Logistics

- Due next Monday, released Friday
  - Quiz 1
- Due next Wed, released already or released today
  - Pre course survey for EC
    - Section 1 ==> EC
    - Section 2 ==> Participation in the pedagogical experiment
  - Practice Assignment on Datahub
- Due next Fri, released this Fri or already released
  - D1 on Dathub
  - #FinAid quiz on Canvas
- Discussion Lab this week:
  - Python review, nothing to turn in

The COGS 108 Final Project will give you the chance to explore a topic of your choice and to expand your analytical skills. By working with real data of your choosing you can examine questions of particular interest to you.

- You are encouraged to work on a topic that matters to the world (your family, your neighborhood, a state/province, country, etc).
- Taboo Topics: Movie Predictions/Recommendation System; YouTube Data Analysis, Kickstarter success prediction/analysis,prediction of what makes a song popular on Spotify Whatever is MOST popular EVER and whatever is HOTTEST RN on Kaggle

Final Project: Objectives

- Identify the problems and goals of a *real* situation and dataset.
- Choose an appropriate approach for formalizing and testing the problems and goals, and be able to articulate the reasoning for that selection.
- Implement your analysis choices on the dataset(s).
- Interpret the results of the analyses.
- Contextualize those results within a greater scientific and social context, acknowledging and addressing any potential issues related to privacy and ethics.
- Work effectively to manage a project as part of a team.

Upcoming Project Components

Groups fixed in week 3

Project Review (5%) - your group will be assigned a previous COGS 108 project to review; A google form will be released to guide your thinking/discussion about and review of what a previous COGS 108 group did for their project. (Week 4)

Project Proposal (8%) - a GitHub repo will be created for your group; ‘submit’ on GitHub (Week 5)

Project Proposal (8%)

Full project guidelines are here:
https://github.com/COGS108/Projects/blob/master/FinalProject_Guidelines.md

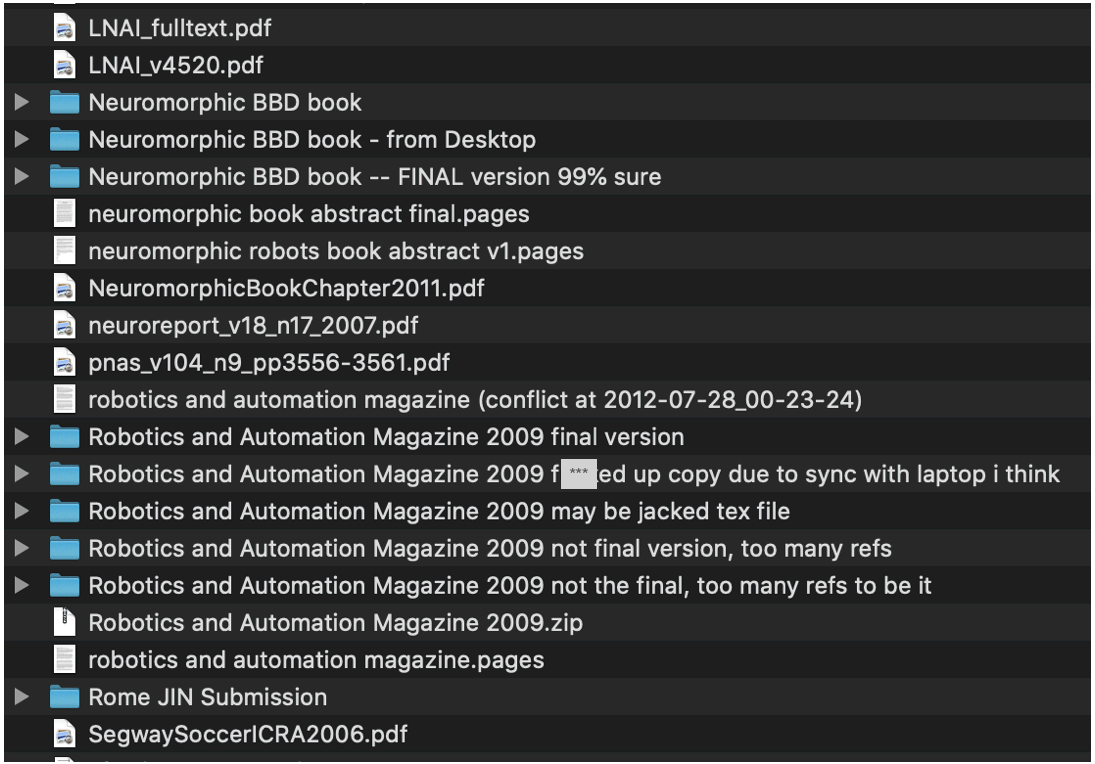
Version Control

Jason G. Fleischer, Ph.D
UC San Diego

Department of Cognitive Science
jfleischer@ucsd.edu
https://jgfleischer.com
@jasongfleischer

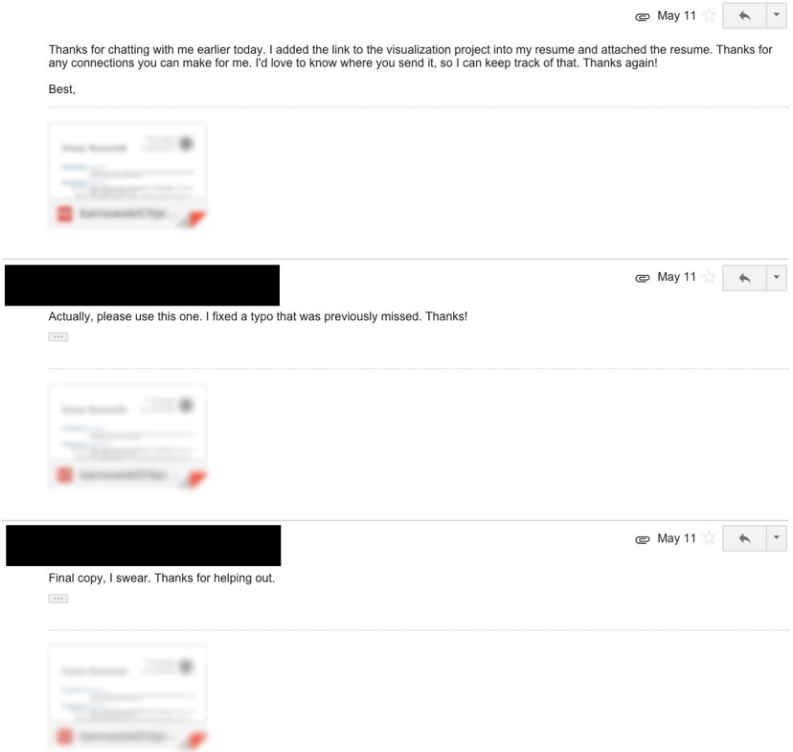
This sucks

archived version of my Documents folder from ~ 2012

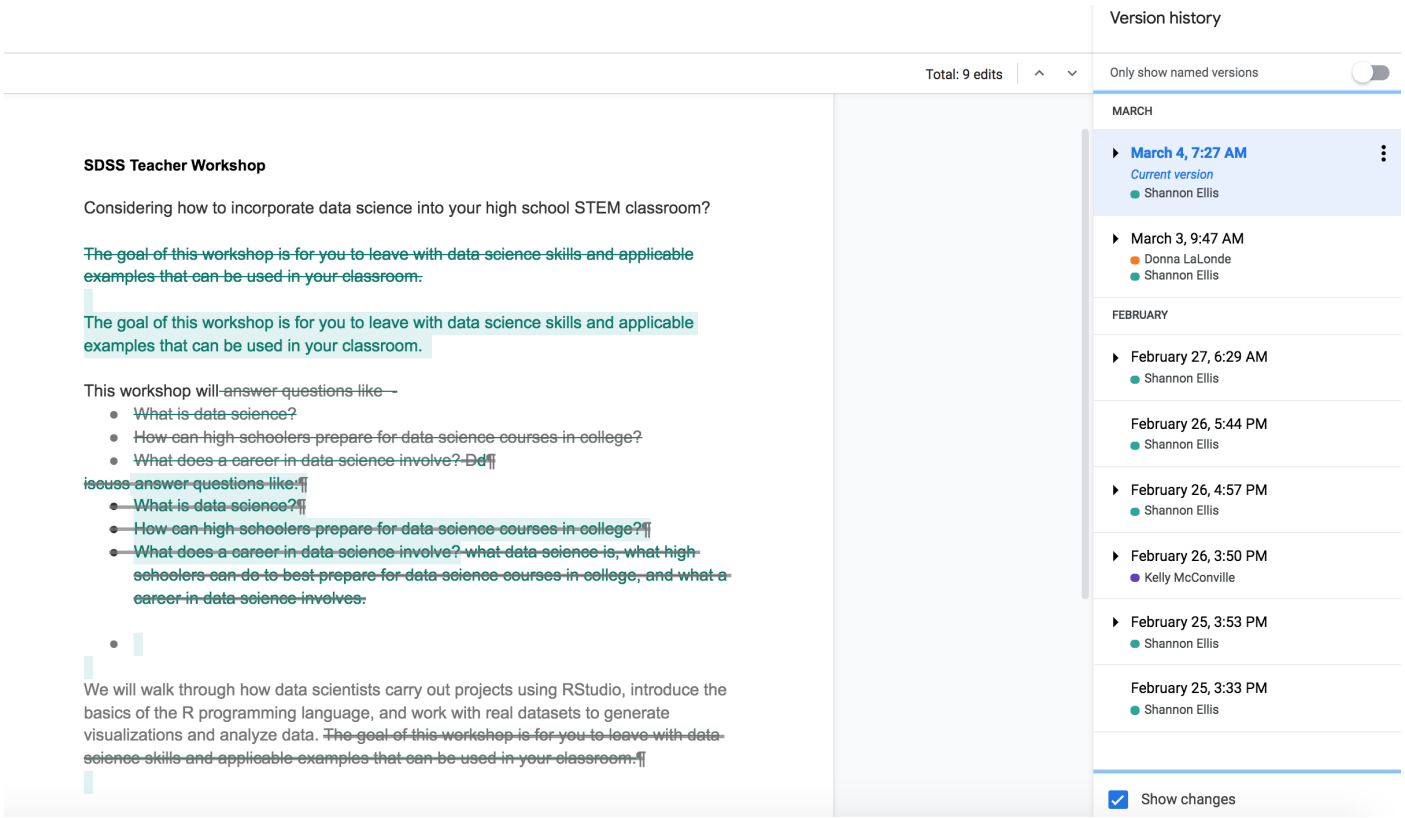


Several months after finishing a writing project, I wanted to keep only the final version of the many different revisions… figuring out which one was the version actually sent to the publisher was hard!

Yup, this sucks too.



This is a step in the right direction



Version Control

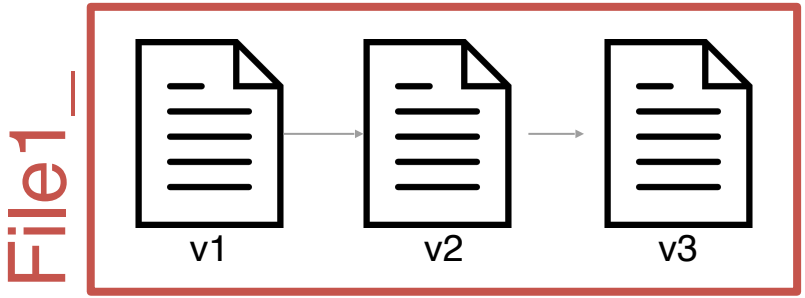
- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.
- Thus, temporary or partial edits by one person do not interfere with another person's work

What is version control?

