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

Reminders

<u>Upcoming due dates</u>

```
Mon Jan 12th Quiz 1
Wed Jan 14th Pre-course survey
Fri Jan 16th Discussion Lab 1
             #FinAid quiz on Canvas
```

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

Version Control 2.0

Data Science in Practice

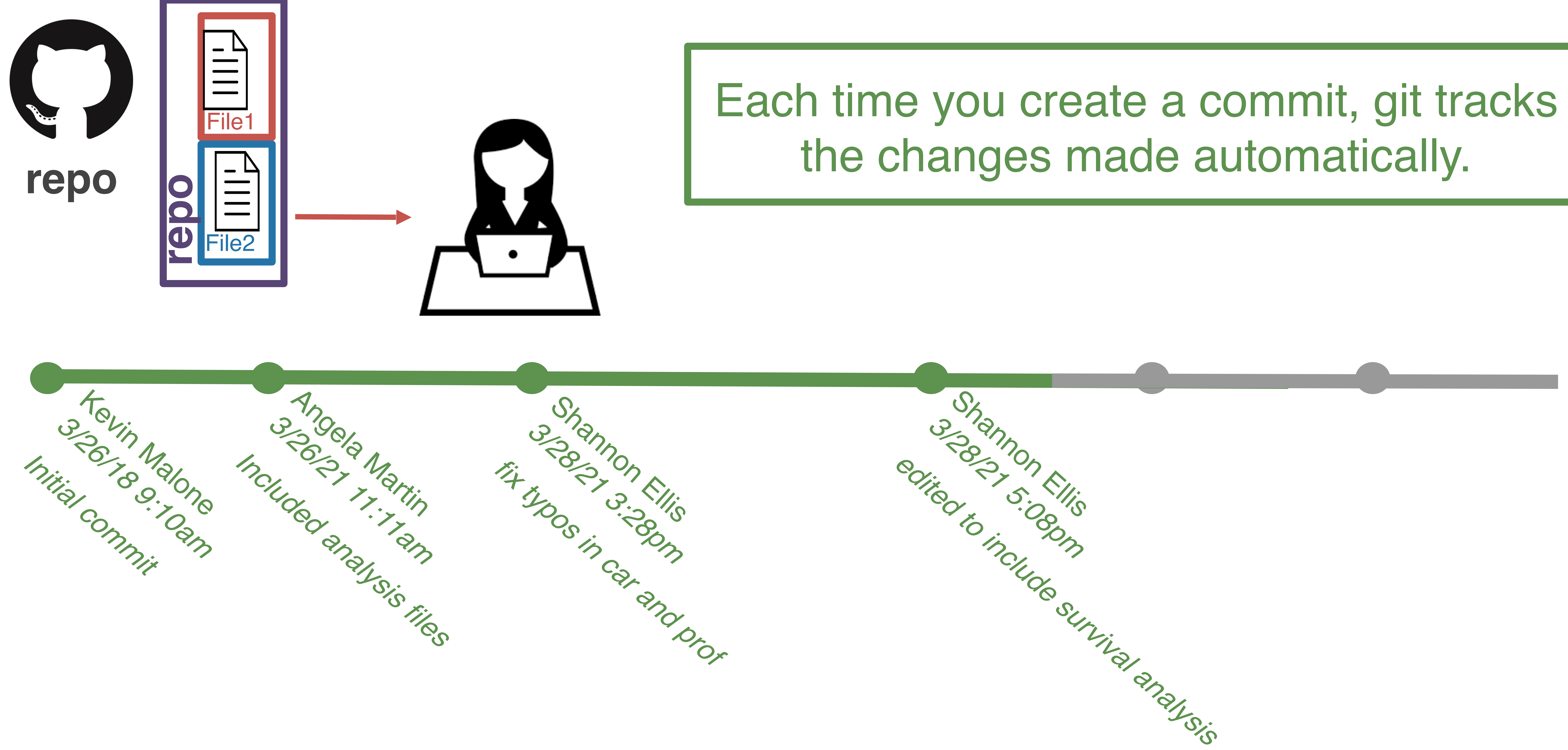
Jason G. Fleischer, PhD **Dept. of Cognitive Science** **UC San Diego** **https://jgfleischer.com**

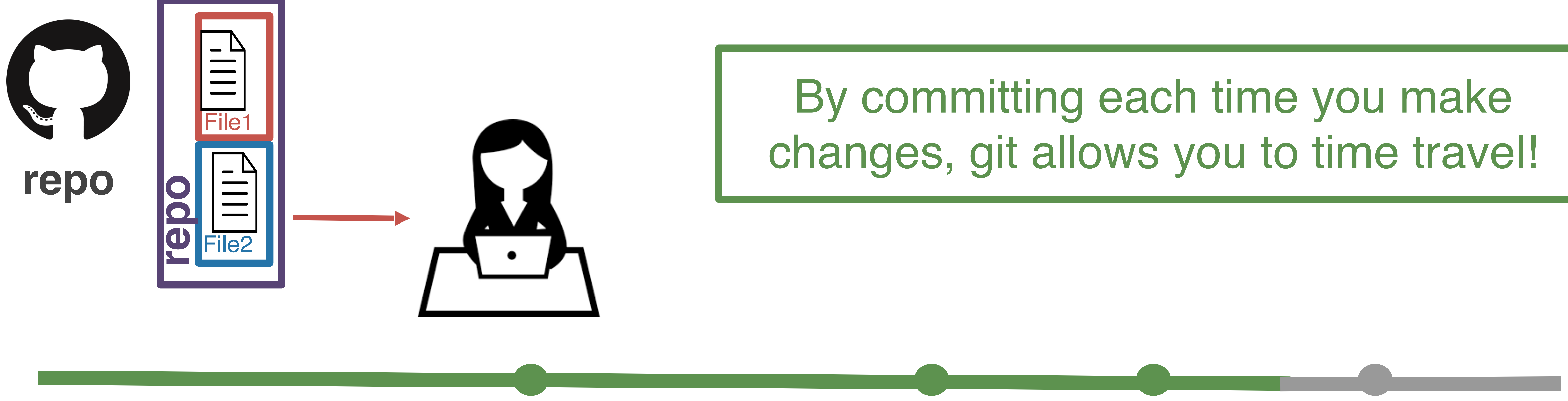
Code
Issue
Pull Request
Open
Comment

Moisture data sharing

What if farmers could share moisture data?!

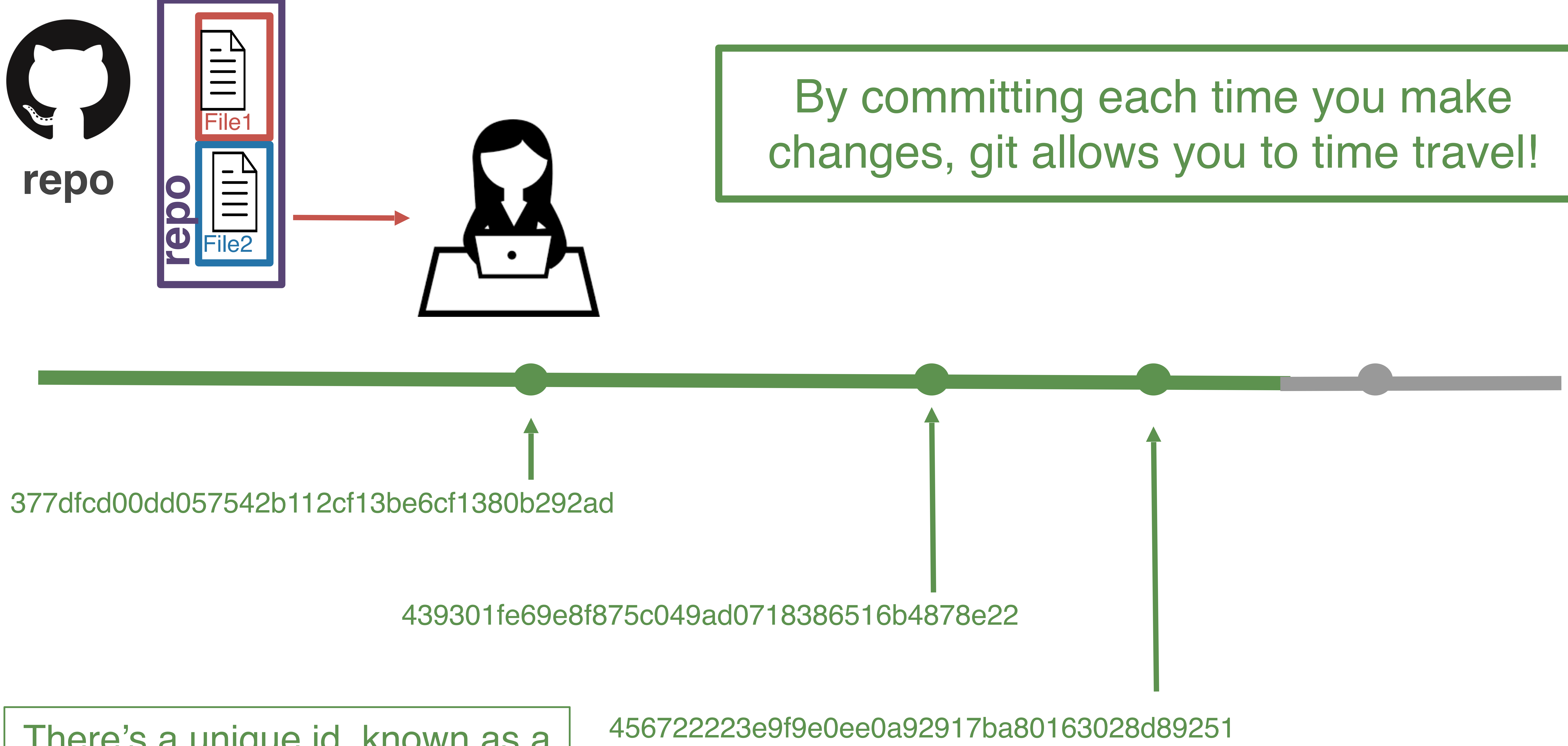
Each time you create a commit, git tracks the changes made automatically.



