Introduction to Git and GitHub
Part 2

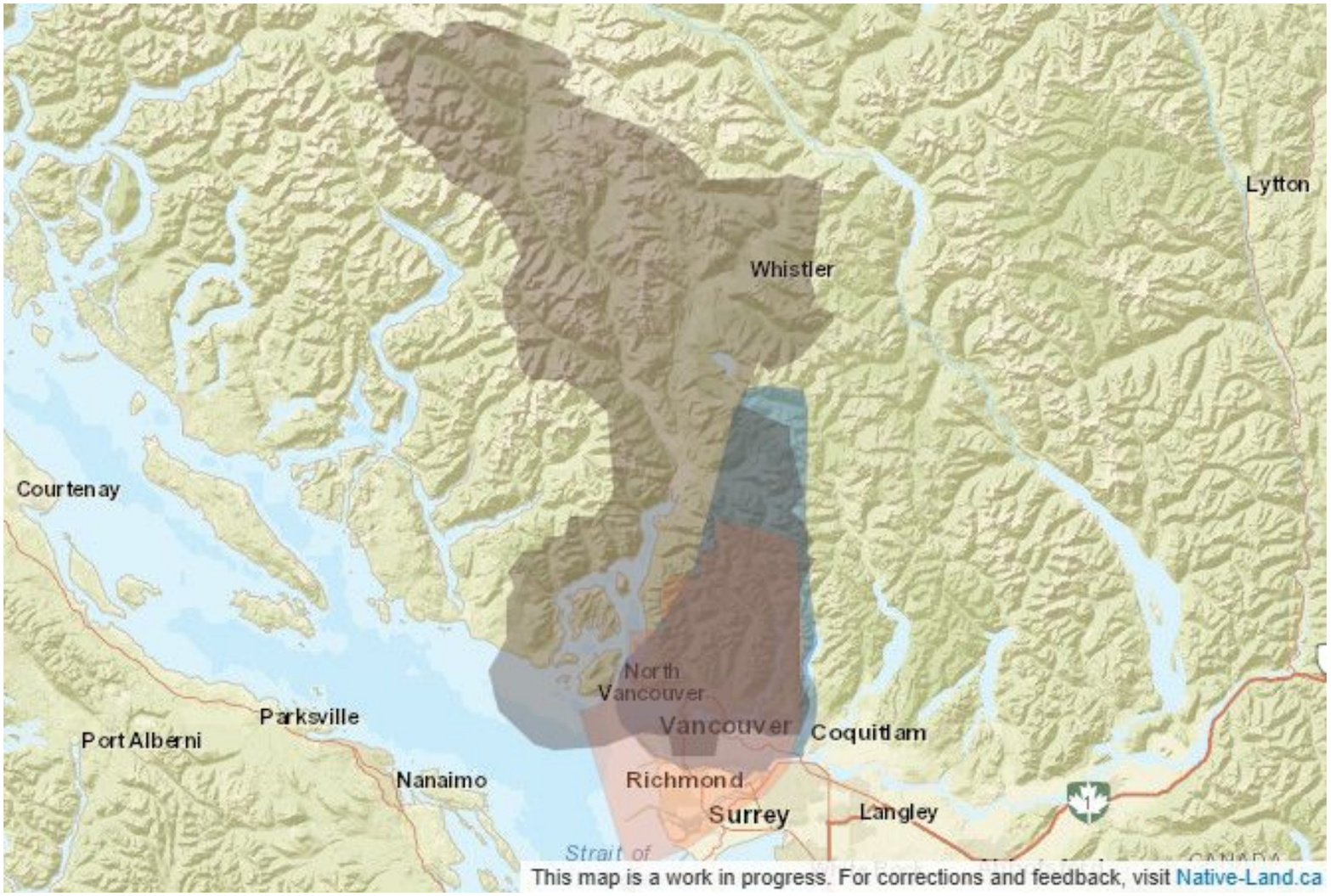
Kendra Oudyk (she/her)

Many parts of this presentation are inspired / based on these great resources

- Chacon, S., & Straub, B. (2014). *Pro git.* Springer Nature. Available at https://git-scm.com/book/en/v2
- The Carpentries. (2021). *Version Control with Git.* https://swcarpentry.github.io/git-novice/.

Land Acknowledgement

The UBC Vancouver campus is located on the traditional, ancestral, and unceded territory of the xʷməθkʷəy̓əm (Musqueam), səlilwətaɬ (Tsleil-Waututh), and Skwxwú7mesh-ulh Temíx̱w (Squamish) peoples.



Goals

Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

Part 2

- Share your work on GitHub
- Collaborate on GitHub
- GitHub features
- Try it out!

These slides complement the **self-paced** materials on the Research Commons website

https://ubc-library-rc.github.io/intro-git/

Who am I?

- Multiple fields of study / research
  - BMus → MA → PhD (Neuroscience) → MLIS (Library Studies)

- I love teaching
  - Certified **Carpentries** instructor
  - Taught Git/GitHub **10x**
  - **New to teaching at UBC**

- Also:
  - Disabled
  - Colour-grapheme synesthesia
  - I have a cat

Who are you?

- Subject areas
  - 1-5 words to explain what you study / research to a teenager

- Programming experience
  - Never / a few times
  - Monthly
  - Weekly
  - Daily

- What kinds of files do you edit in a text editor?
  - E.g., .txt, .md, .py, .r, .tex

Goals

Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

Part 2

- Share your work on GitHub
- Collaborate on GitHub
- GitHub features
- Try it out!

Goals

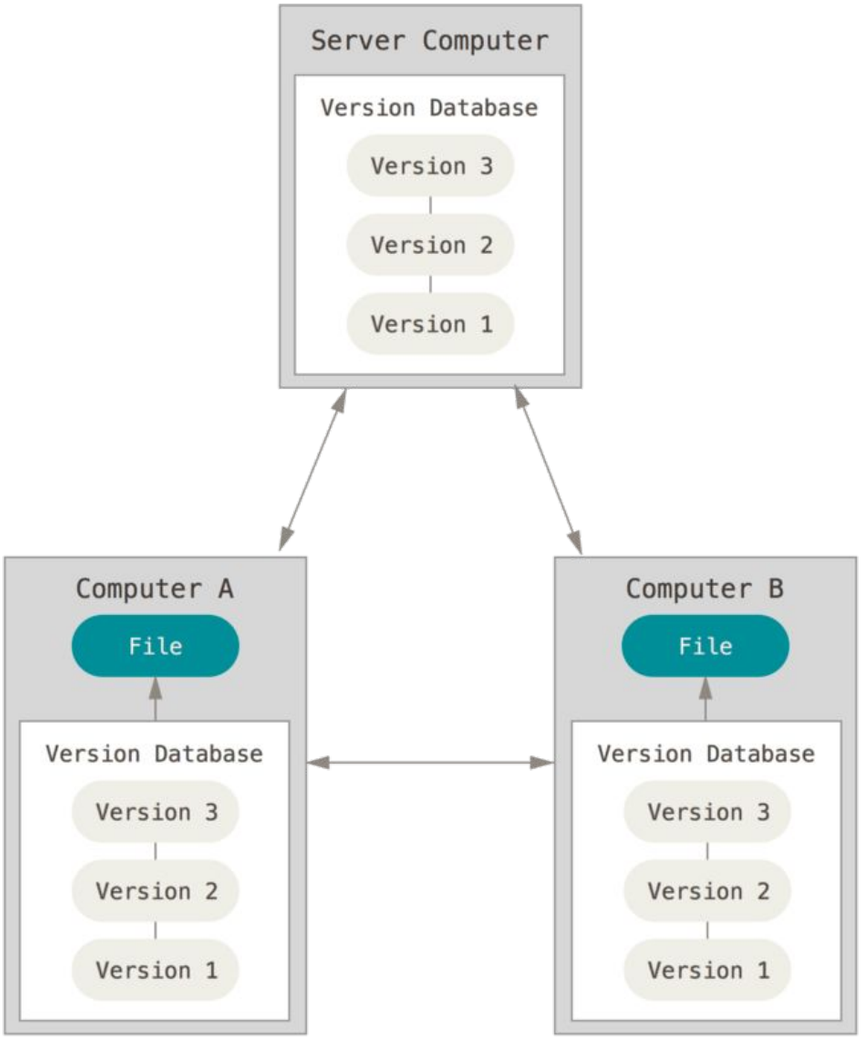
Part 1 (last week)

- **What is distributed version control?**
- Why is Git useful?
- **Track your own work with Git**