A way to manage the evolution of a set of files

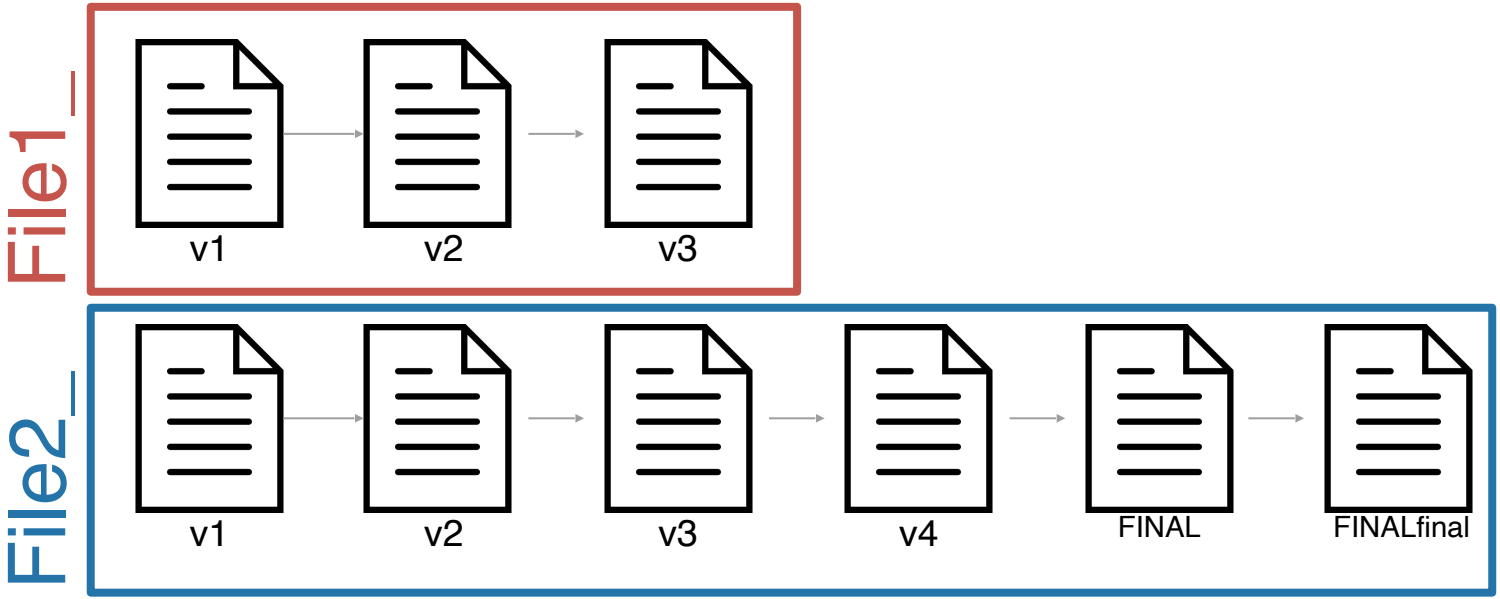
What is version control?

A way to manage the evolution of a set of files



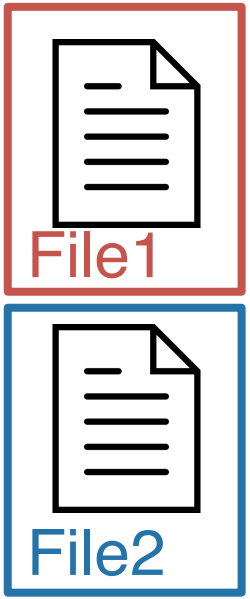
What is version control?

A way to manage the evolution of a set of files



What is version control?

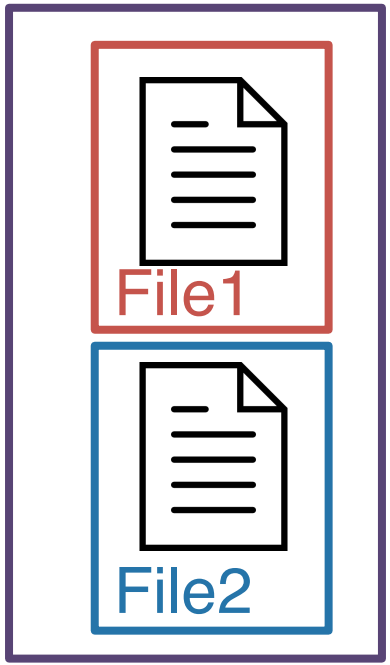
A way to manage the evolution of a set of files



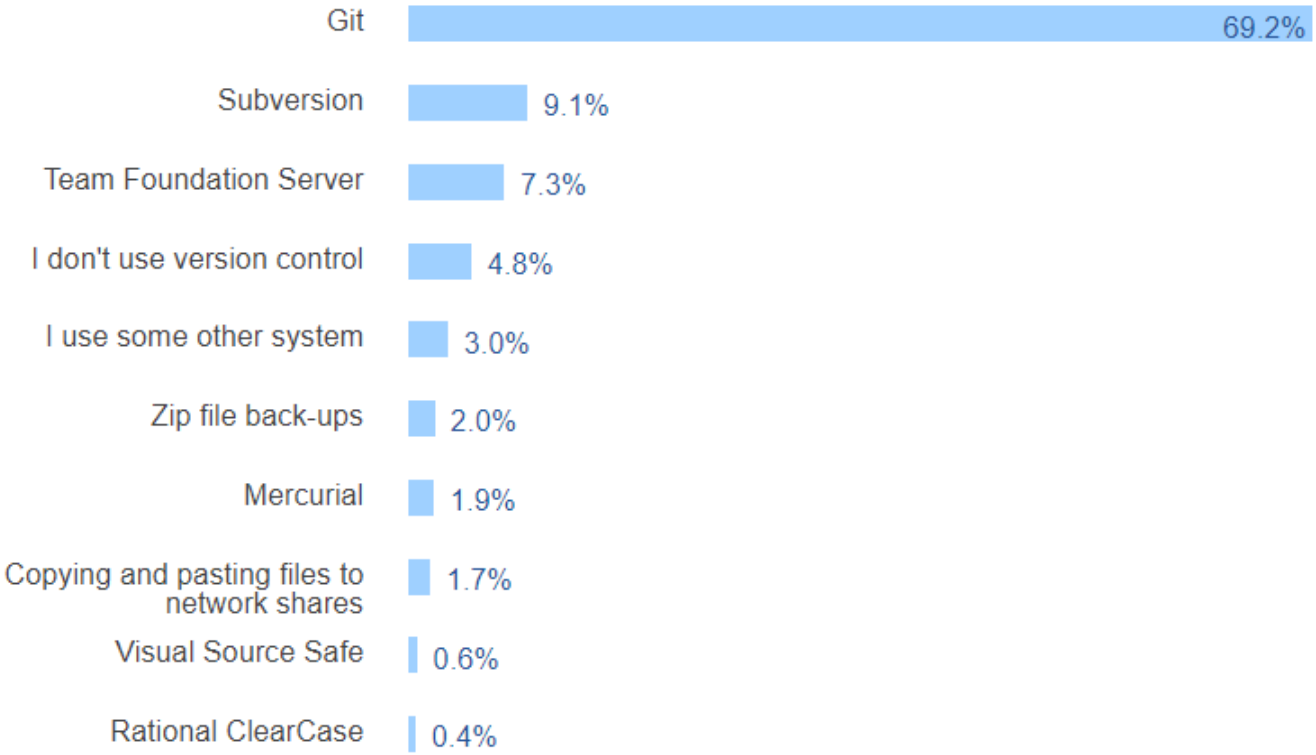
When using a version control system, you have **one copy of each file** and the *version control system tracks the changes* that have occurred over time

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)**

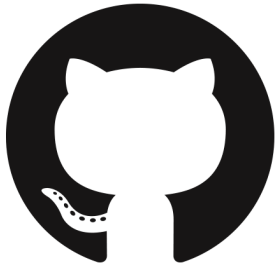

Git
69.2%
Subversion
9.1%
Team Foundation Server
7.3%
I don't use version control
4.8%
I use some other system
3.0%
Zip file back-ups
2.0%
Mercurial
1.9%
Copying and pasting files to network shares
1.7%
Visual Source Safe
0.6%
Rational ClearCase
0.4%

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.

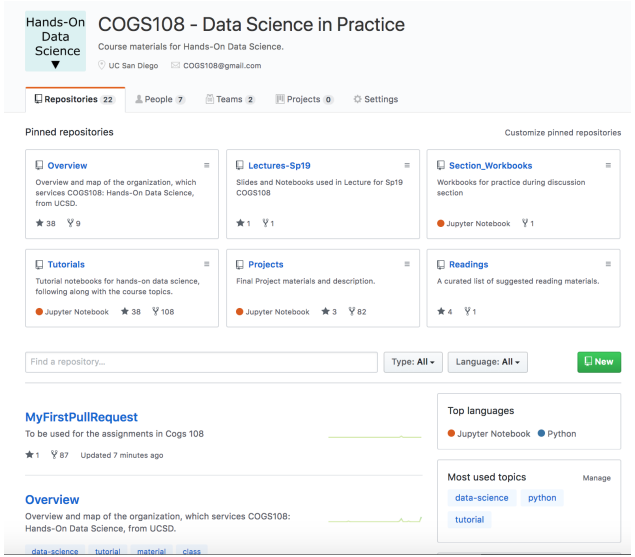
~ Dropbox + social media for programmers

What version control looks like

```
$ git clone https://www.github.com/username/repo.git
$ git pull
$ git add -A
$ git commit -m "informative commit message"
$ git push
```

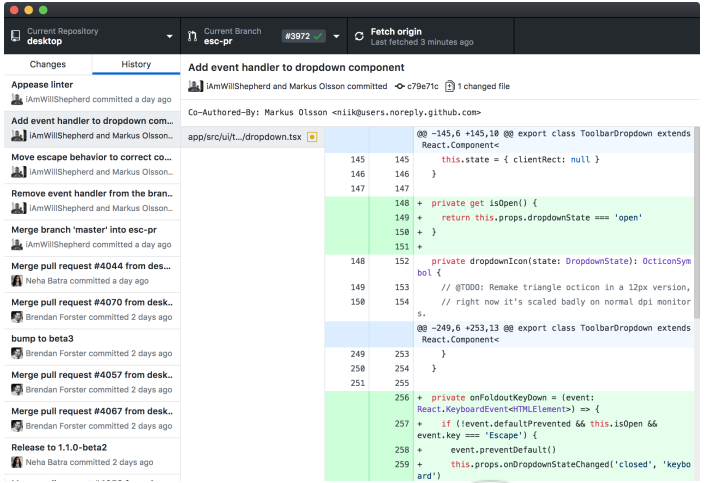
Terminal

git

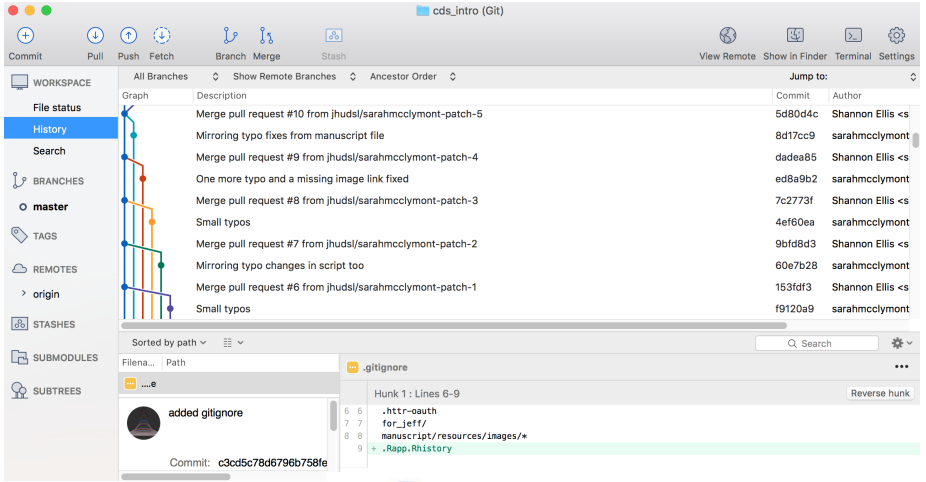


GitHub

GUIs can be helpful when working with version control



GitHub Desktop



SourceTree

https://forms.gle/8UeUL2Ux4YtG2CVr8

Version Controller

How do you typically interact with git?