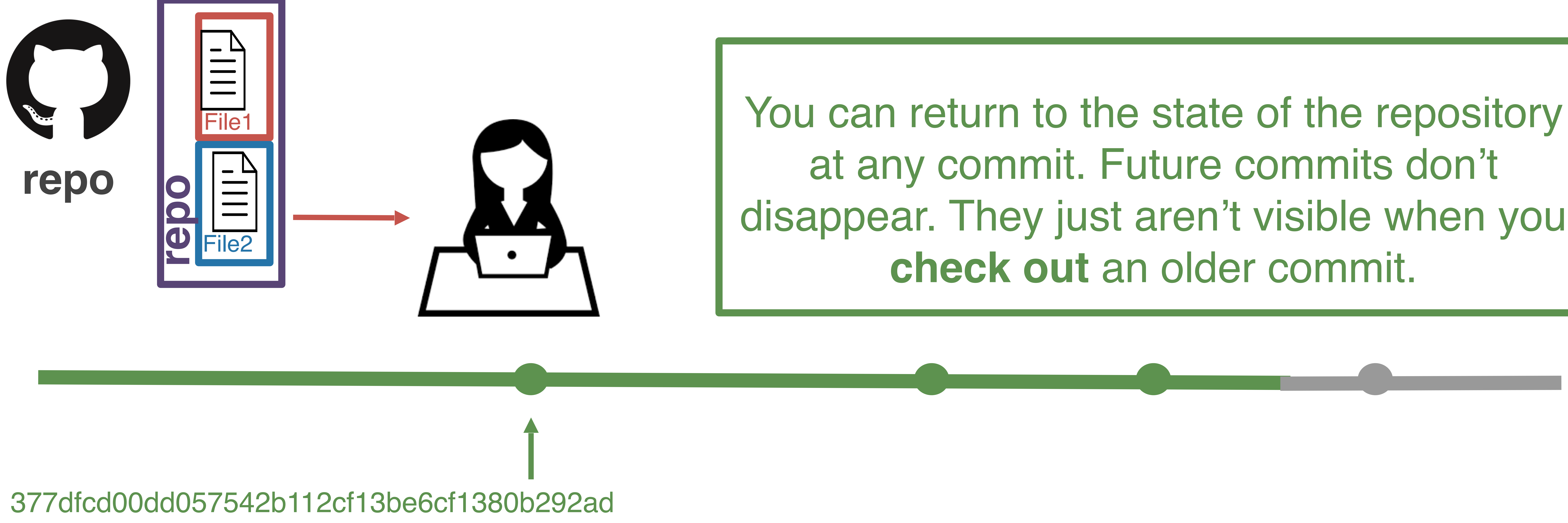


By committing each time you make changes, git allows you to time travel!

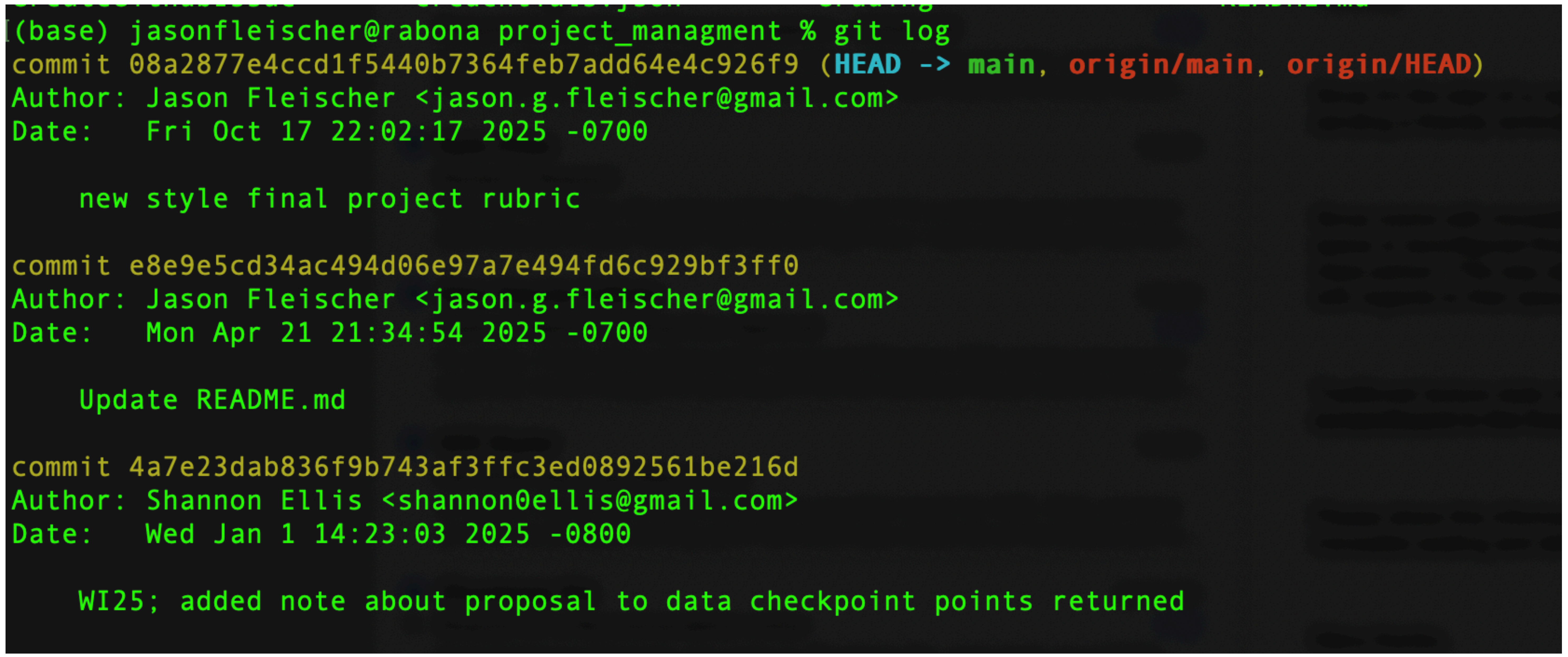
By committing each time you make changes, git allows you to time travel!


repo
repo
File1
File2
377dfcd00dd057542b112cf13be6cf1380b292ad
439301fe69e8f875c049ad0718386516b4878e22
456722223e9f9e0ee0a92917ba80163028d89251

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

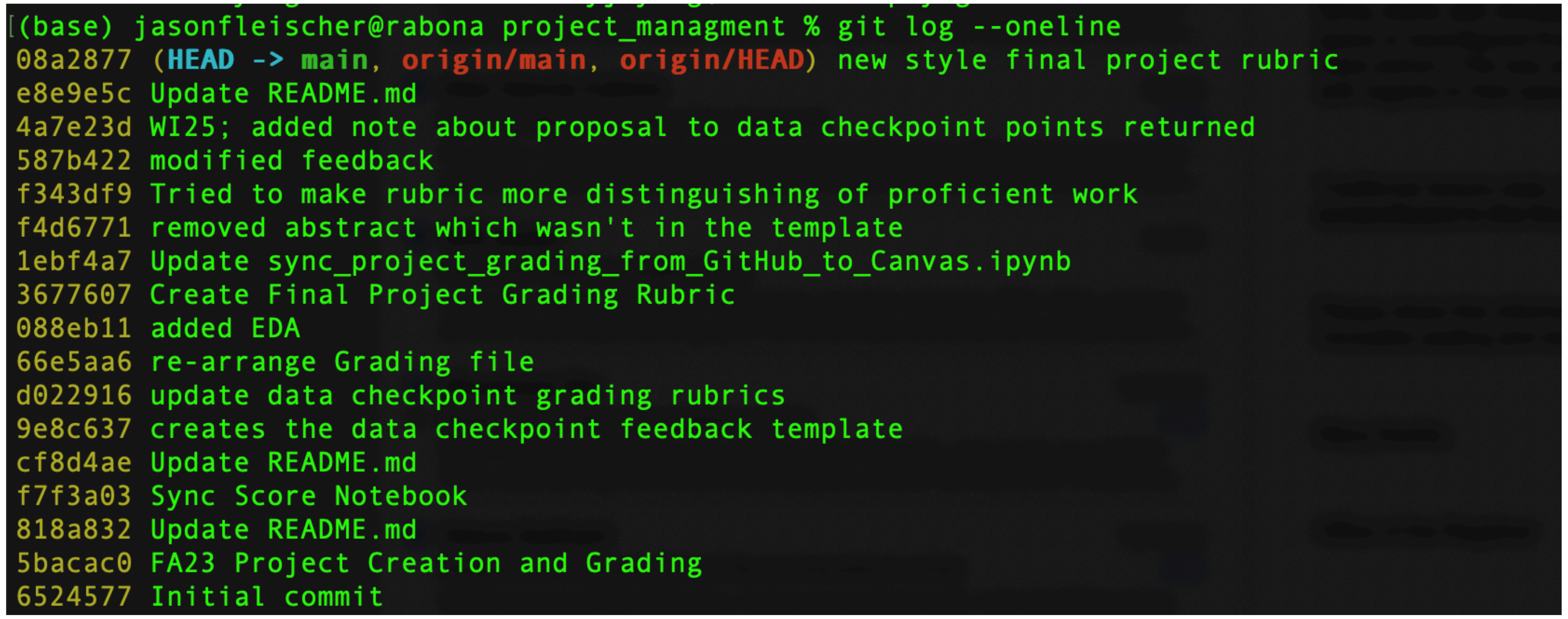


You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.



