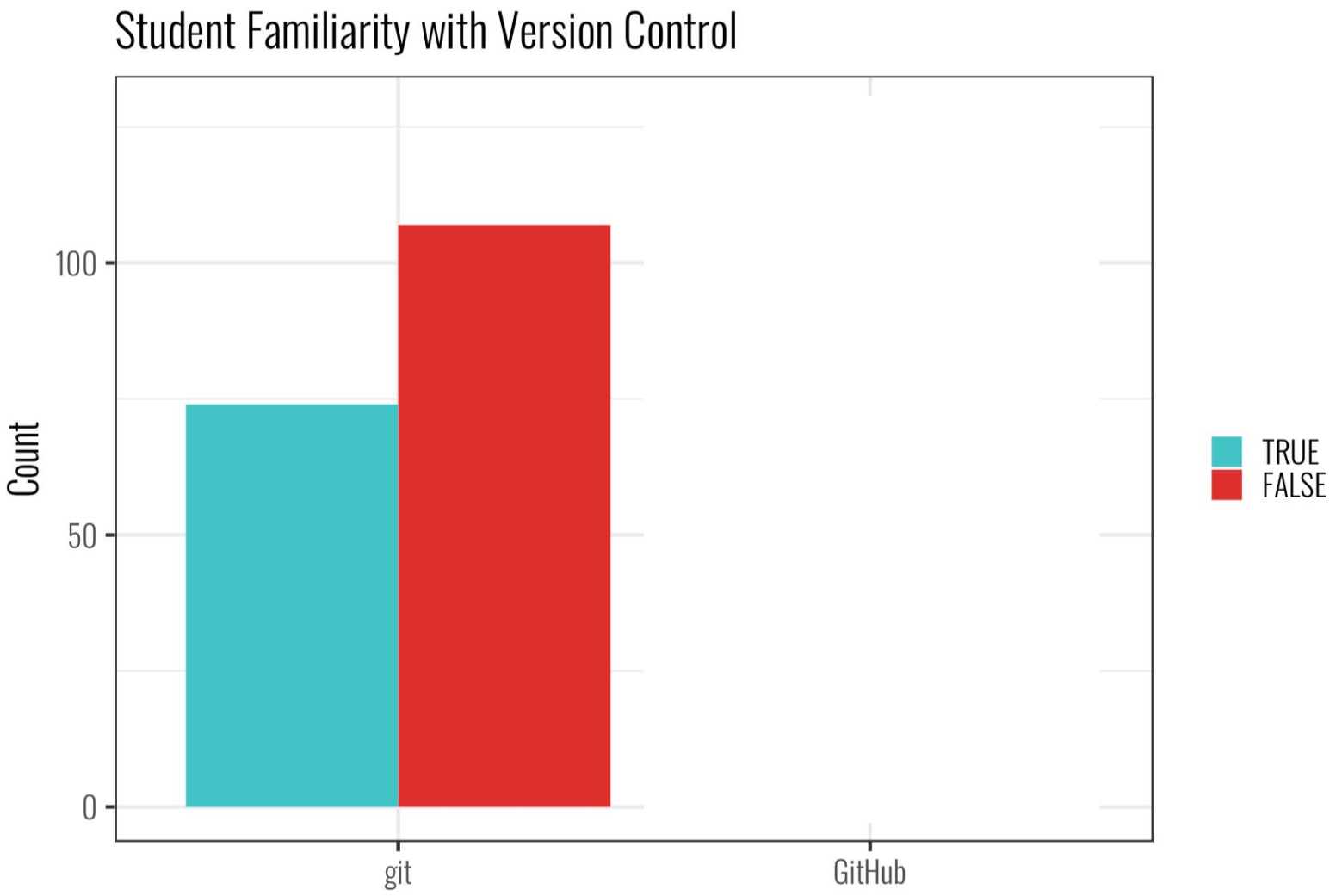
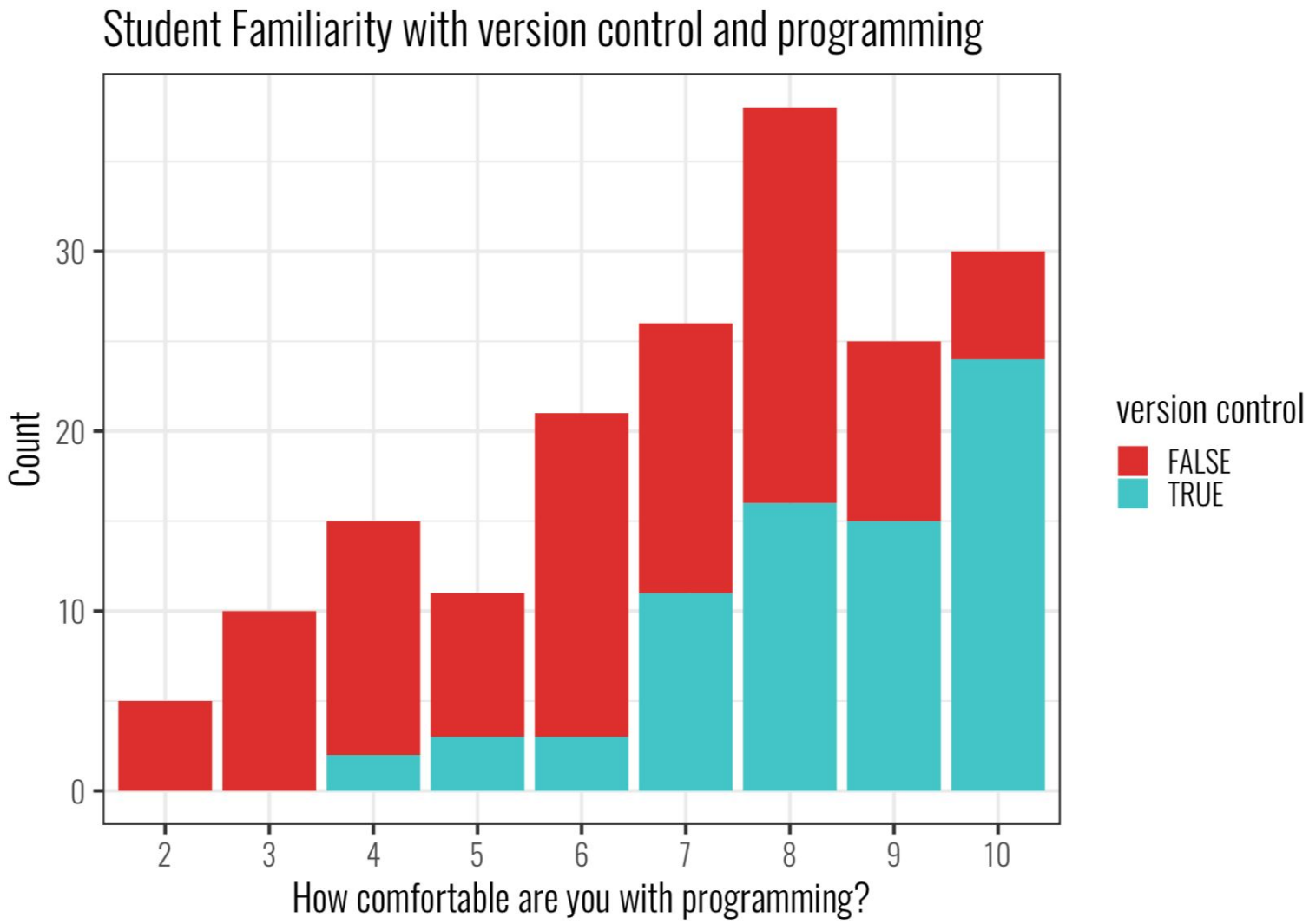
# Version Control

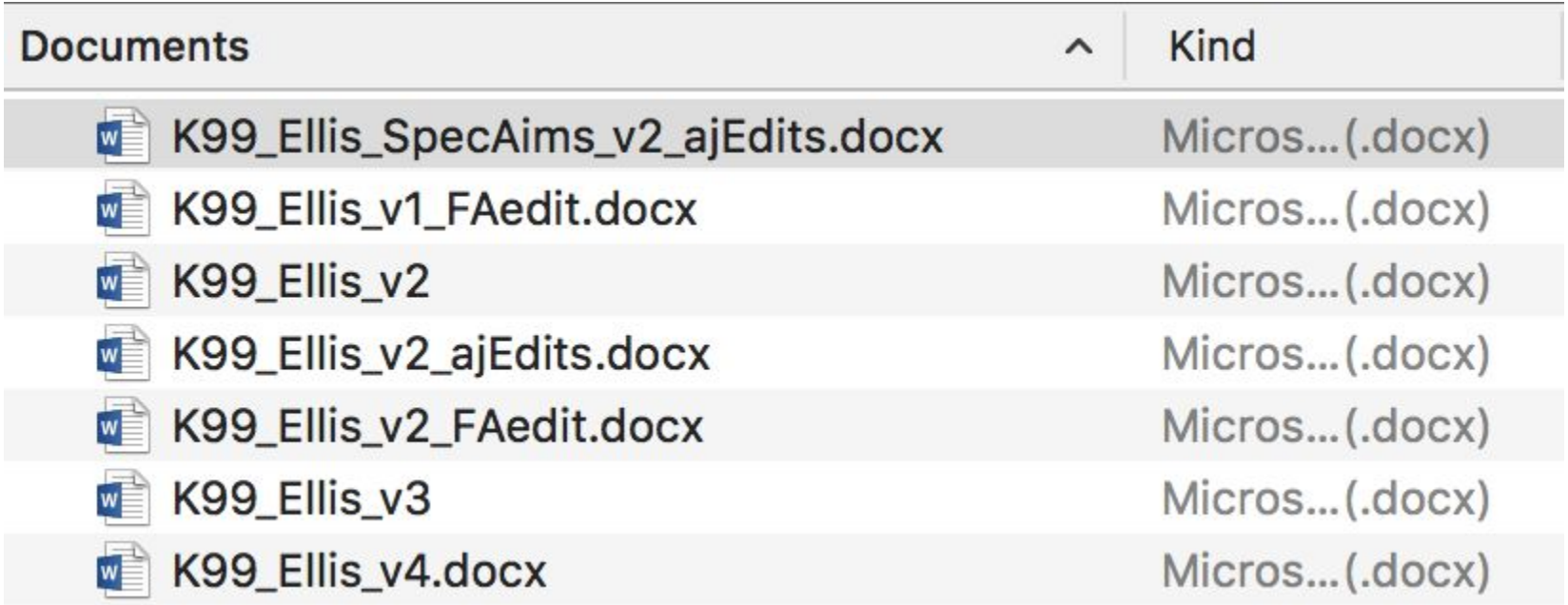
Shannon E. Ellis, Ph.D
UC San Diego

Department of Cognitive Science
sellis@ucsd.edu

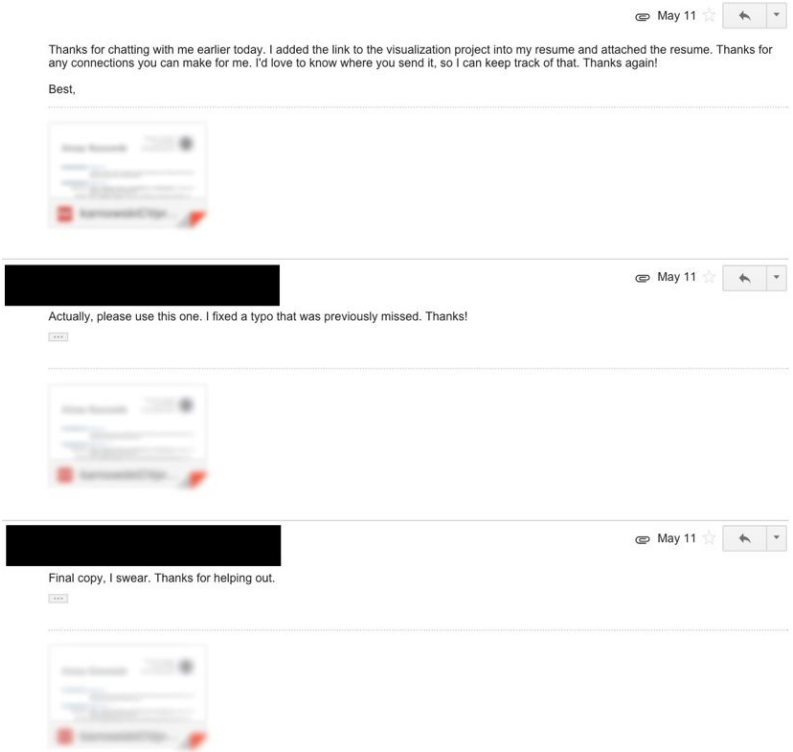

Student Familiarity with Version Control
Count
100
50
0
git
GitHub
TRUE
FALSE


Student Familiarity with version control and programming
Count
30
20
10
0
2
3
4
5
6
7
8
9
10
How comfortable are you with programming?
version control
FALSE
TRUE