A. I don't

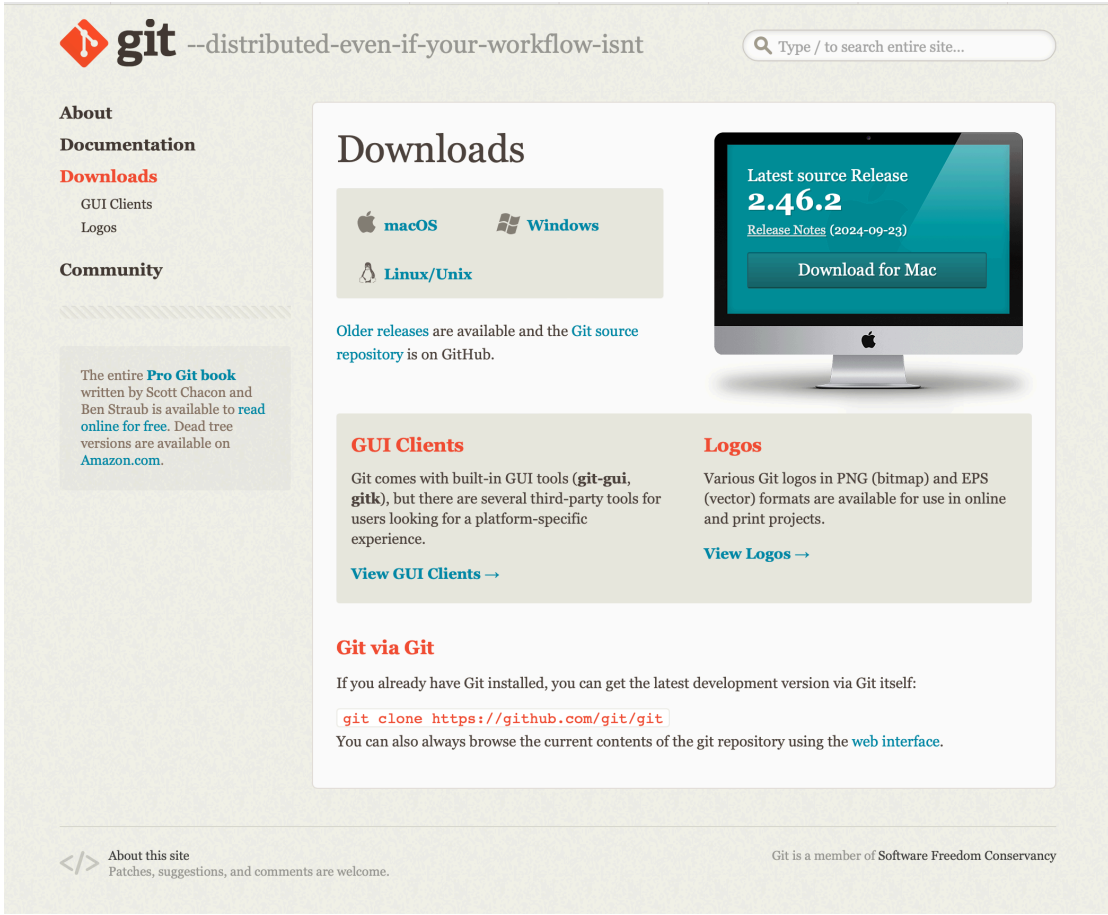
B. command line

C. GUI GitHub Desktop

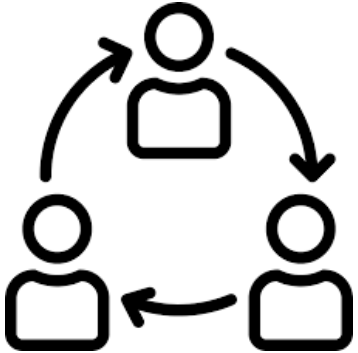
D. GUI: SourceTree

E. GUI: other

https://git-scm.com/downloads



Why version control with git and GitHub?



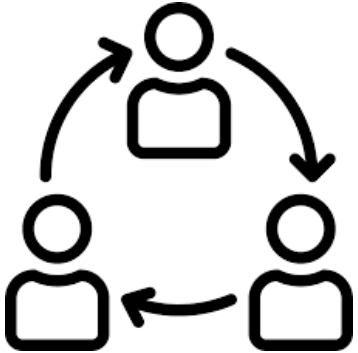
Collaboration

Returning to a safe state

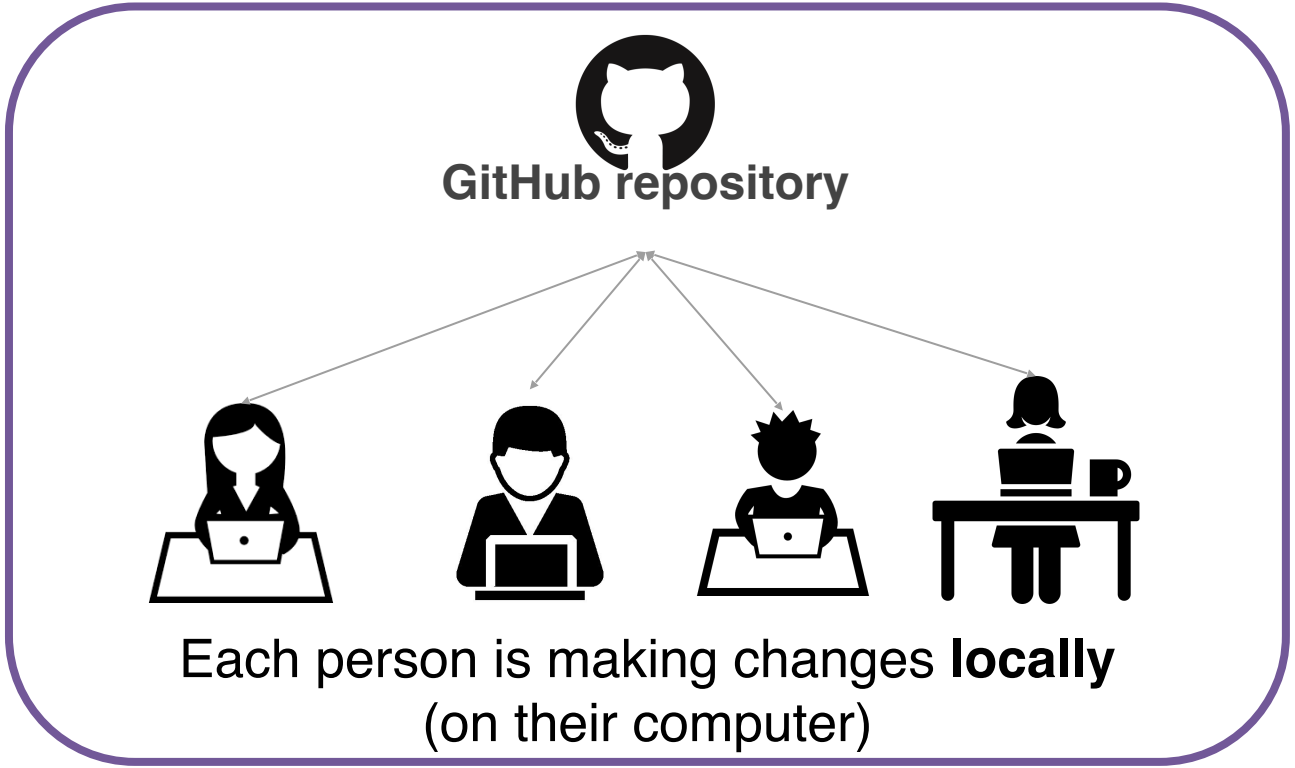
Exposure for your work

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

Code Issues Pull requests

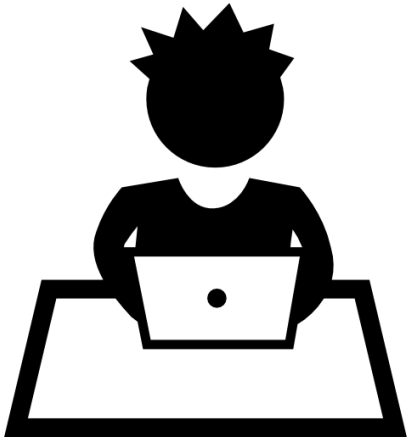
A simple bot to rickroll your friends on Discord!

