

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 Datahub
 - #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](https://github.com/COGS108/Projects/blob/master/FinalProject_Guidelines.md)

Version Control

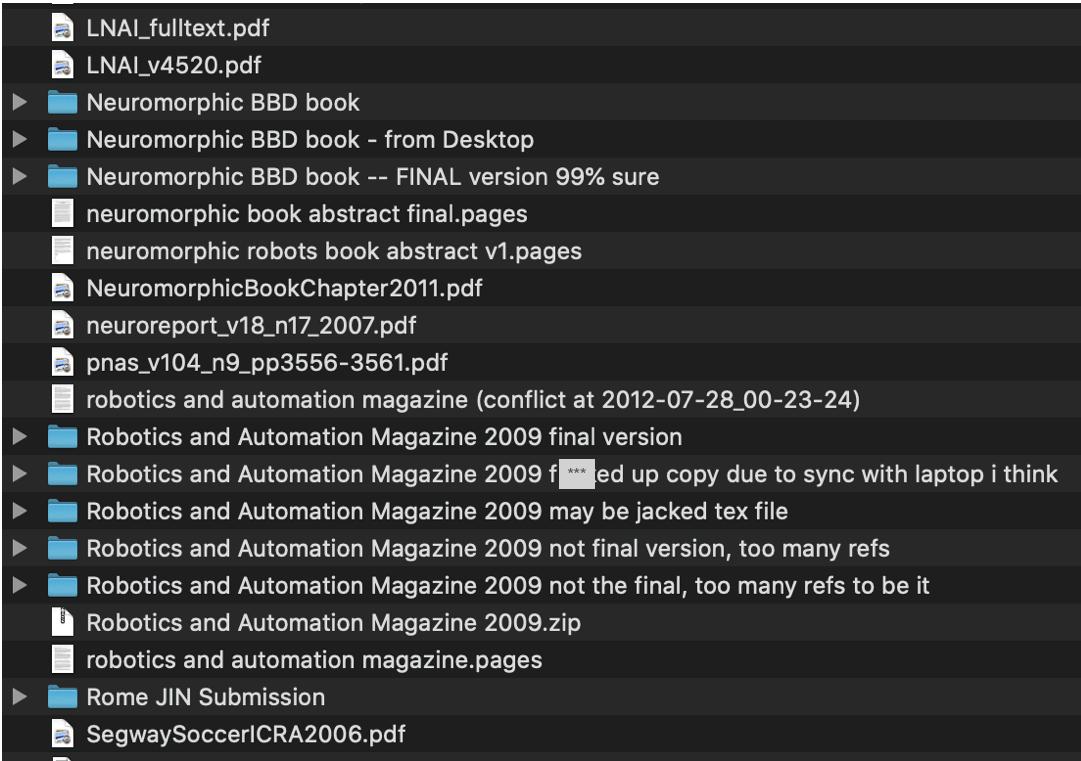
Jason G. Fleischer, Ph.D
UC San Diego

• • •

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.

✉ May 11 ★ ↻ ▾

Thanks for chatting with me earlier today. I added the link to the visualization project into my resume and attached the resume. Thanks for any connections you can make for me. I'd love to know where you send it, so I can keep track of that. Thanks again!

Best,



✉ May 11 ★ ↻ ▾

Actually, please use this one. I fixed a typo that was previously missed. Thanks!



✉ May 11 ★ ↻ ▾

Final copy, I swear. Thanks for helping out.



This is a step in the right direction

Total: 9 edits | ^ v

Version history

Only show named versions

SDSS Teacher Workshop

Considering how to incorporate data science into your high school STEM classroom?

The goal of this workshop is for you to leave with data science skills and applicable examples that can be used in your classroom.

The goal of this workshop is for you to leave with data science skills and applicable examples that can be used in your classroom.

This workshop will answer questions like—

- What is data science?
- How can high schoolers prepare for data science courses in college?
- What does a career in data science involve? Donna LaLonde

is use answer questions like:

- What is data science?
- How can high schoolers prepare for data science courses in college?
- What does a career in data science involve? what data science is, what high schoolers can do to best prepare for data science courses in college, and what a career in data science involves.

We will walk through how data scientists carry out projects using RStudio, introduce the basics of the R programming language, and work with real datasets to generate visualizations and analyze data. The goal of this workshop is for you to leave with data science skills and applicable examples that can be used in your classroom.

MARCH

▶ March 4, 7:27 AM Current version Shannon Ellis

▶ March 3, 9:47 AM Donna LaLonde Shannon Ellis

FEBRUARY

▶ February 27, 6:29 AM Shannon Ellis

February 26, 5:44 PM Shannon Ellis

▶ February 26, 4:57 PM Shannon Ellis

▶ February 26, 3:50 PM Kelly McConville

▶ February 25, 3:53 PM Shannon Ellis

February 25, 3:33 PM Shannon Ellis

Show changes

Version Control

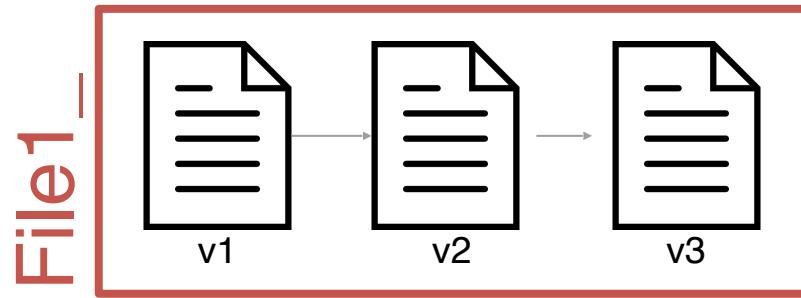
- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.
- Thus, temporary or partial edits by one person do not interfere with another person's work

What is version control?

A way to manage the evolution of a set of files

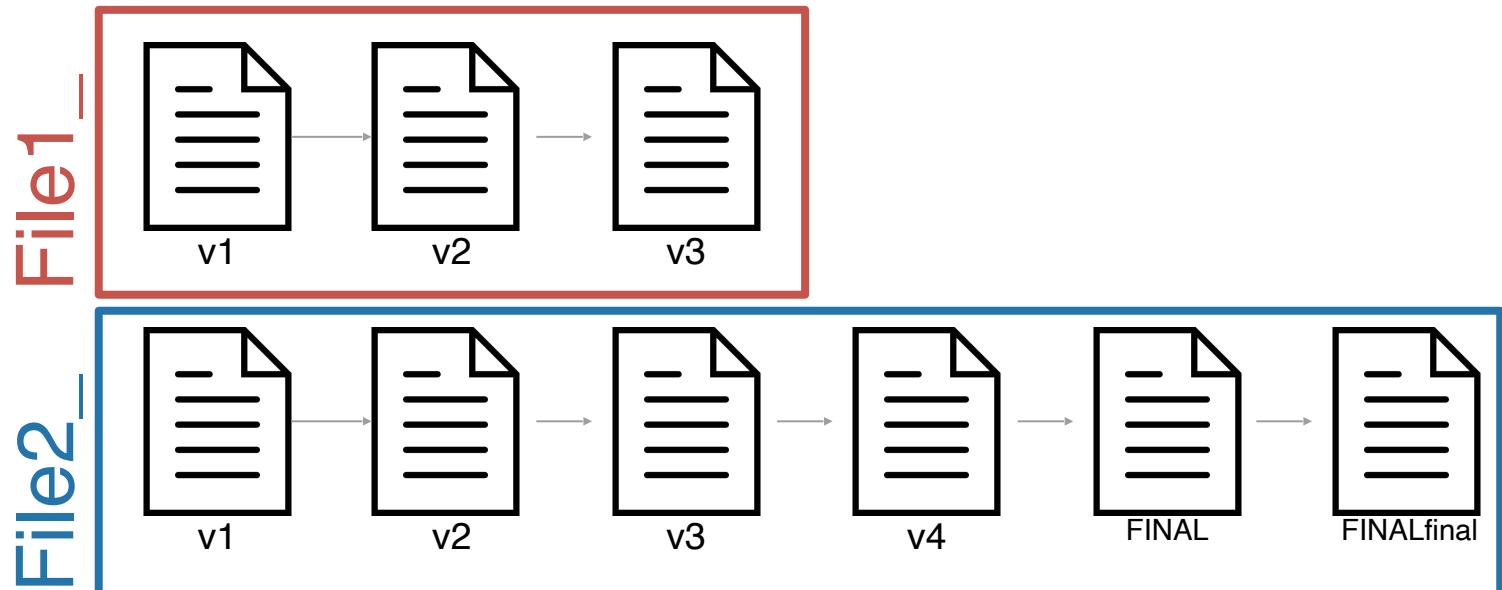
What is version control?

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?

