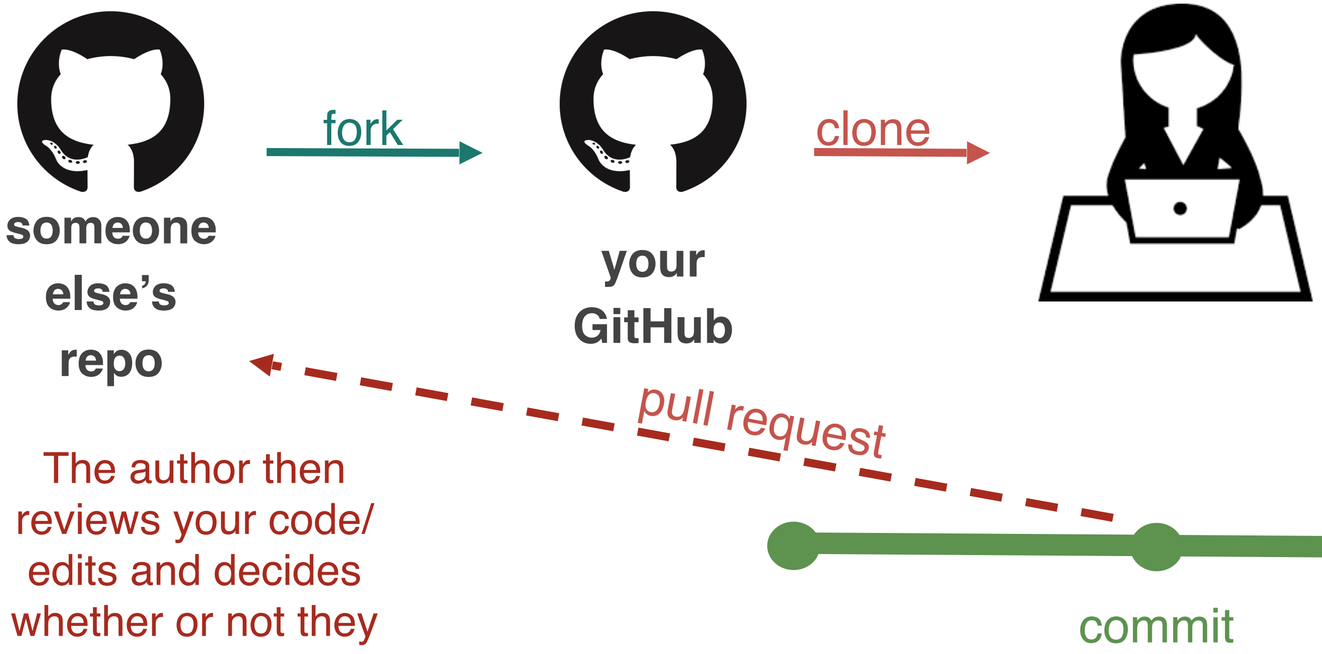
What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.



After you fork their repo, you can play around with it however you want, using the workflow we've already discussed.



But what if you think you've found a bug in their code, a typo, or want to add a new feature to their software? For this, you'll submit a **pull request** (aka **PR**).



The author then reviews your code/ edits and decides whether or not they want to **merge your pull request**.

But what if you think you've found a bug in their code, a typo, or want to add a new feature to their software? For this, you'll submit a **pull request** (aka **PR**).

someone else's repo

Last but not least...what if you find a bug in someone else's code OR you want to make a suggestion but aren't going to submit a suggestion with a PR. For this, you can file an **issue** on GitHub.