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

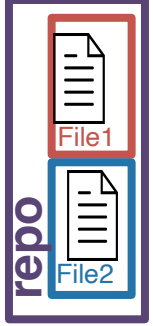
Keep up with others' work easily

Tracking others' work



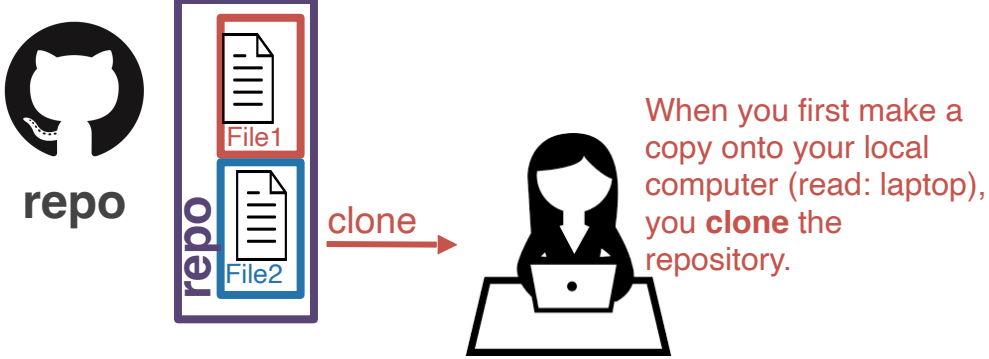
As a social platform, you can see others' work too!

repo

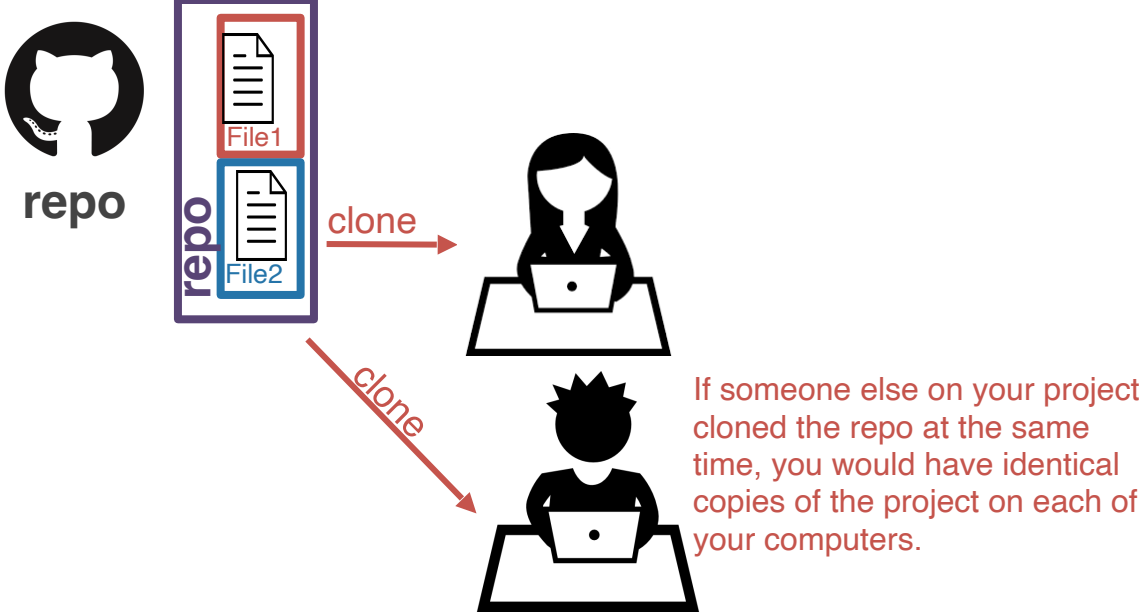


A **GitHub repo** contains all the files and folders for your project.

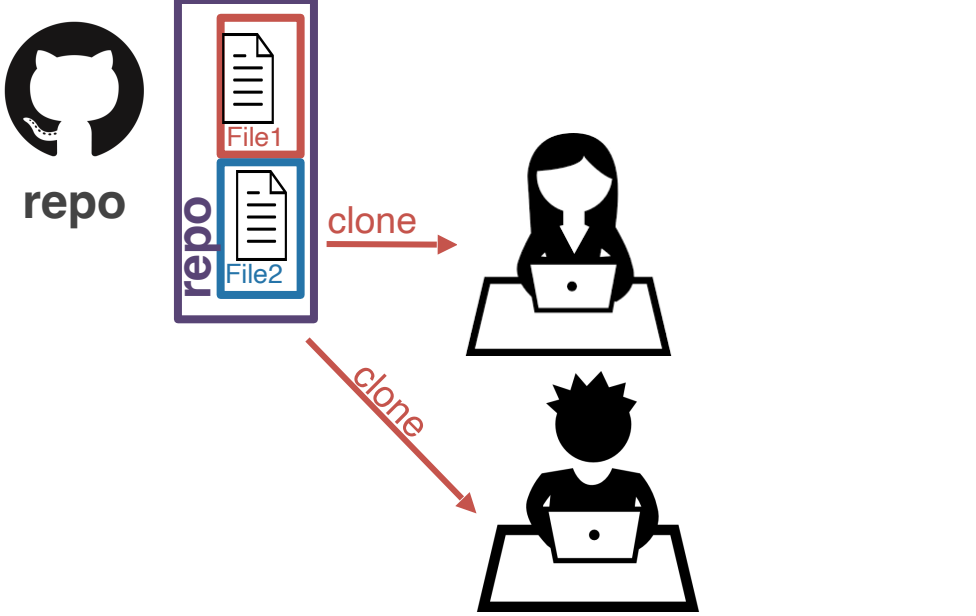
GitHub is a **remote host**. The files are geographically distant from any files on your computer.



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



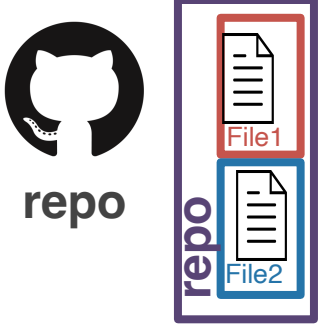
If someone else on your project cloned the repo at the same time, you would have identical copies of the project on each of your computers.

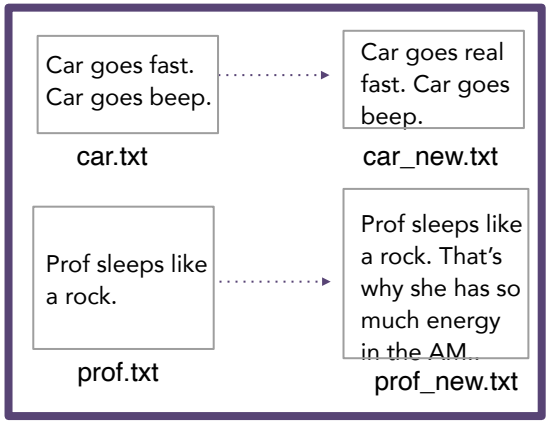


Yay! Everyone can work on the project!

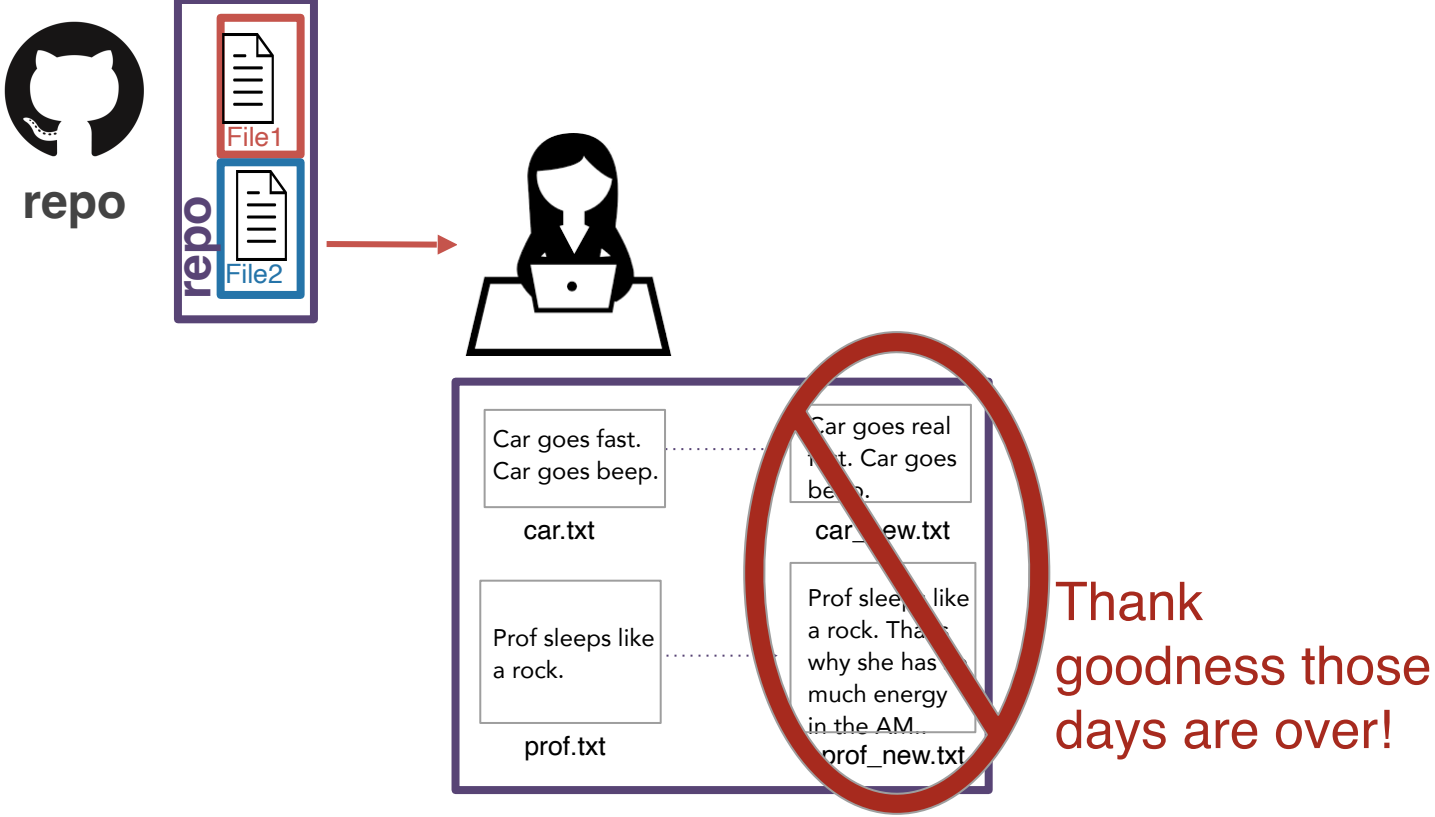

repo
repo
File1
File2
You decide you want to change some of the text in the project.

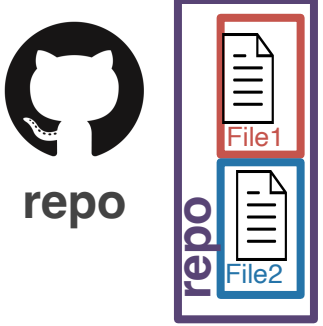

repo
repo
File1
File2
You decide you want to change some of the text in the project.