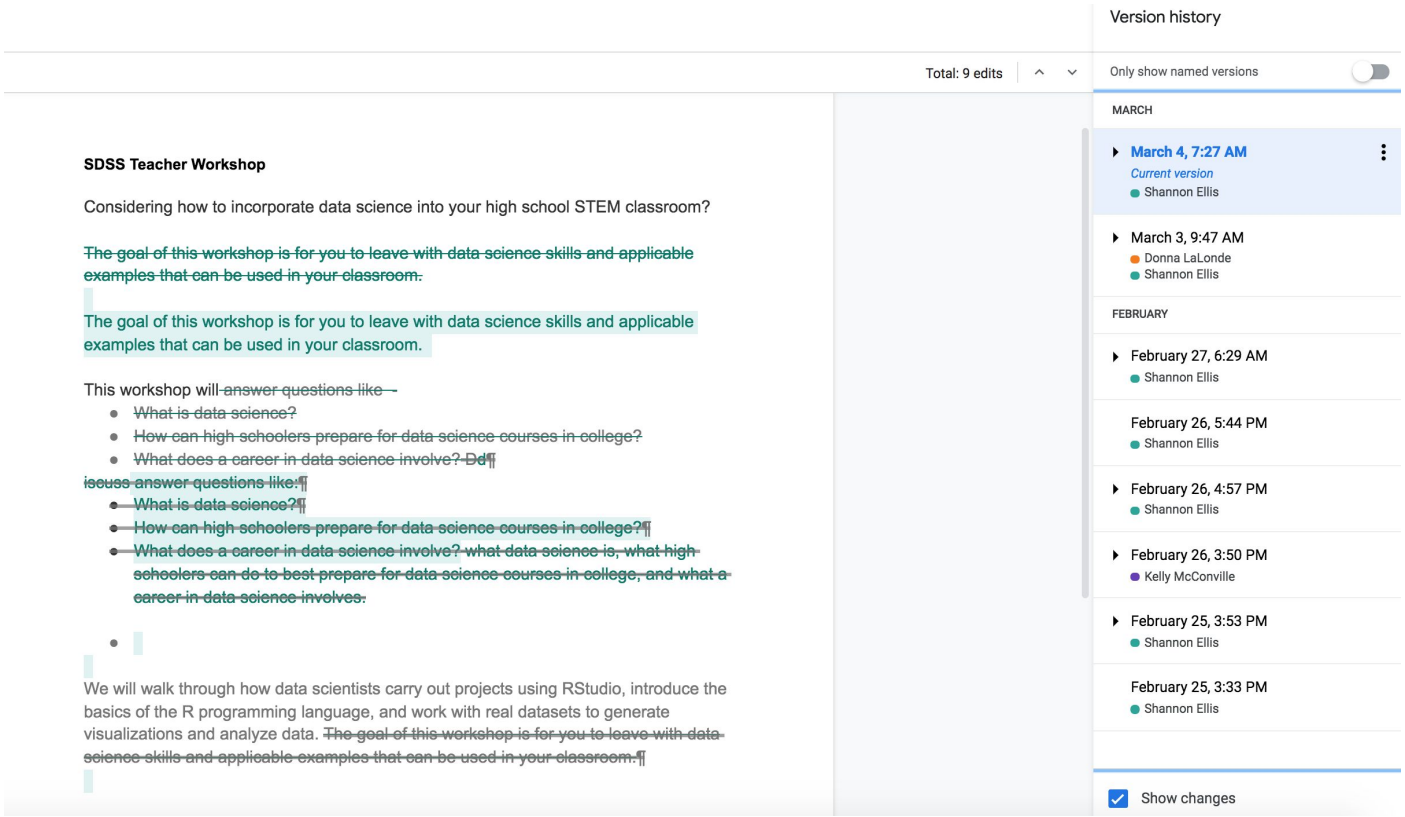
# This sucks


Documents
Kind
K99_Ellis_SpecAims_v2_ajEdits.docx
Micros...(.docx)
K99_Ellis_v1_FAedit.docx
Micros...(.docx)
K99_Ellis_v2
Micros...(.docx)
K99_Ellis_v2_ajEdits.docx
Micros...(.docx)
K99_Ellis_v2_FAedit.docx
Micros...(.docx)
K99_Ellis_v3
Micros...(.docx)
K99_Ellis_v4.docx
Micros...(.docx)

# Yup, this sucks too.



# This is a step in the right direction



# Version Control

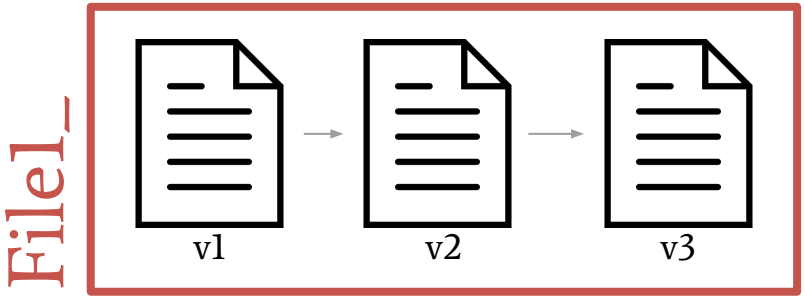
- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.
- Thus, temporary or partial edits by one person do not interfere with another person's work

# What is version control?

A way to manage the evolution of a set of files

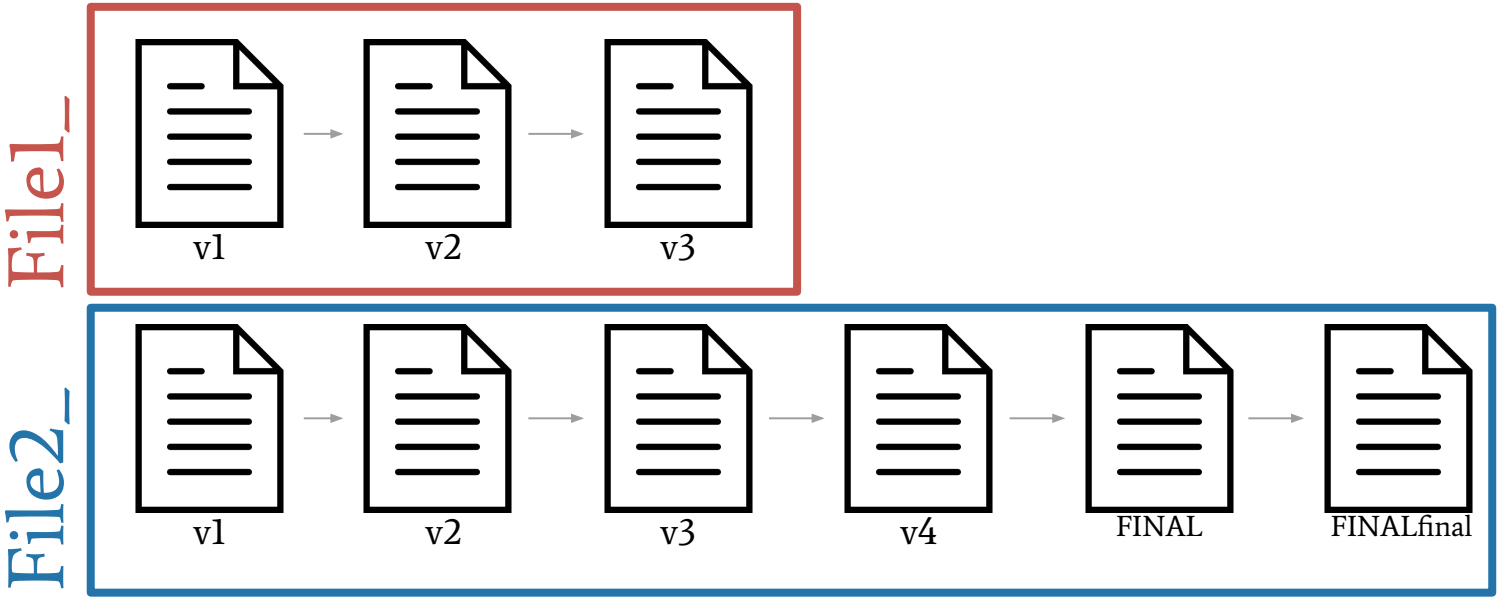
# What is version control?

A way to manage the evolution of a set of files



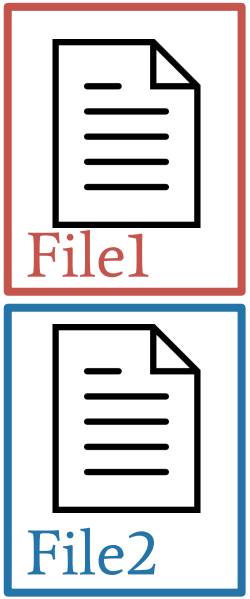
# What is version control?