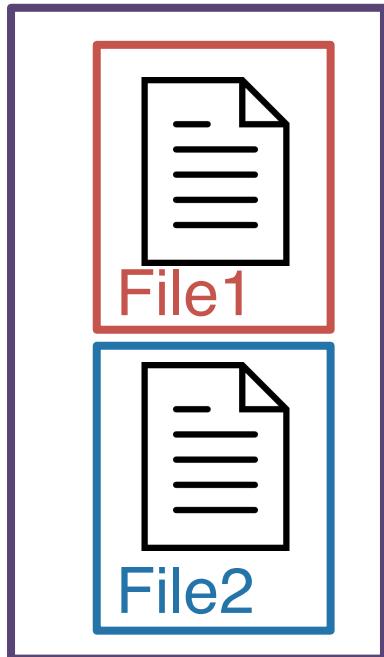
A way to manage the evolution of a set of files



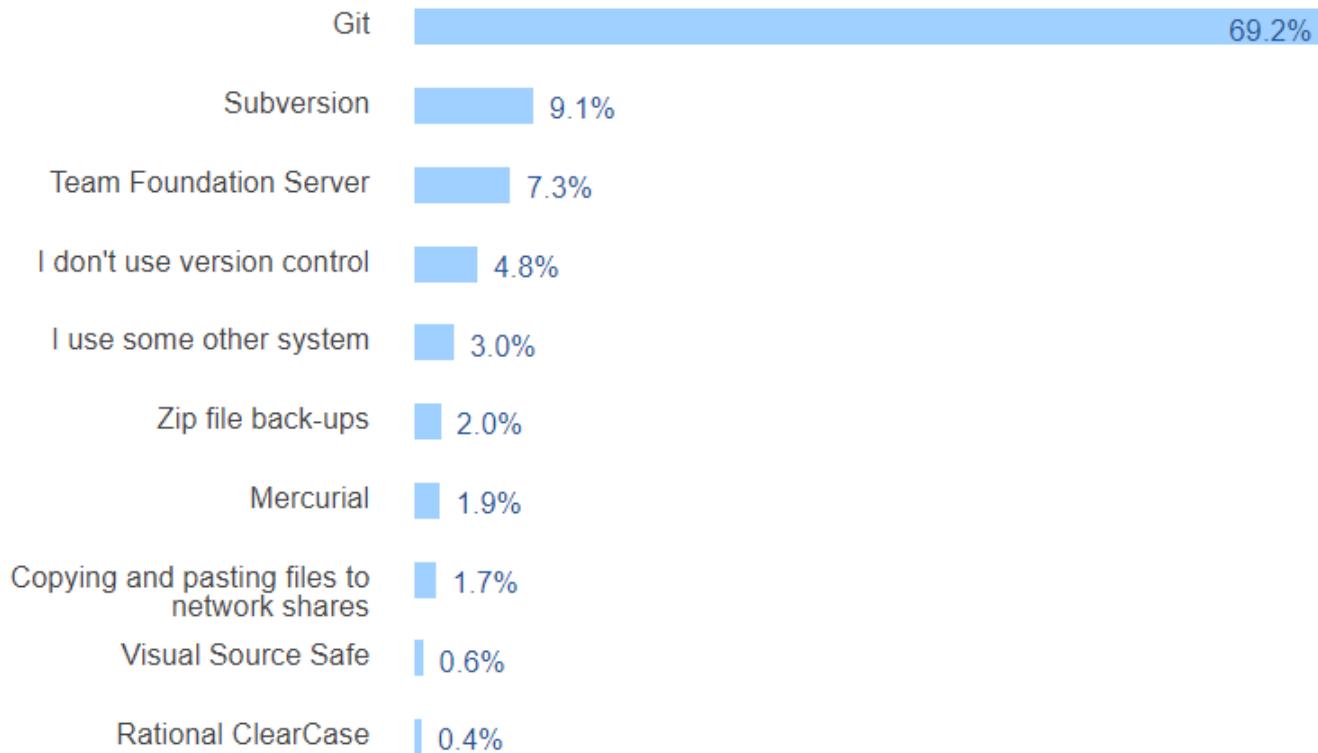
When using a version control system,
you have **one copy of each file** and the
*version control system tracks the
changes* that have occurred over time

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)**

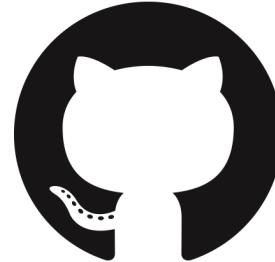


git & GitHub

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
```

The screenshot shows the GitHub organization page for COGS108. At the top, there's a header with the organization name, location (UC San Diego), and email (COGS108@gmail.com). Below the header, there are tabs for 'Repositories' (22), 'People' (7), 'Teams' (2), 'Projects' (0), and 'Settings'. A section titled 'Pinned repositories' lists several items:

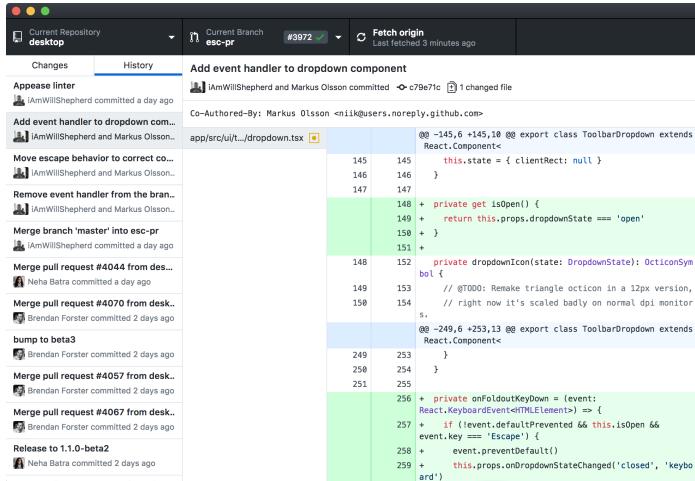
- Overview**: Overview and map of the organization, which services COGS108: Hands-On Data Science, from UCSD. (1 star, 9 views)
- Lectures-Sp19**: Slides and Notebooks used in Lecture for Sp19 COGS108. (1 star, 1 view)
- Section_Workbooks**: Workbooks for practice during discussion section. (1 Jupyter Notebook, 1 view)
- Tutorials**: Tutorial notebooks for hands-on data science, following along with the course topics. (1 Jupyter Notebook, 38 views, 108 forks)
- Projects**: Final Project materials and description. (1 Jupyter Notebook, 3 views, 82 forks)
- Readings**: A curated list of suggested reading materials. (4 views)

Below the pinned repositories, there's a search bar ('Find a repository...'), filters ('Type: All', 'Language: All'), and a 'New' button. Further down, there's a 'MyFirstPullRequest' section and an 'Overview' section.

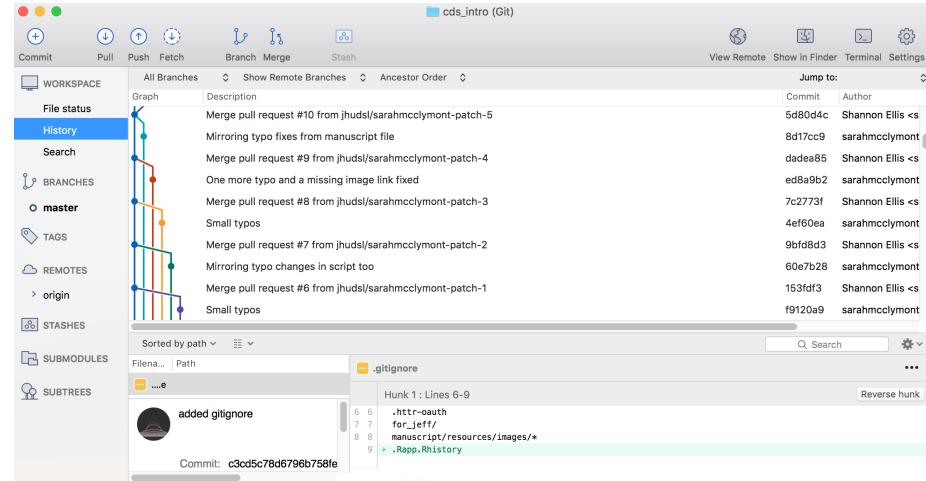
Terminal
git



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?

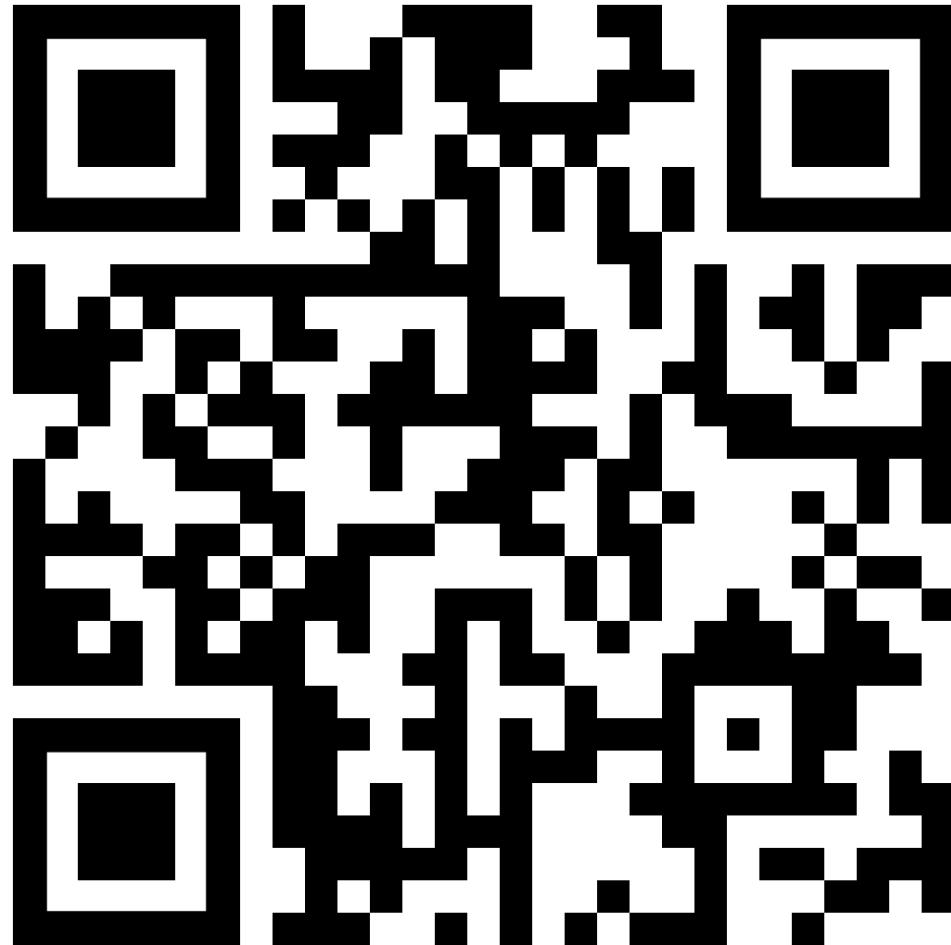
A. I don't

B. command line

C. GUI GitHub Desktop

D. GUI: SourceTree

E. GUI: other



<https://git-scm.com/downloads>



git

--distributed-even-if-your-workflow-isnt

Type / to search entire site...

About

Documentation

Downloads

GUI Clients
Logos

Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

 macOS  Windows
 Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.



GUI Clients

Git comes with built-in GUI tools ([git-gui](#), [gitk](#)), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

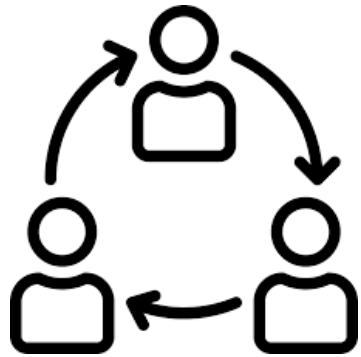
```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

</> About this site
Patches, suggestions, and comments are welcome.