someone else's repo

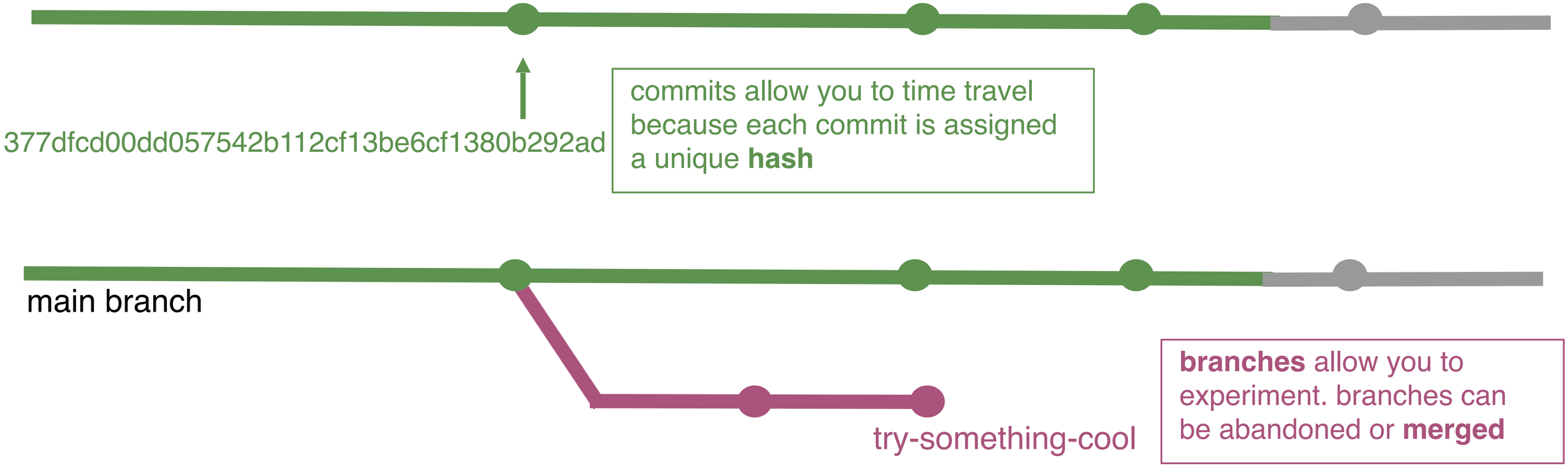
Last but not least...what if you find a bug in someone else's code OR you want to make a suggestion but aren't going to submit a suggestion with a PR. For this, you can file an **issue** on GitHub.

Issues are *bug trackers*. While, they can include bugs, they can also include feature requests, to-dos, whatever you want, really!

They can be assigned to people.

They can be closed once addressed ….or if the software maintainer doesn't like the suggestion

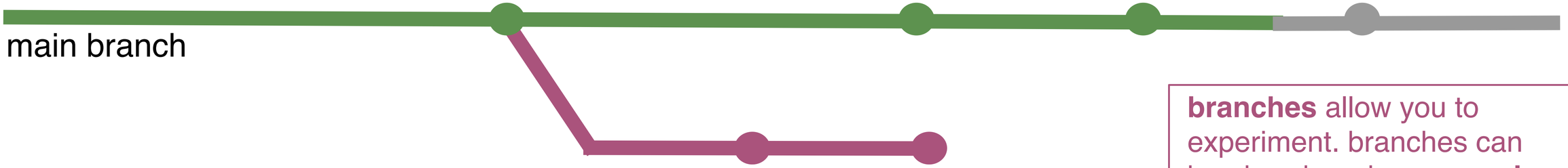
One more git recap...



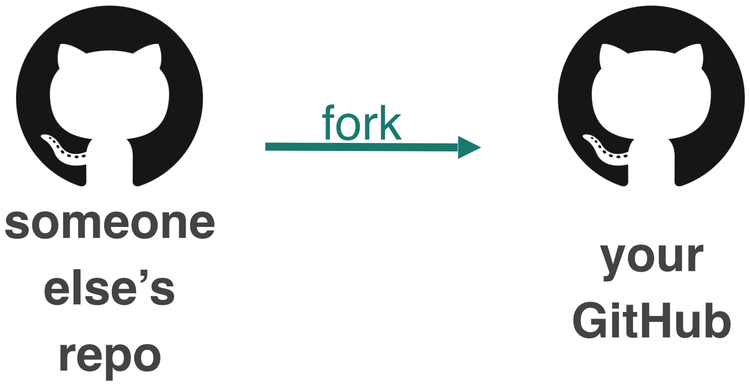
One more git recap...