A way to manage the evolution of a set of files



# What is version control?

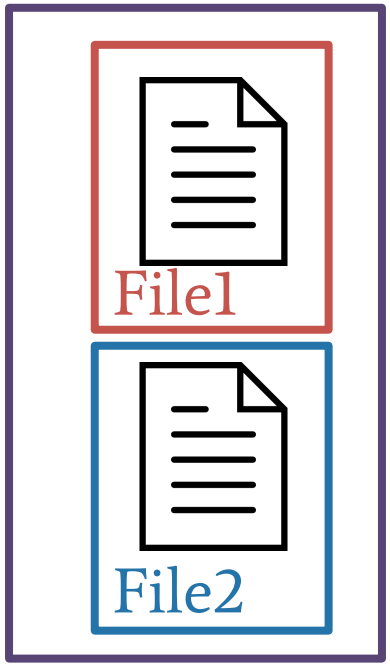
A way to manage the evolution of a set of files



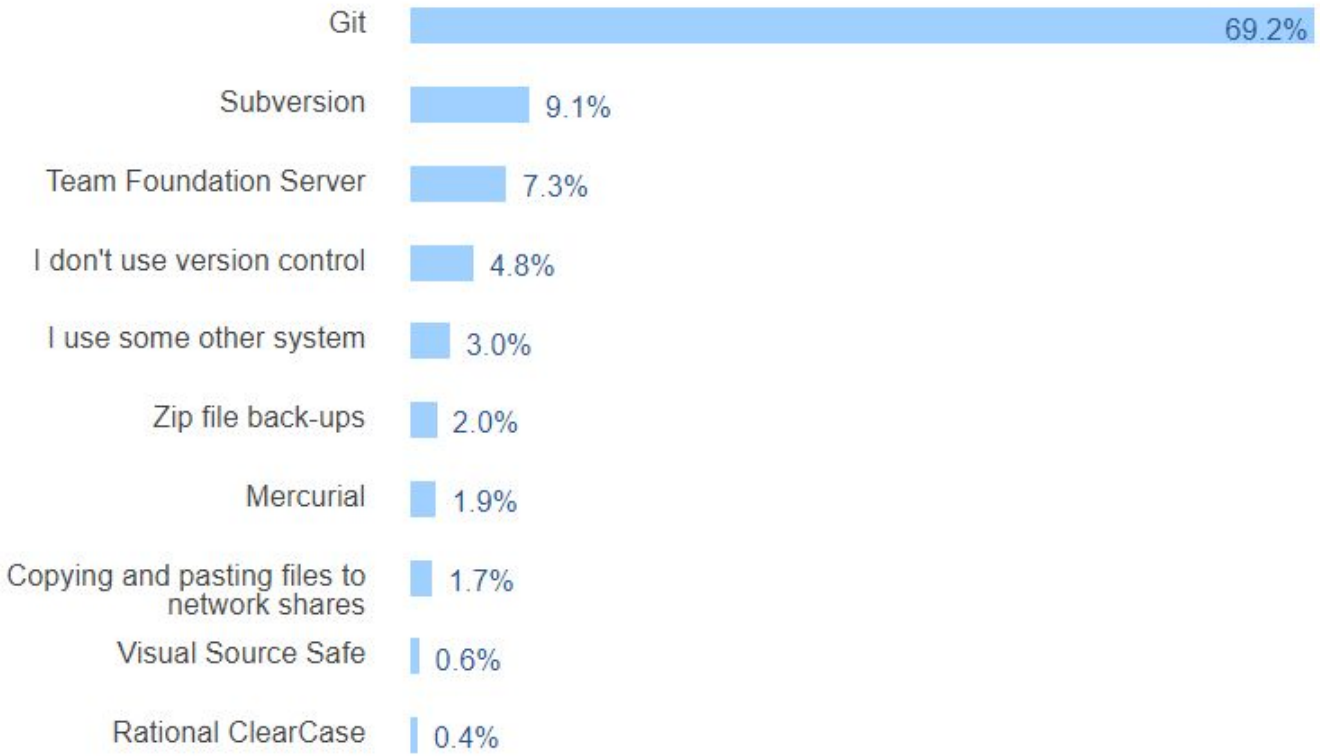

When using a version control system, you have **one copy of each file** and the *version control system tracks the changes* that have occurred over time