Git is a member of Software Freedom Conservancy

Why version control with git and GitHub?



Collaboration



Returning to
a safe state

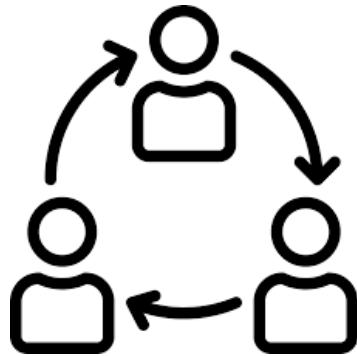


Exposure
for your
work

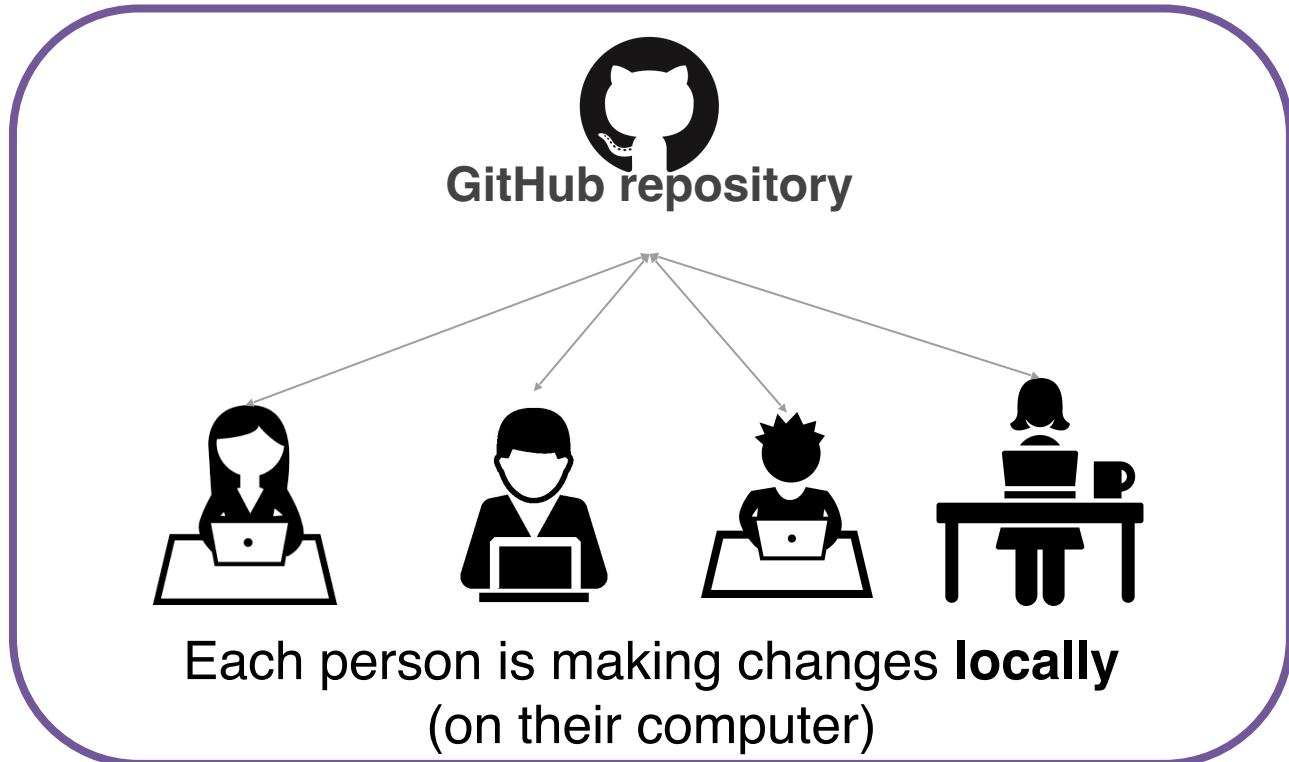


Tracking
others' work

Collaborate like you do with Google Docs



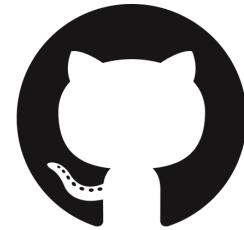
Collaboration



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>



Search or jump to...



Pull requests Issues Marketplace Explore

Explore

Topics

Trending

Collections

Events

GitHub Sponsors

#

april-fools

DuckMasterAI / **rickroll-bot**

Star 5



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



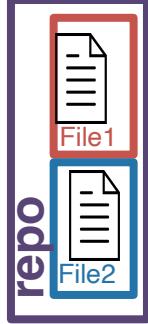
Tracking
others' work



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



repo

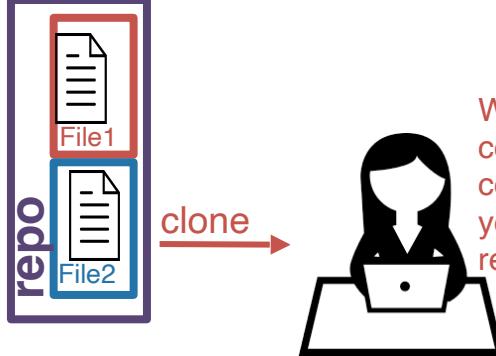


A **GitHub repo** contains all the files and folders for your project.

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



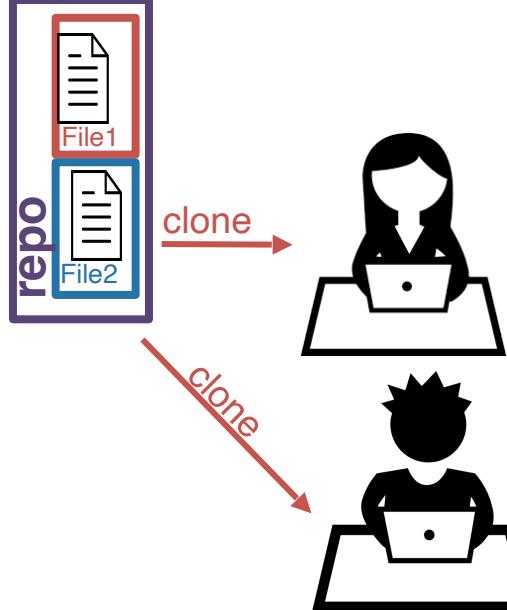
repo



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



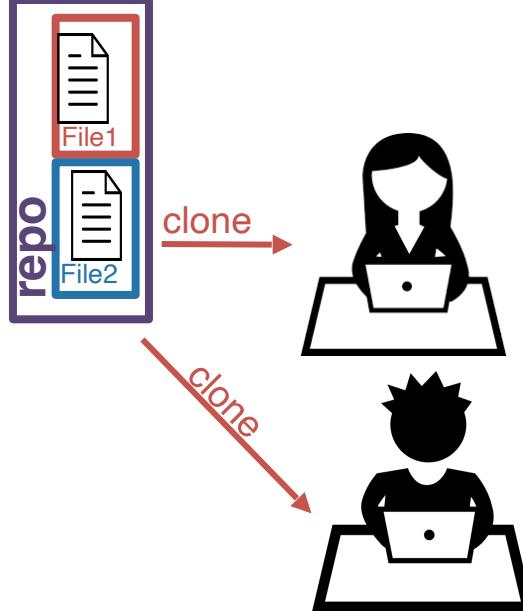
repo



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.



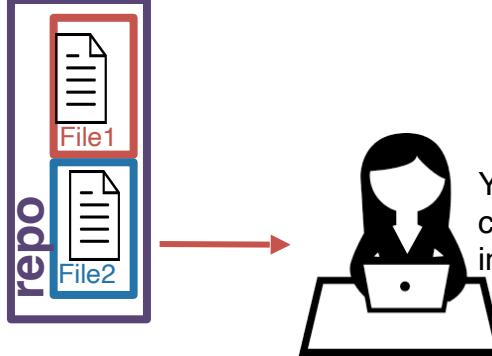
repo



Yay! Everyone can
work on the project!



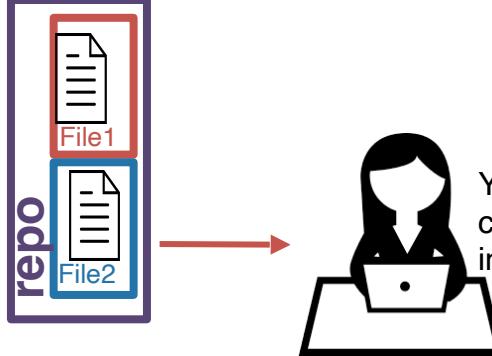
repo



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



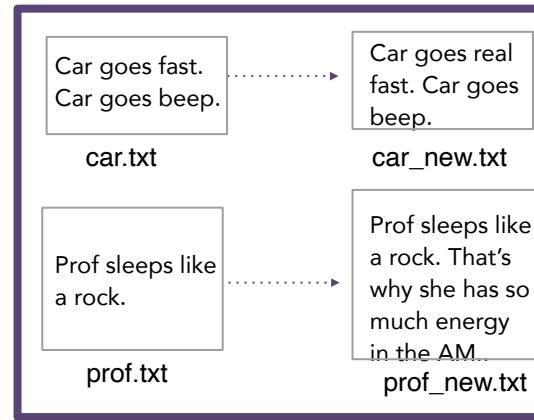
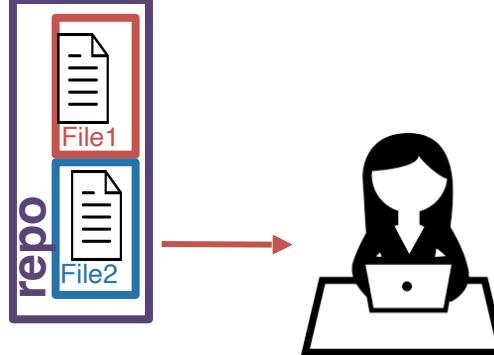
repo



