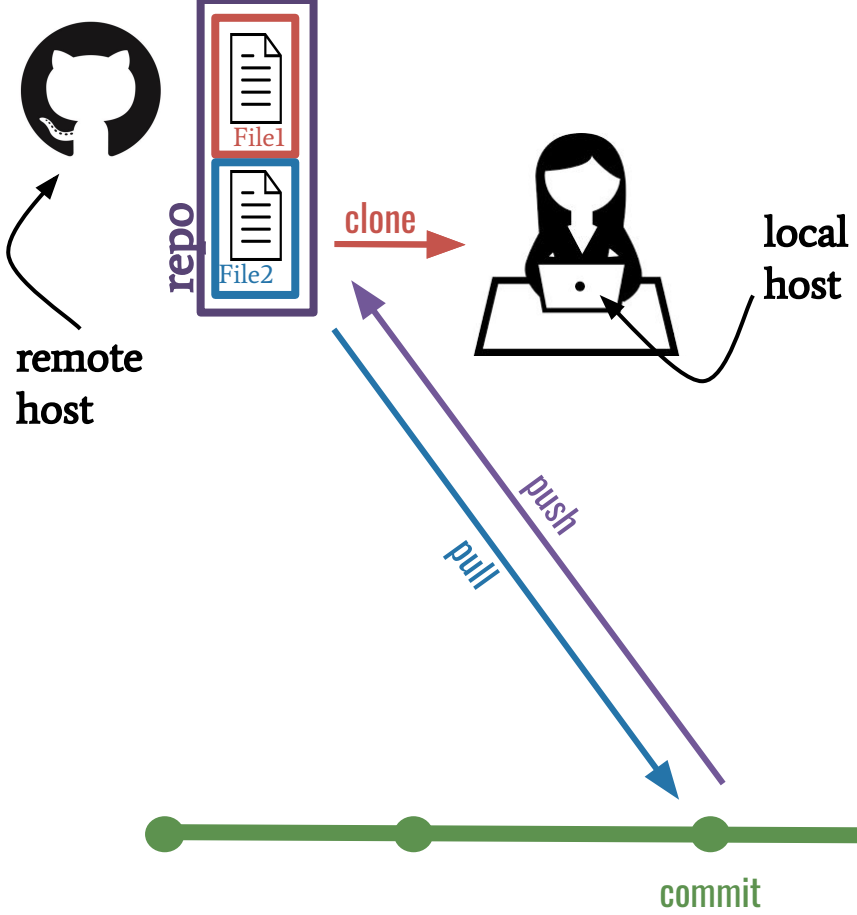
Plan:

1. Gain familiarity with basic version control processes
2. Introduce (more of) the basic version control verbs

Version Control: Verbs II

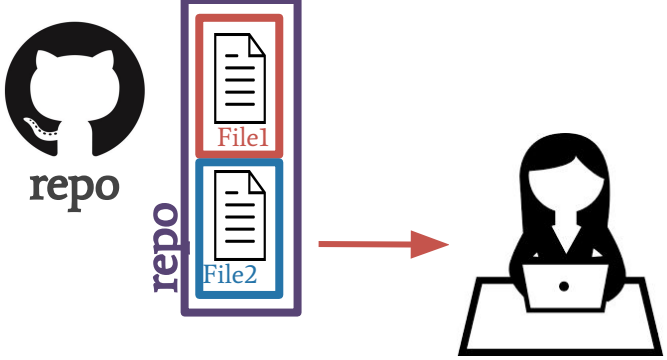
Shannon E. Ellis, Ph.D
UC San Diego

Department of Cognitive Science
sellis@ucsd.edu

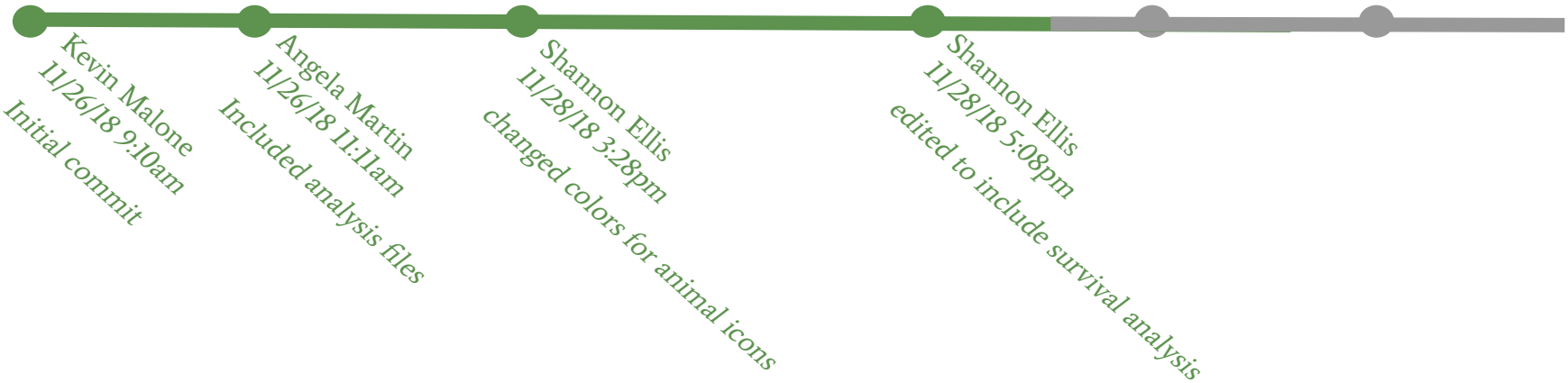


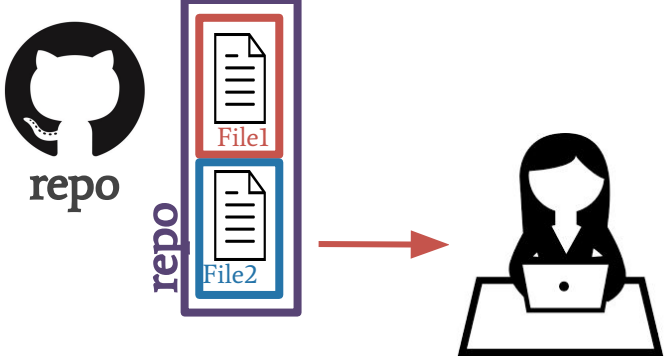
Let's recap real quick!

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

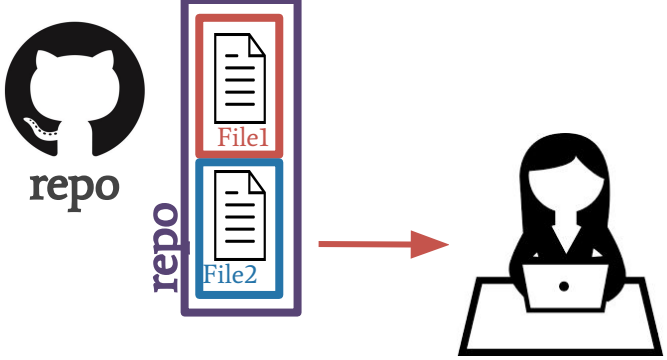


Each time you create a commit, git tracks the changes made automatically.





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



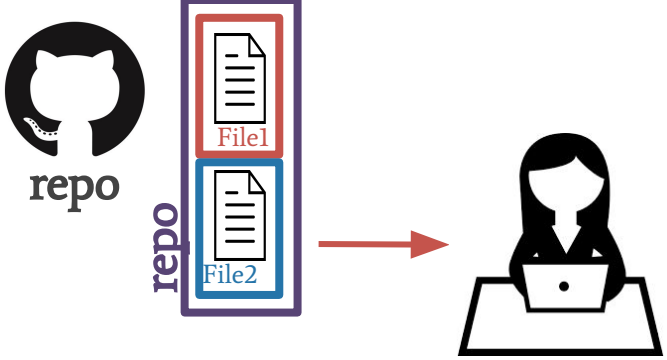
By committing each time you make changes, git allows you to time travel!