# 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)**



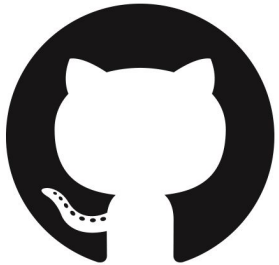

https://insights.stackoverflow.com/survey/2017#work-version-control

# 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.

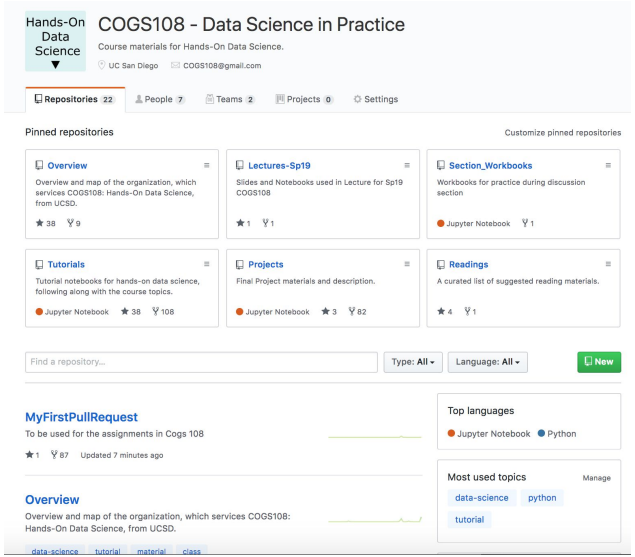
~ Dropbox....but way better

# 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
```

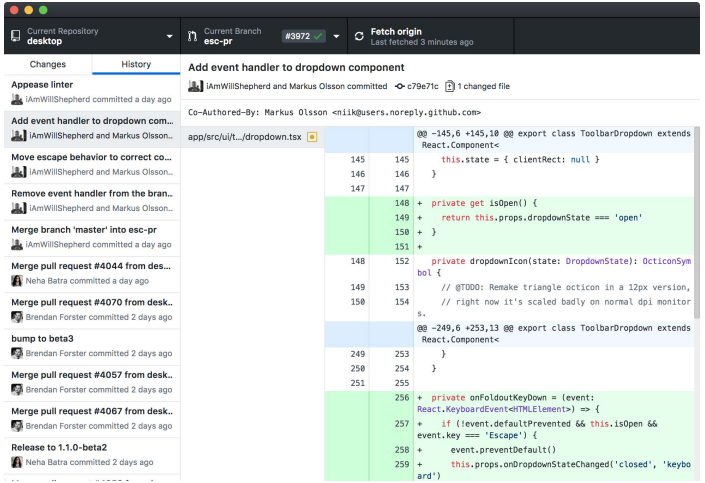
Terminal

**git**

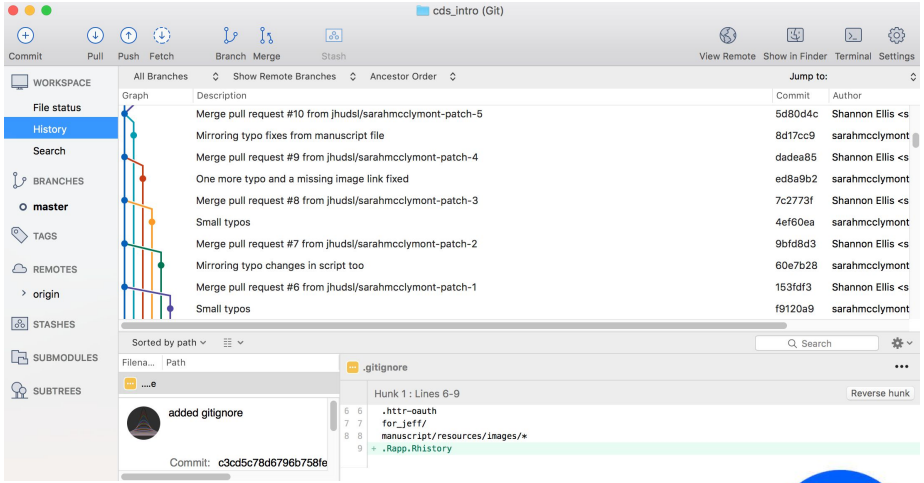



GitHub

# GUIs can be helpful when working with version control

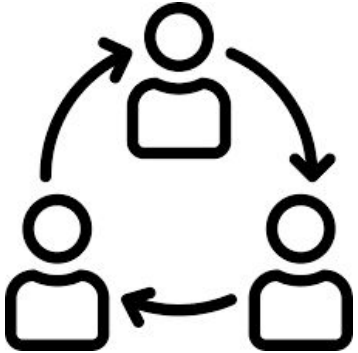



GitHub Desktop




SourceTree

# Why version control with git and GitHub?



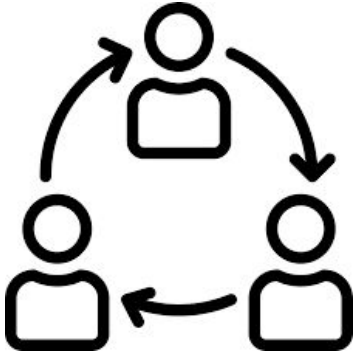
Collaboration

Returning to a safe state

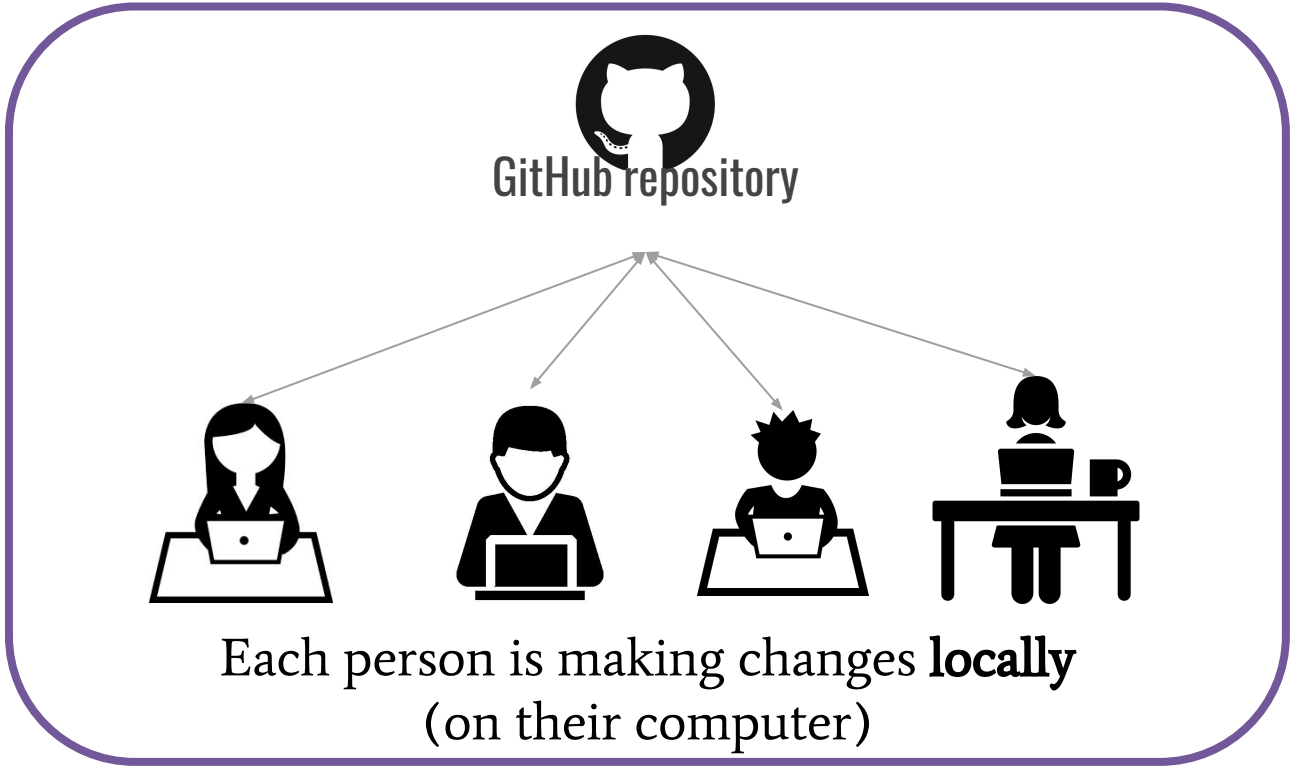
Exposure for your work

Tracking others' work

# Collaborate like you do with Google Docs

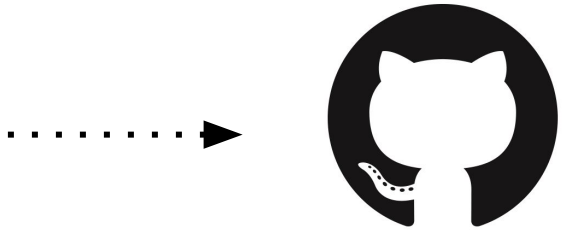


Collaboration



# 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

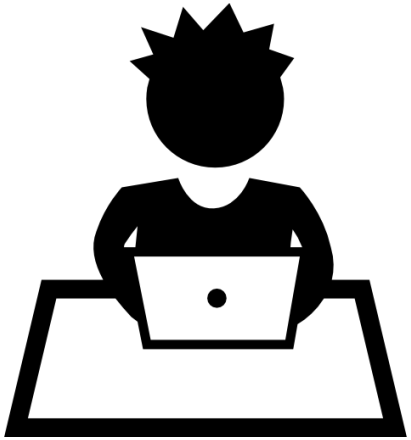
# Your repositories will be visible to others!

Exposure for your work

Your public GitHub repos are your coding social media

# Keep up with others' work easily

Tracking others' work

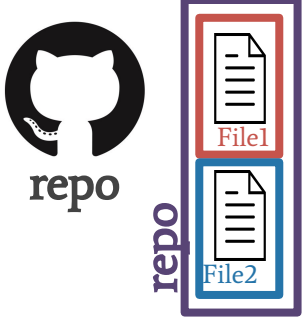


As a social platform, you can see others' work too!

# When you'll have to use git/GitHub in this course:

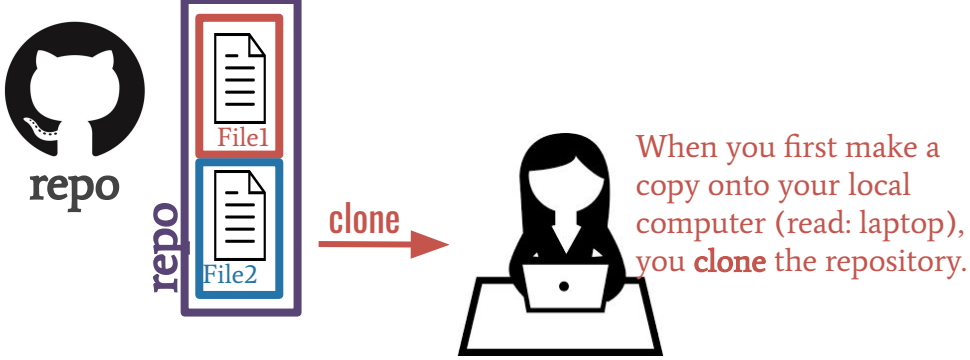
- Course materials
- Completing A1 (individual)
- Final Project (one submission per group)
  - Proposal
  - Check-In
  - Submission

Note: You're encouraged to put projects on GitHub.
Please do not put assignments on GitHub.

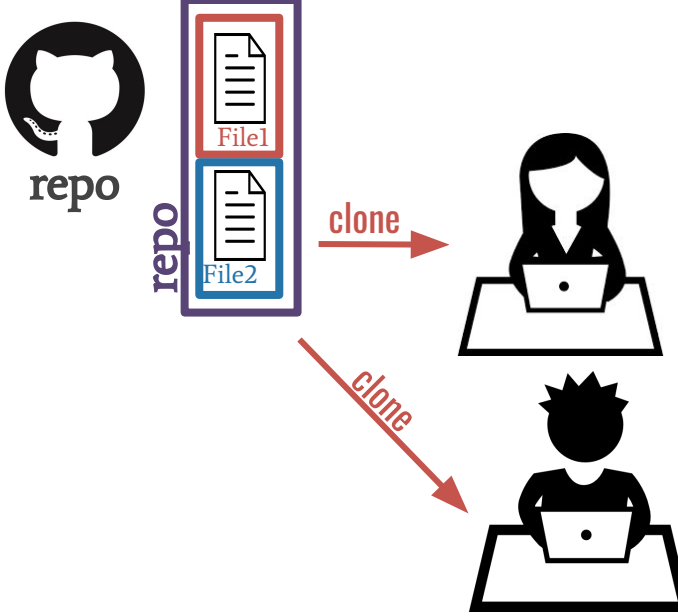