Part 2

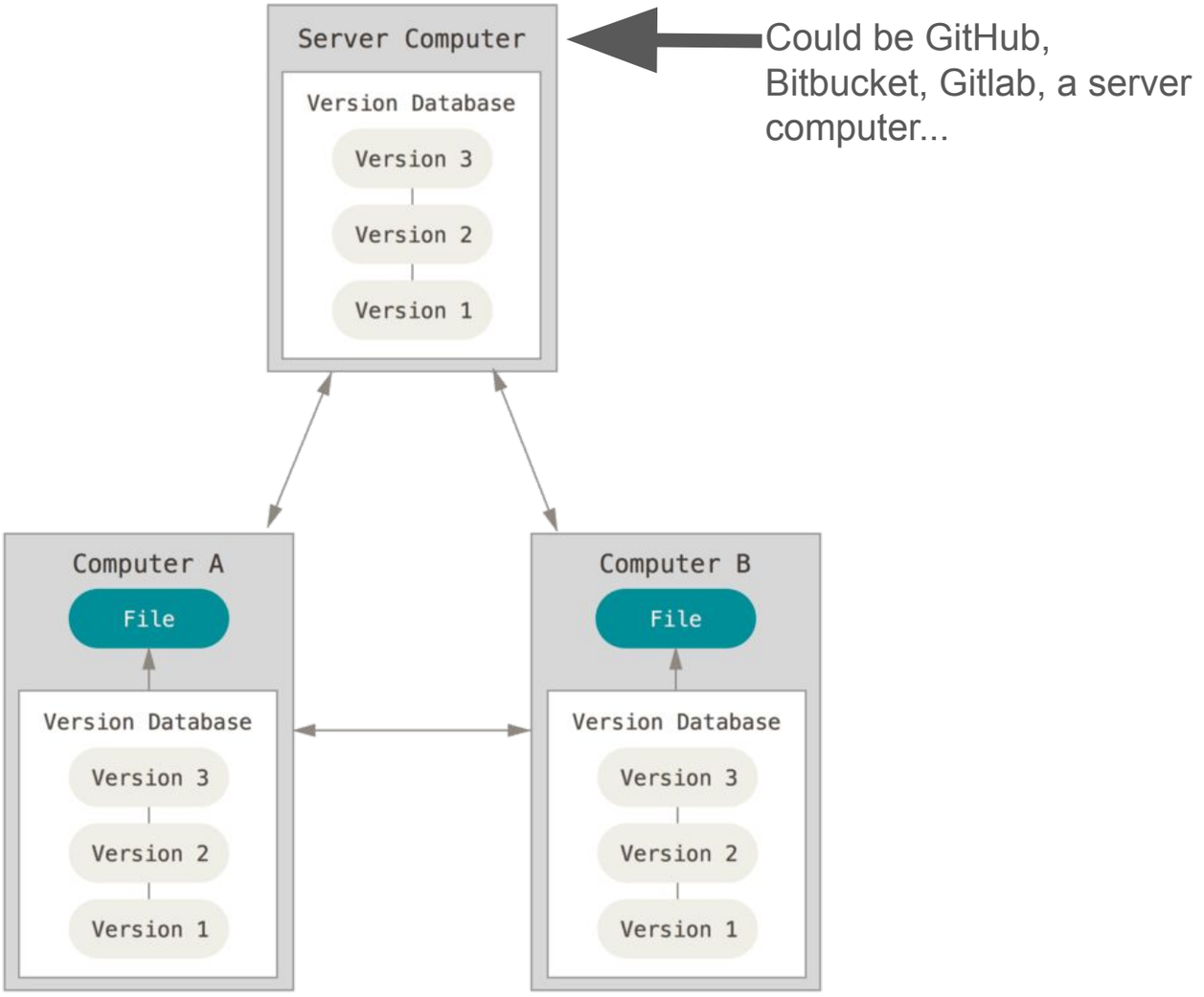
- Share your work on GitHub
- Collaborate on GitHub
- GitHub features
- Try it out!