377dfcd00dd057542b112cf13be6cf1380b292ad

439301fe69e8f875c049ad0718386516b4878e22

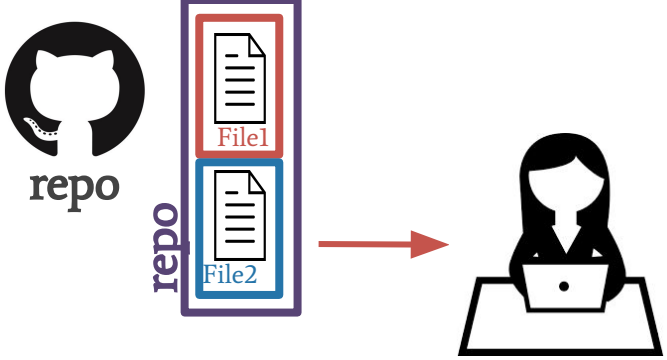
456722223e9f9e0ee0a92917ba80163028d89251

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

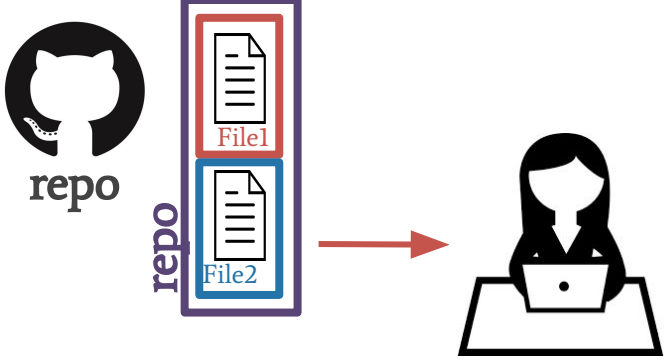


You can return to the state of the repository at any commit. Future commits don't disappear. They just aren't visible when you **check out** an older commit.

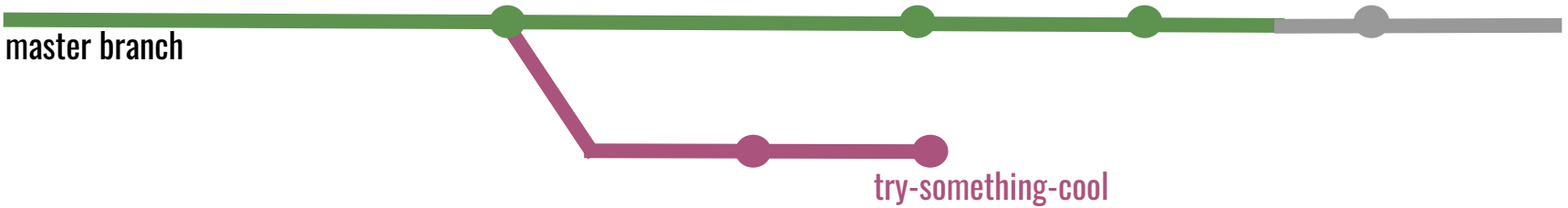
377dfcd00dd057542b112cf13be6cf1380b292ad

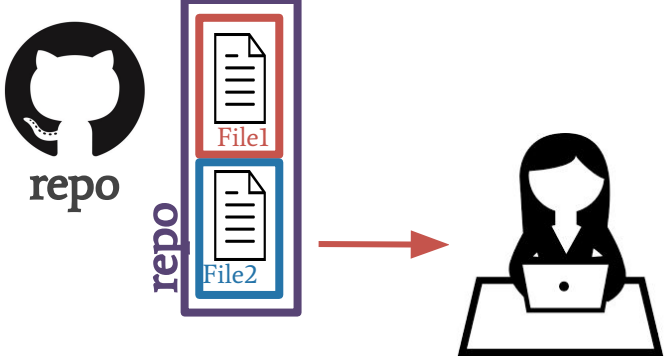


But...not everything is always linear. Sometimes you want to try something out and you're not sure it's going to work. This is where you'll want to use a **branch**.