A **GitHub repo** contains all the files and folders for your project.

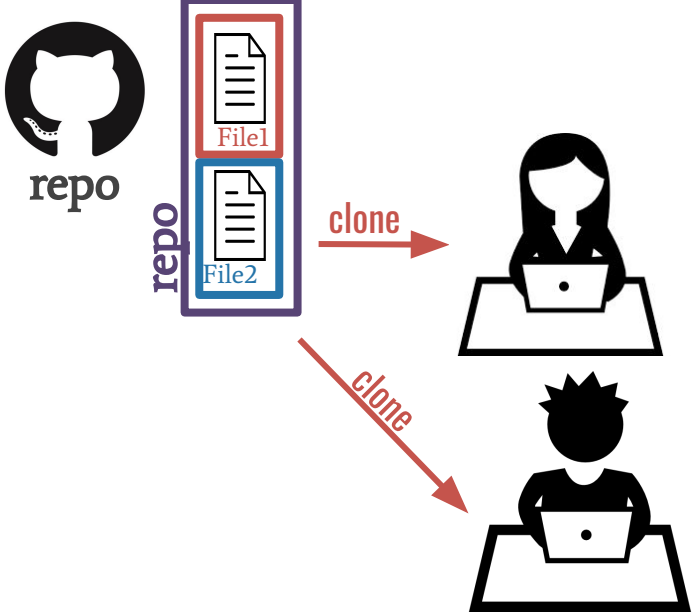
GitHub is a **remote host**. The files are geographically distant from any files on your computer.



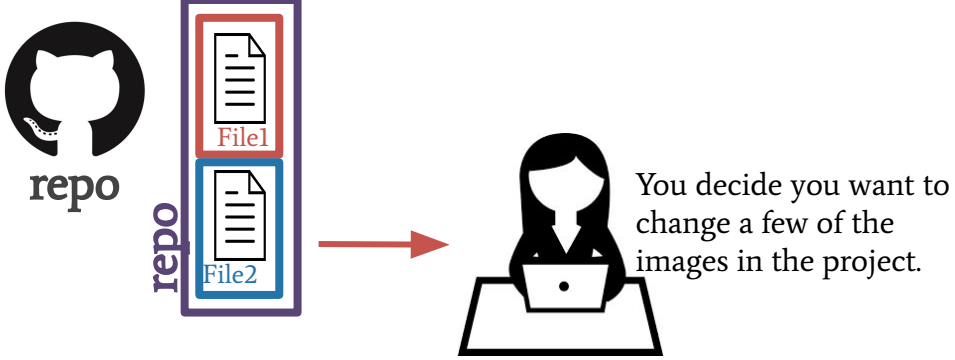

When you first make a copy onto your local computer (read: laptop), you **clone** the repository.




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!

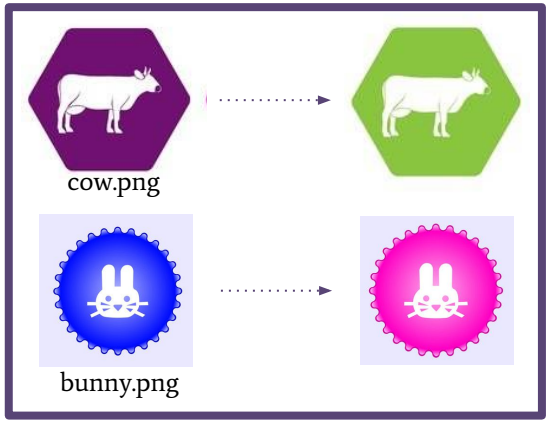


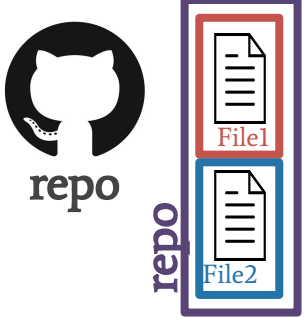

You decide you want to change a few of the images in the project.

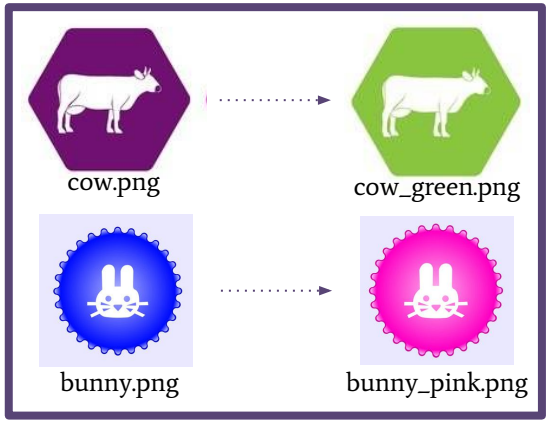




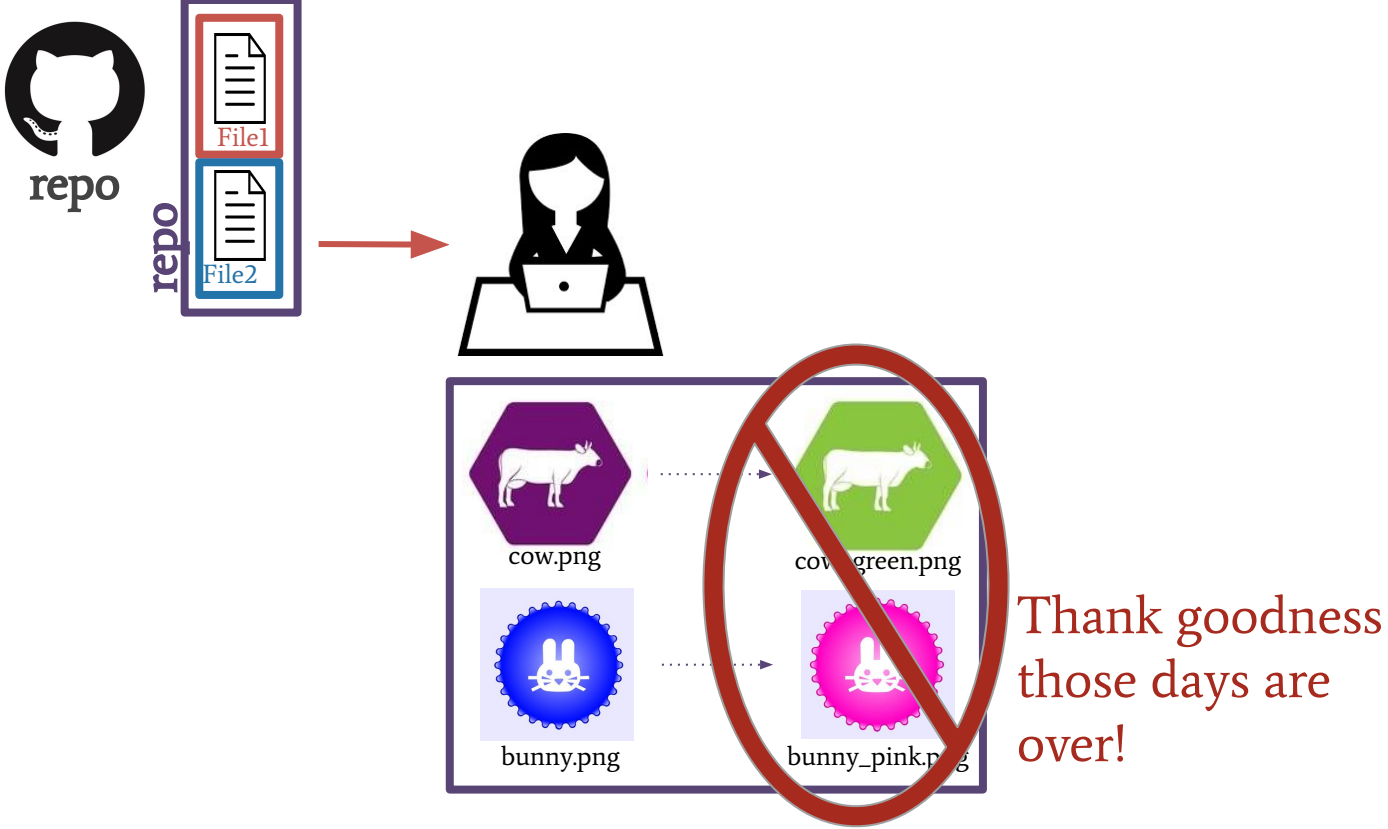
You decide you want to change a few of the images in the project.



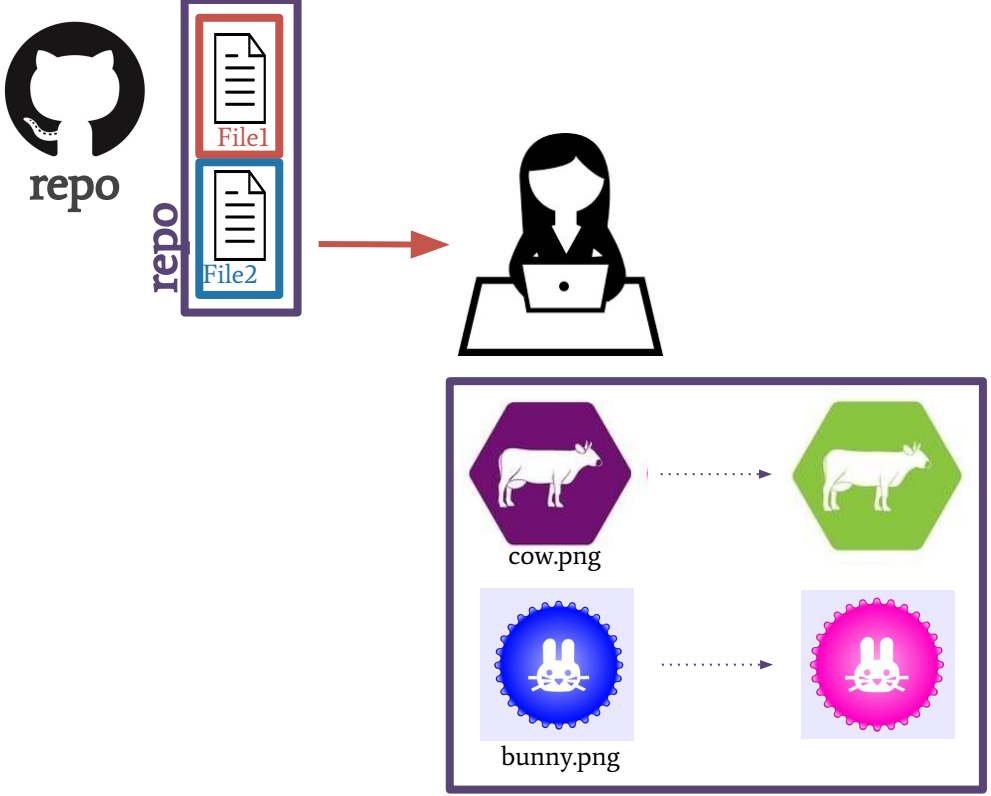


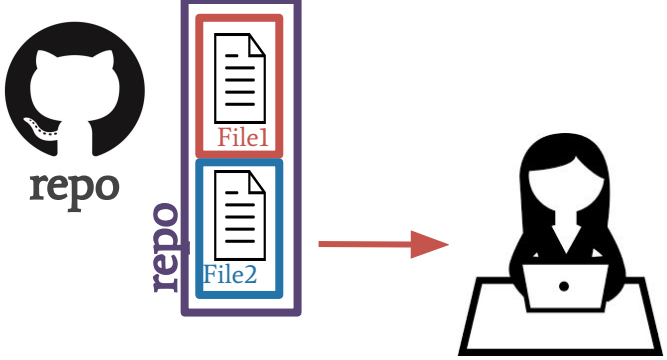

without git...you'd likely rename these files....



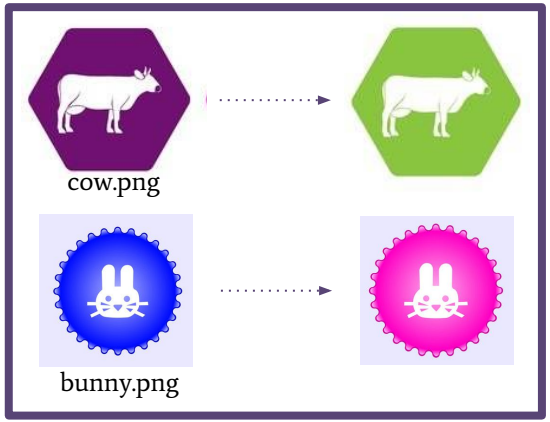

Thank goodness those days are over!



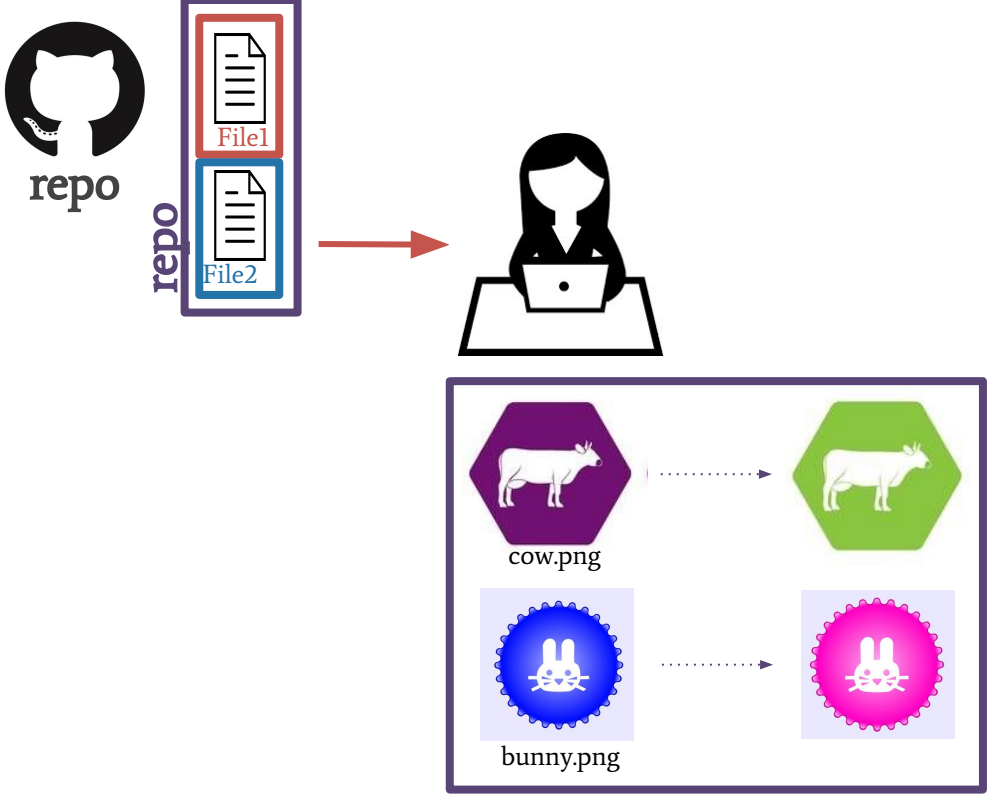

Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.




| | |
|---|---|
| `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 |



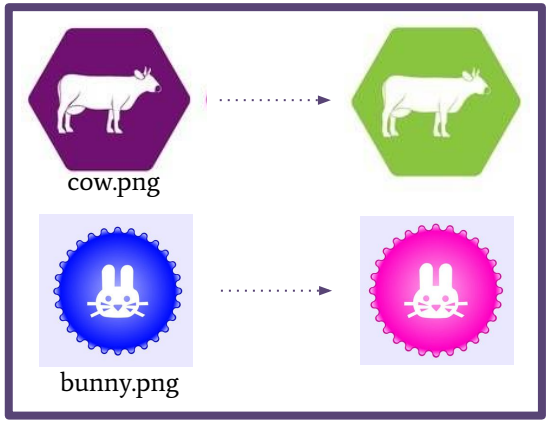

Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.




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

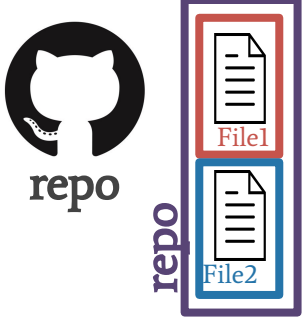


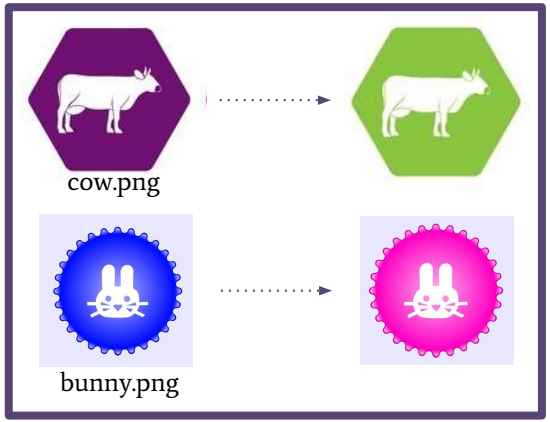





Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

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



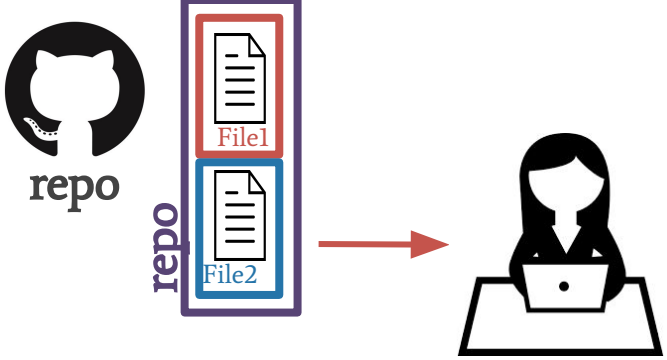




You can make commits more informative by adding a **commit message**.

Example: `git commit -m "changed colors for animal icons"`

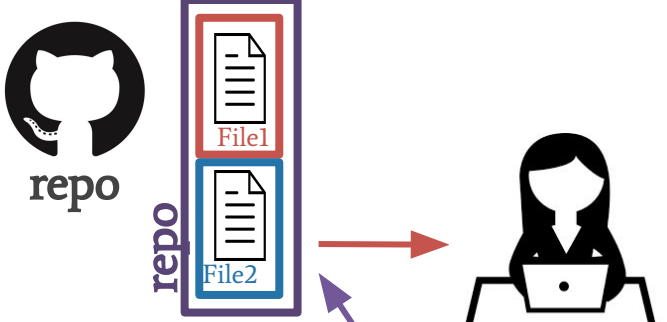