Distributed version control

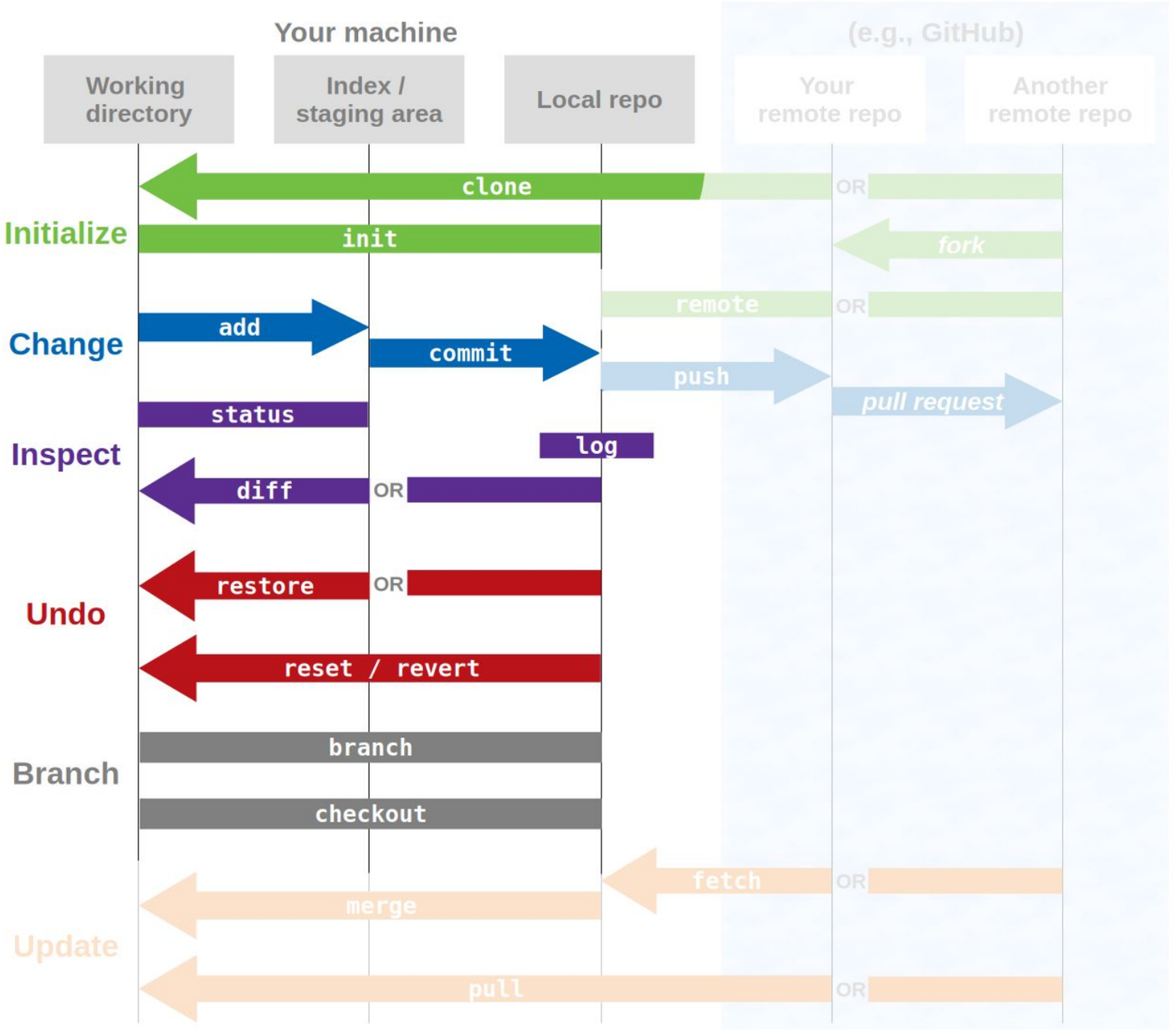


Git vs. GitHub

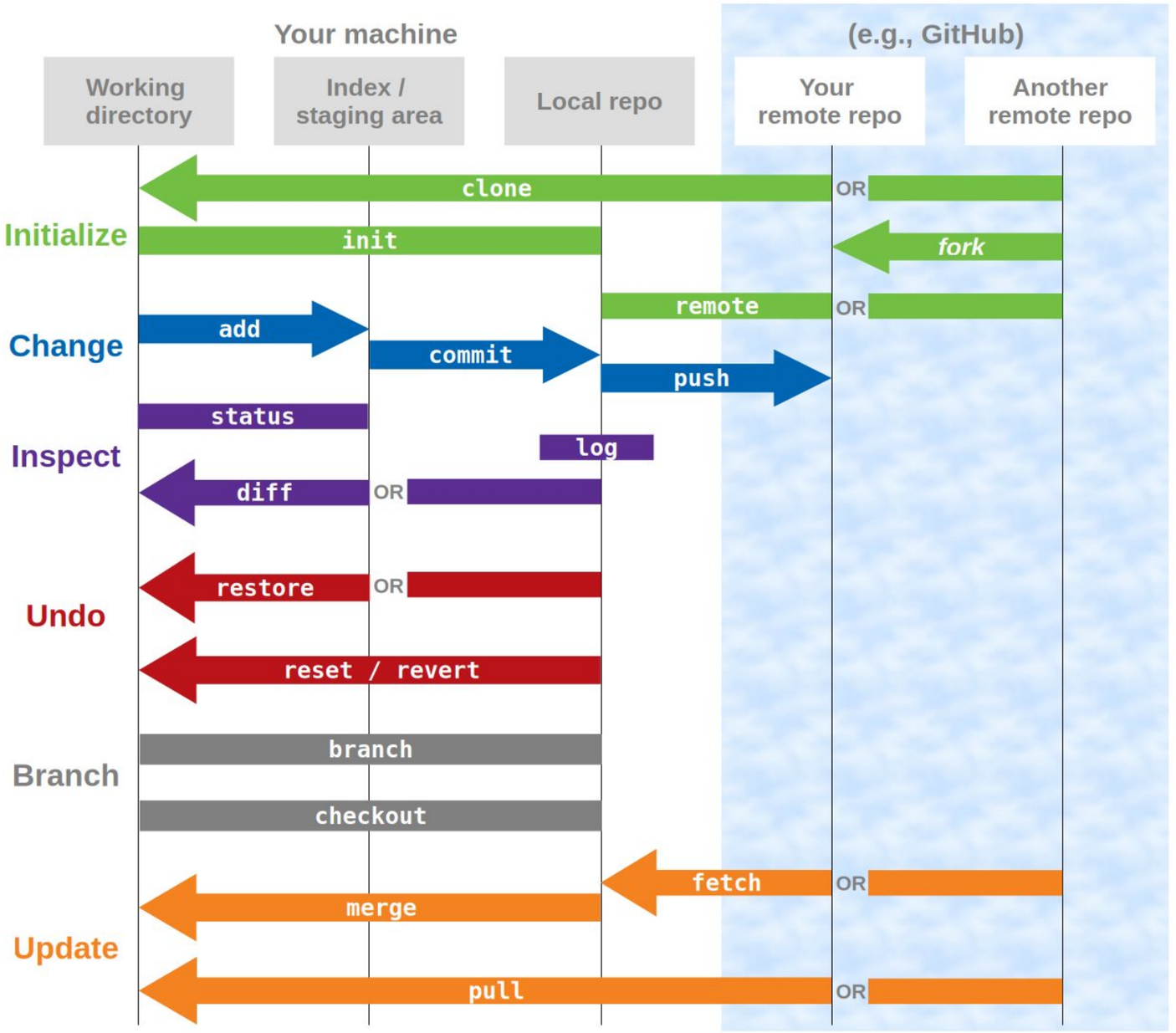
- Git is the "**language**" we use to do version control.
- GitHub **hosts** git repositories online.



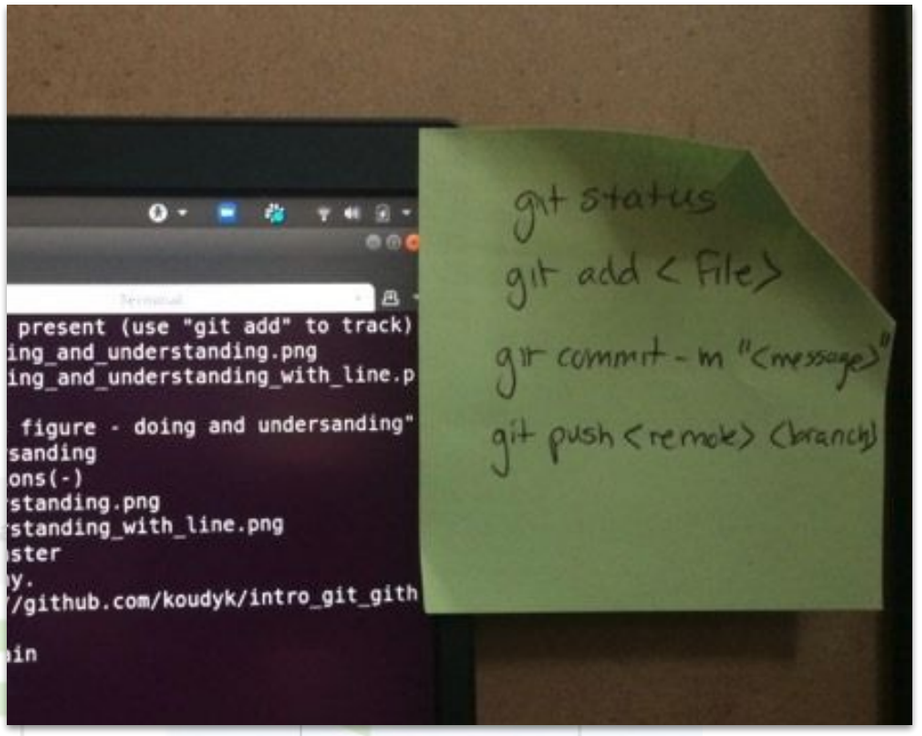
The commands we covered **last week**



The commands we'll have covered **by the end of this lecture**



The commands **you might want to remember**



(Everything else you can look up when you need it)

Goals

Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

Part 2

- **Share your work on GitHub**
- Collaborate on GitHub
- GitHub features
- Try it out!

Basic Bash

- `cd <directory path>` changes directories (aka folders)
  - `.` represents the current directory
  - `..` represents the directory one level up

- `mkdir <directory path>` makes a new directory

- `touch <filename>` creates a new file

- `ls` lists the directory's contents
  - `ls -a` includes hidden files

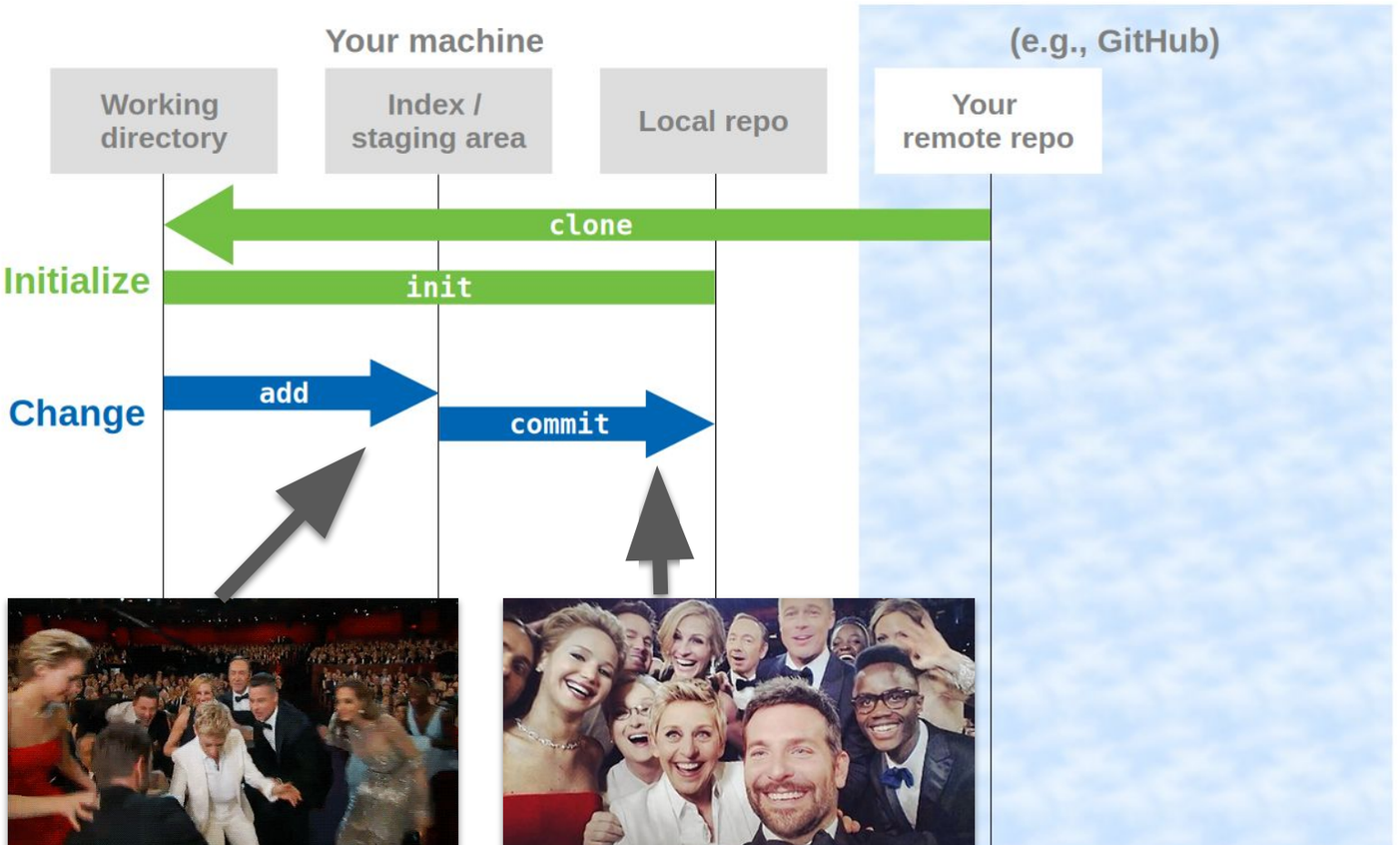
- `<text editor name>` opens a text editor
  - `<text editor name> <filename>` opens a file in a text editor