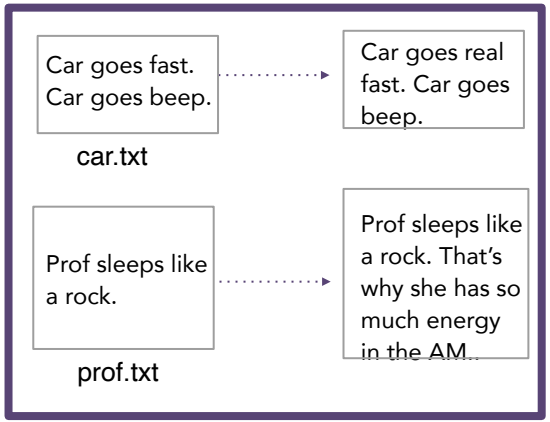




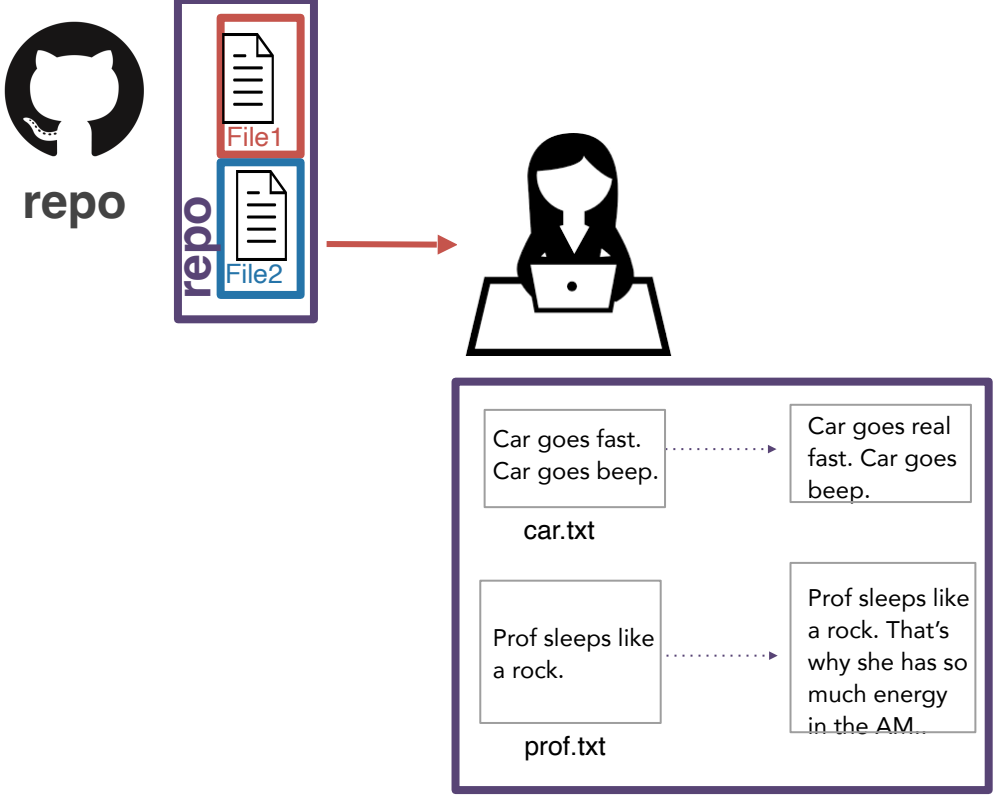
without git...you'd likely rename these files….


repo
repo
File1
File2
Car goes fast.
Car goes beep.
car.txt
Prof sleeps like
a rock.
prof.txt
Thank
goodness those
days are over!



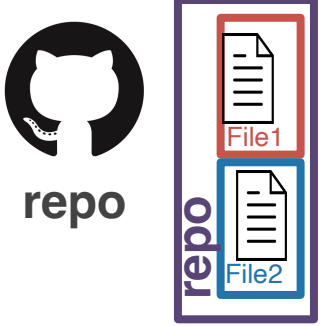


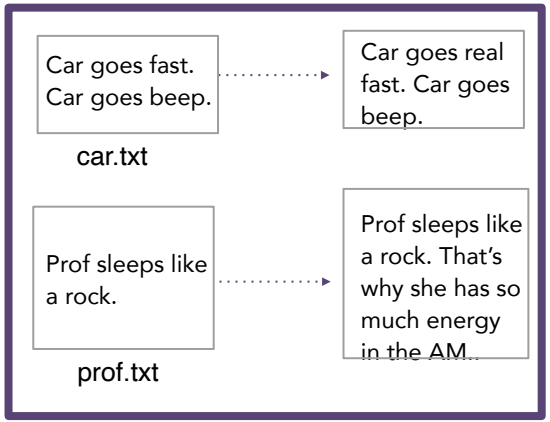
Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.



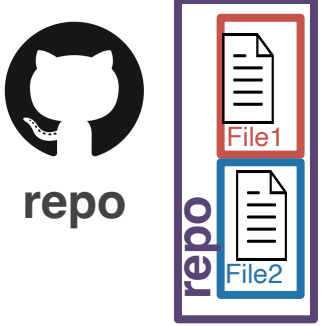
`git add file`	stages specified file (or folder)
`git add .`	stages new and modified files
`git add -u`	stages modified and deleted files
`git add -A`	stages new, modified, and deleted files
`git add *.csv`	Stages any files with .csv extension
`git add *`	Use with caution: stages everything

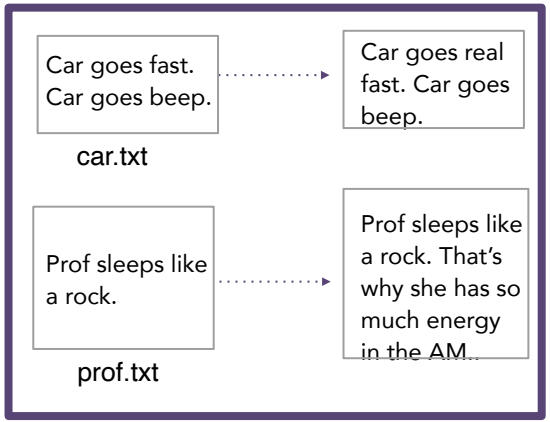
Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.





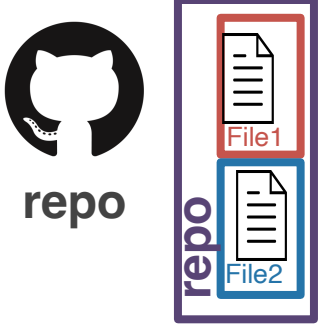
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

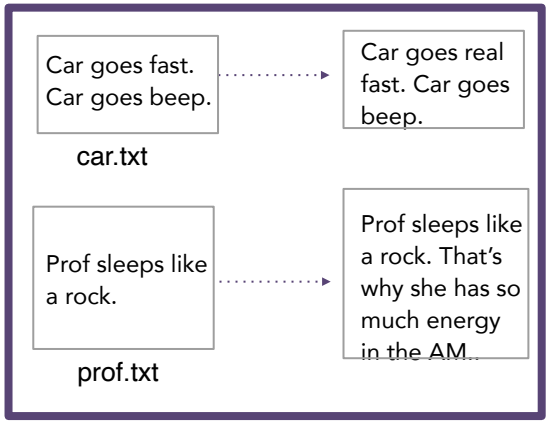




Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

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





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

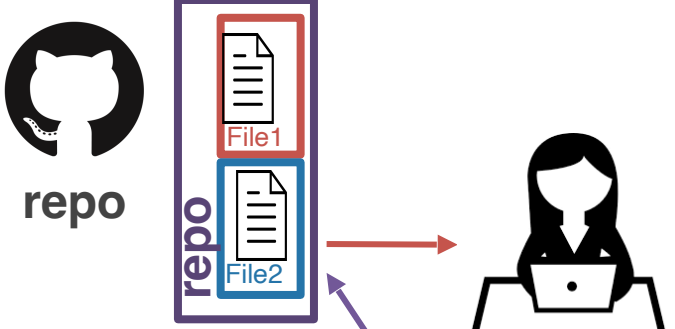
Example: `git commit -m 'fix typos in car and prof'`