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



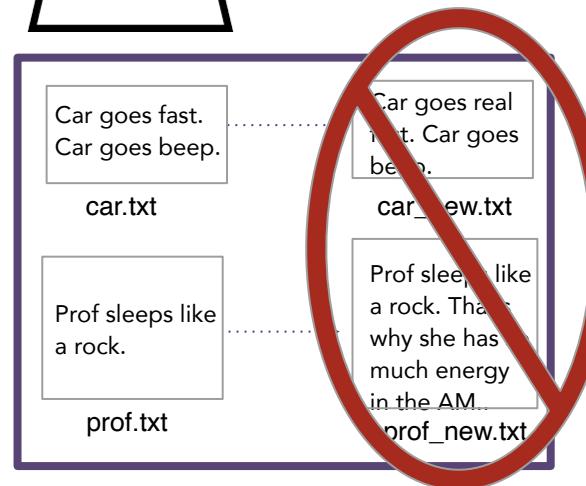
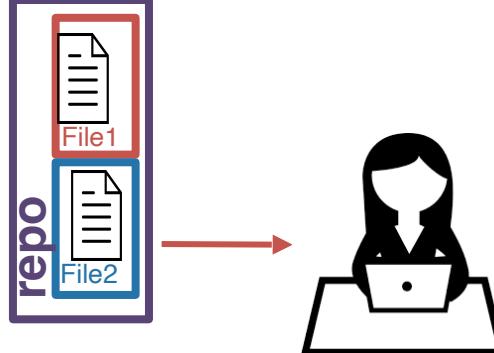
repo



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



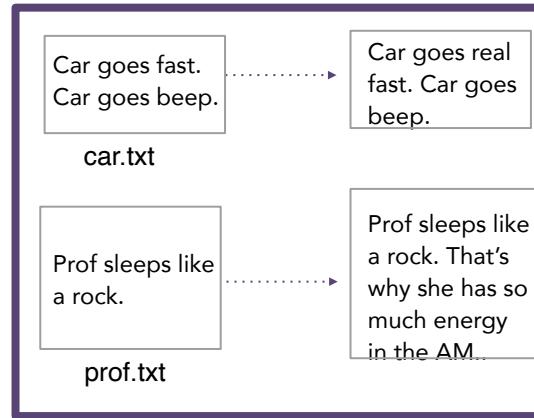
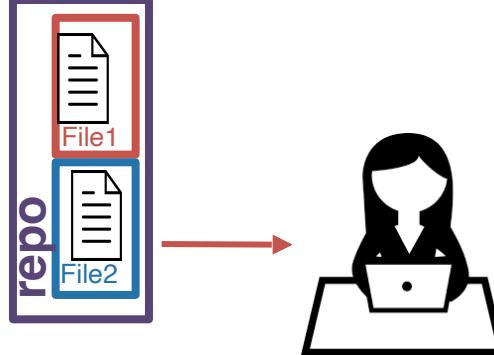
repo



Thank
goodness those
days are over!



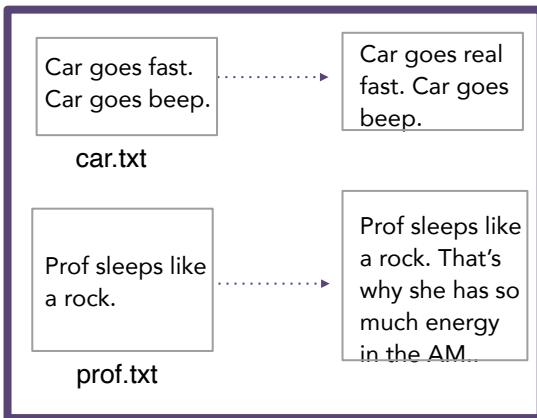
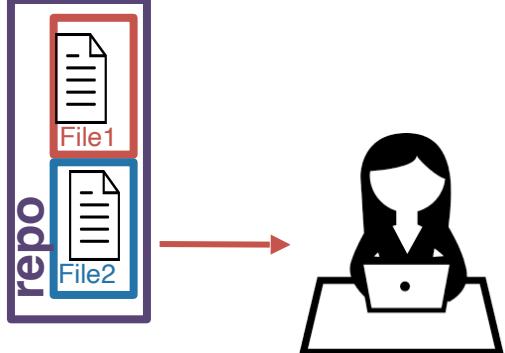
repo



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



repo

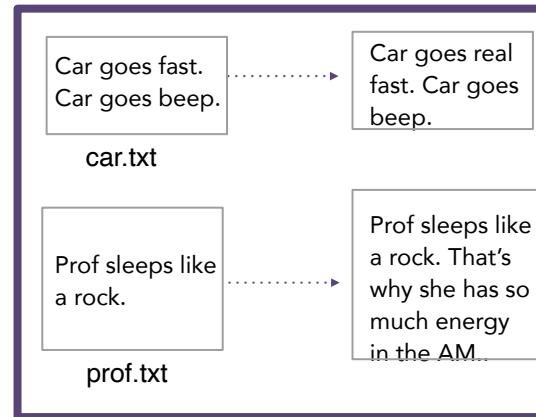
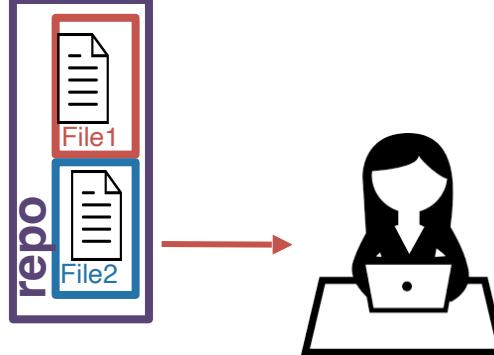


<code>git add file</code>	stages specified file (or folder)
<code>git add .</code>	stages new and modified files
<code>git add -u</code>	stages modified and deleted files
<code>git add -A</code>	stages new, modified, and deleted files
<code>git add *.csv</code>	Stages any files with .csv extension
<code>git add *</code>	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*.



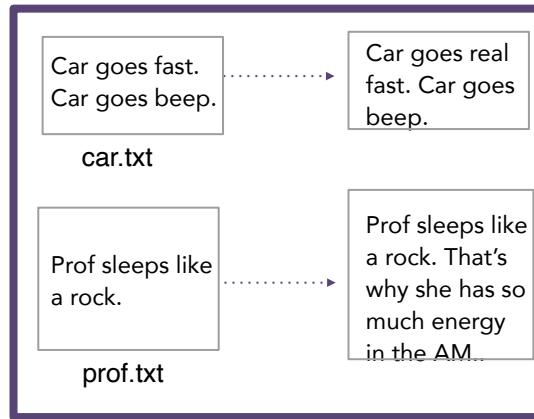
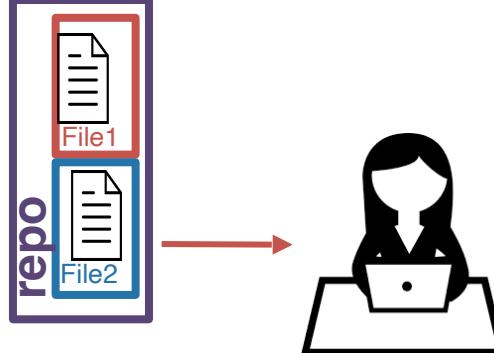
repo



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



repo



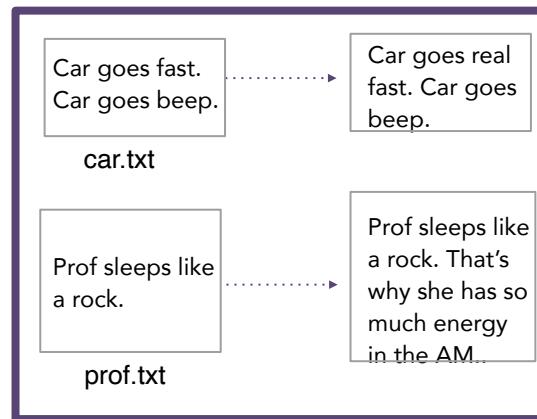
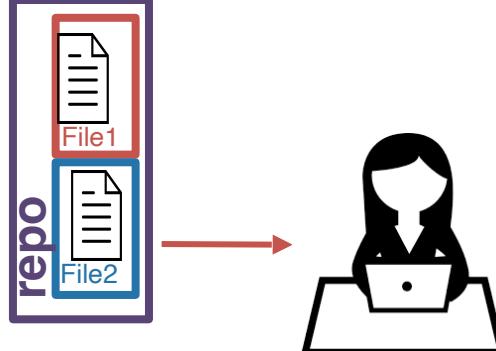
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.



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



repo



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

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

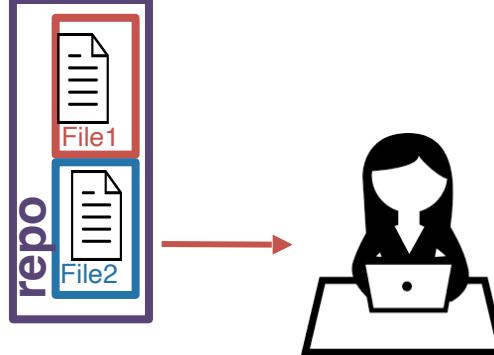
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.



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



repo



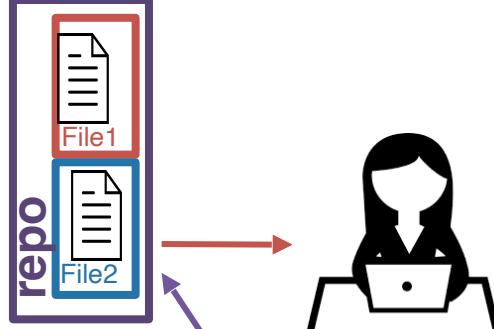
Shannon Ellis

3/28/21 3:28pm

fix typos in car and prof



repo



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.