3-step basic workflow for tracking your work

1. **Modify**
   Change a file in your working tree

2. **Stage**
   `git add <filename>`

3. **Commit**
   `git commit -m "<short, informative commit message>"`



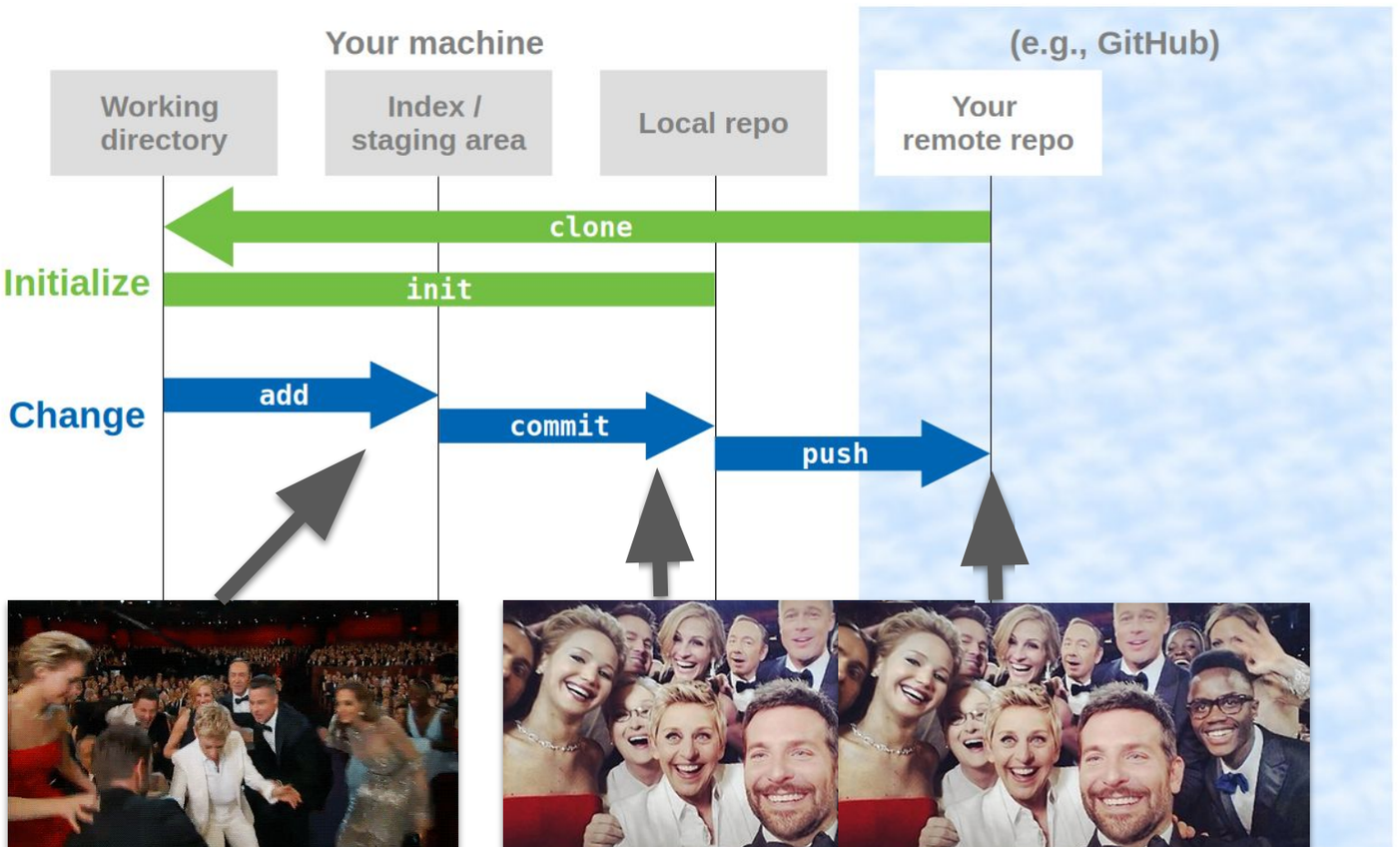
4-step basic workflow for tracking your work **and sharing it**

1. **Modify**
   Change a file in your working tree

2. **Stage**
   `git add <filename>`

3. **Commit**
   `git commit -m "<short, informative commit message>"`

4. **Push**
   `git push <remote> <branch>`

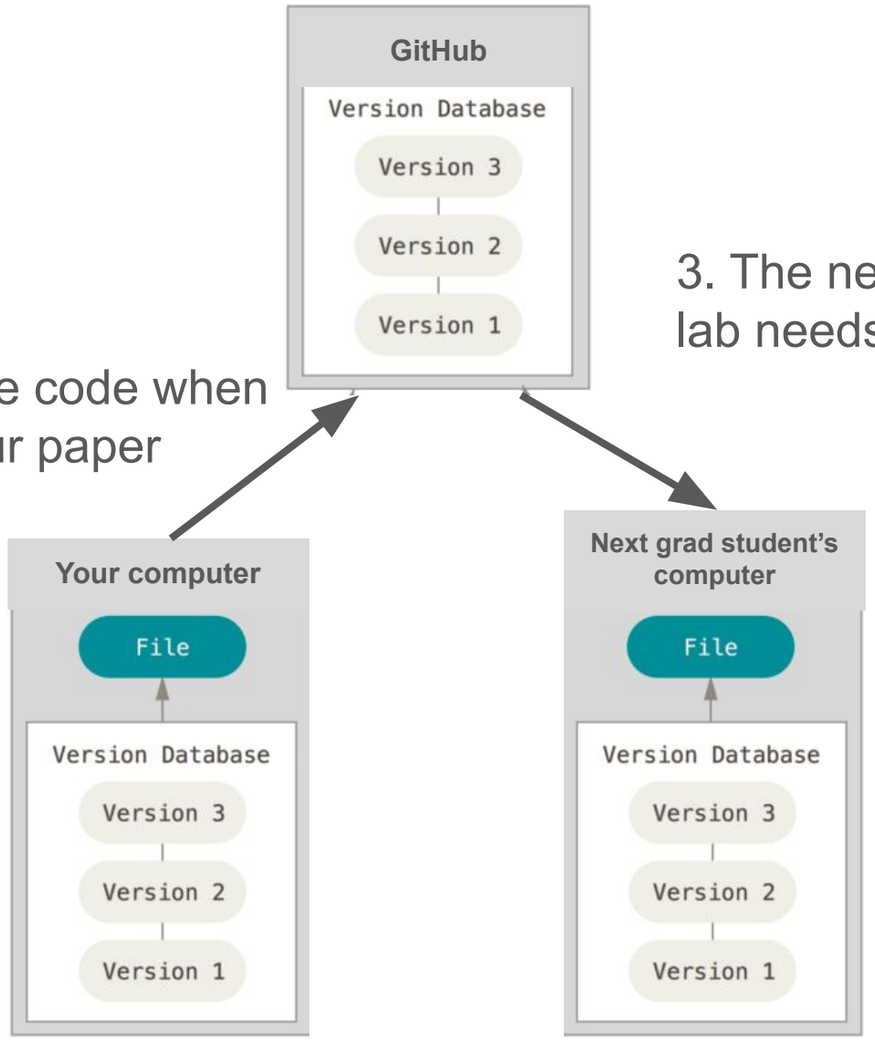


Sharing your work

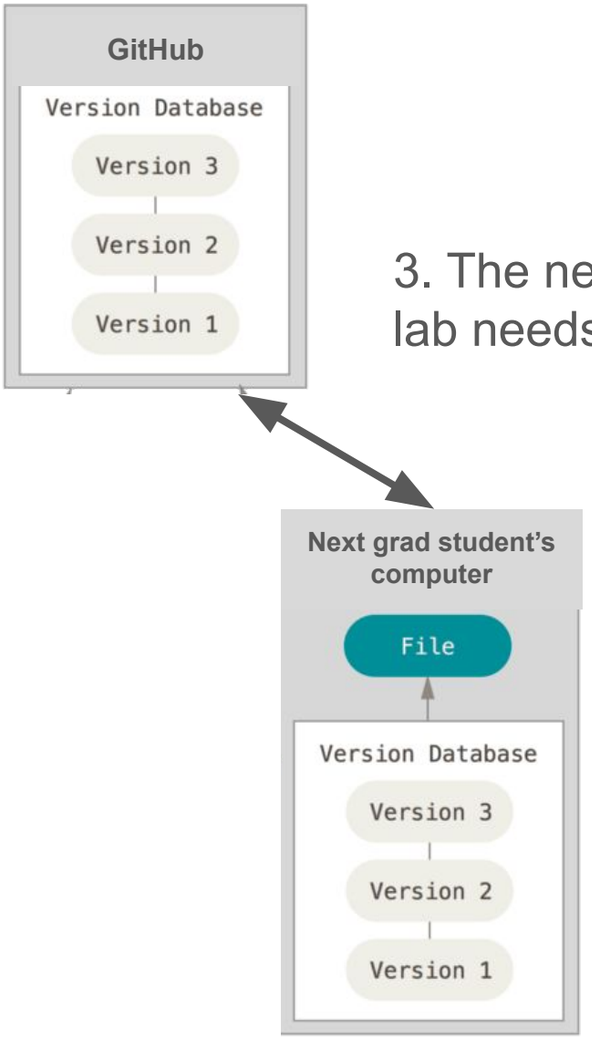
1. You write code to run an analysis for your paper