377dfcd00dd057542b112cf13be6cf1380b292ad

commits allow you to time travel because each commit is assigned a unique **hash**



branches allow you to experiment. branches can be abandoned or **merged**

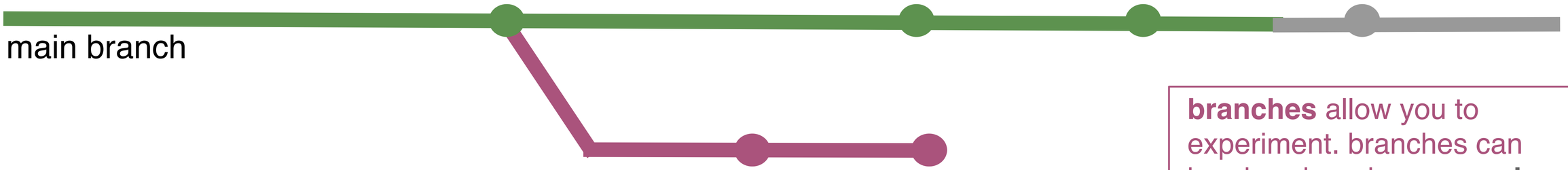


You can work on others' repos by first **forking** their repository onto your GitHub

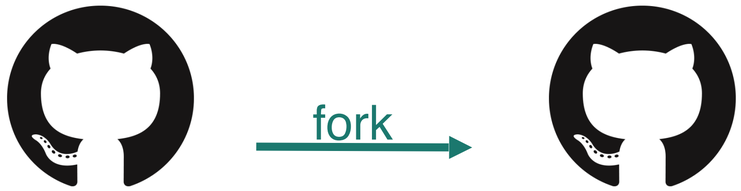
One more git recap...

377dfcd00dd057542b112cf13be6cf1380b292ad

commits allow you to time travel because each commit is assigned a unique **hash**



branches allow you to experiment. branches can be abandoned or **merged**



someone else's repo

your GitHub

You can work on others' repos by first **forking** their repository onto your GitHub

Pull requests allow you to make specific edits to others' repos

Issues allow you to make general suggestions to your/others' repos

One more git recap...

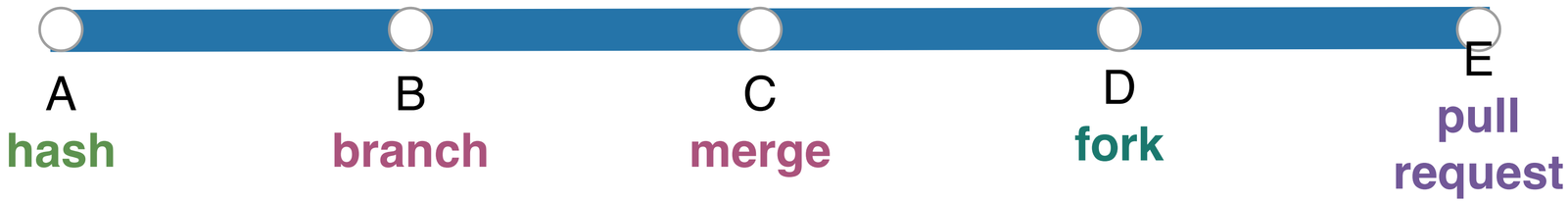
Review & Question Time

Version Controller III

https://forms.gle/eyxgHB3wvqmy17uR9

To experiment within your own repo (test out a new feature, make some changes you're not sure will work)...

what should you do?