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

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




Shannon Ellis
*11/28/18 3:28pm*

*changed colors for animal icons*



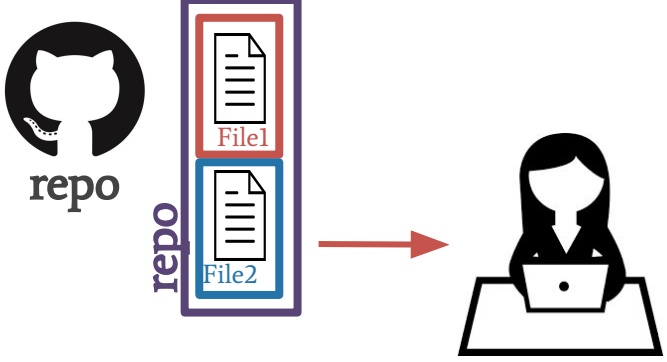



push

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.

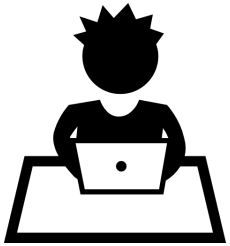
Shannon Ellis
*11/28/18 3:28pm*

*changed colors for animal icons*




Shannon Ellis
*11/28/18 3:28pm*

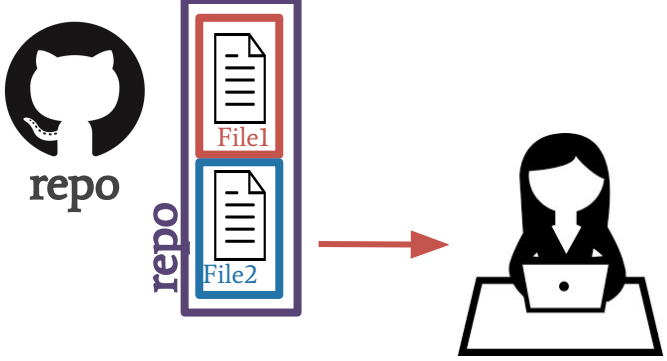
*changed colors for animal icons*



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

Shannon Ellis
*11/28/18 3:28pm*

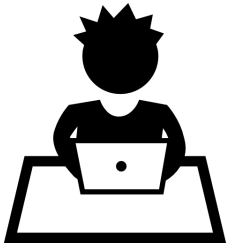
*changed colors for animal icons*




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

Shannon Ellis
*11/28/18 3:28pm*

*changed colors for animal icons*



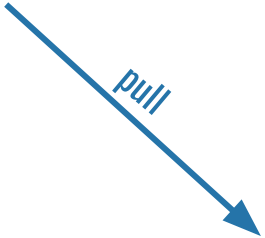
Your teammate is still working with the (out-of-date) copy he cloned earlier!

Shannon Ellis
*11/28/18 3:28pm*

*changed colors for animal icons*



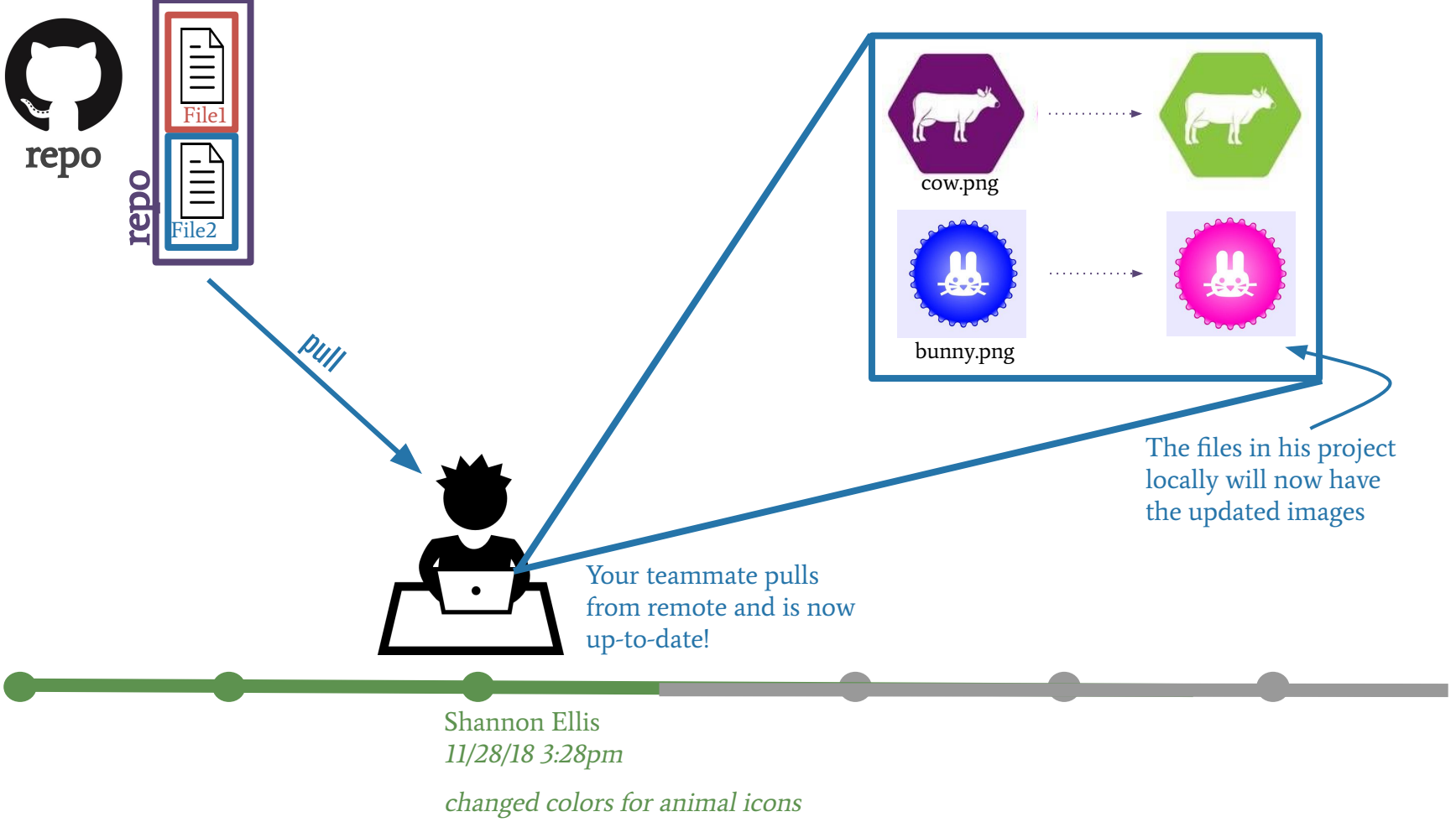


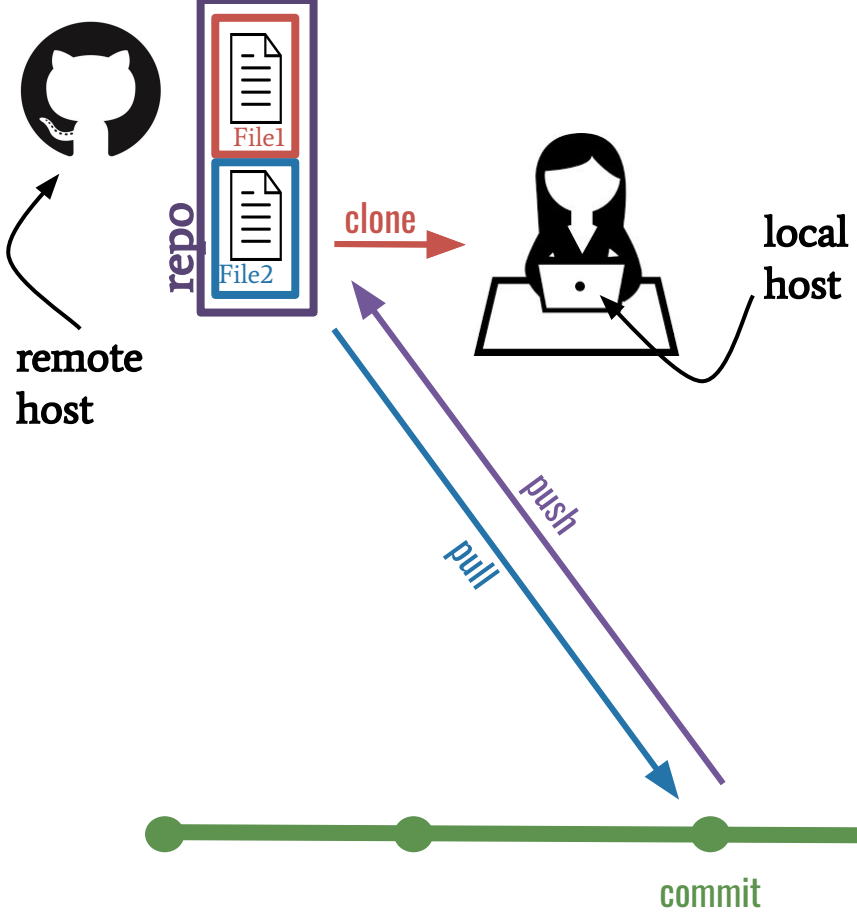



Your teammate pulls from remote and is now up-to-date!

Shannon Ellis
*11/28/18 3:28pm*

*changed colors for animal icons*


repo
repo
File1
File2
pull
cow.png
bunny.png
The files in his project locally will now have the updated images
Your teammate pulls from remote and is now up-to-date!
Shannon Ellis
11/28/18 3:28pm
changed colors for animal icons




# 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

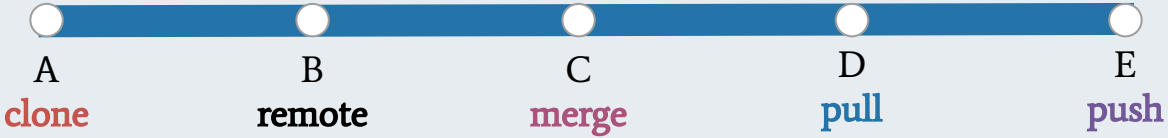
```
(base) sellis:Projects shannonellis$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        FinalProject_Guidelines.pdf