2. You share the code when you publish your paper

3. The next grad student in your lab needs to use your code



Sharing your work



3. The next grad student in your lab needs to use your code

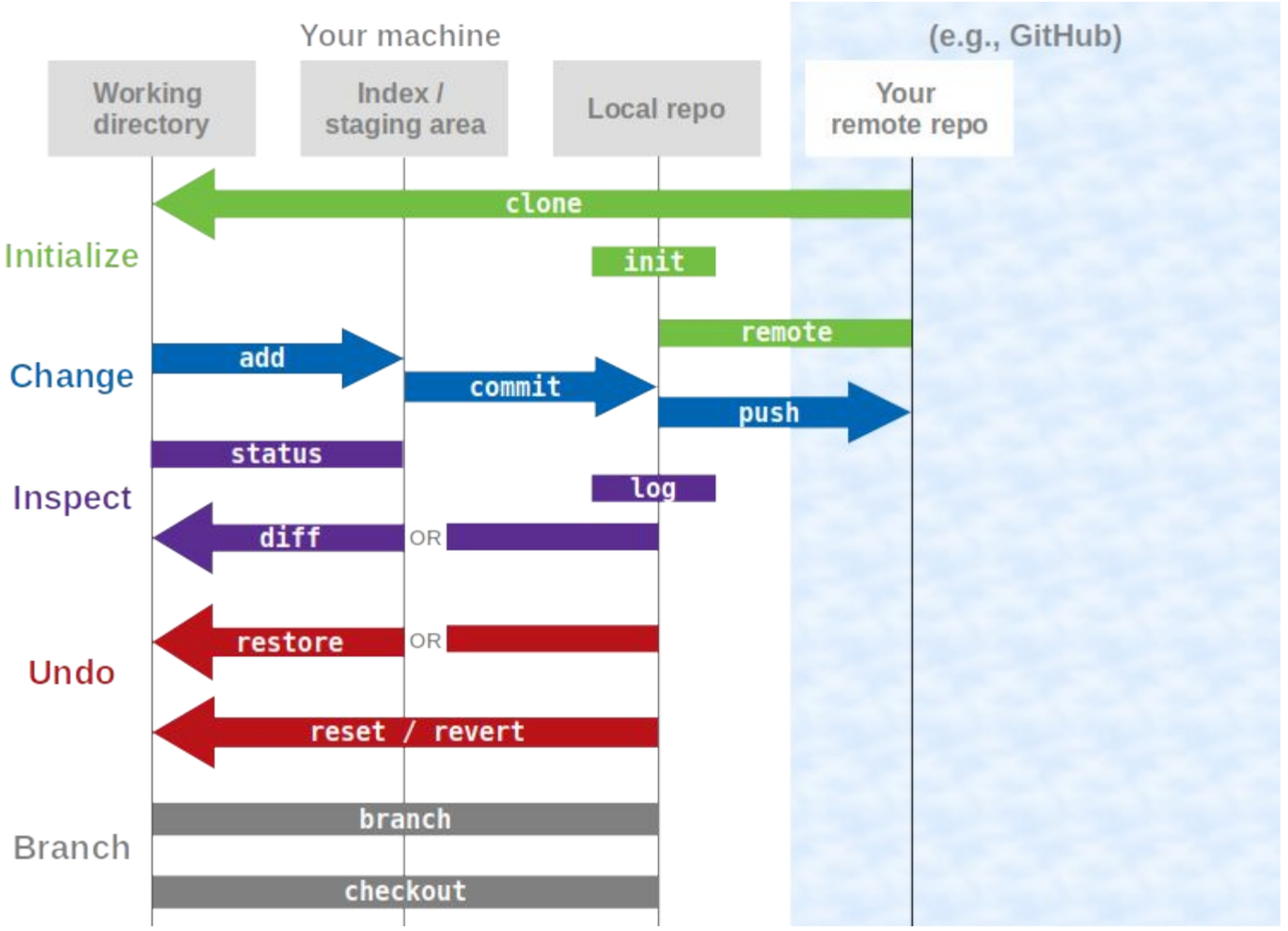
4. The next grad student changes / fixes the code

Git commands for sharing your work

- Show your remote repos
  `git remote -v`

- Push commits to a remote repo
  `git push <remote name> <branch>`

Remotes - sharing your work



Goals

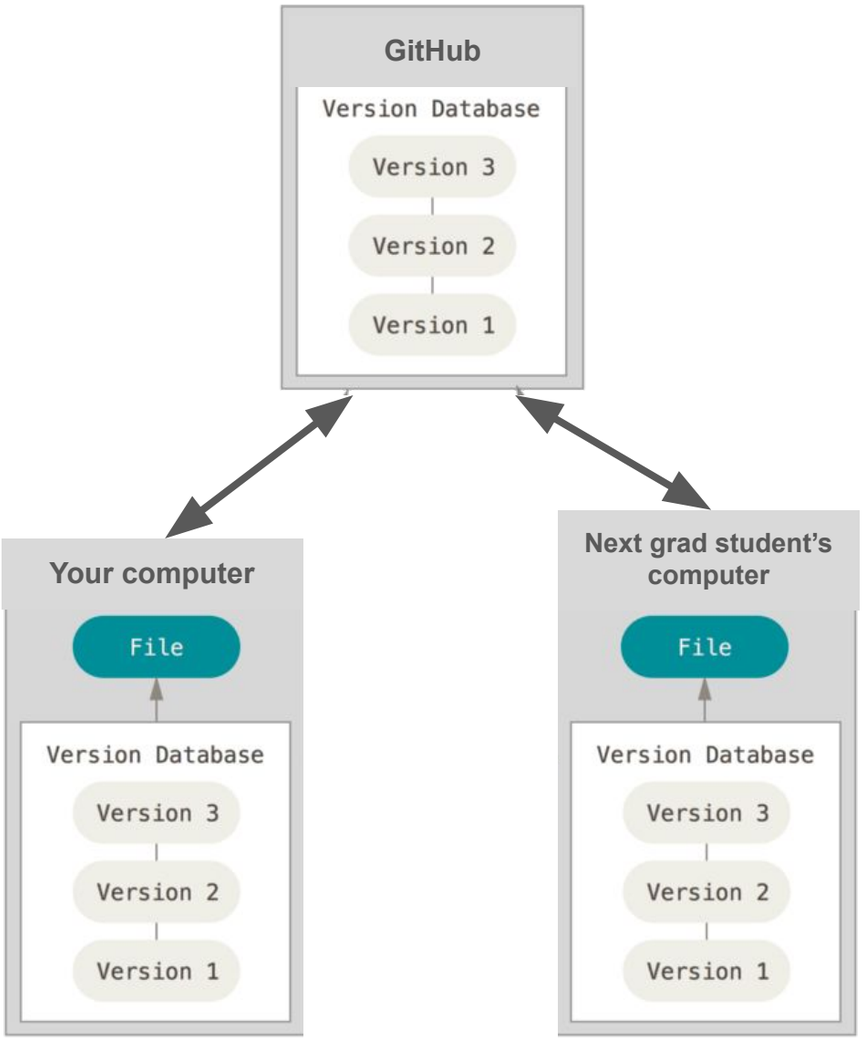
Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