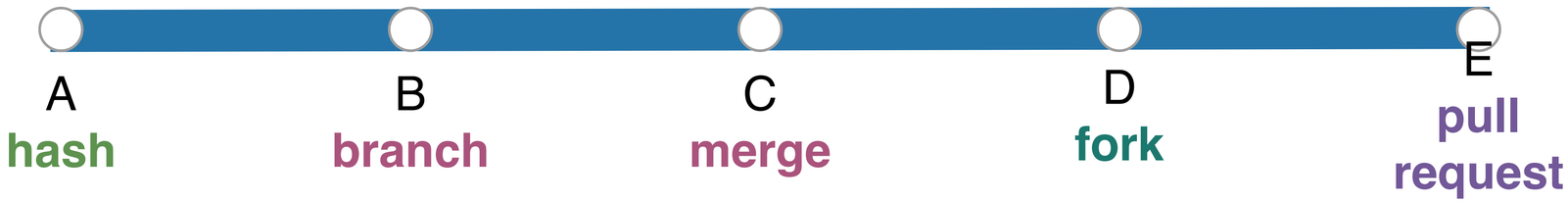


Version Controller IV

https://forms.gle/eyxgHB3wvqmy17uR9

If you've made edits to someone else's repo that you're not a collaborator on…

what would *they* have to do to incorporate your changes?



😎 Good to Git

Use it for…

- Plain text files
- < 100MB per file limit
- Total repository size < 5 GB

⚠️ Careful

Easy to mess up

- Metadata rich text files that change over time, e.g.
  - Jupyter Notebook .ipynb
  - Word processor formats like .docx
  - JSON data

🙅 Don’t Git

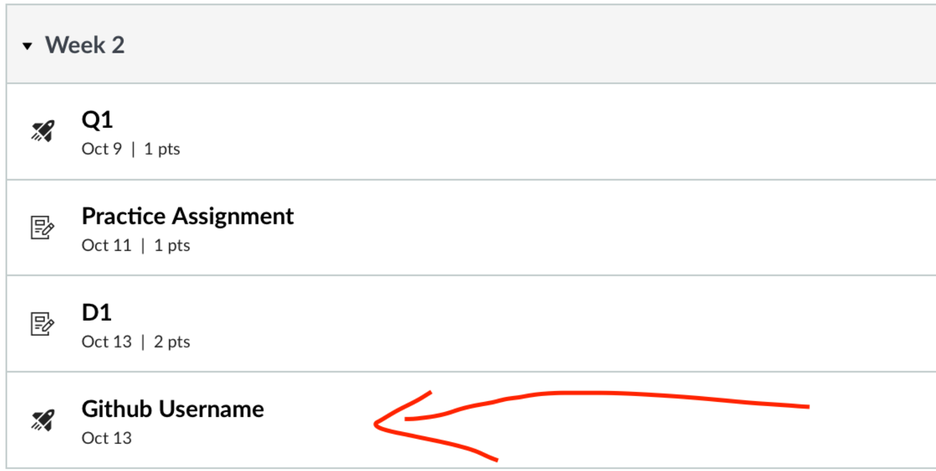
Avoid using for

- Large text files
  ❌ 648MB .
- Vast directory structures
  ❌ 36,142,087 small text files in a nested subdirectory tree up to 14 levels deep
- Binary files
  ❌ .jpg, .mp4, .sql, etc

✅ Git LFS can help!

Version Control: Practice

- GO TO GITHUB AND GET A USERNAME! We need it for many things in the class
- You will use git & Github when you do
  - Discussion Lab 1
  - Assignment 1
- Understand what you're doing in the assignment!
- You may have to google, ask others, spend some time with this!