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

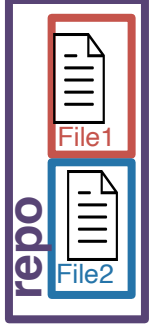
Shannon Ellis
3/28/21 3:28pm

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

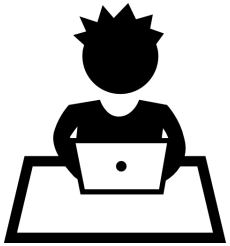
fix typos in car and prof



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

Shannon Ellis
3/28/21 3:28pm

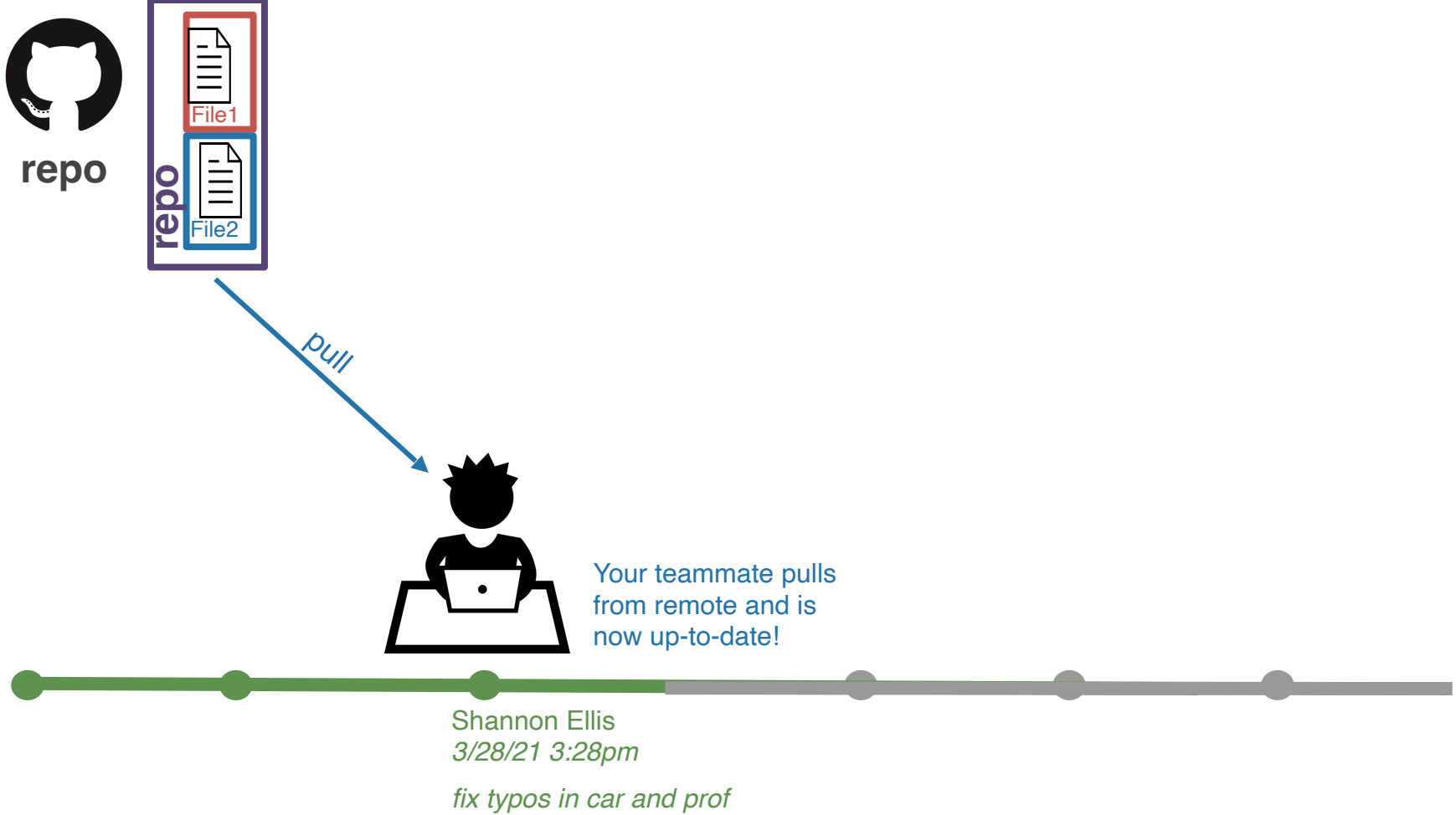
fix typos in car and prof

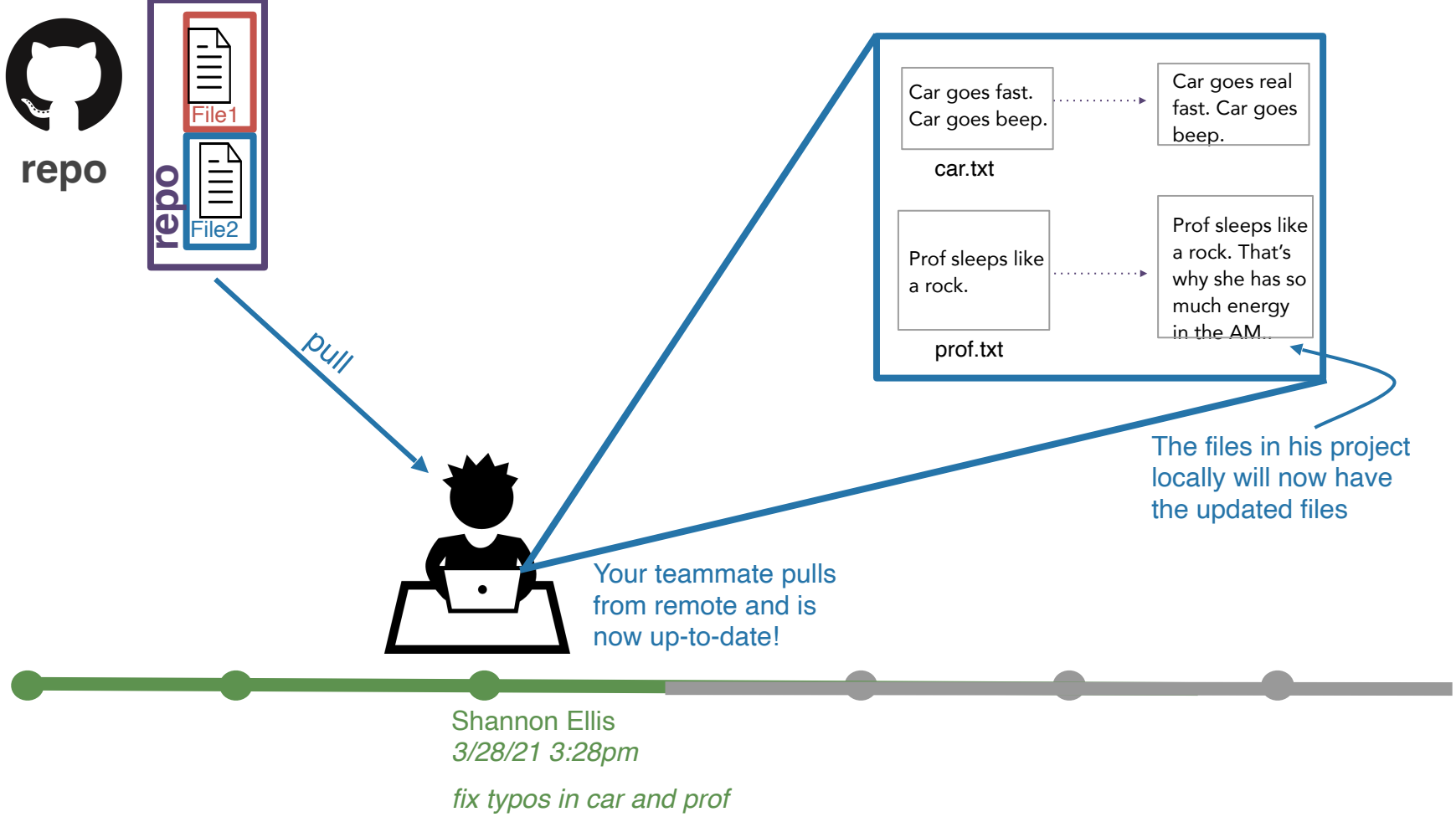


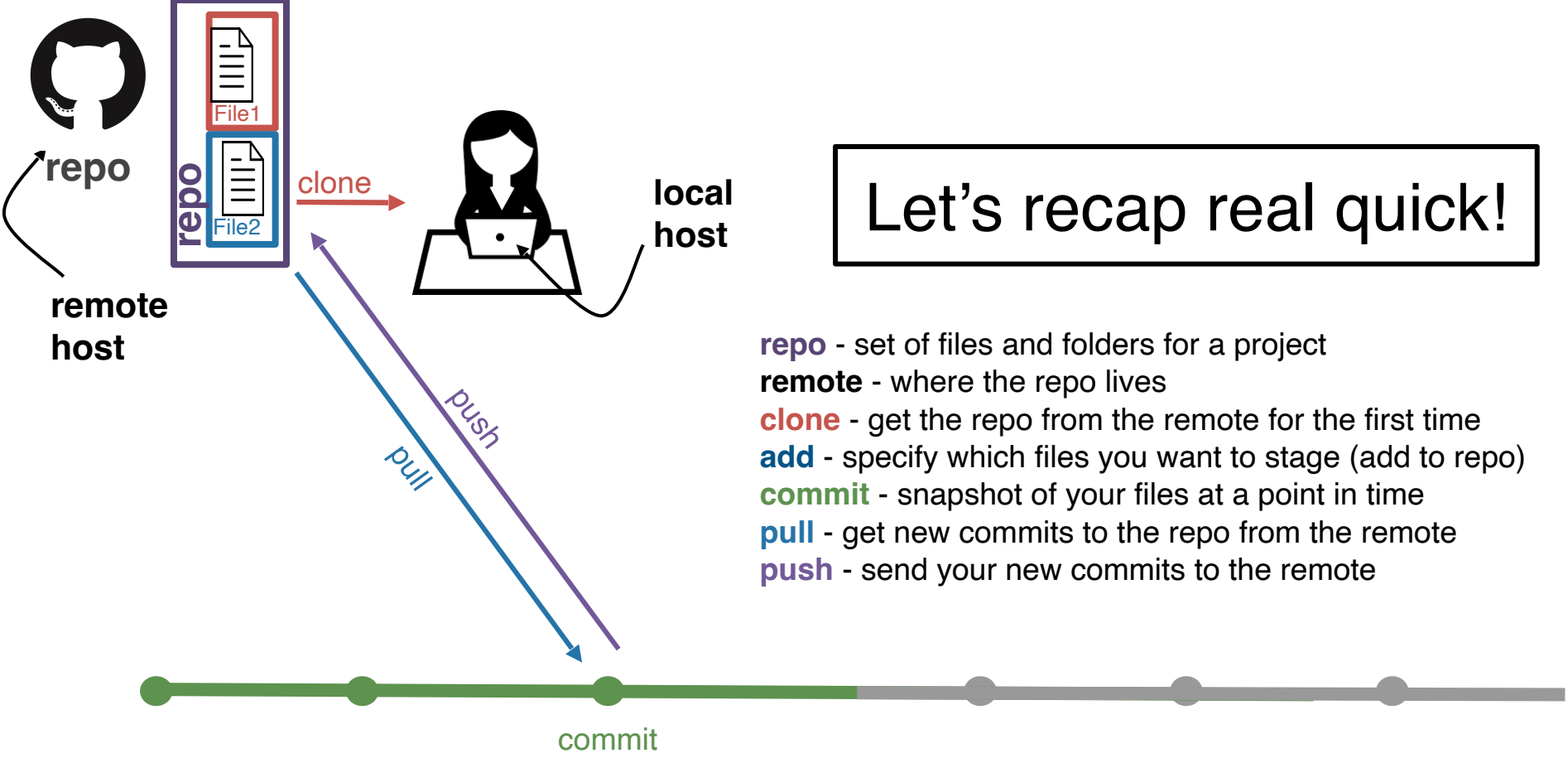
Your teammate is still working with the (out-of-date) copy he cloned earlier!

Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof


repo
repo
File1
File2
pull
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 goes real fast. Car goes beep.
car.txt
Prof sleeps like a rock.
Prof sleeps like a rock. That's why she has so much energy in the AM.
prof.txt
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

```
(base) jasonfleischer@rabona COGS108 % cd Lectures-Fa23
(base) jasonfleischer@rabona Lectures-Fa23 % git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Week_01/.ipynb_checkpoints/
        Week_01/01_03_and_04_version_control.pdf

nothing added to commit but untracked files present (use "git add" to track)
(base) jasonfleischer@rabona Lectures-Fa23 % git add Week_01/01_03_and_04_version_control.pdf
(base) jasonfleischer@rabona Lectures-Fa23 % git commit -m "lectures 3 and 4"
[main 08fae3a] lectures 3 and 4
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Week_01/01_03_and_04_version_control.pdf
(base) jasonfleischer@rabona Lectures-Fa23 % git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 973.36 KiB | 24.96 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/COGS108/Lectures-Fa23.git
   a6d9a0e..08fae3a  main -> main
(base) jasonfleischer@rabona Lectures-Fa23 %
```

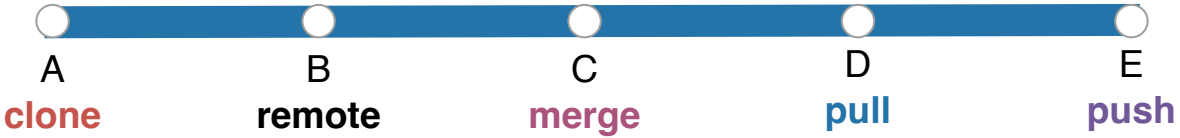
Review & Question Time

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.

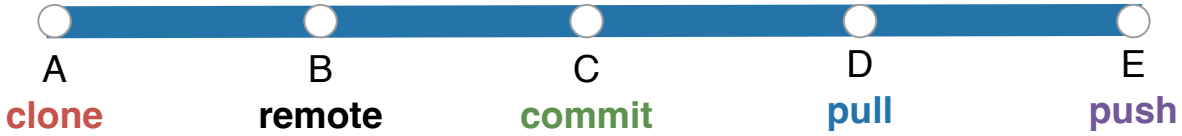
What do you do to get them on the remote?