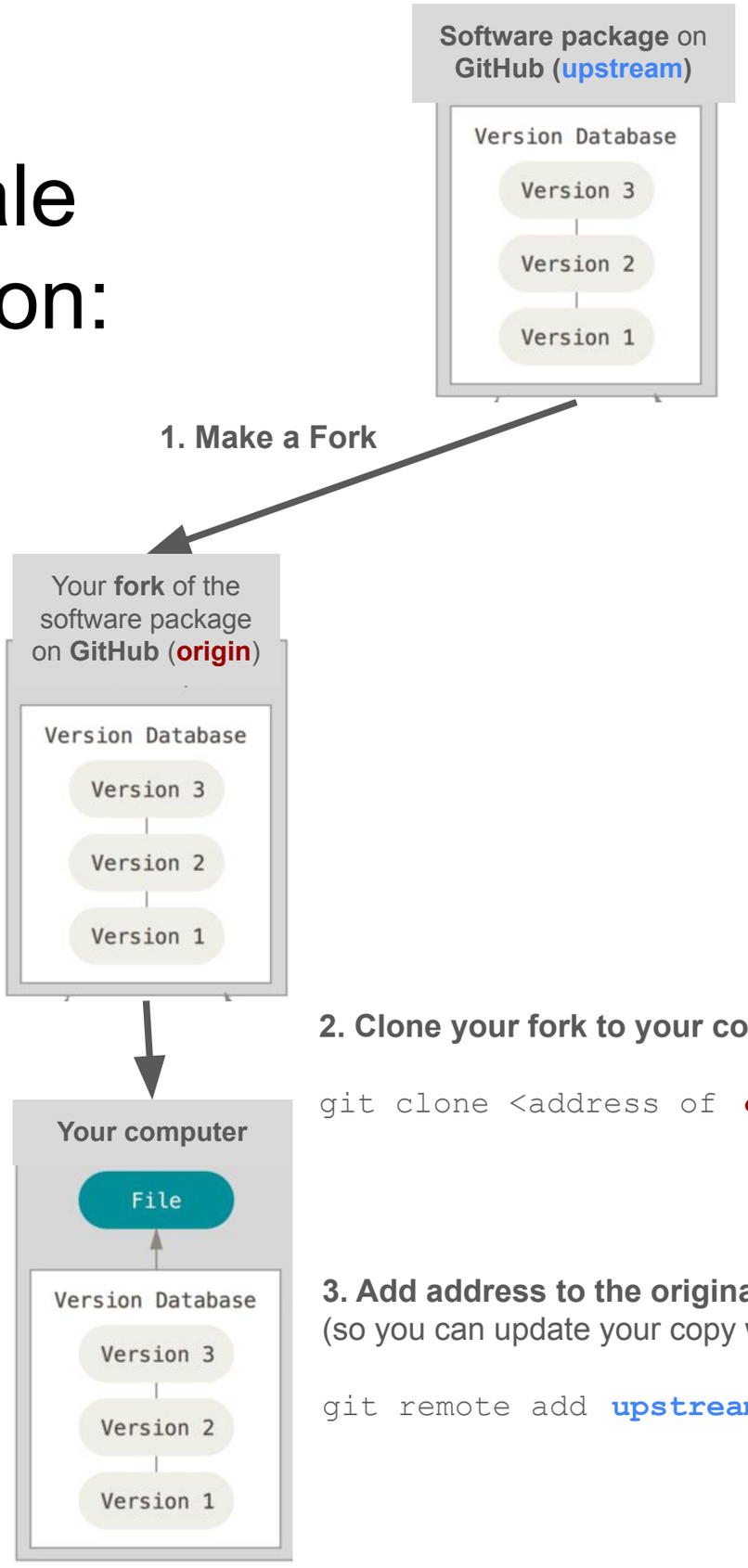
Part 2

- Share your work on GitHub
- **Collaborate on GitHub**
- GitHub features
- Try it out!

Small-scale collaboration



Larger-scale collaboration: **Setup**



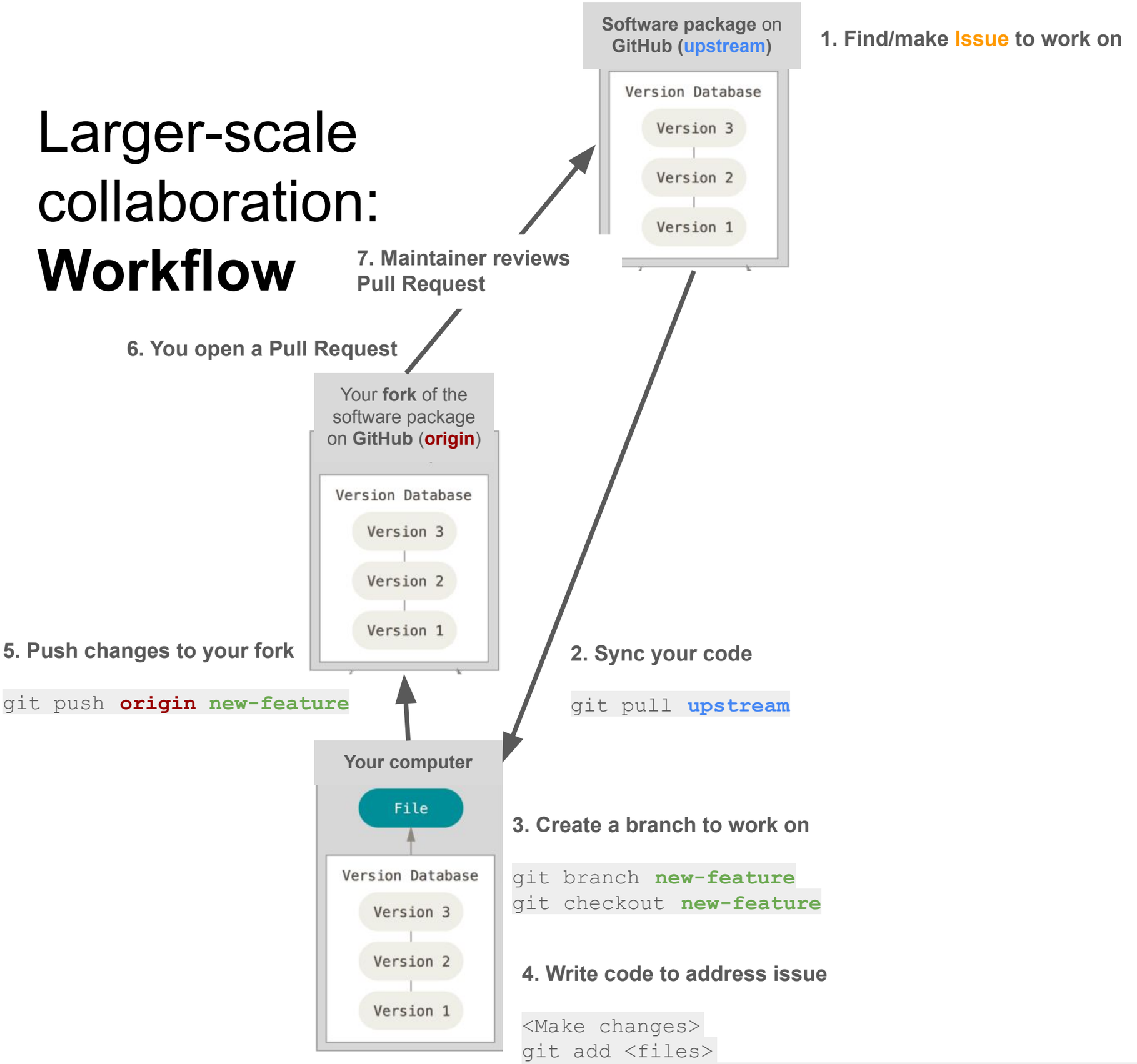
2. Clone your fork to your computer

```
git clone <address of origin>
```

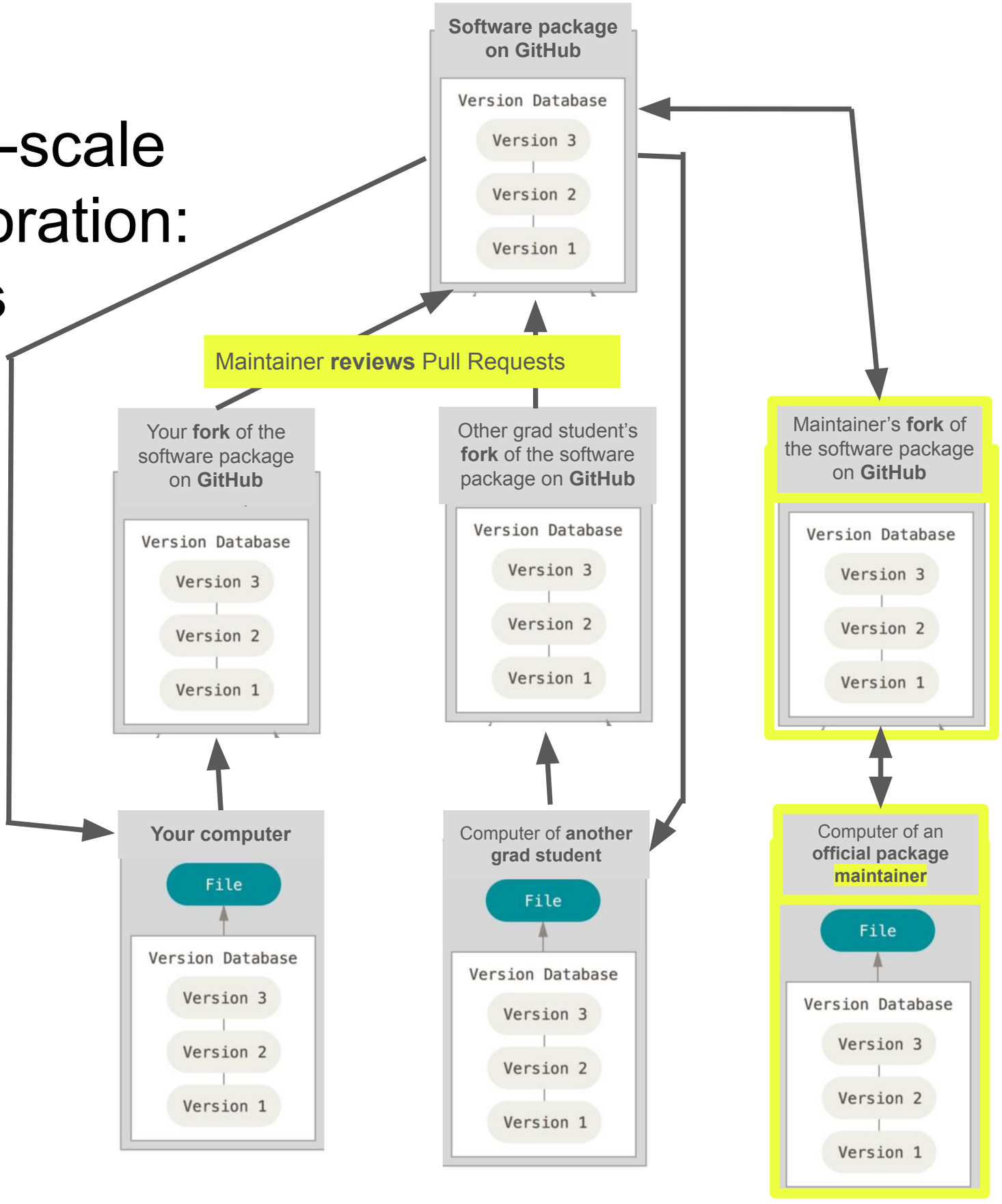
3. Add address to the original repo
(so you can update your copy with others' changes)

```
git remote add upstream <address of upstream>
```