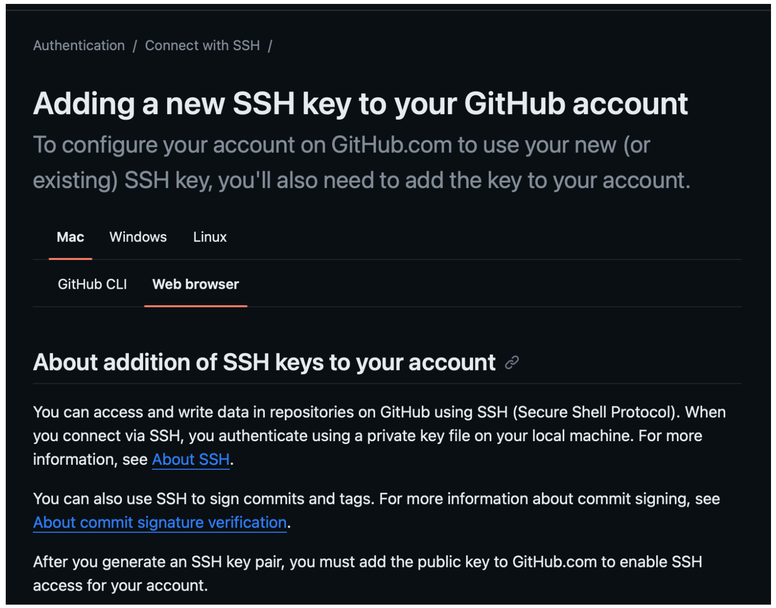
- git & Github == How to get the course lectures/materials
  - Assignment 1 will have you fork the Lectures and Project repos
  - You can keep the lectures up-to-date throughout the quarter
- you'll be using GitHub for your final projects
- Fill in the quiz before the end of week 2 so we have your username for creating project repos!

Using SSH to contribute work to your project repo

You can’t write or access private repos without authentication!

Click here, follow instructions:

https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account



You will need to do this for each local computer you will use (e.g., on Datahub and again on your laptop!)

GitFu level: advanced

Installing git
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Installing GH GLI
https://cli.github.com/