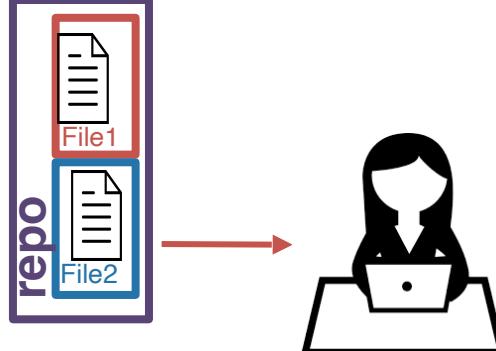
Shannon Ellis

3/28/21 3:28pm

fix typos in car and prof



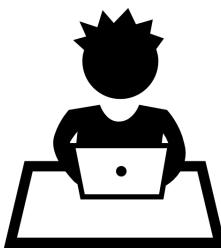
repo



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!

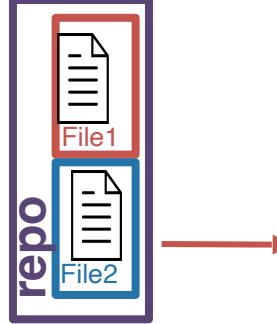
Shannon Ellis

3/28/21 3:28pm

fix typos in car and prof



repo

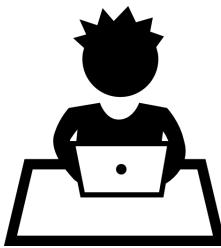


To catch up, your teammate will have to
pull the changes from GitHub (remote)



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!

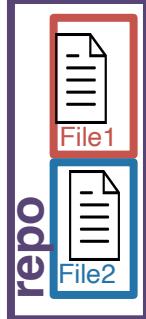


Shannon Ellis
3/28/21 3:28pm

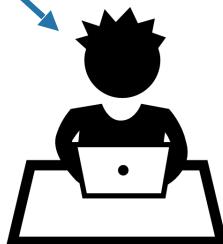
fix typos in car and prof



repo



pull



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

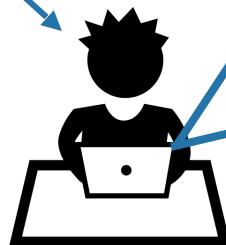
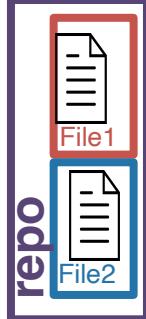


Shannon Ellis
3/28/21 3:28pm

fix typos in car and prof



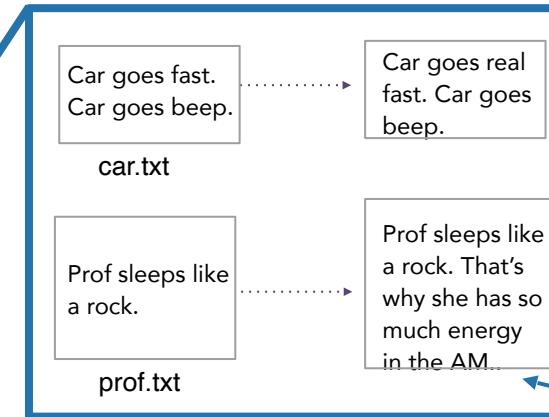
repo



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

Shannon Ellis
3/28/21 3:28pm

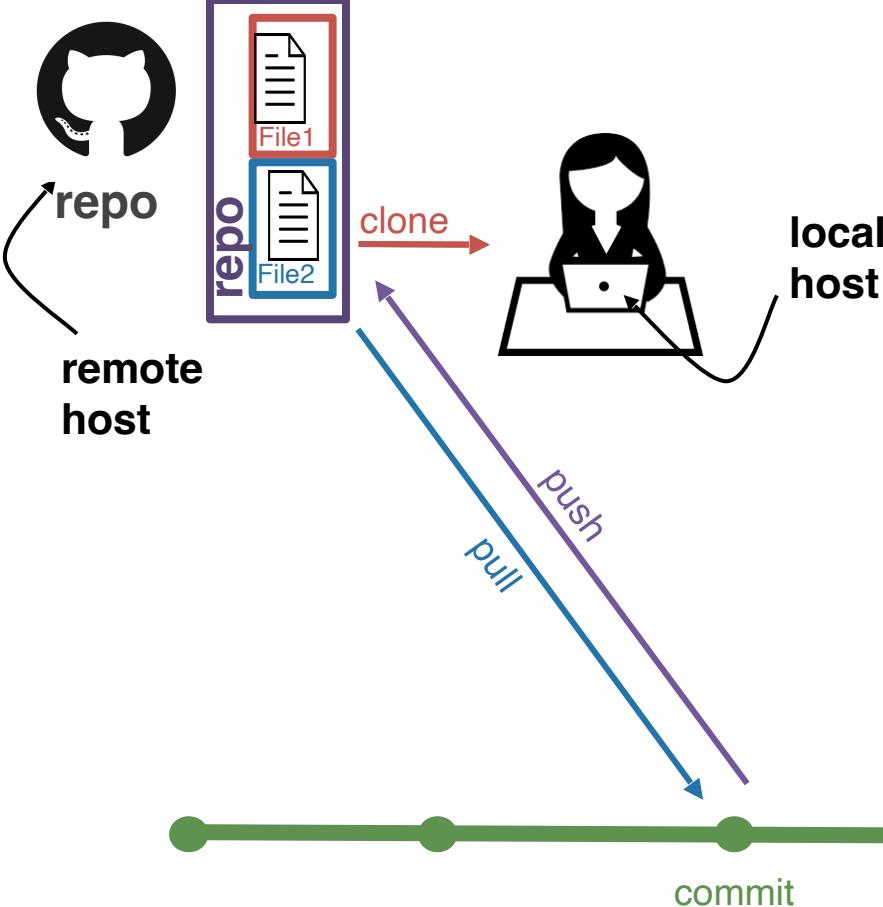
fix typos in car and prof



Car goes real
fast. Car goes
beep.