Larger-scale collaboration: **Workflow**



Larger-scale collaboration: **Repos**



Git commands for collaboration

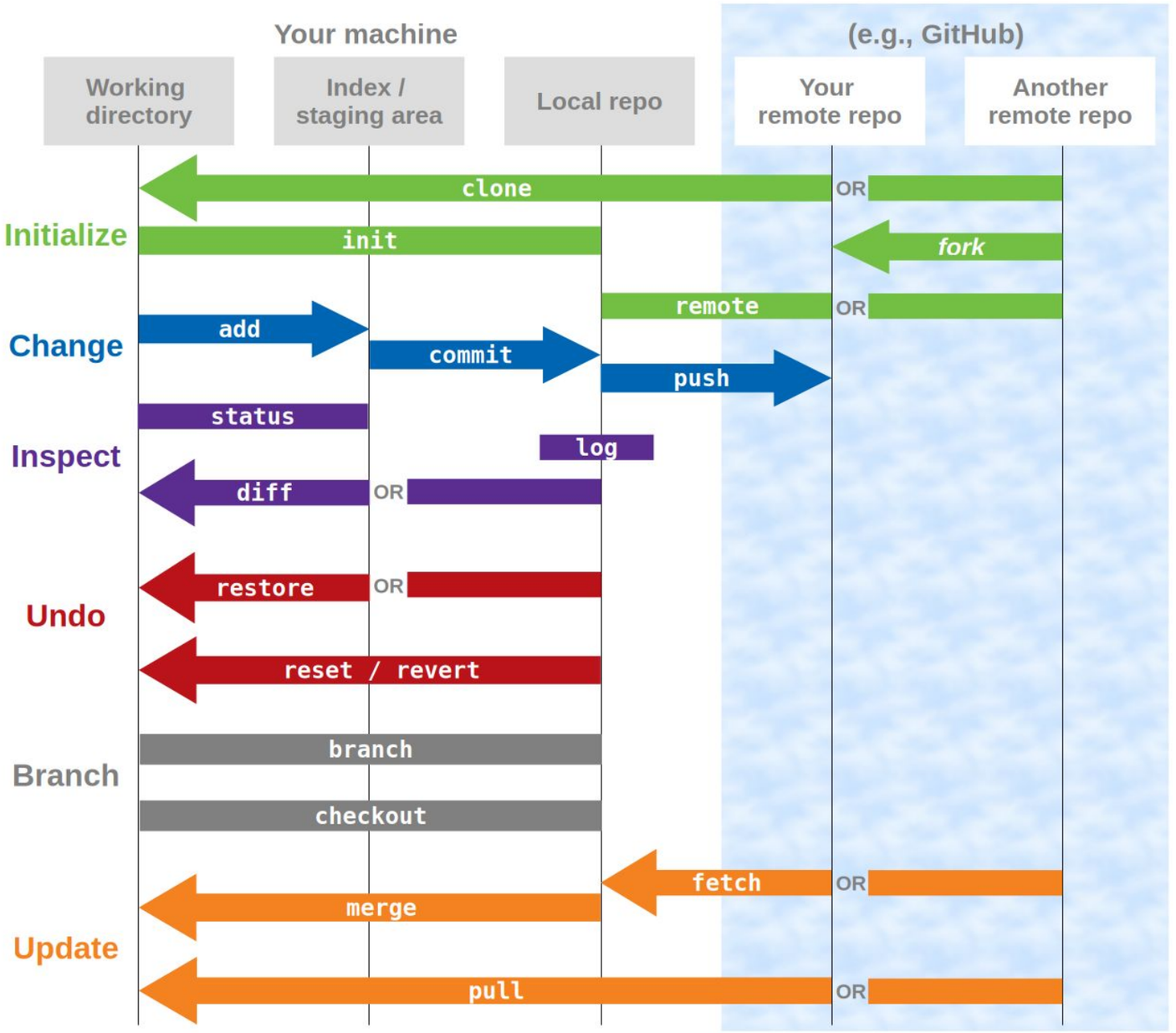
- Show your remote repos
  `git remote -v`

- Add a remote repo
  `git remote add <remote name> <remote address>`

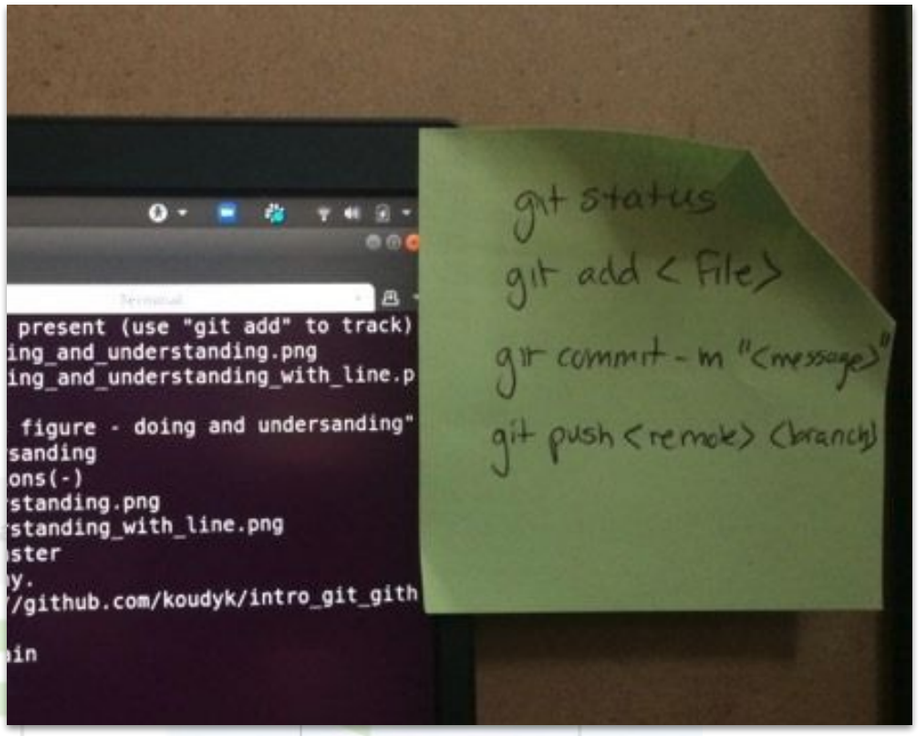
- Push commits to a remote repo
  `git push <remote name> <branch>`

- Pull (fetch and merge) commits from a remote repo
  `git pull <remote name> <branch>`

The commands we covered **today and last week**



The commands **you might want to remember**



(Everything else you can look up when you need it)