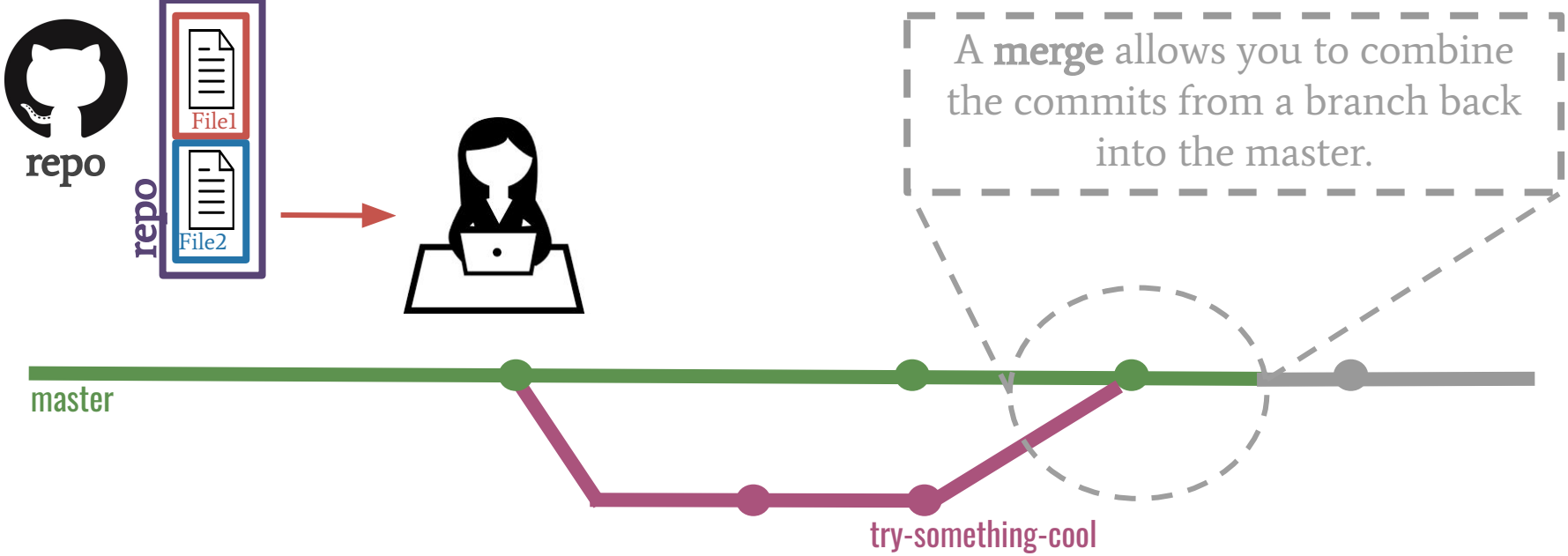


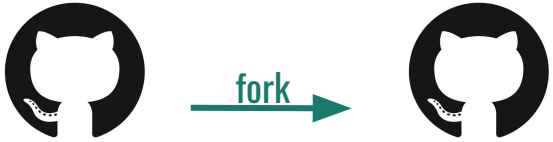
It's a good way to experiment. It's pretty easy to get rid of a branch later on should you not want to include the commits on that branch.