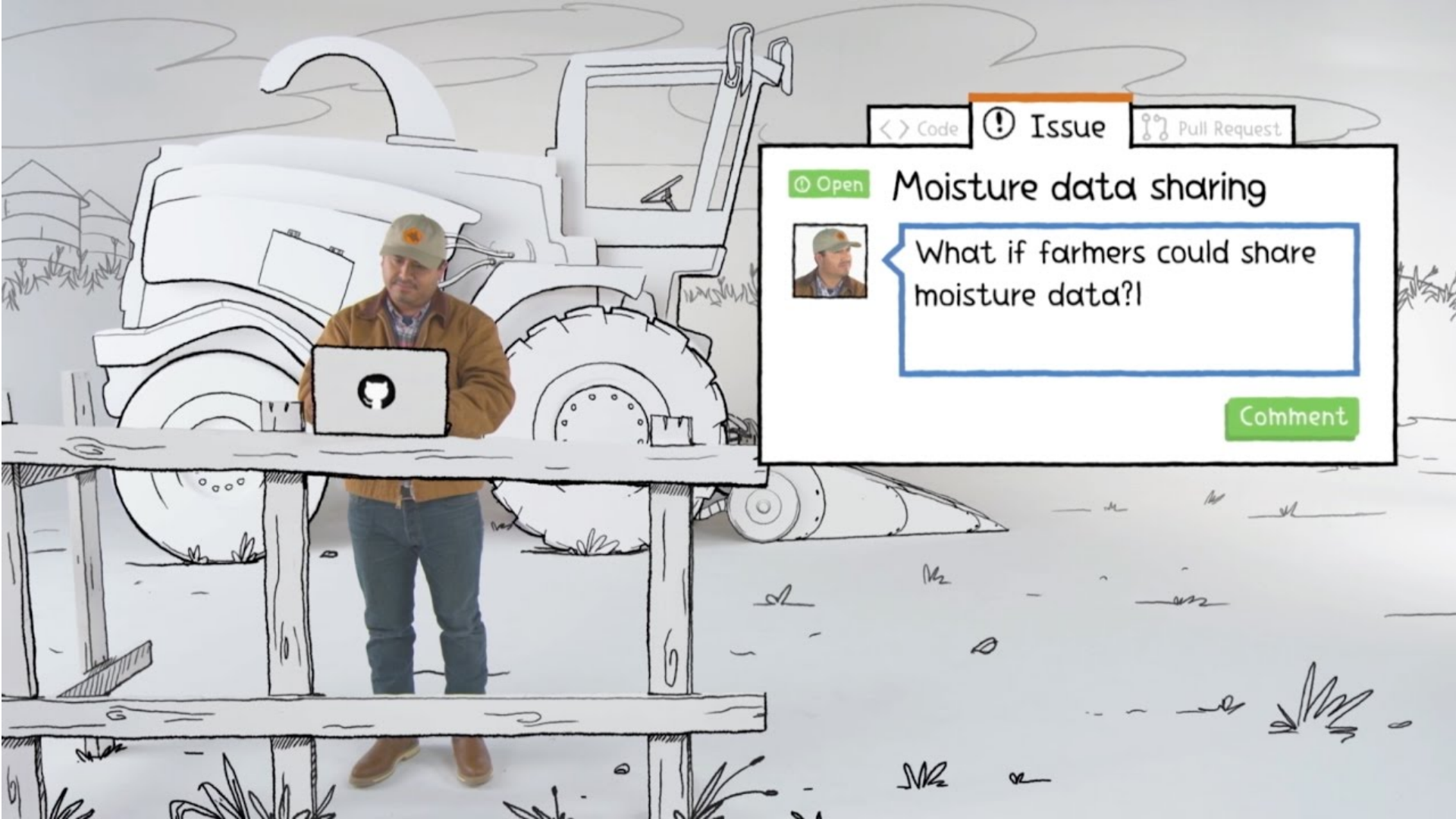
Version Controller II

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?



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

Git exercise #1

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

   `git clone` https://github.com/COGS108/Lectures-Sp25
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!

Usually stop here for time

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!

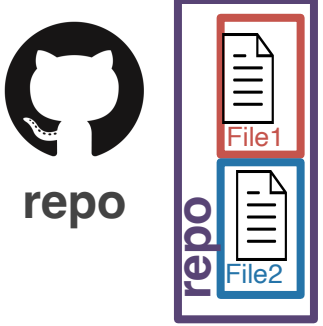
By committing each time you make changes, git allows you to time travel!

377dfcd00dd057542b112cf13be6cf1380b292ad

439301fe69e8f875c049ad0718386516b4878e22

456722223e9f9e0ee0a92917ba80163028d89251

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



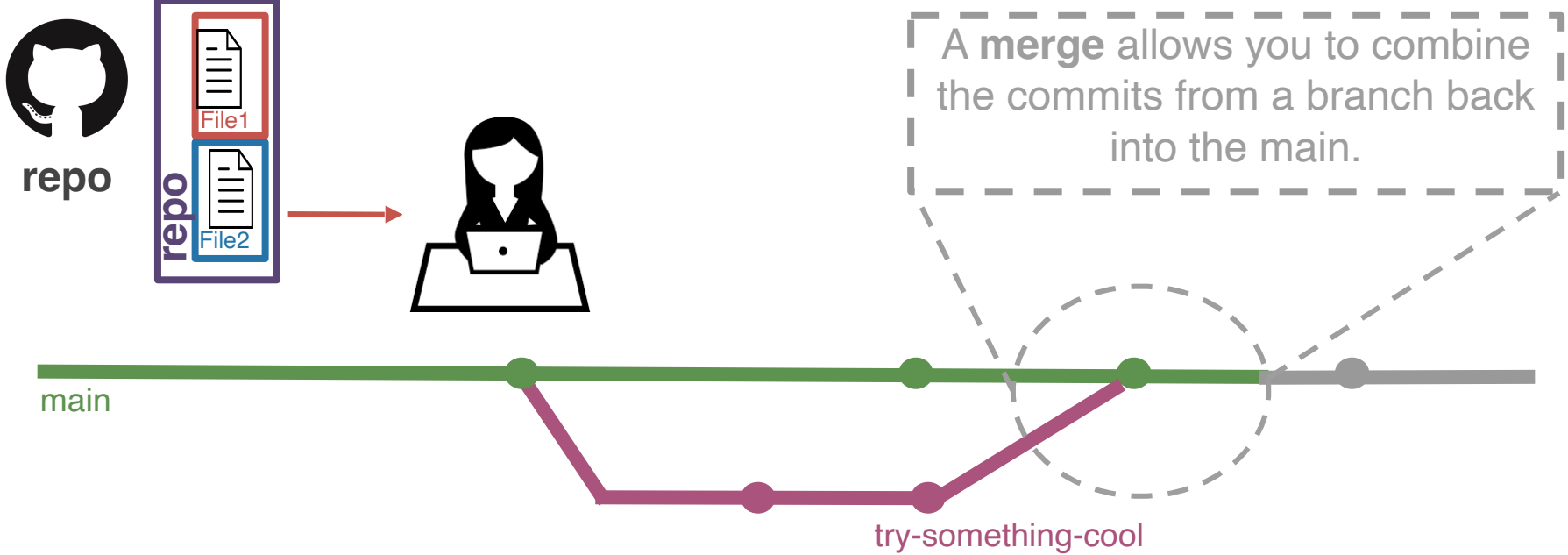
You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.

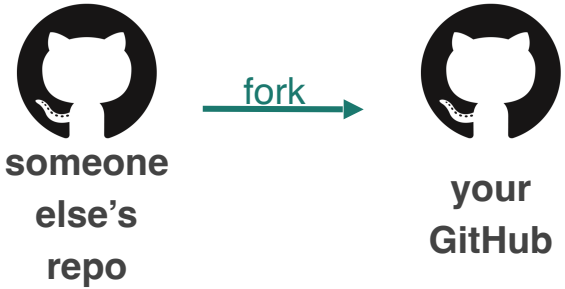
377dfcd00dd057542b112cf13be6cf1380b292ad

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**.

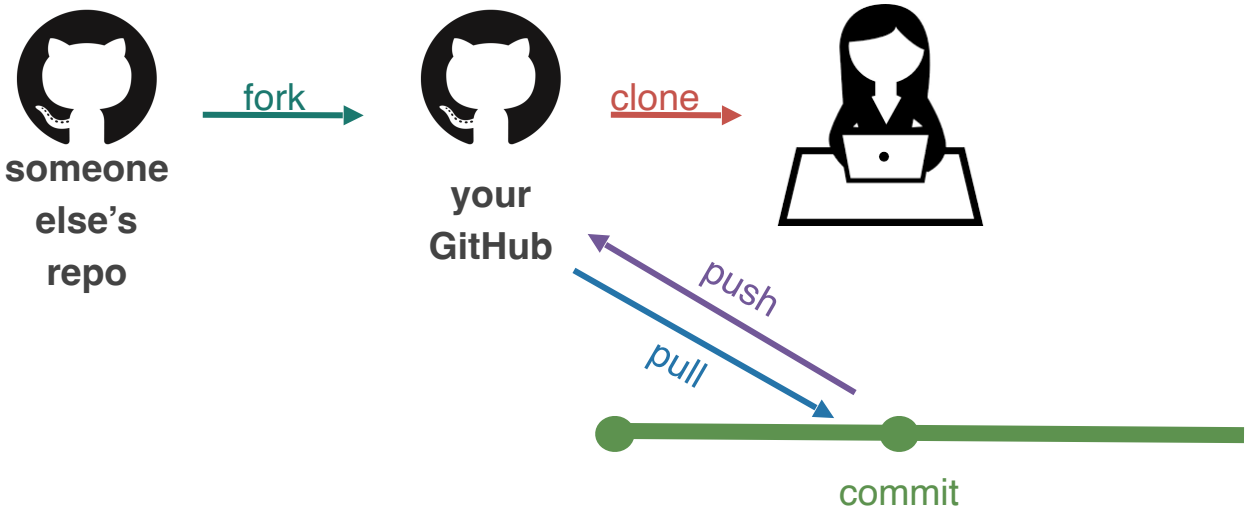
It's a good way to experiment. It's pretty easy to get rid of a branch later on should you not want to include the commits on that branch.

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

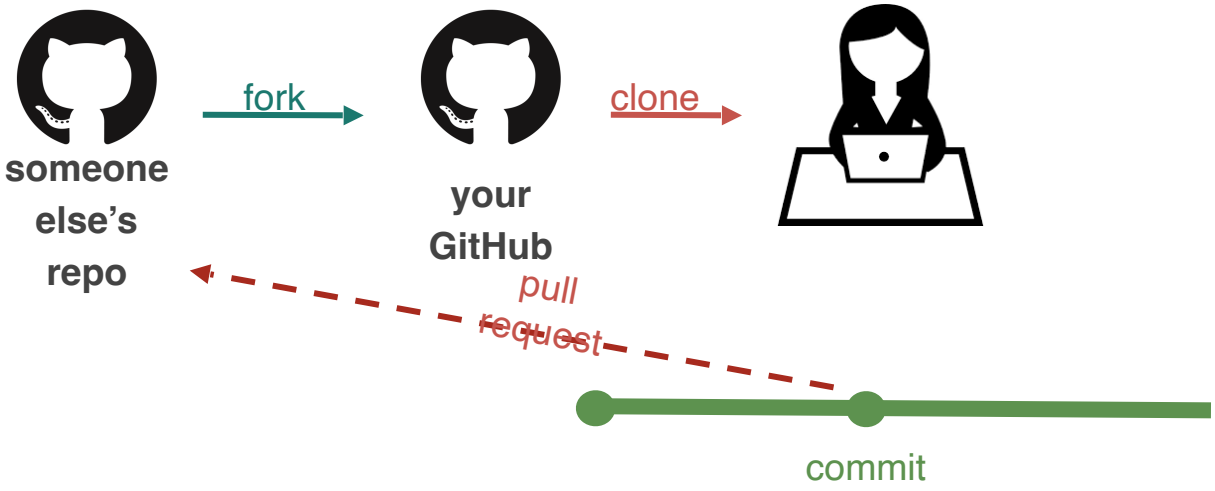

repo
repo
File1
File2
A **merge** allows you to combine the commits from a branch back into the main.
main
try-something-cool



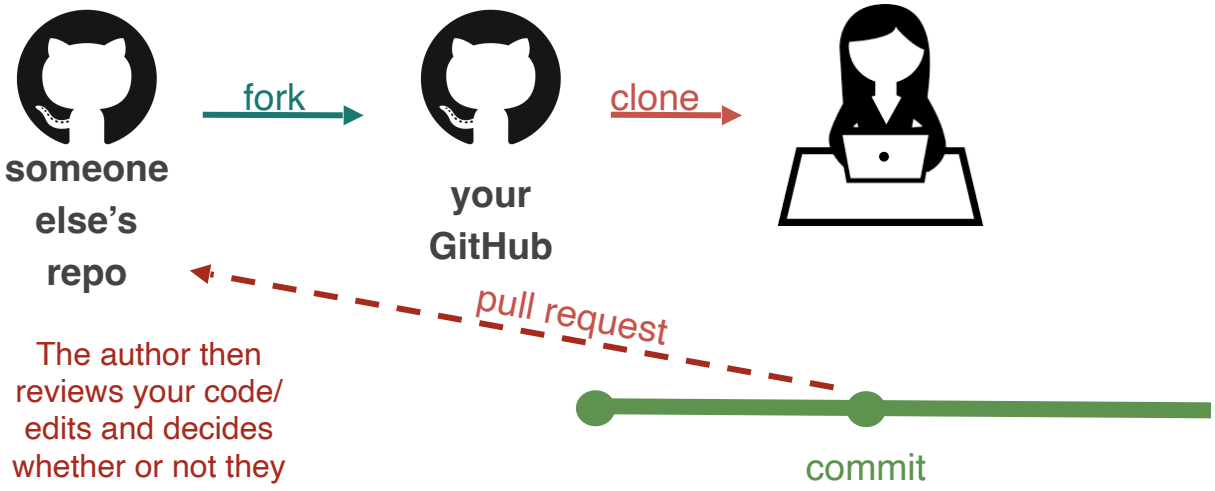
What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.