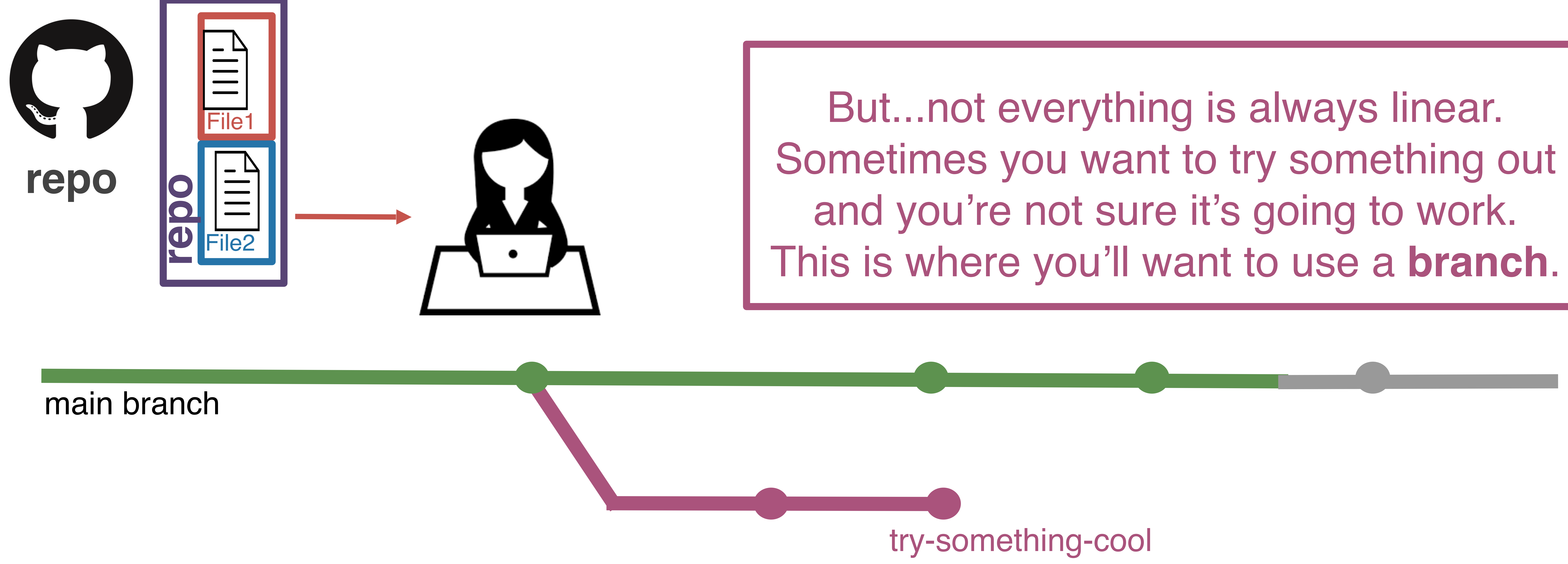
```
(base) jasonfleischer@rabona project_managment % git log
commit 08a2877e4ccd1f5440b7364feb7add64e4c926f9 (HEAD -> main, origin/main, origin/HEAD)
Author: Jason Fleischer <jason.g.fleischer@gmail.com>
Date:   Fri Oct 17 22:02:17 2025 -0700

    new style final project rubric

commit e8e9e5cd34ac494d06e97a7e494fd6c929bf3ff0
Author: Jason Fleischer <jason.g.fleischer@gmail.com>
Date:   Mon Apr 21 21:34:54 2025 -0700

    Update README.md

commit 4a7e23dab836f9b743af3ffc3ed0892561be216d
Author: Shannon Ellis <shannon0ellis@gmail.com>
Date:   Wed Jan 1 14:23:03 2025 -0800

    WI25; added note about proposal to data checkpoint points returned
```

Use git log
to find the hash…

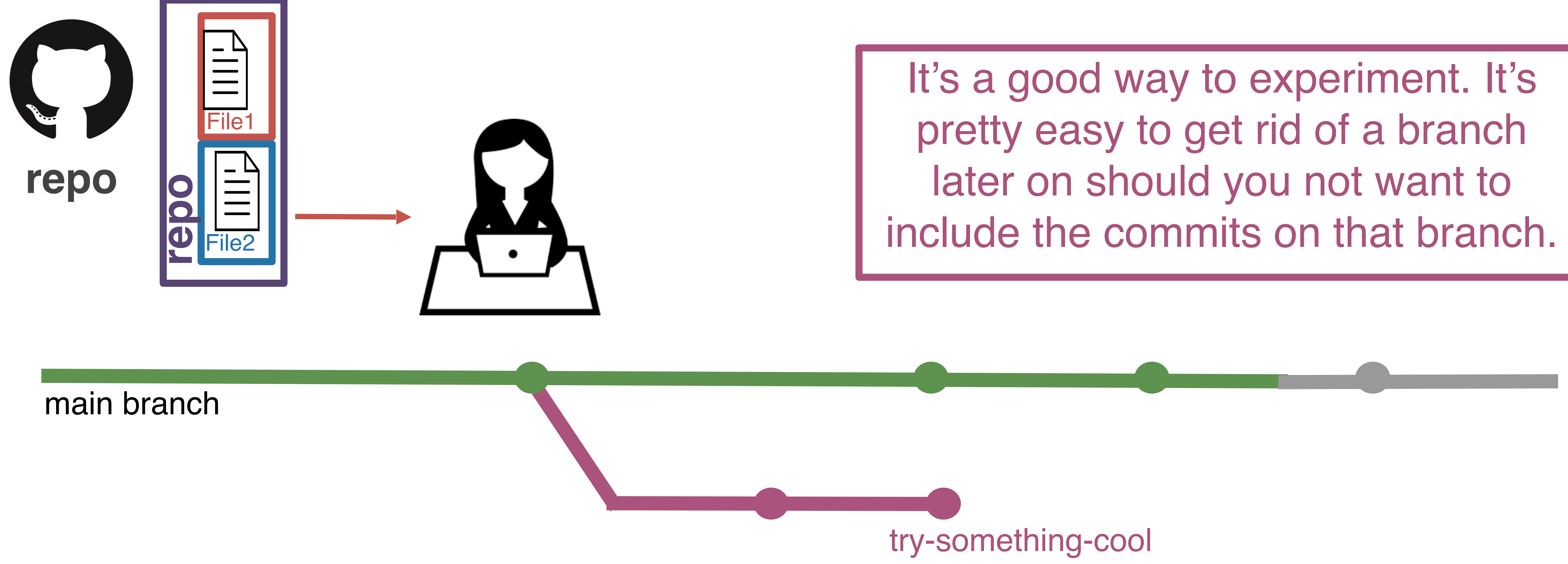


```
(base) jasonfleischer@rabona project_managment % git log --oneline
08a2877 (HEAD -> main, origin/main, origin/HEAD) new style final project rubric
e8e9e5c Update README.md
4a7e23d WI25; added note about proposal to data checkpoint points returned
587b422 modified feedback
f343df9 Tried to make rubric more distinguishing of proficient work
f4d6771 removed abstract which wasn't in the template
1ebf4a7 Update sync_project_grading_from_GitHub_to_Canvas.ipynb
3677607 Create Final Project Grading Rubric
088eb11 added EDA
66e5aa6 re-arrange Grading file
d022916 update data checkpoint grading rubrics
9e8c637 creates the data checkpoint feedback template
cf8d4ae Update README.md
f7f3a03 Sync Score Notebook
818a832 Update README.md
5bacac0 FA23 Project Creation and Grading
6524577 Initial commit
```

And I'll do a little
live demo of checkout
in our current lecture repo

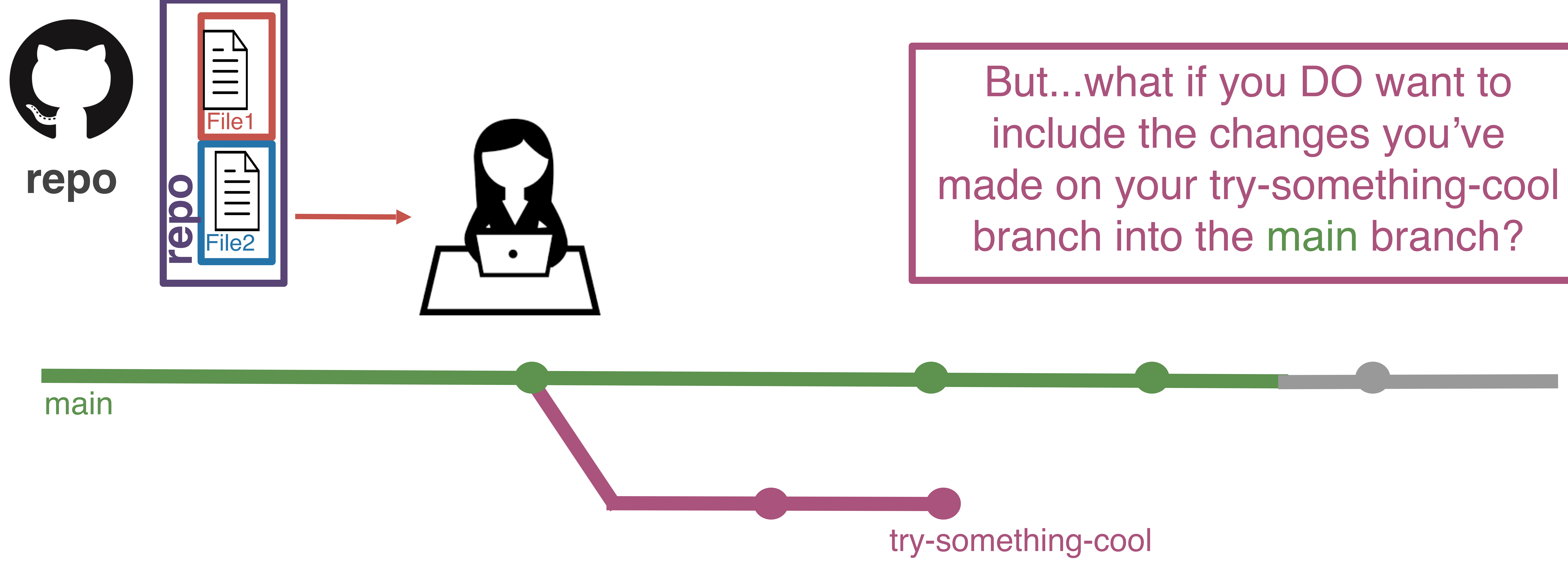


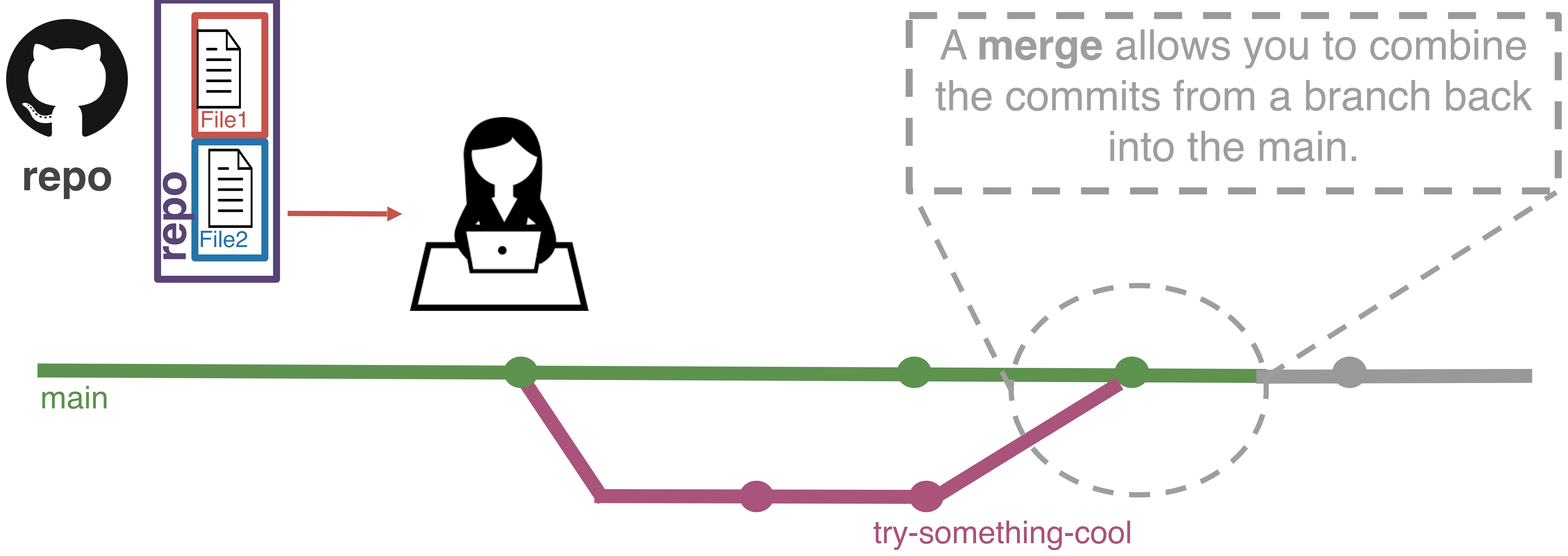
But...not everything is always linear.
Sometimes you want to try something out
and you’re not sure it’s going to work.
This is where you’ll want to use a **branch**.