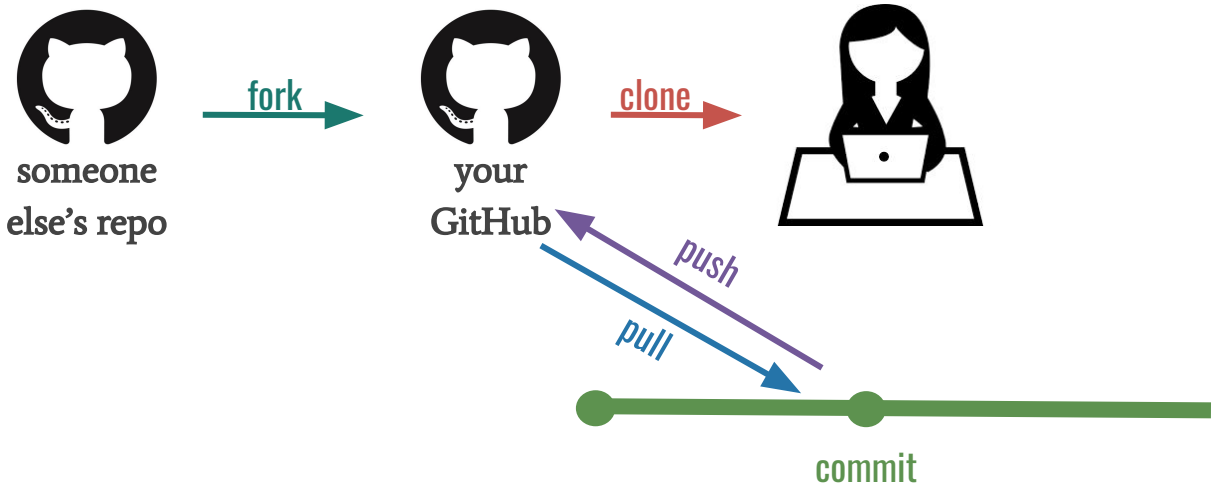


But...what if you DO want to include the changes you've made on your **try-something-cool** branch into the **master** branch?

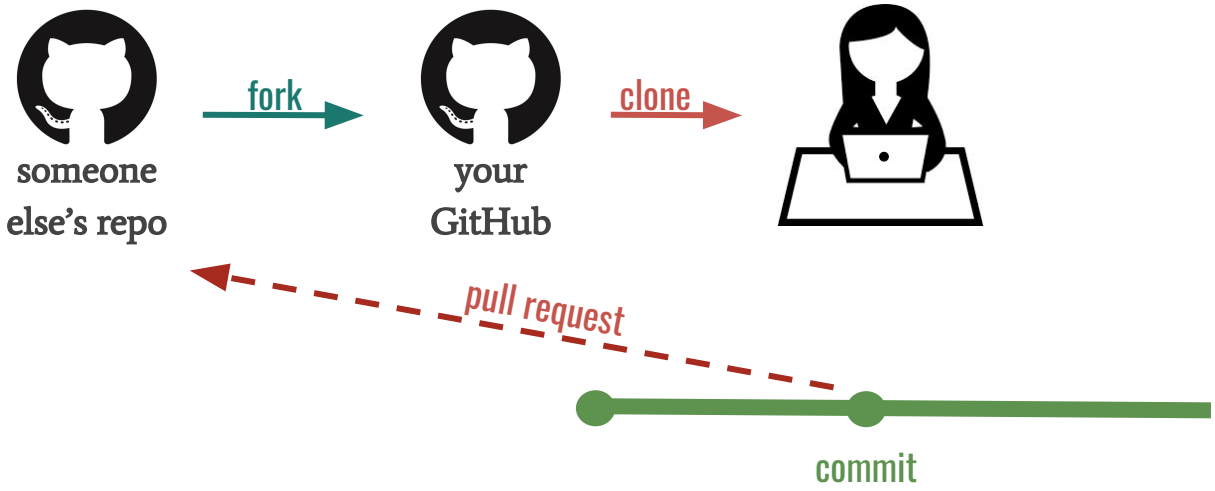

repo
repo
File1
File2
A merge allows you to combine the commits from a branch back into the master.
master
try-something-cool