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.

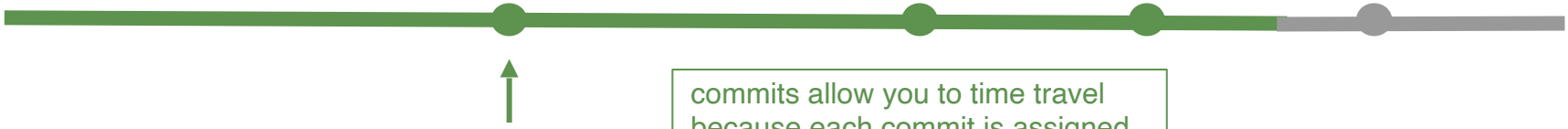
someone else's repo

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!

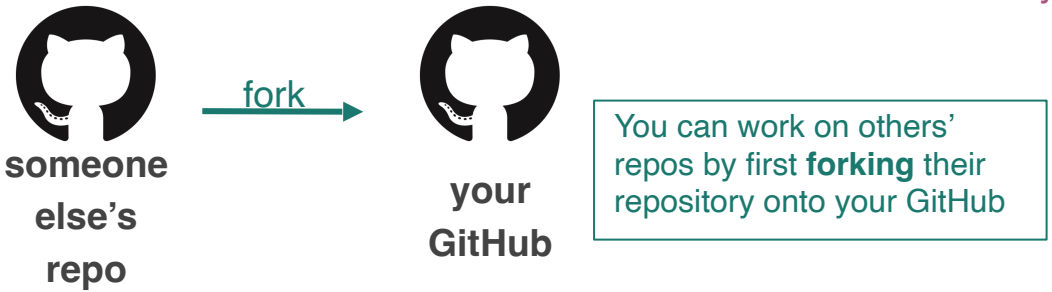
They can be assigned to people.

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

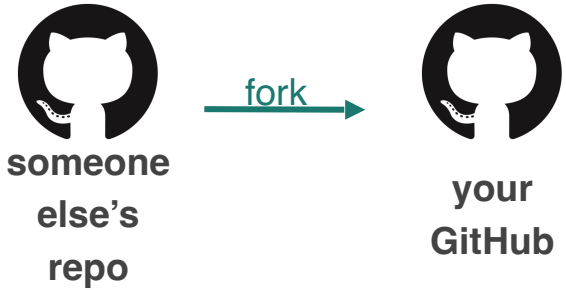


One more git recap...

One more git recap...



One more git recap...



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...

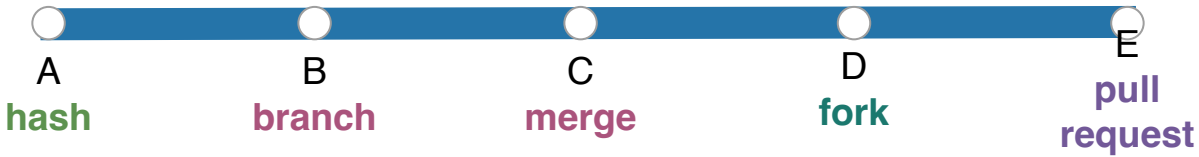
Review & Question Time

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?

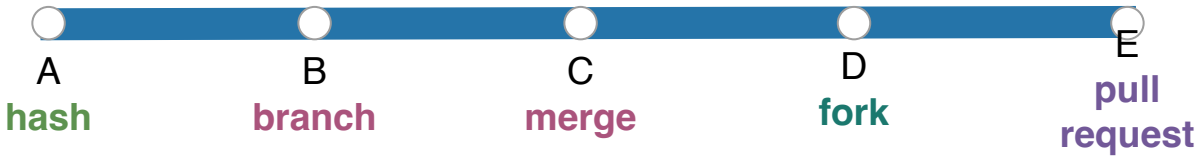


Version Controller IV

https://forms.gle/eyxgHB3wvqmy17uR9

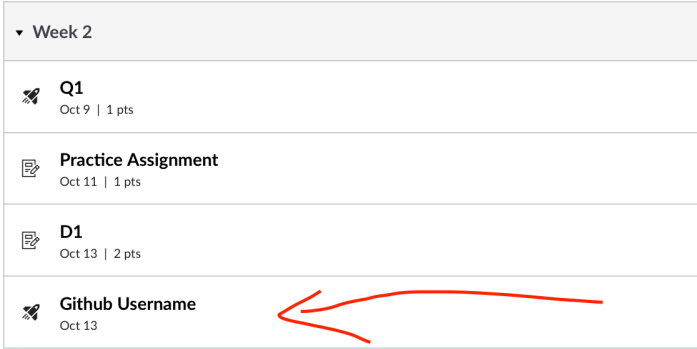
If you've made edits to someone else's repo that you're not a collaborator on…

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



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!



github personal access token

All | Videos | News | Images | Books | More | Tools

About 282,000 results (0.44 seconds)

www.youtube.com › watch

How to Create a Personal Access Token in GitHub - YouTube

Personal access tokens (PATs) are an alternative to using passwords for authentication to **GitHub** when using ...

YouTube · CoderDave · Mar 11, 2021

7 key moments in this video

www.youtube.com › watch

Using Personal Access Tokens with GIT and GitHub - YouTube

A short walk-through of how to use **Personal Access Tokens** to work with **GitHub**.Written instructions can be ...

YouTube · Ed Goad · Feb 9, 2021

3 key moments in this video

www.youtube.com › watch

GitHub password no longer works? GitHub Personal Access ...

Enroll to the 23-hours long **Git** and **GitHub** course ... Now for proper interaction with **GitHub** you need to ...

YouTube · Bogdan Stashchuk · Sep 2, 2021

7 key moments in this video

Tokens are

- More secure (no dictionary attacks)
- Unique per person or per device
- You can have lots of them, different PATs for different roles in different projects

Our Scott Yang wrote this great HOWTO
https://docs.google.com/document/d/1Sb6tQwUVBhzcmBGWw4UnhGlYcMDdyUy3gaRKcQzYur4/edit

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

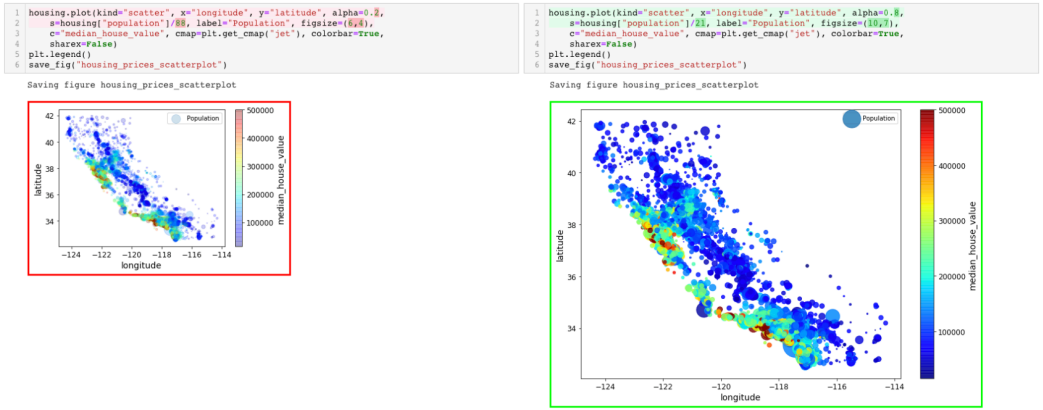
GO TO GITHUB

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.



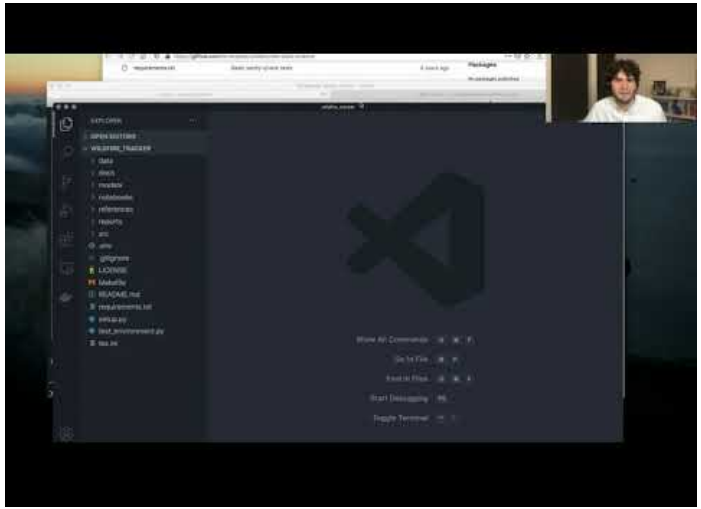
ReviewNB example courtesy of the ReviewNB website

More options

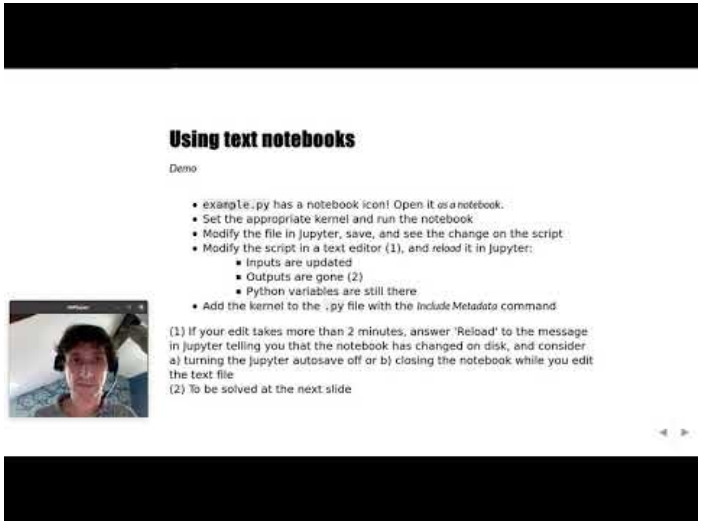
nbautoexport

docs stable | pypi v0.5.0 | conda-forge v0.5.0 | tests passing | codecov 99%

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



-jupy
+text



Let’s practice!

1. Open Datahub in one tab
2. In another tab open https://github.com/COGS108/GitExercise_Conflicts
3. Follow instructions in the repo’s README

Even more 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!