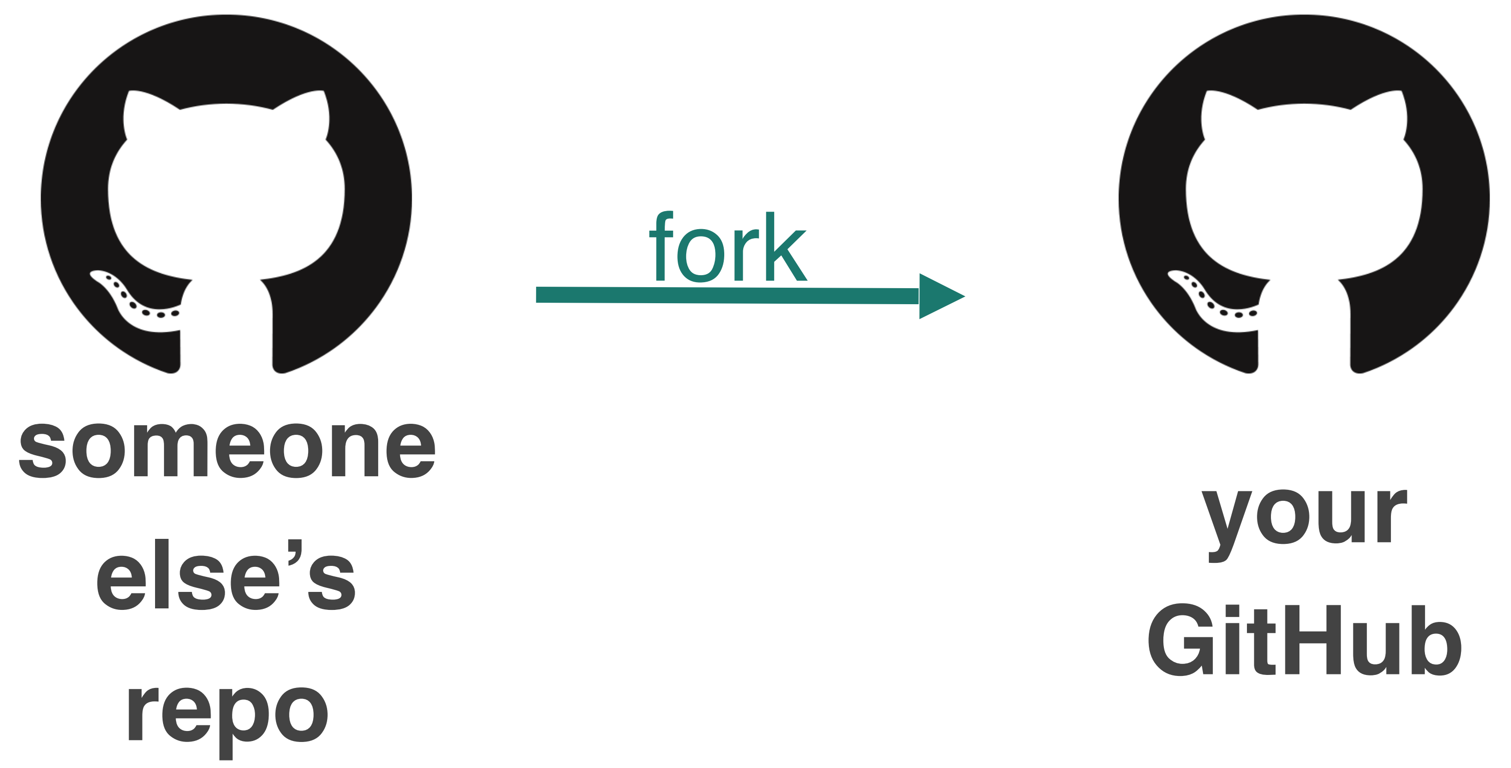
It's a good way to experiment. It's pretty easy to get rid of a branch later on should you not want to include the commits on that branch.

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

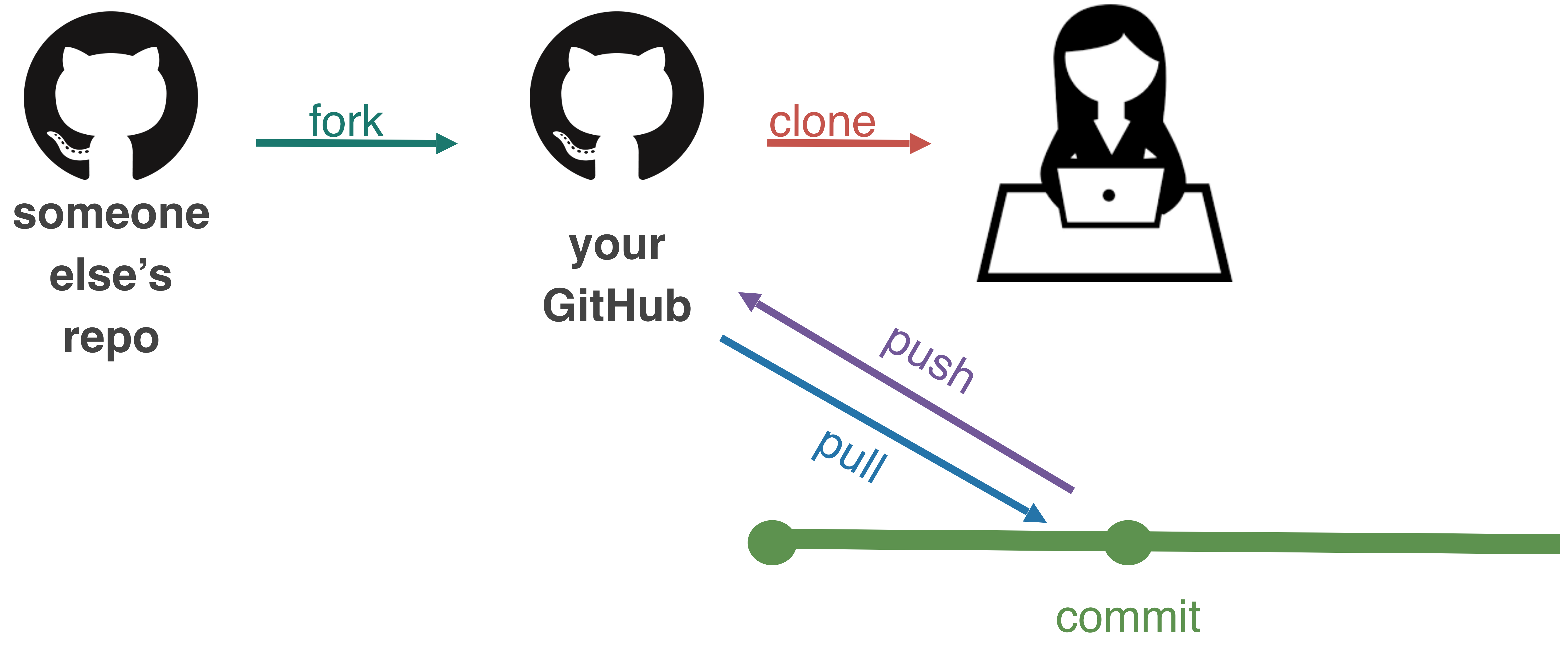




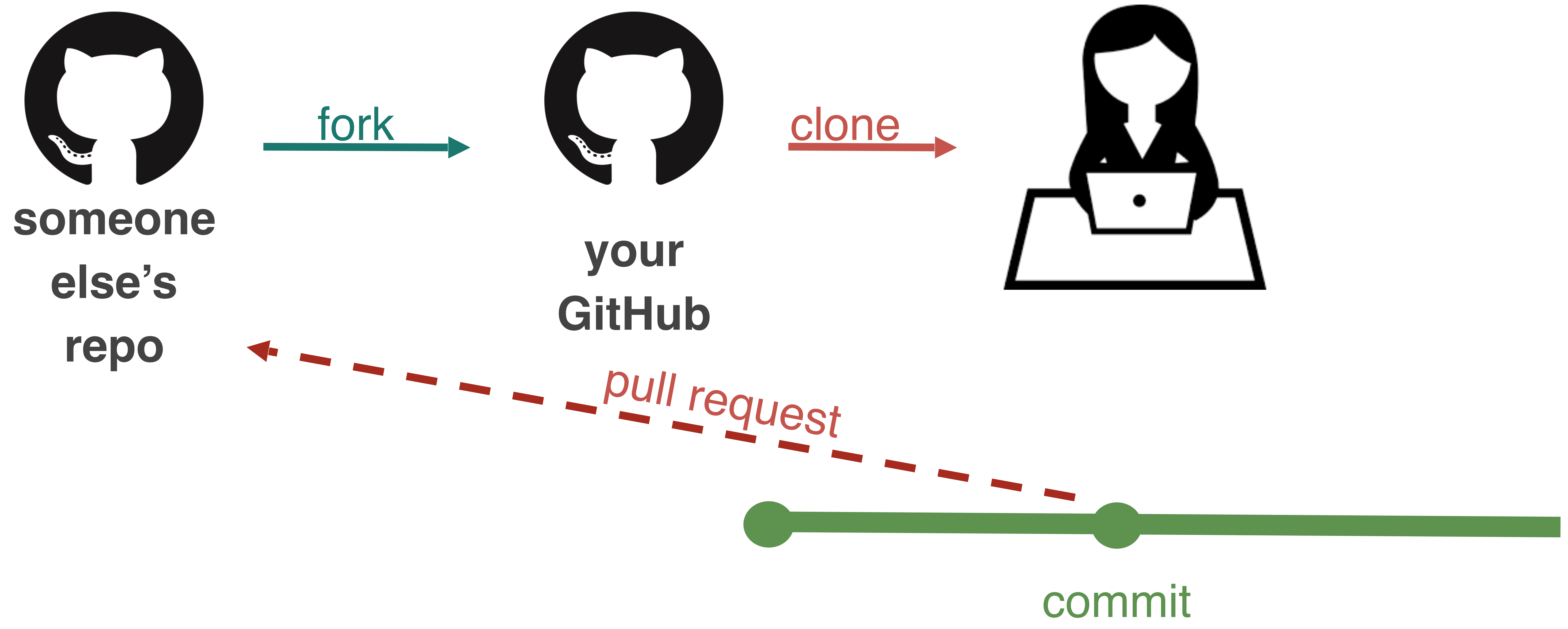
A **merge** allows you to combine the commits from a branch back into the main.