nothing added to commit but untracked files present (use "git add" to track)
(base) sellis:Projects shannonellis$ git add FinalProject_Guidelines.pdf
(base) sellis:Projects shannonellis$ git commit -m "update Project Guidelines"
[master 264e91a] update Project Guidelines
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 FinalProject_Guidelines.pdf
(base) sellis:Projects shannonellis$ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 148.21 KiB | 29.64 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/COGS108/Projects.git
   6931768..264e91a  master -> master
```

# Review & Question Time

# Version Controller I

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.

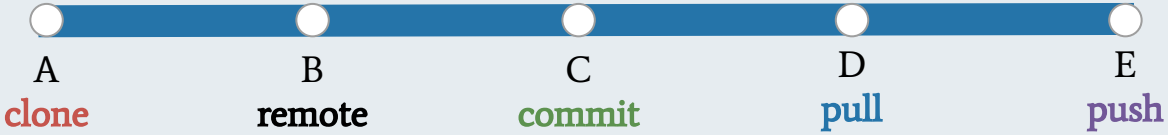
## What do you do to get them on the remote?

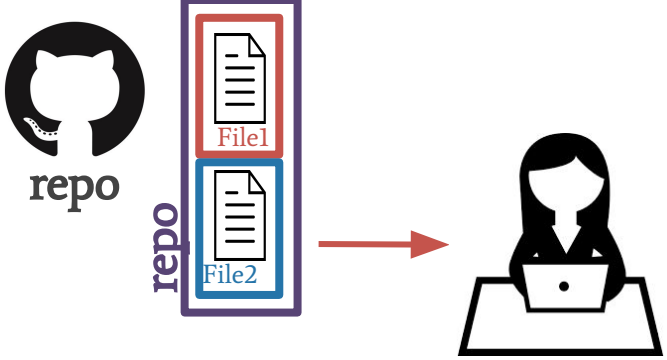


# Version Controller II

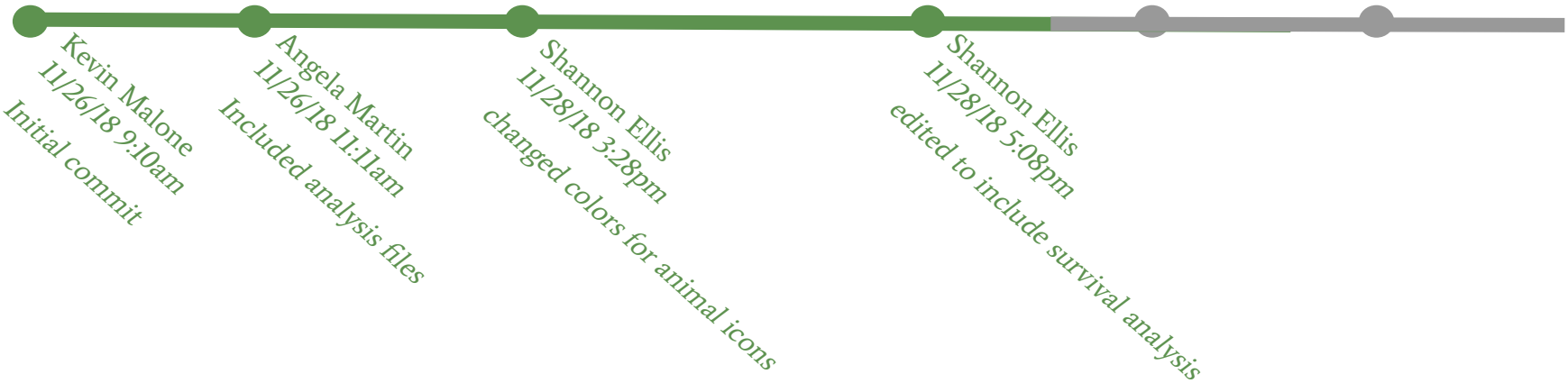
Your teammate has given you access to a GitHub repository to work on a project together. You want to get them for the first time on your computer locally.

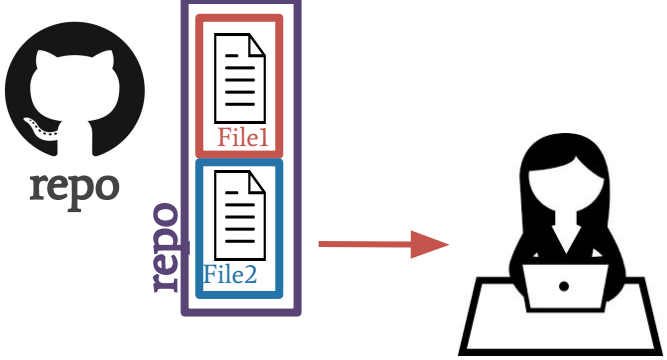
## What do you do to get the repo on your computer?



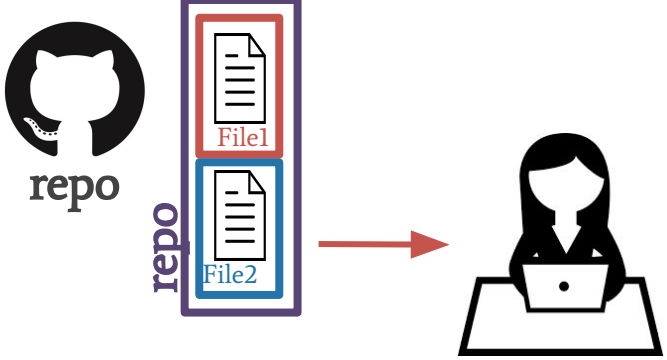



Each time you create a commit, git tracks the changes made automatically.






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



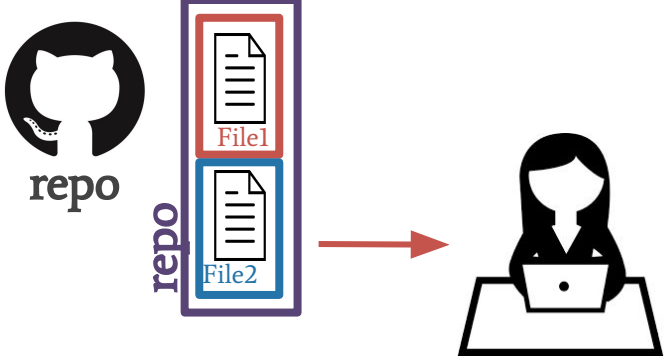

By committing each time you make changes, git allows you to time travel!

377dfcd00dd057542b112cf13be6cf1380b292ad

439301fe69e8f875c049ad0718386516b4878e22

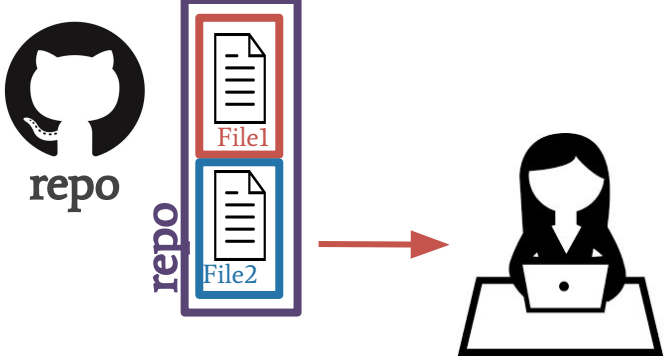
456722223e9f9e0ee0a92917ba80163028d89251

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

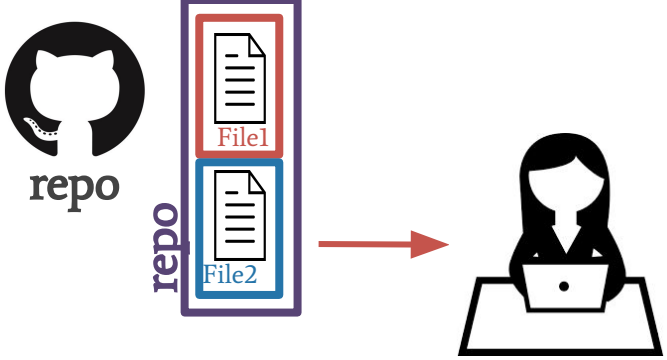



You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.

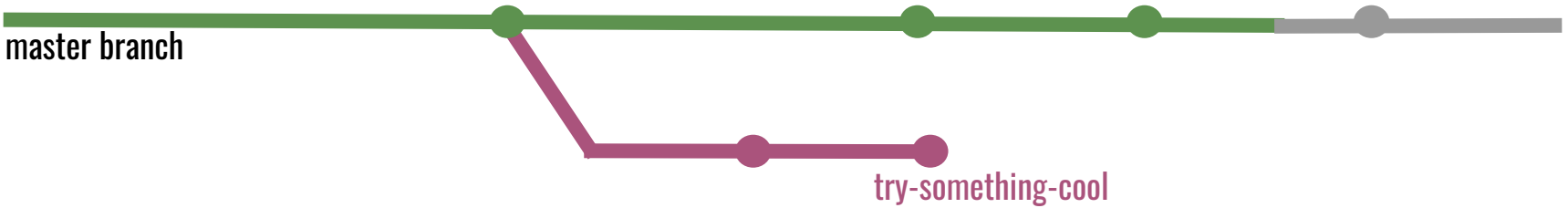
377dfcd00dd057542b112cf13be6cf1380b292ad

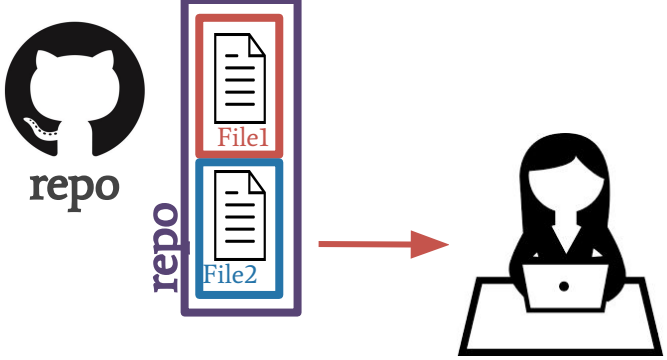



But...not everything is always linear. Sometimes you want to try something out and you're not sure it's going to work. This is where you'll want to use a **branch**.

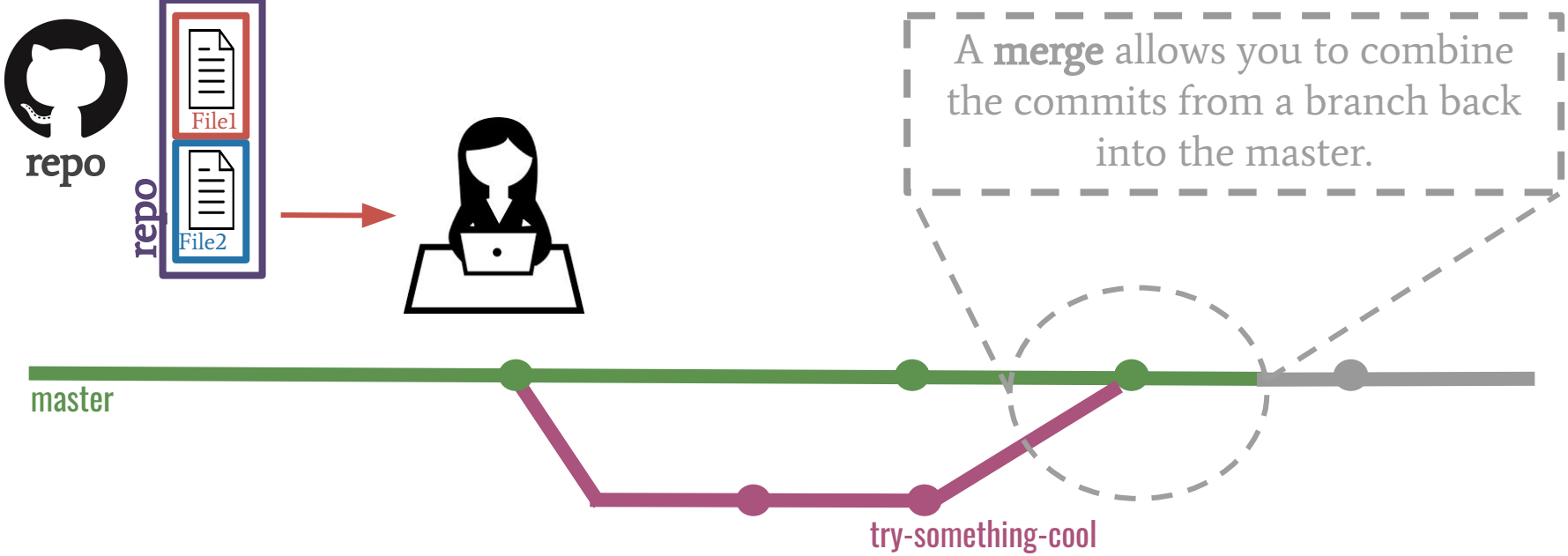


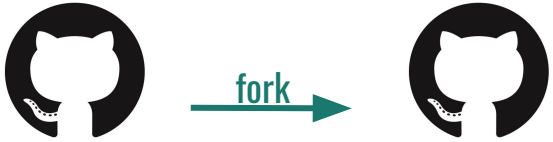