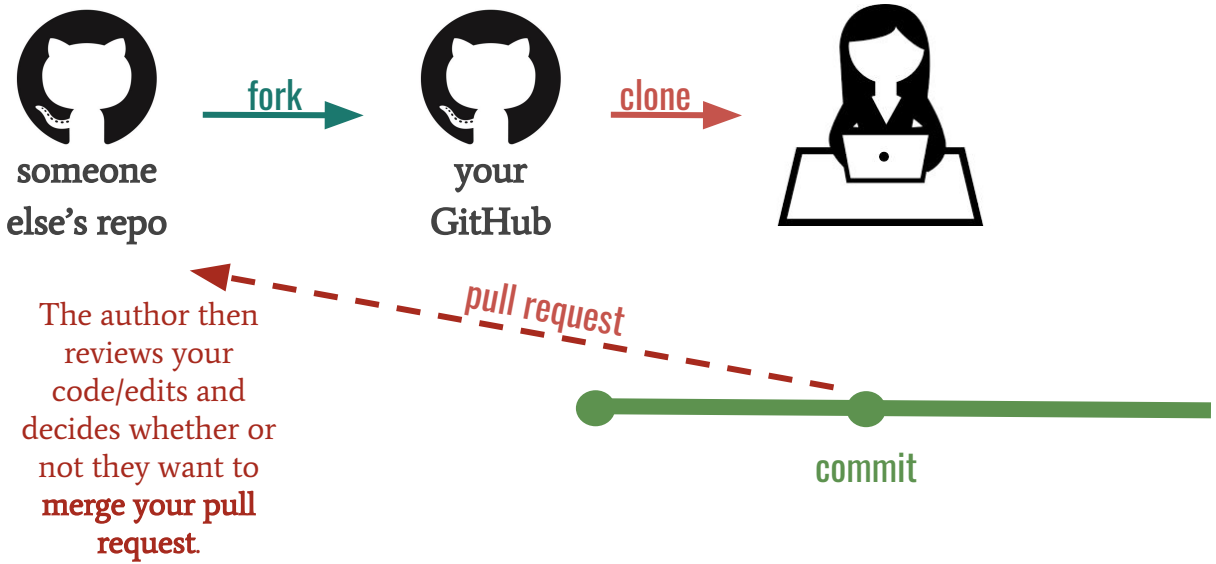
What if someone else is working on something cool and you want to play around with it? You'll have to **fork** their repo.



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.

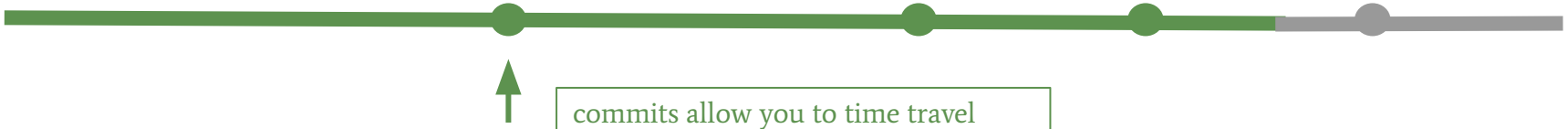
someone else's repo

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.

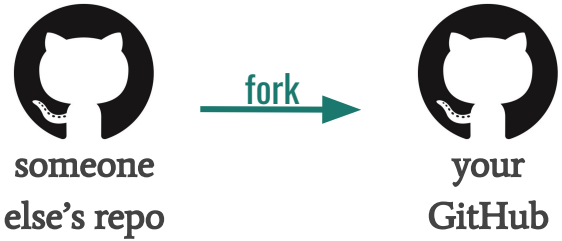
They can be closed once addressed ....or if the software maintainer doesn't like the suggestion



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...