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



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

commit

```
(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

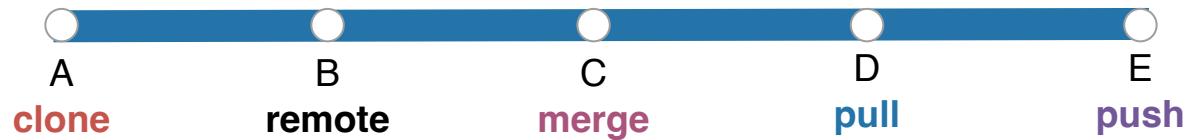


Version Controller I

<https://forms.gle/wHA2GSyucFre5qr6>

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.

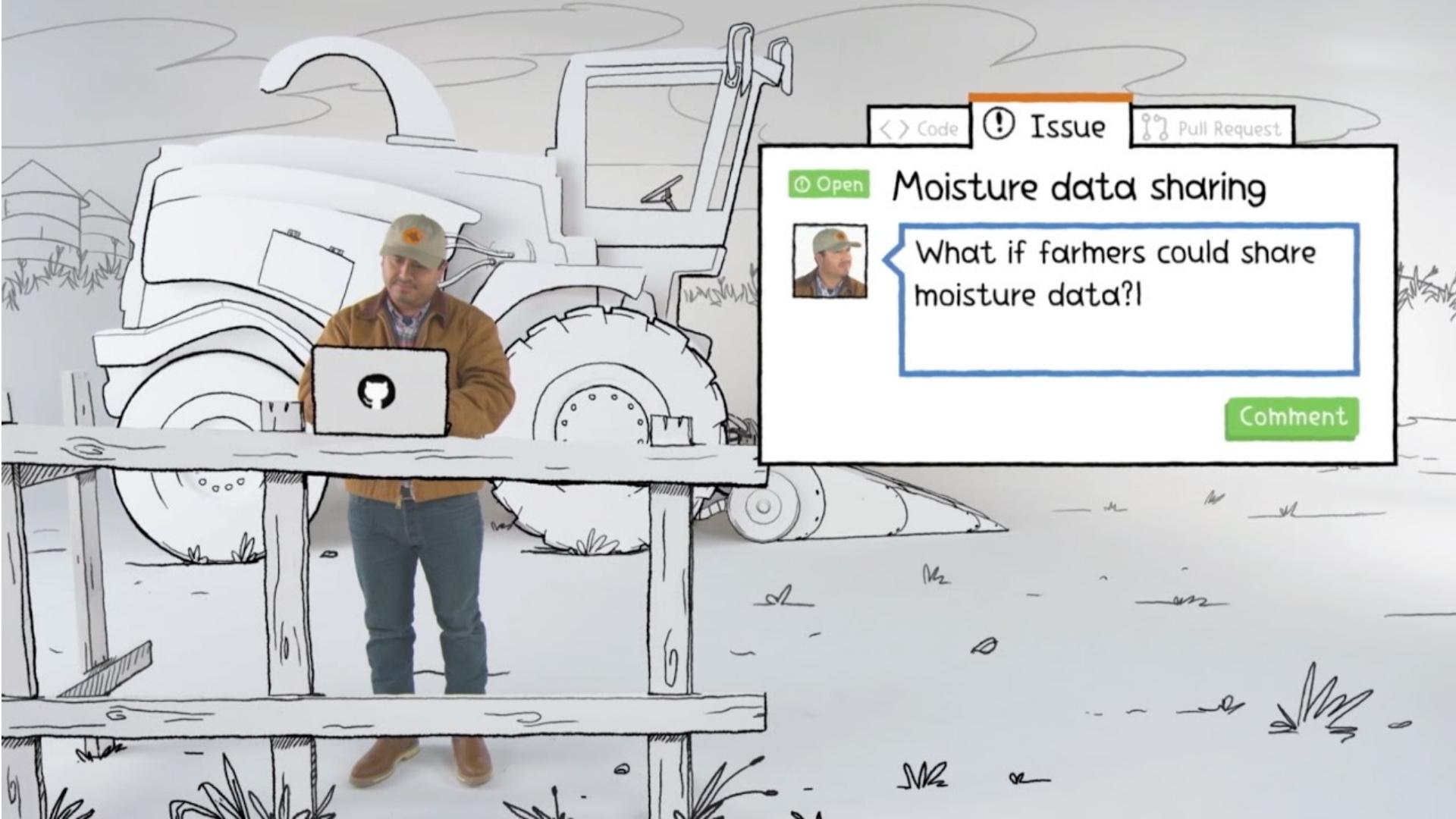
What do you do to get the repo on your computer?





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:

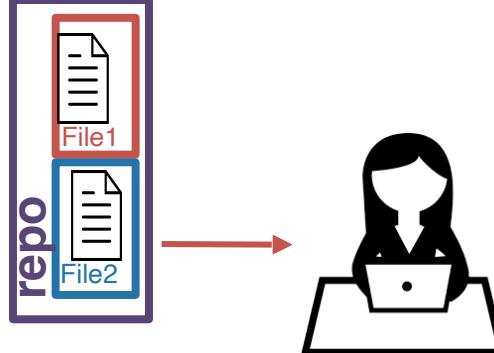
```
git clone https://github.com/COGS108/Lectures-Sp25
```

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!

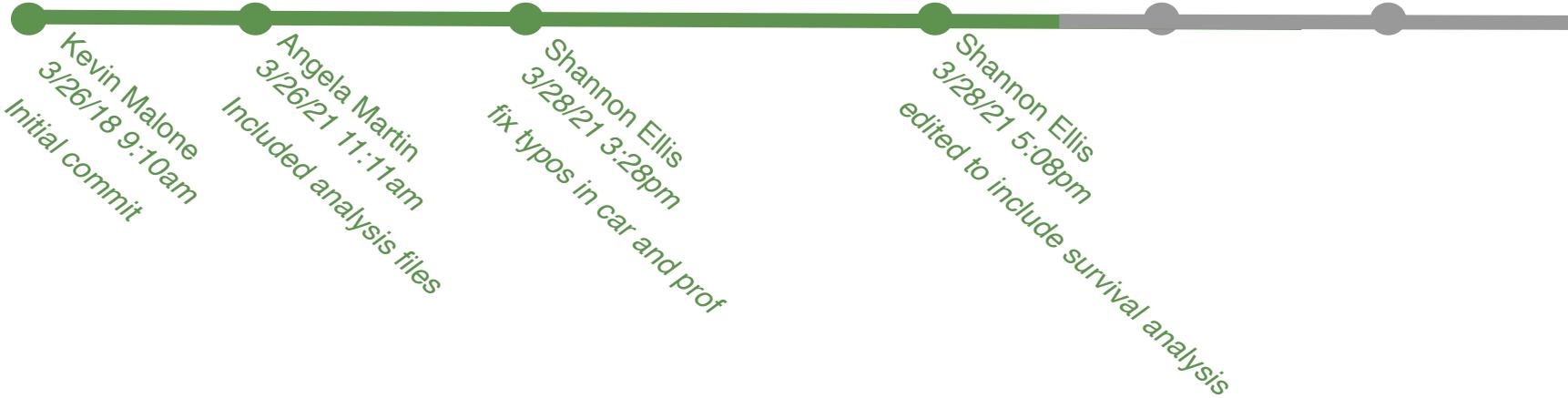
Usually stop here for time



repo

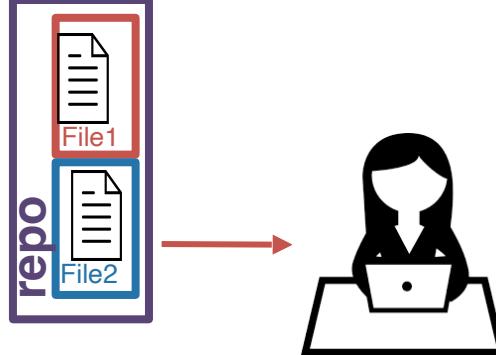


Each time you create a commit, git tracks the changes made automatically.





repo

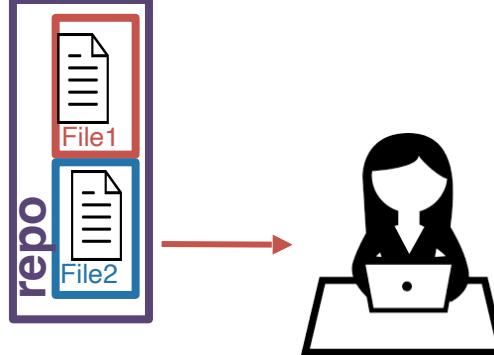


By committing each time you make changes, git allows you to time travel!

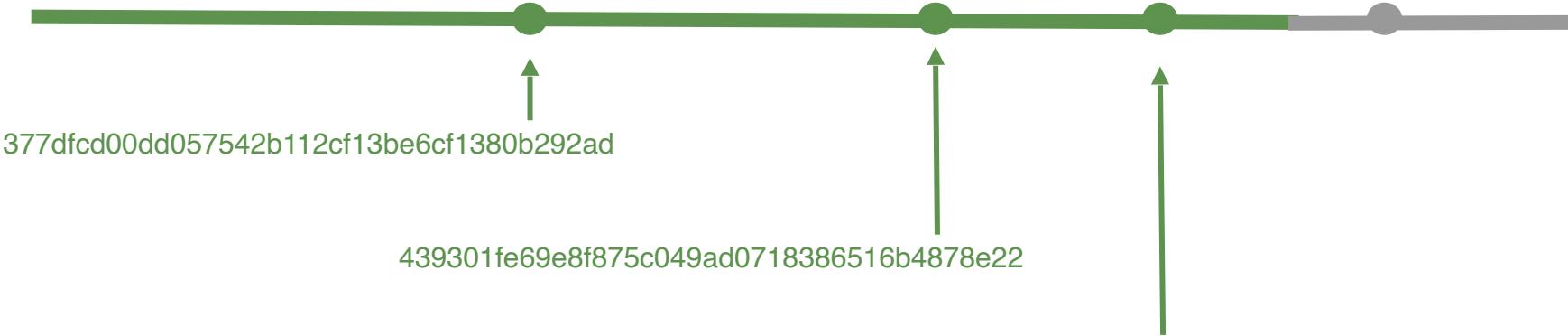




repo



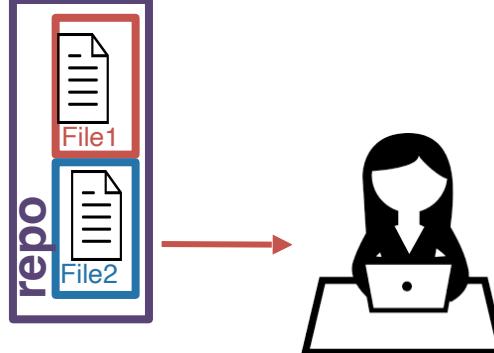
By committing each time you make changes, git allows you to time travel!



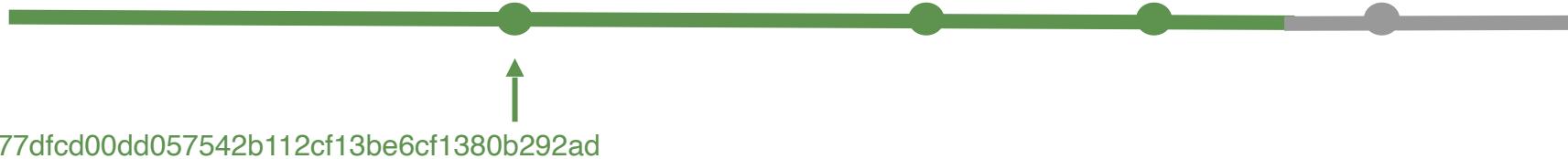
There's a unique id, known as a **hash**, associated with each commit.



repo

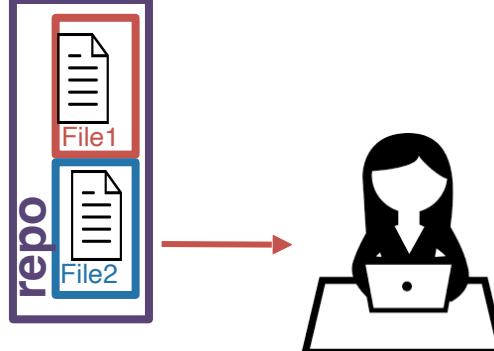


You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.

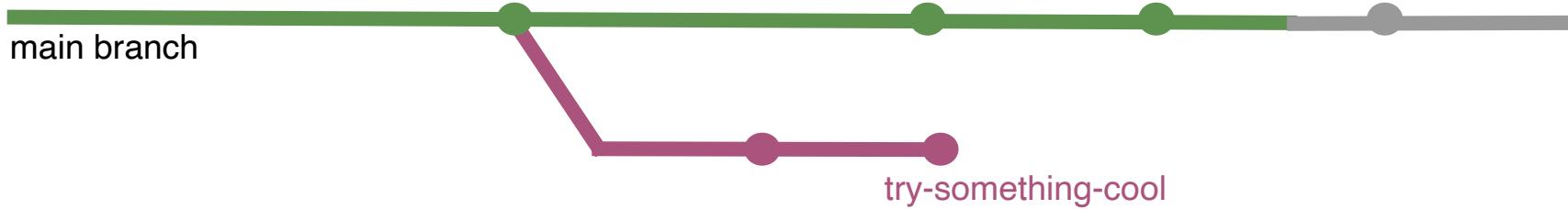




repo

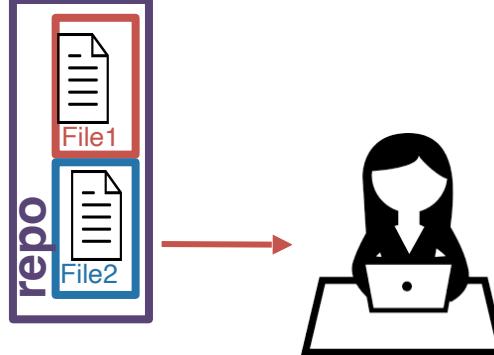


But...not everything is always linear.
Sometimes you want to try something out
and you're not sure it's going to work.
This is where you'll want to use a **branch**.

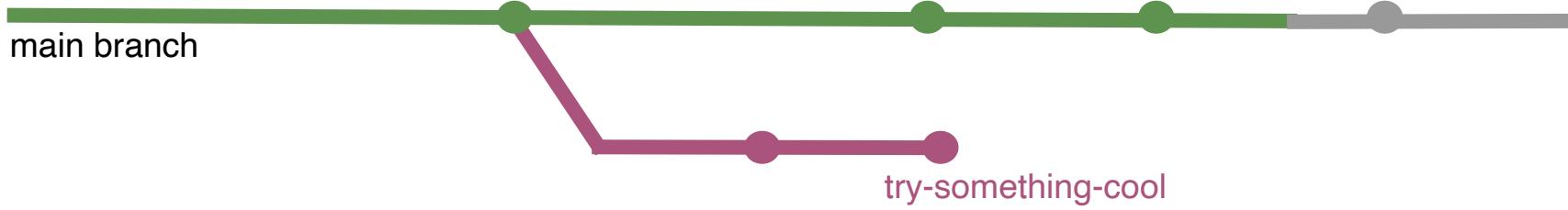




repo

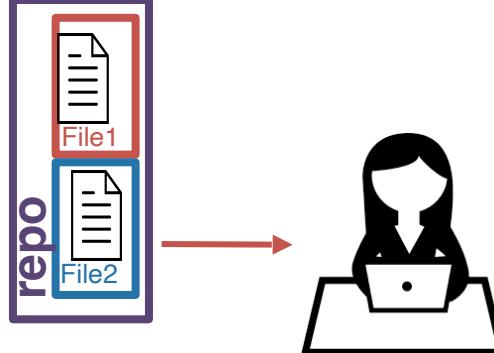


It's a good way to experiment. It's pretty easy to get rid of a branch later on should you not want to include the commits on that branch.

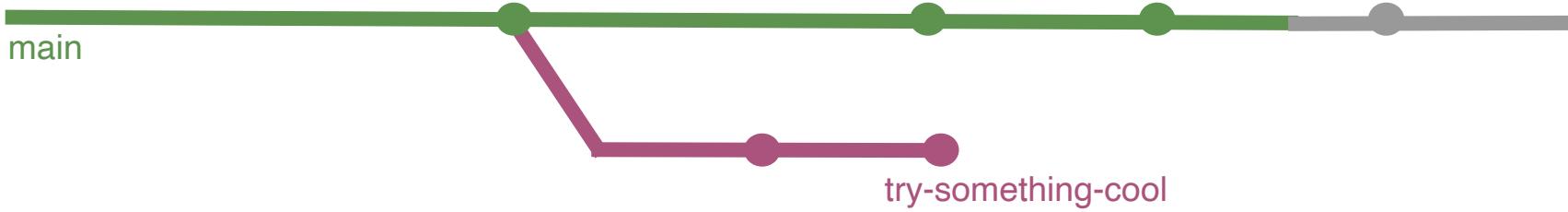




repo

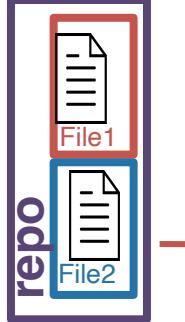


But...what if you DO want to include the changes you've made on your try-something-cool branch into the main branch?