It's a good way to experiment. It's pretty easy to get rid of a branch later on should you not want to include the commits on that branch.



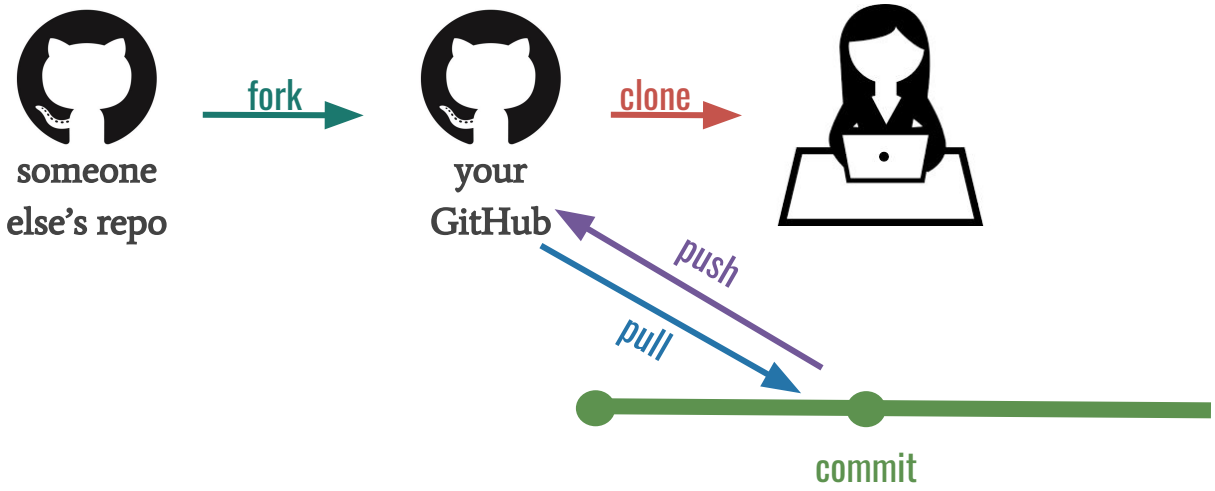



But...what if you DO want to include the changes you've made on your **try-something-cool** branch into the **master** branch?

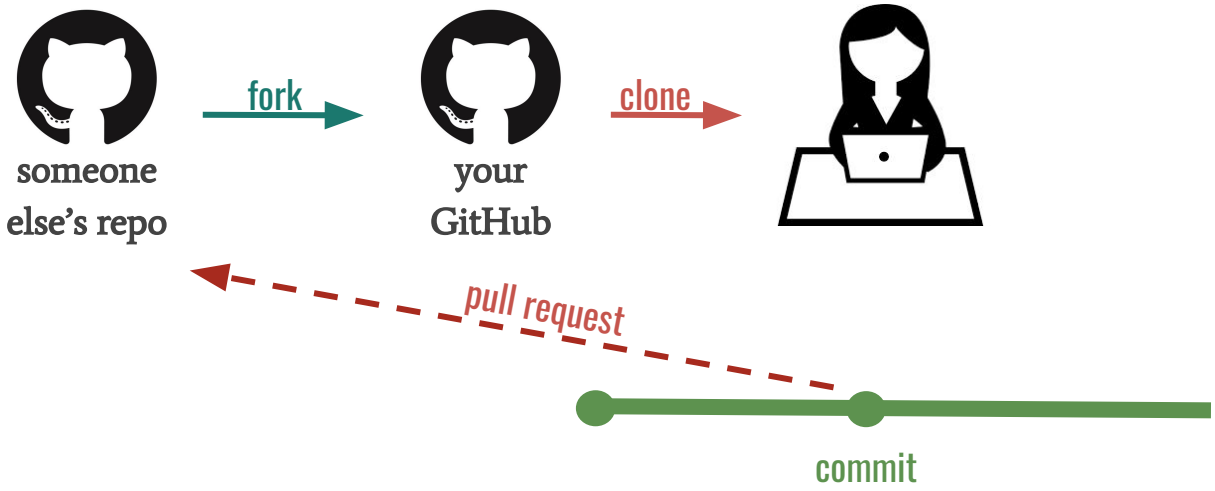

repo
repo
File1
File2
A merge allows you to combine the commits from a branch back into the master.
master
try-something-cool



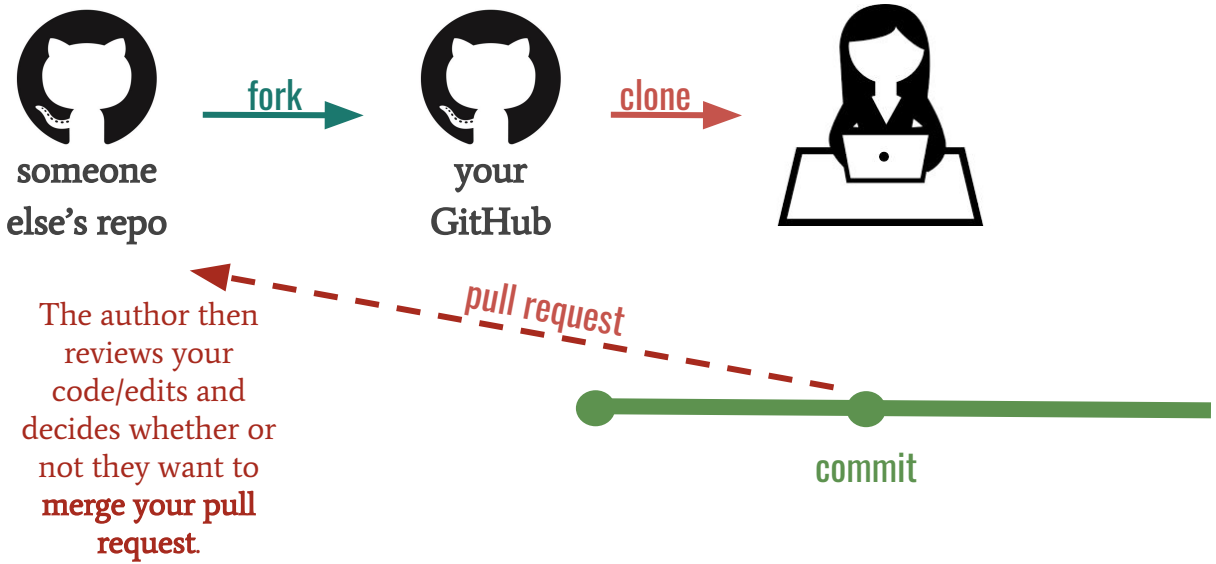

What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.




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.

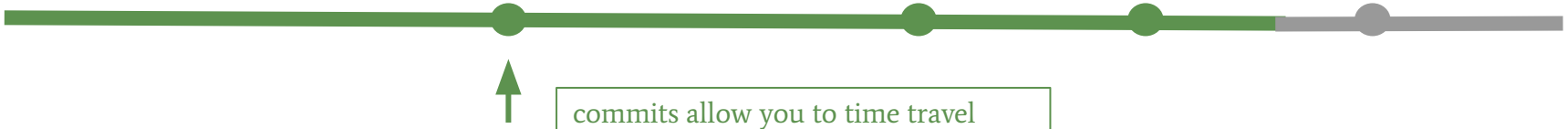
someone else's repo

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.

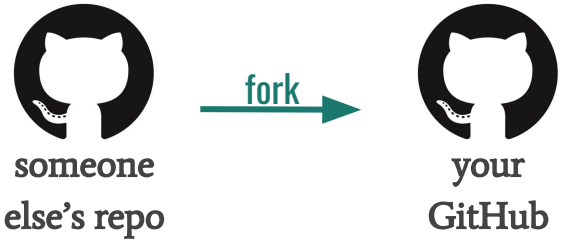
They can be closed once addressed ....or if the software maintainer doesn't like the suggestion




# 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

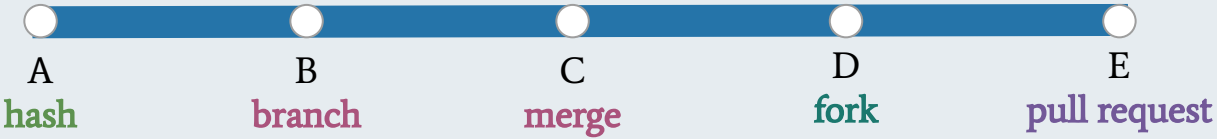
# One more git recap...

# Review & Question Time

# Version Controller III

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

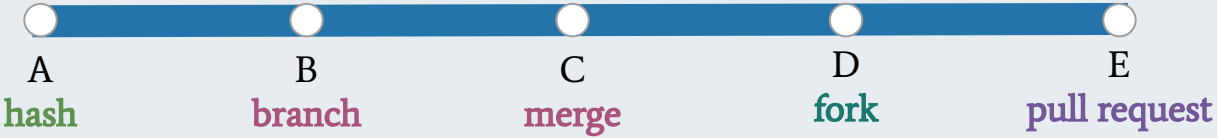
## what should you do?



# Version Controller IV

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

- Assignment 1: Part 1
  - This will get you practice with git & GitHub
  - Understand what you're doing in the assignment!
  - You may have to google, ask others, spend some time with this!
  - Part II is a Python review; each part of this assignment is self-contained
  - Do this part of the assignment ASAP
- 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

# COGS 108 Final Projects

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

# 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.

# Deadlines & Action Items

**Project Proposal**: Due Fri, Jan 31st @ 11:59 PM (Week 4)

**Project Check-In**: Due Fri, Feb 21st @ 11:59 PM (Week 7)

**Project Survey**: Due Fri, Mar 13st @ 11:59 PM (Week 10)

**Final Project**: Due THURS, Mar 19th @ 11:59 PM (Finals Week)

# Project Proposal (10%)

https://github.com/COGS108/Projects/blob/master/ProjectProposal.ipynb

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