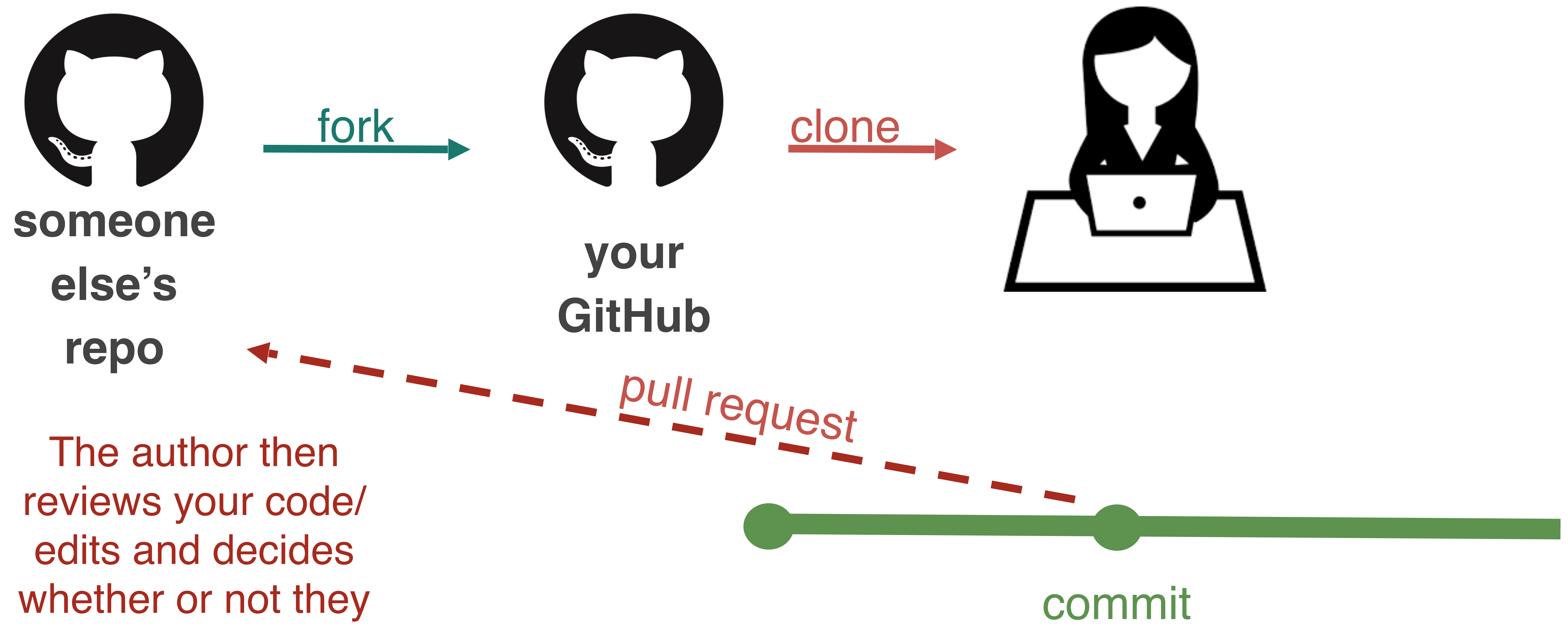
What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.



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.

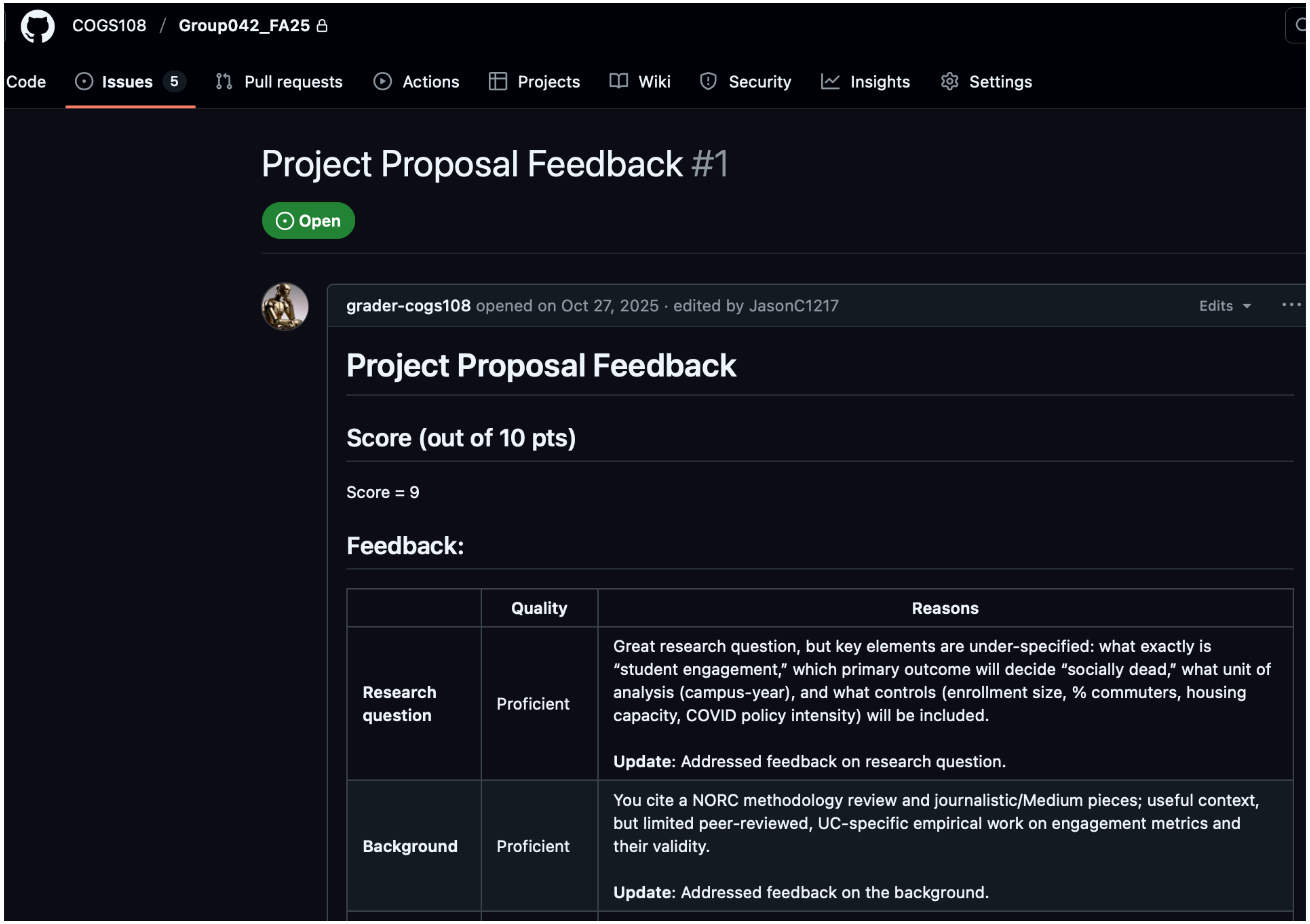
Issues are *bug trackers*. While, they can include bugs, they can also include feature requests, to-dos, whatever you want, really!

They can be assigned to people.

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

GitHub Issues

- Can also be used for team coordination like you saw in the video
- Will be used for your grade feedback!!



Project Proposal Feedback #1

grader-cogs108 opened on Oct 27, 2025 · edited by JasonC1217

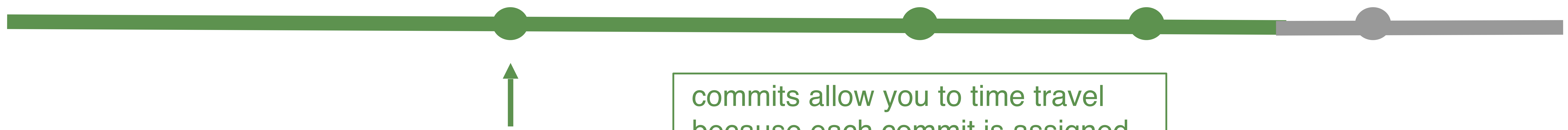
Project Proposal Feedback

Score (out of 10 pts)

Score = 9

Feedback:

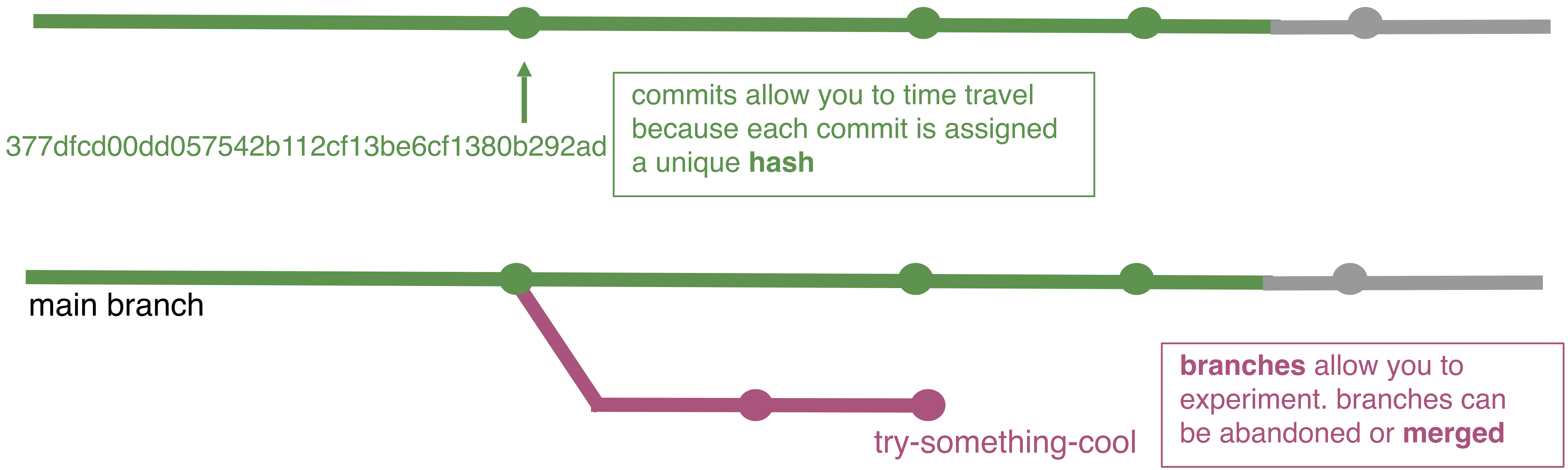
	Quality	Reasons
Research question	Proficient	Great research question, but key elements are under-specified: what exactly is “student engagement,” which primary outcome will decide “socially dead,” what unit of analysis (campus-year), and what controls (enrollment size, % commuters, housing capacity, COVID policy intensity) will be included. **Update:** Addressed feedback on research question.
Background	Proficient	You cite a NORC methodology review and journalistic/Medium pieces; useful context, but limited peer-reviewed, UC-specific empirical work on engagement metrics and their validity. **Update:** Addressed feedback on the background.