Other options include Github Desktop, Source Tree, VSCode, or any other IDE

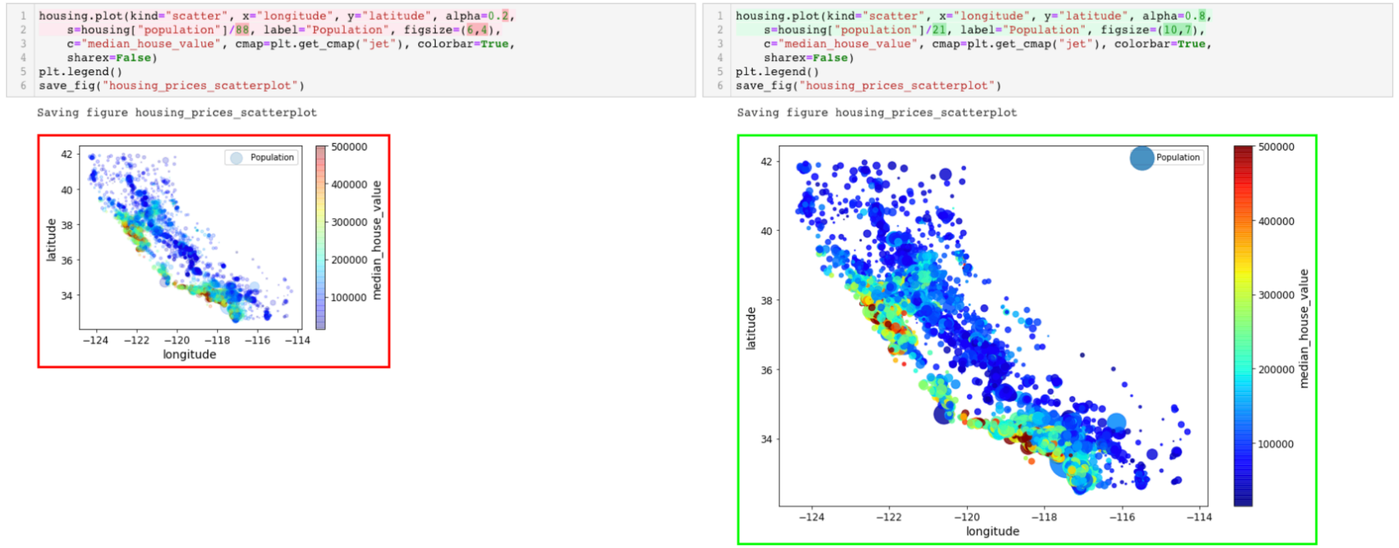
GO TO GITHUB

Jupyter notebooks suck to version control

https://nextjournal.com/schmudde/how-to-version-control-jupyter

ReviewNB

ReviewNB is a GitHub app that also offers visual diffing with an interface that looks similar to the traditional Jupyter IDE. Because the outputs are visualized, problems associated with committing binary blobs disappear.

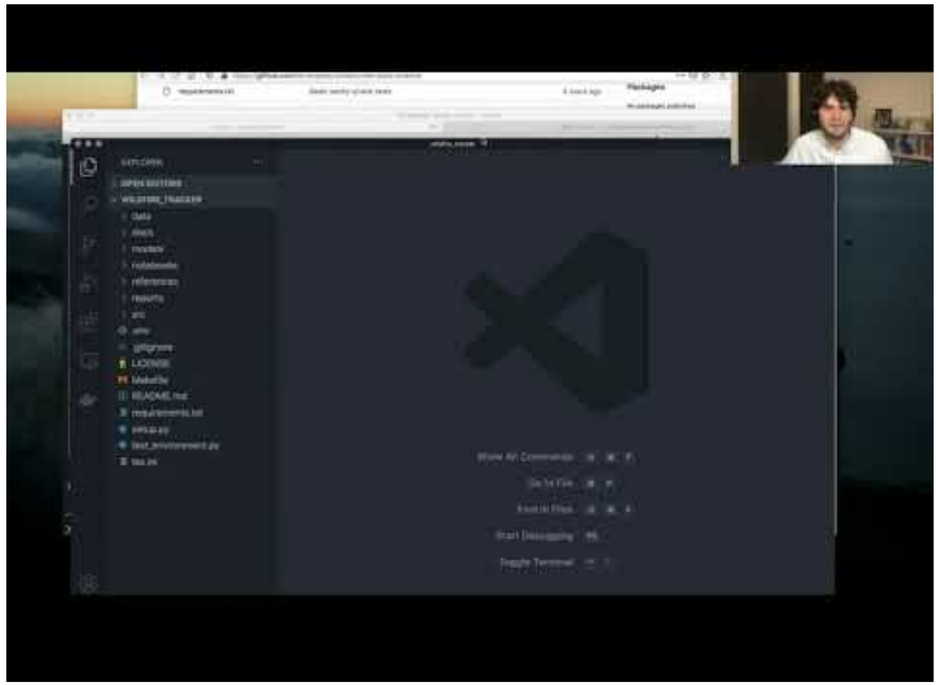


ReviewNB example courtesy of the ReviewNB website

More options

nbautoexport

> Making it easier to code review Jupyter notebooks, one script at a time.



-jupy
+text

Lets practice!

At least if we have enough time right now :)

- Create repo
- Create file.md
- Create a branch “my_work”
- On “my_work”: Add a new file, prog.py
- On “my_work”: Change contents of file.md
- On “master”: Change contents of file.md in a way that conflicts with “my_work”
- Merge “my_work” into “master”
- Resolve conflicts, finish merge

Practice

You need to learn by doing. Listening to me blah blah is only going to go so far. Here's some choices that seem to me like they would be good for beginner to intermediate levels, but note I have not actually used them.

- Free 16 hour Coursera course by Google covering common Git usage patterns https://www.coursera.org/learn/introduction-git-github
- Katas for Git https://github.com/eficode-academy/git-katas

- If you have other suggestions please let me know!
- If you use these and like or hate them please give me feedback!