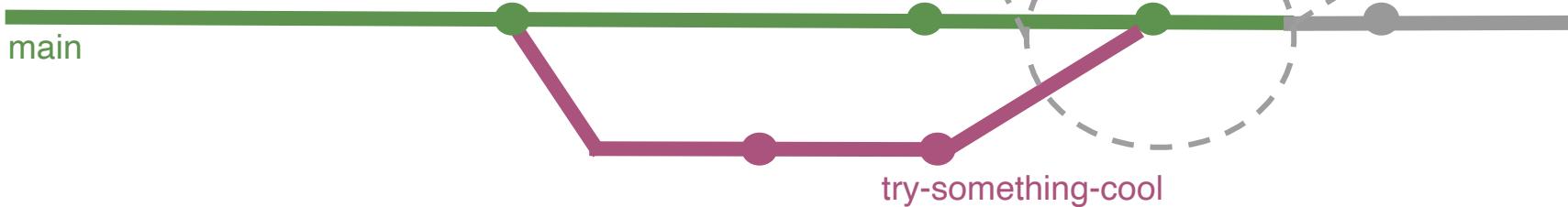




repo



main



A merge allows you to combine the commits from a branch back into the main.

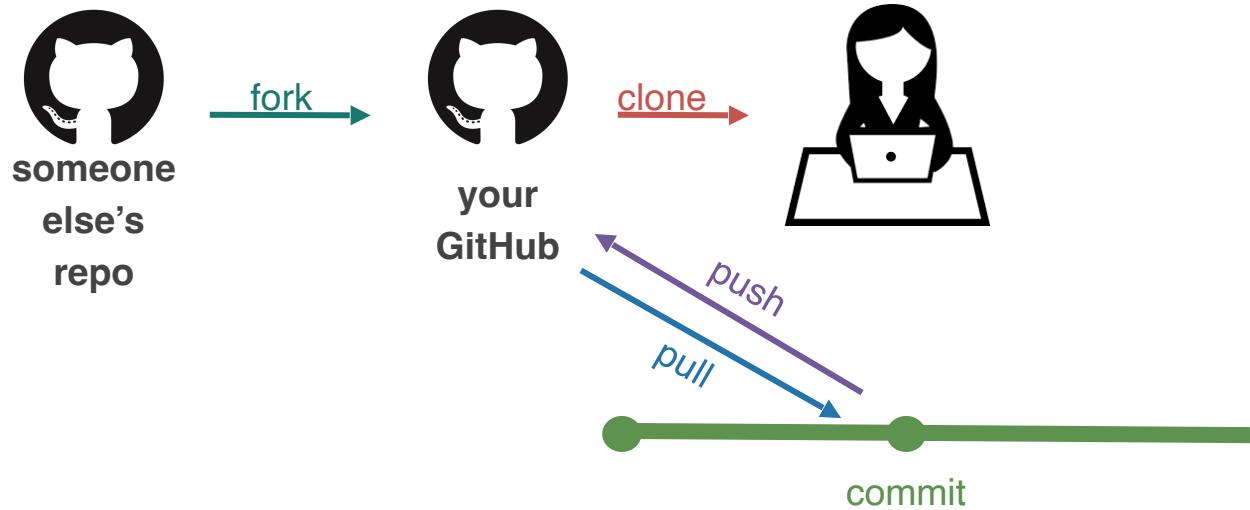


someone
else's
repo

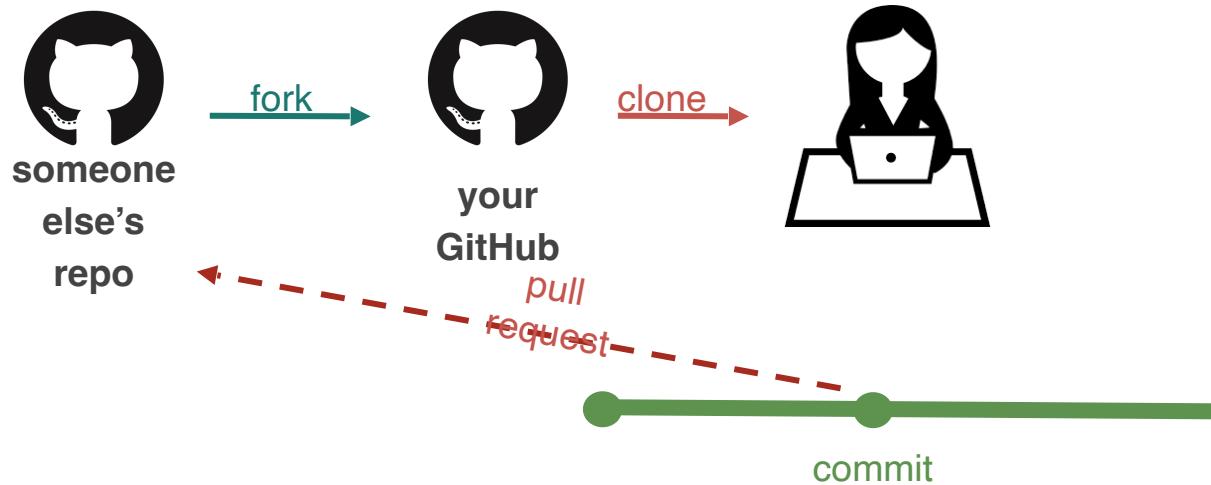


your
GitHub

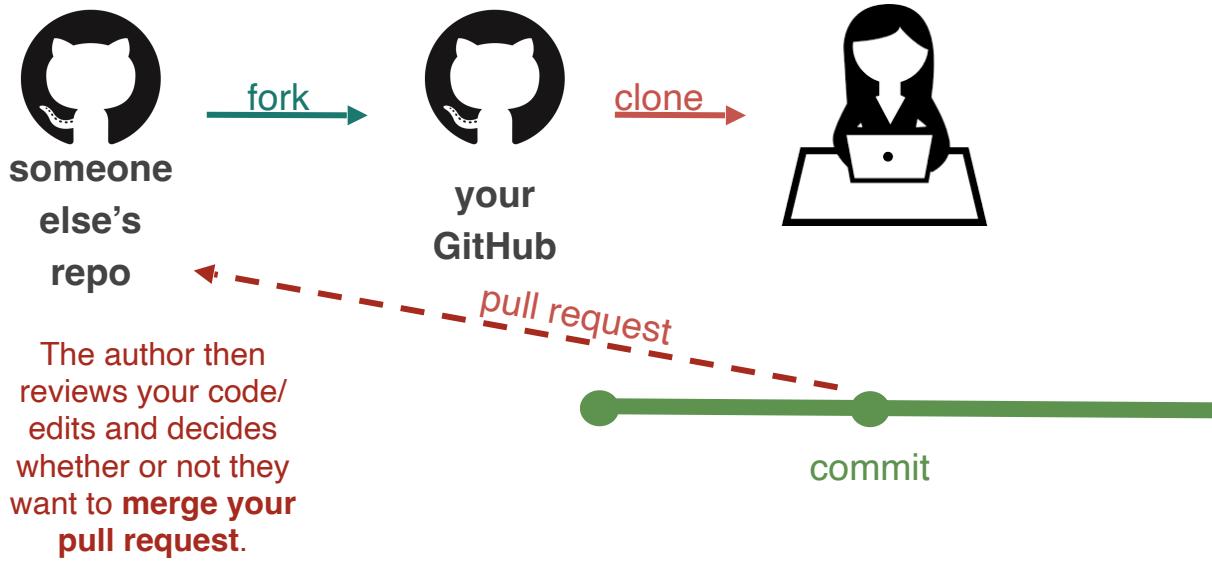
What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.



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.



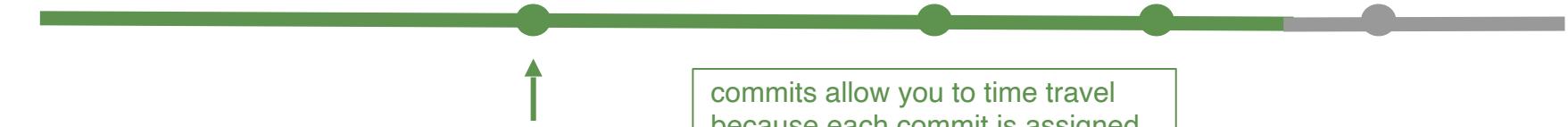
someone
else's
repo

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 addressedor if the software maintainer doesn't like the suggestion

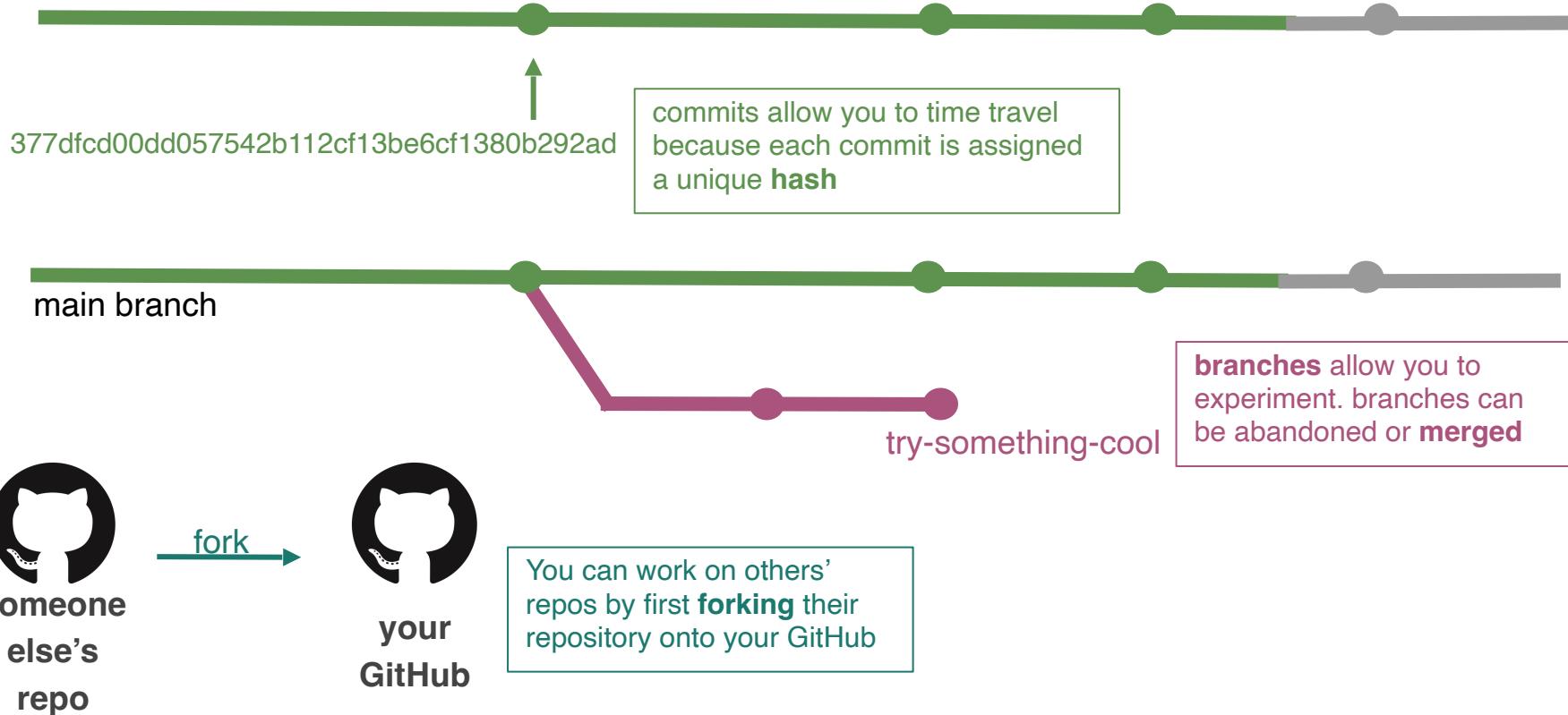


commits allow you to time travel
because each commit is assigned
a unique **hash**

One more git recap...



One more git recap...



One more git recap...

377dfcd00dd057542b112cf13be6cf1380b292ad

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

main branch

try-something-cool

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



someone
else's
repo

fork



your
GitHub

You can work on others'
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

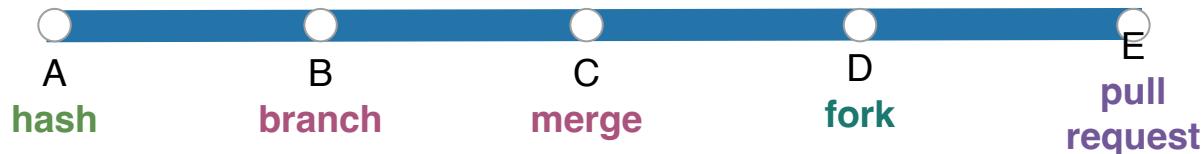


Version Controller III

<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?

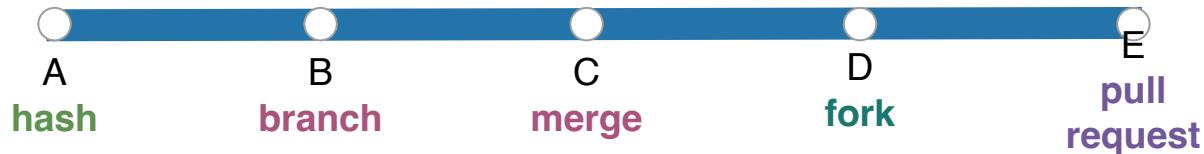




Version Controller IV

<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!

Week 2

- Q1 Oct 9 | 1 pts
- Practice Assignment Oct 11 | 1 pts
- D1 Oct 13 | 2 pts
- Github Username Oct 13

Q 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

9:30 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

6:49 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/1Sb6tQwUVBhzcmBGWw4UnhGIYcMDdyUy3gaRKcQzYur4/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

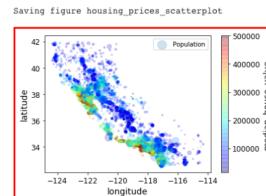
Jupyter notebooks suck to version control

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

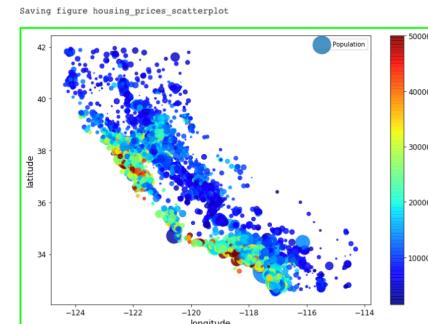
ReviewNB

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.