377dfcd00dd057542b112cf13be6cf1380b292ad

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

One more git recap...

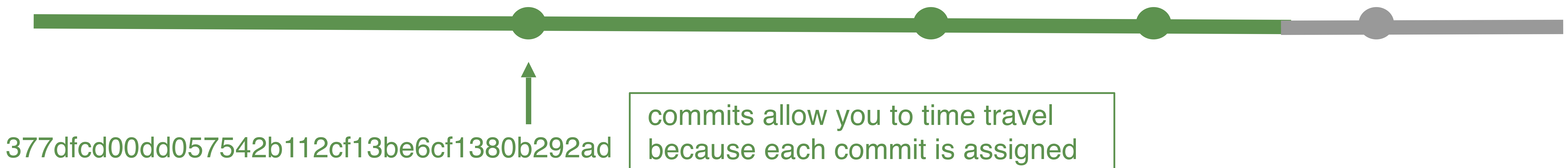


commits allow you to time travel because each commit is assigned a unique **hash**

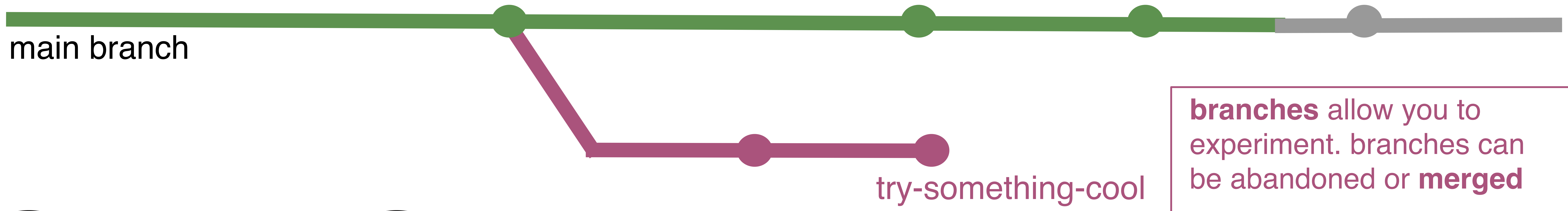
branches allow you to experiment. branches can be abandoned or **merged**

One more git recap...

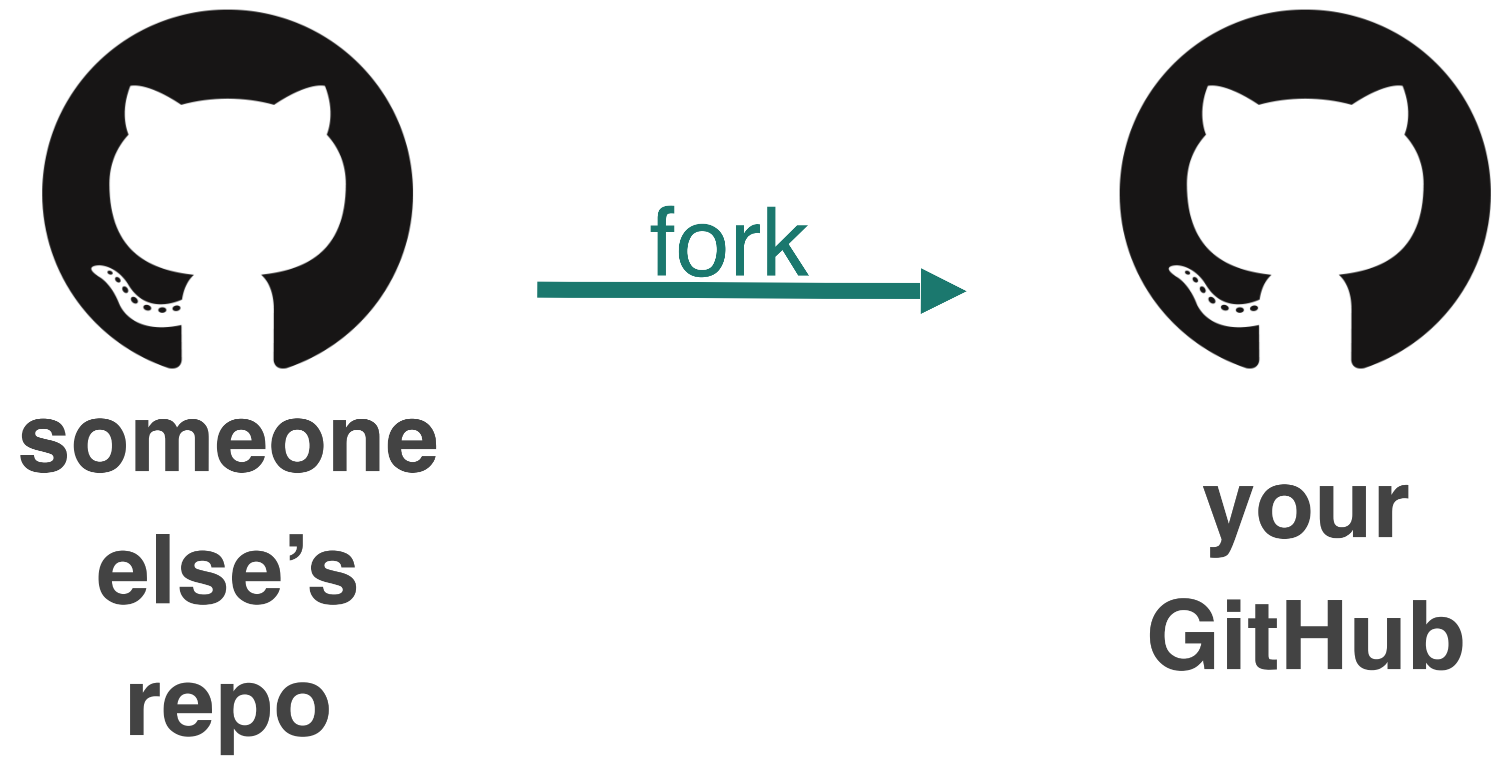
One more git recap...



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

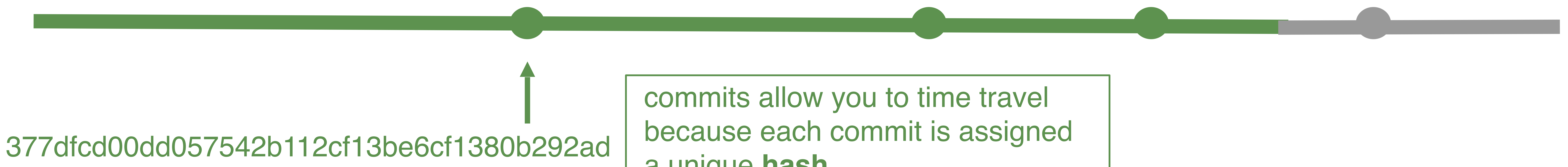


branches allow you to experiment. branches can be abandoned or **merged**

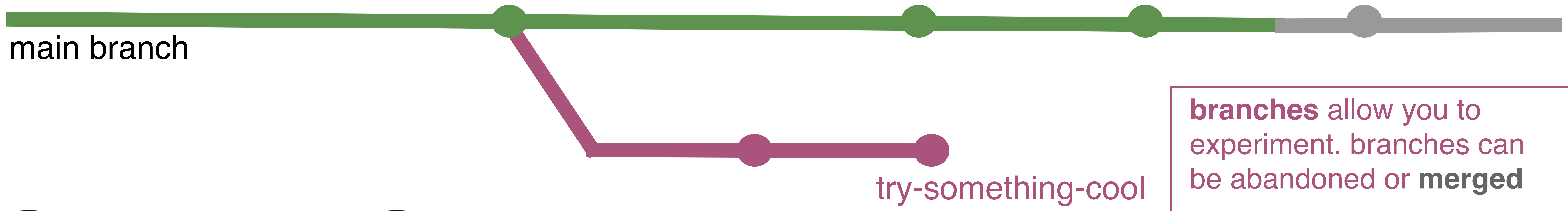


You can work on others' repos by first **forking** their repository onto your GitHub