```
1 housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.8,
2   s=housing['population']/25, label="Population", figsize=(10,7),
3   c="median_house_value", cmap=plt.get_cmap('jet'), colorbar=True,
4   sharex=False)
5 plt.legend()
6 save_fig("housing_prices_scatterplot")
```



```
1 housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.8,
2   s=housing['population']/25, label="Population", figsize=(10,7),
3   c="median_house_value", cmap=plt.get_cmap('jet'), colorbar=True,
4   sharex=False)
5 plt.legend()
6 save_fig("housing_prices_scatterplot")
```



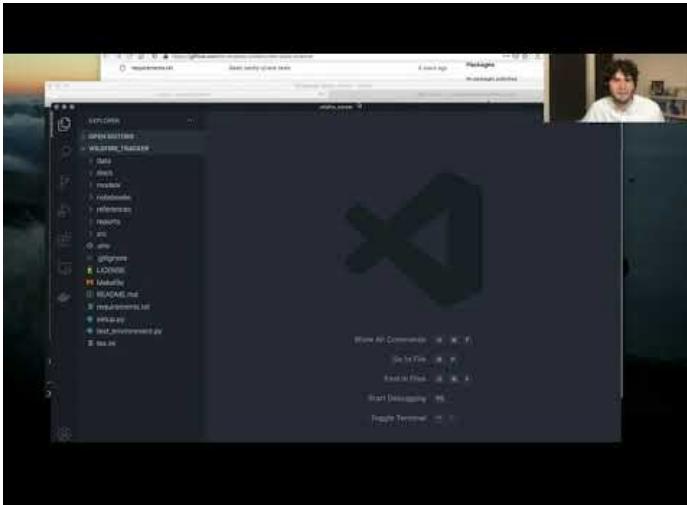
ReviewNB example courtesy of the [ReviewNB website](#)

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

Using text notebooks

Demo

- example.py has a notebook icon! Open it as a notebook
 - Set the appropriate kernel and run the notebook
 - Modify the file in Jupyter, save, and see the change on the script
 - Modify the script in a text editor (1), and reload it in Jupyter:
 - Inputs are updated
 - Outputs are gone (2)
 - Python variables are still there
 - Add the kernel to the .py file with the `IncludeMetadata` command
- (1) If your edit takes more than 2 minutes, answer 'Reload' to the message in Jupyter telling you that the notebook has changed on disk, and consider:
a) turning the jupyter autosave off or b) closing the notebook while you edit the text file
- (2) To be solved at the next slide



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!