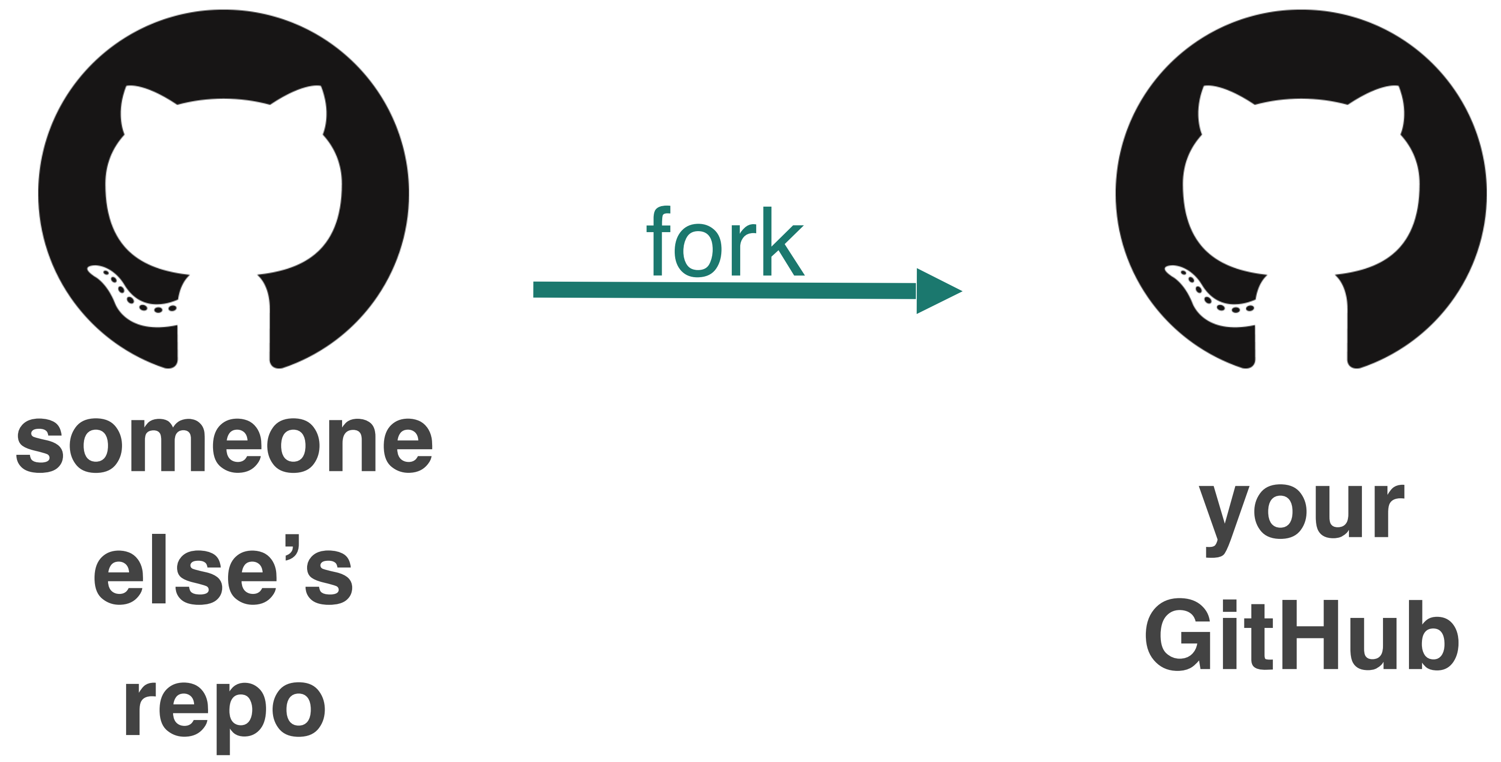
One more git recap...



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



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



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

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

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

Working with others in git

https://forms.gle/eyxgHB3wvqmy17uR9

😎 Good to Git

Use it for…

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

⚠️ Careful

Easy to mess up

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

🙅 Don't Git

Avoid using for

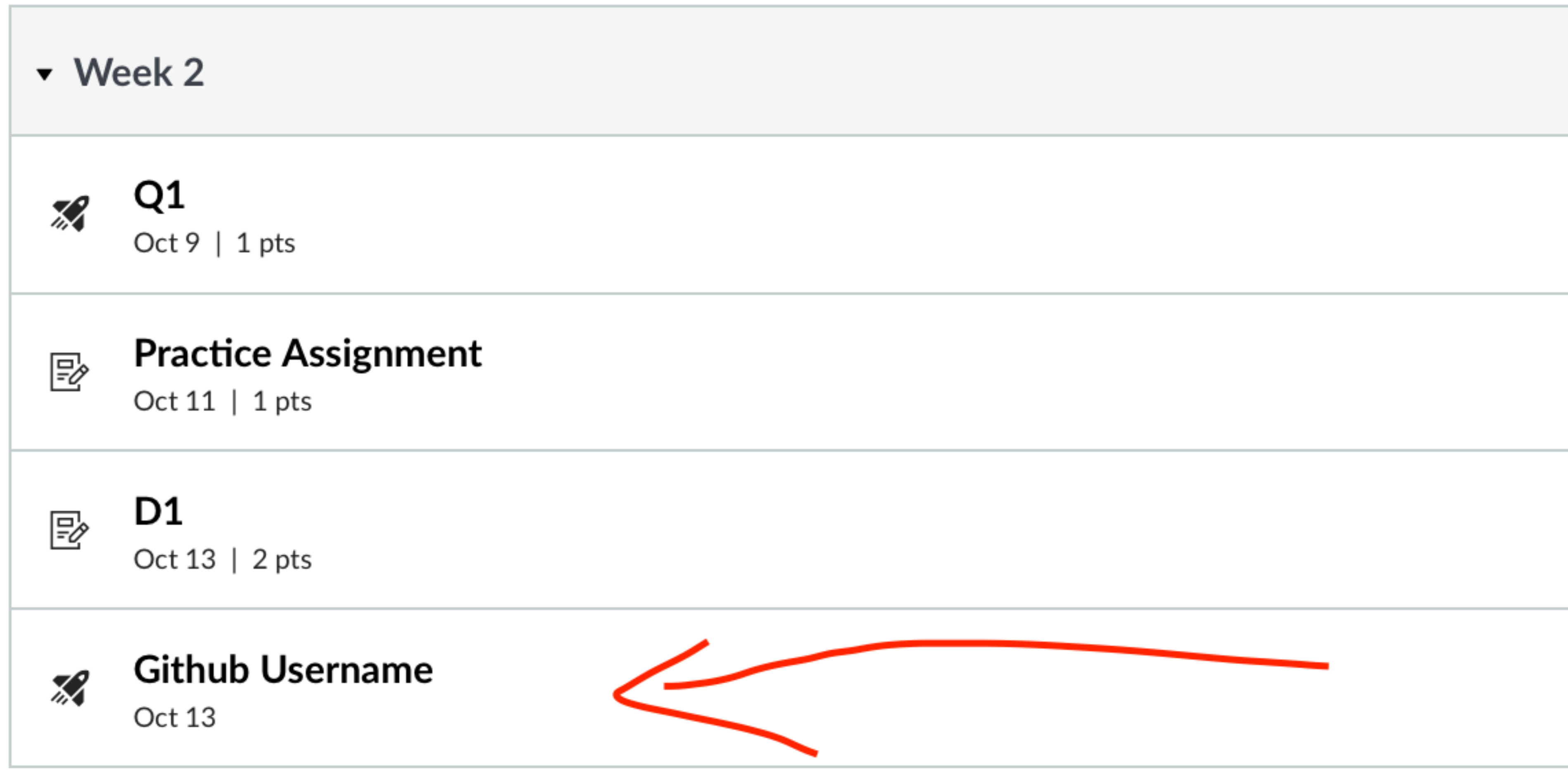
- Large text files
  ❌ 648MB .

✅ Git LFS can help!

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

Version Control in this course

- GO TO GITHUB AND GET A USERNAME! We need it for many things in the class
- You will get practice using 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

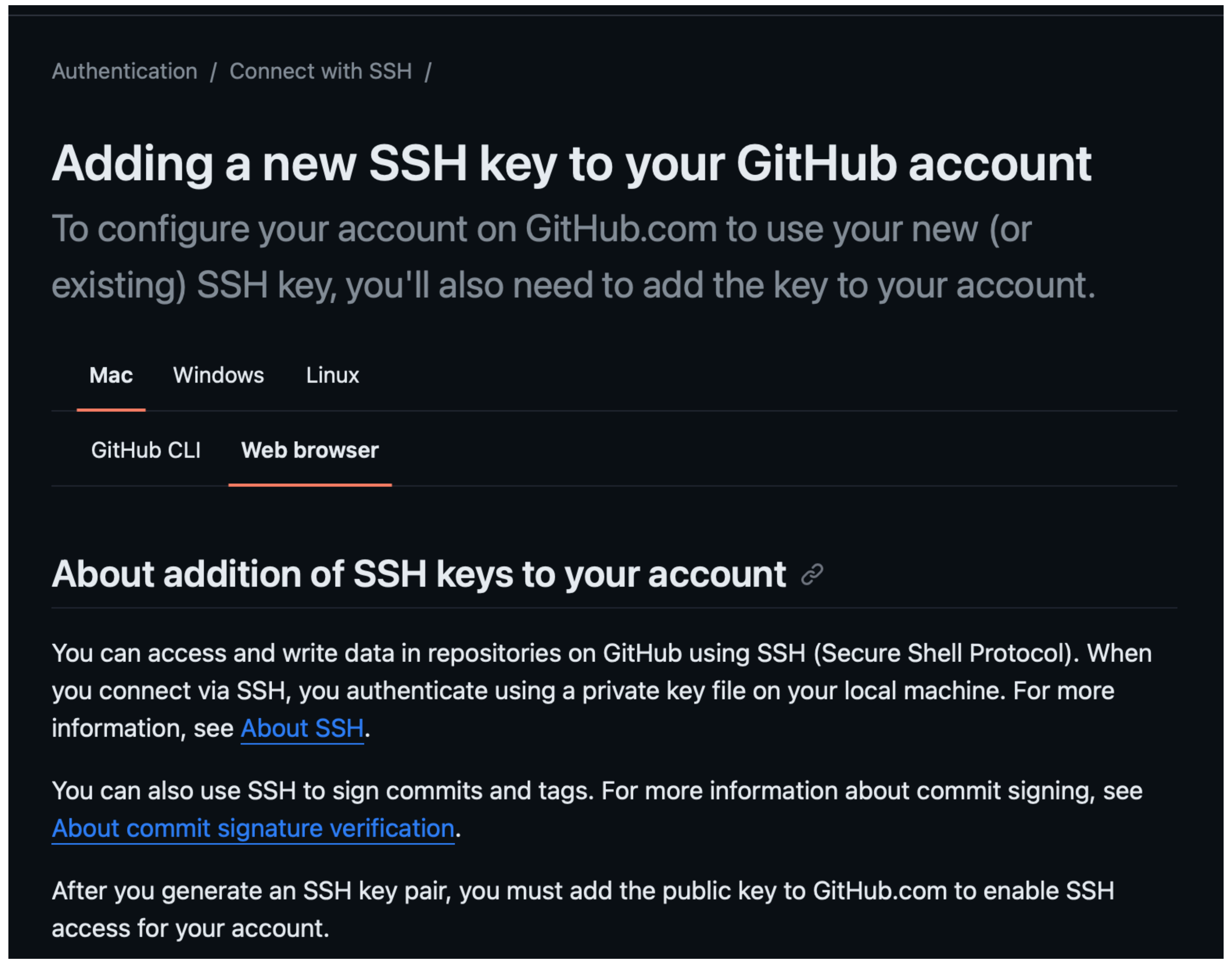
Github Username
Oct 13

Using SSH to contribute work to your project repo

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

Click here, follow instructions:

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


Authentication / Connect with SSH /

Adding a new SSH key to your GitHub account

To configure your account on GitHub.com to use your new (or existing) SSH key, you'll also need to add the key to your account.

About addition of SSH keys to your account

You can access and write data in repositories on GitHub using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine. For more information, see About SSH.

You can also use SSH to sign commits and tags. For more information about commit signing, see About commit signature verification.

After you generate an SSH key pair, you must add the public key to GitHub.com to enable SSH access for your account.

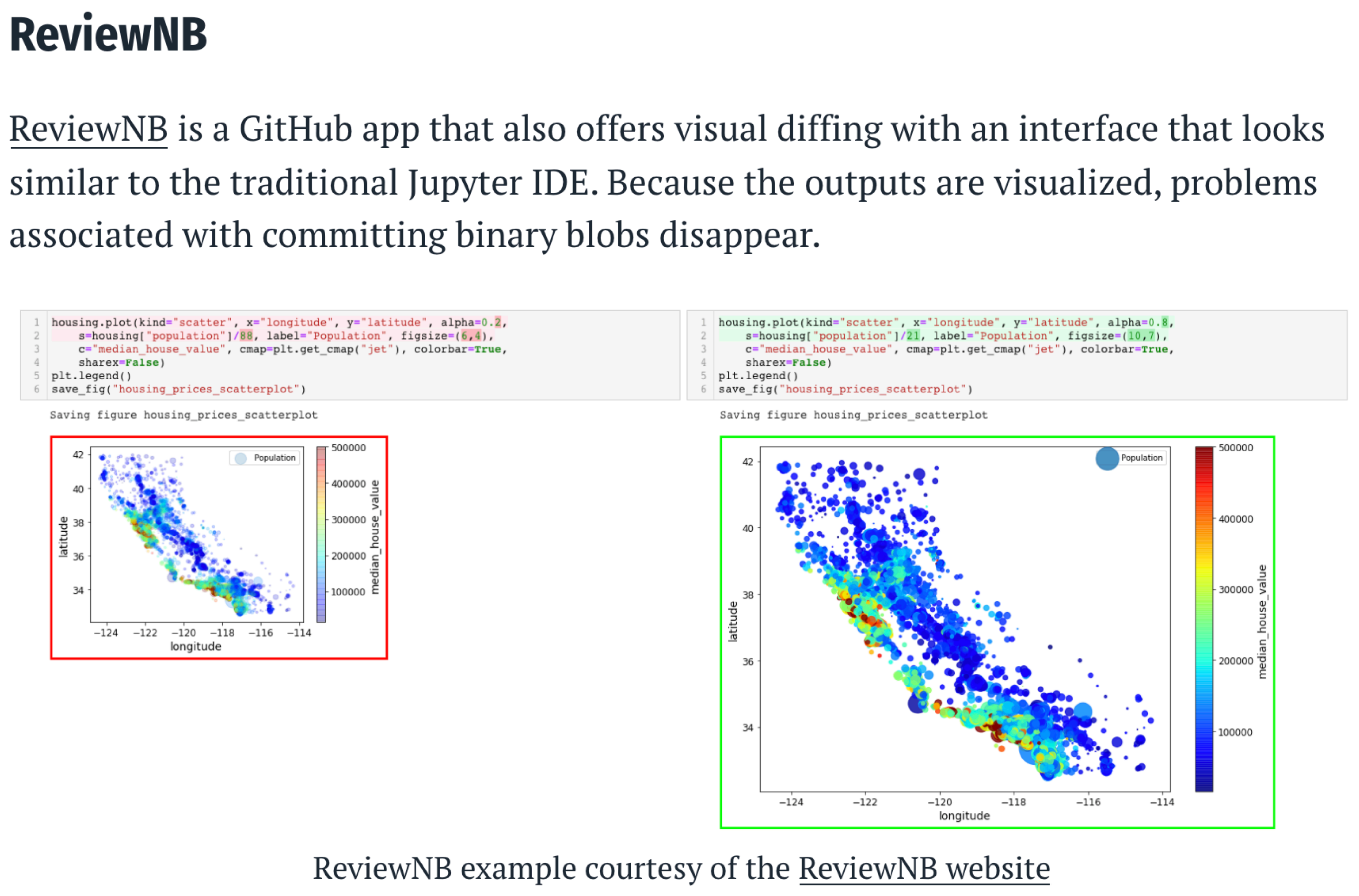
You will need to do this for each local computer you will use
(e.g., on Datahub and again on your laptop!)

Skills you can use

- Checking for difference from the last commit
- Branching patterns for keeping track of different projects
- Rebasing?
-

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.

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.2,
    s=housing["population"]/88, label="Population", figsize=(6,4),
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
    sharex=False)
plt.legend()
save_fig("housing_prices_scatterplot")
```

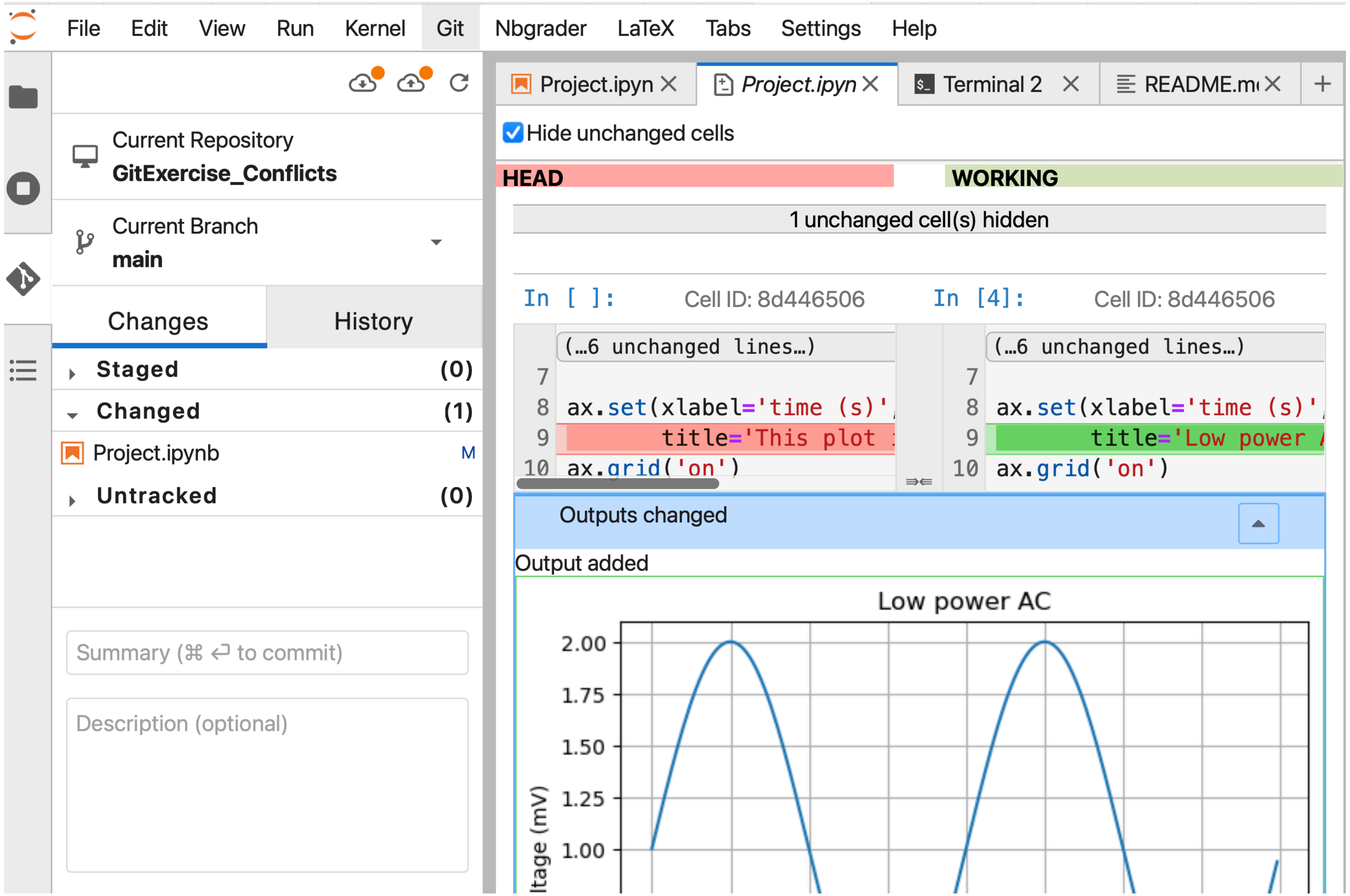
```
Saving figure housing_prices_scatterplot
```

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.8,
    s=housing["population"]/21, label="Population", figsize=(10,7),
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
    sharex=False)
plt.legend()
save_fig("housing_prices_scatterplot")
```

```
Saving figure housing_prices_scatterplot
```

ReviewNB example courtesy of the ReviewNB website

What version control looks like



File Edit View Run Kernel Git Nbgrader LaTeX Tabs Settings Help

Current Repository
GitExercise_Conflicts

Current Branch
main

Changes | History

Staged (0)
Changed (1)
Project.ipynb M
Untracked (0)

Summary (⌘ ⏎ to commit)
Description (optional)

Project.ipyn × | *Project.ipyn* × | Terminal 2 × | README.m ×

☑ Hide unchanged cells

HEAD | **WORKING**

1 unchanged cell(s) hidden

In [ ]: Cell ID: 8d446506 | In [4]: Cell ID: 8d446506

```
(…6 unchanged lines…)
7
8  ax.set(xlabel='time (s)',
9         title='This plot
10 ax.grid('on')
```

```
(…6 unchanged lines…)
7
8  ax.set(xlabel='time (s)',
9         title='Low power A
10 ax.grid('on')
```

Outputs changed

Output added

Datahub @ UCSD

Lets practice!

At least if we have enough time right now :)

- Create repo
- Create file.md
- Create a branch “my_work”
- On “my_work”: Add a new file, prog.py
- On “my_work”: Change contents of file.md by adding newlines
- Switch to “master” and merge “my_work” into “master” (problem free)
- On “master”: Change contents of file.md header
- On “my_work”: Change contents of file.md header in a way that conflicts with master
- Switch to “master” and merge “my_work”
- Resolve conflicts, finish merge

More practice

For outside of lecture

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!

Things you may wish to install on your laptop

Git (if not already installed!)
https://git-scm.com/downloads

Python and Jupyter via Anaconda
https://www.anaconda.com/download

NBDime is how to get ReviewNB-style diffs
in your laptop's Jupyter notebook
https://nbdime.readthedocs.io/en/latest/installing.html

Other things to consider according to your preferences:

- GitHub Desktop, Sourcetree, or some other GUI?
- VSCode and GitHub extensions?