Goals

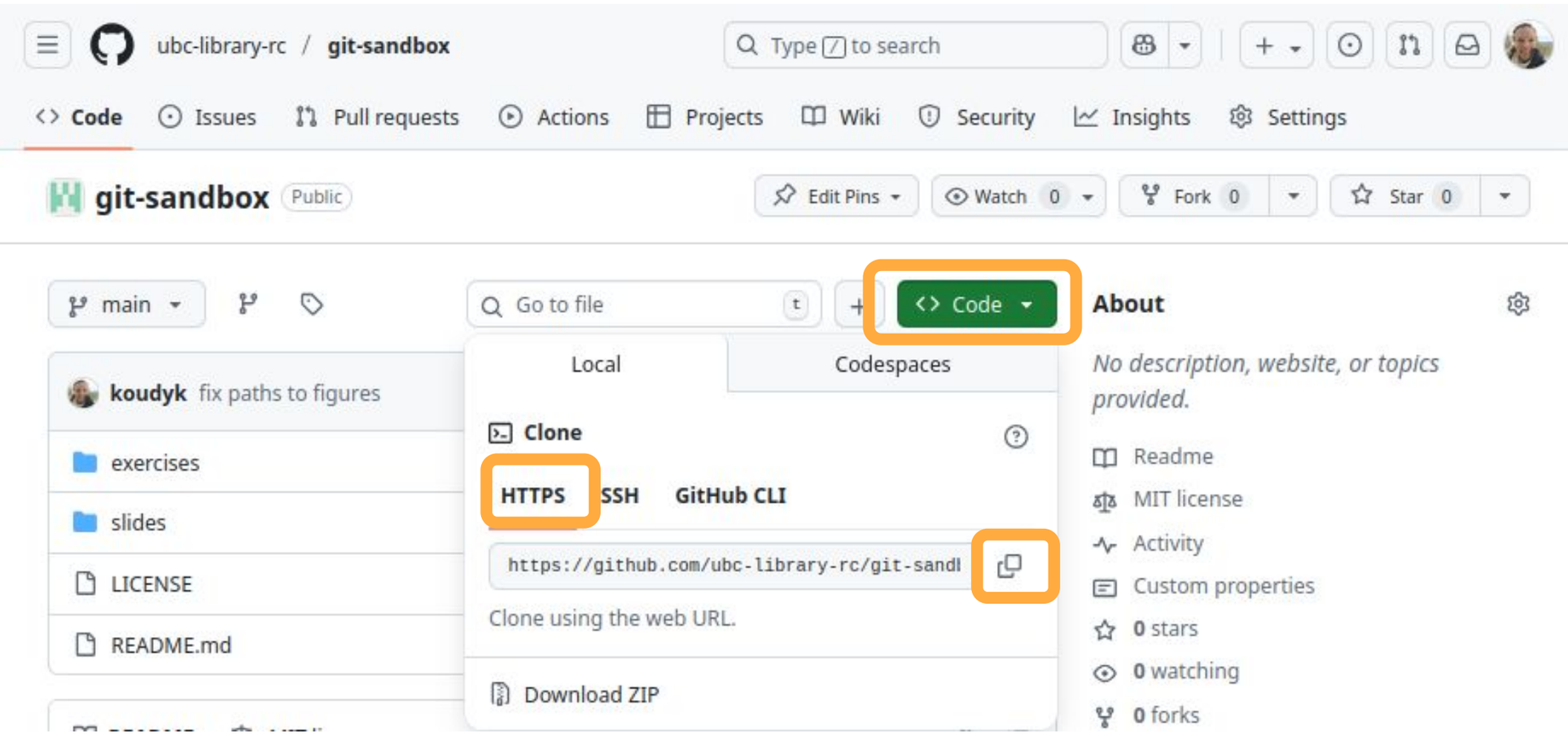
Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

Part 2

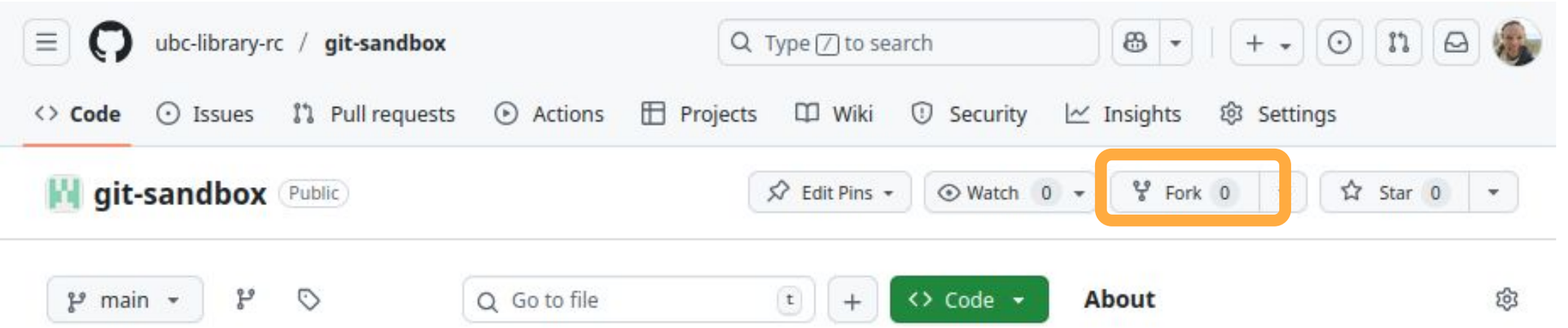
- Share your work on GitHub
- Collaborate on GitHub
- **GitHub features**
- Try it out!

GitHub features



Where to find address for cloning a repo to your computer

GitHub features



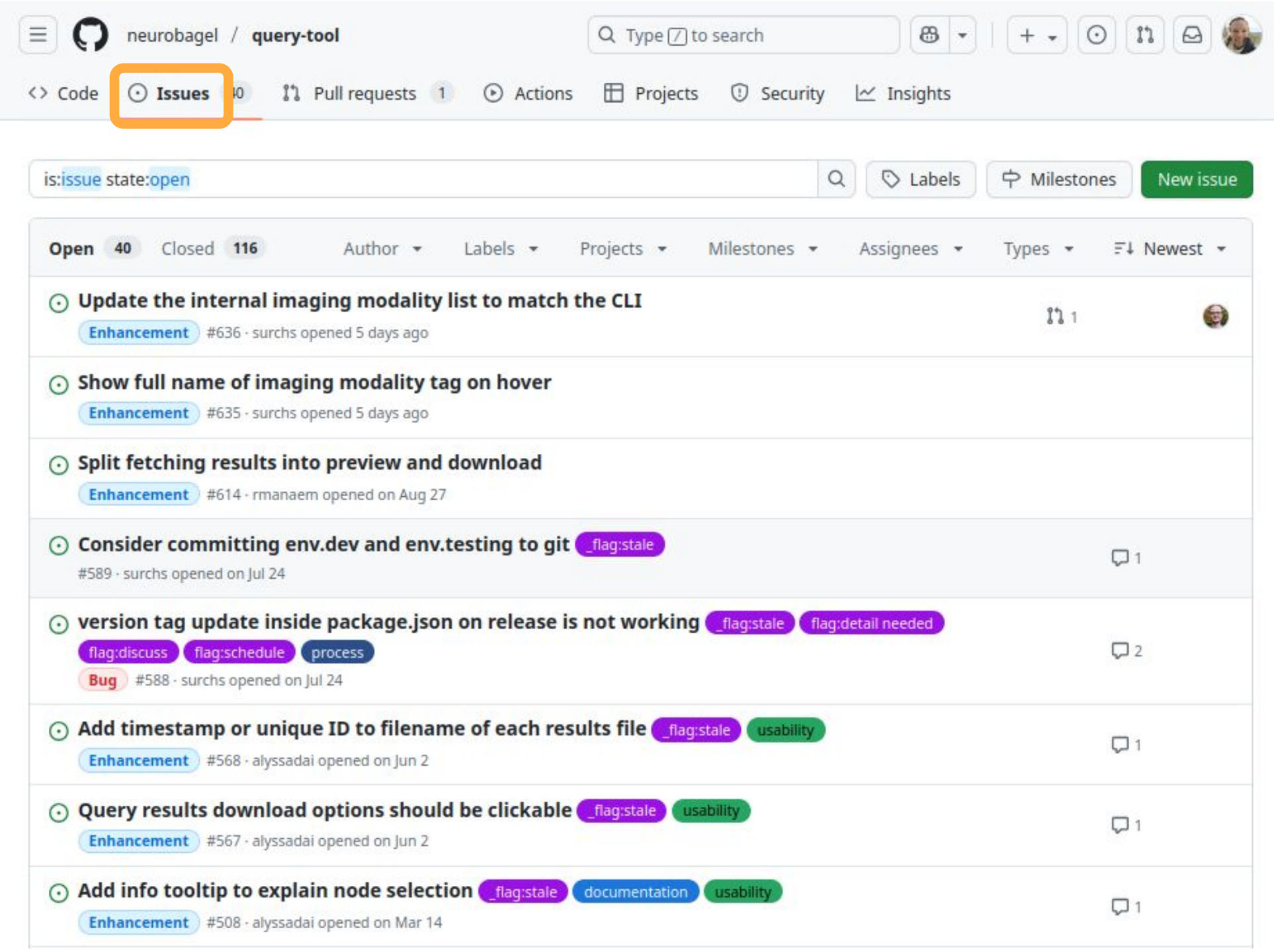
Forks

- Like a clone, but on GitHub
- Most collaborators will have their own fork where they work

GitHub features

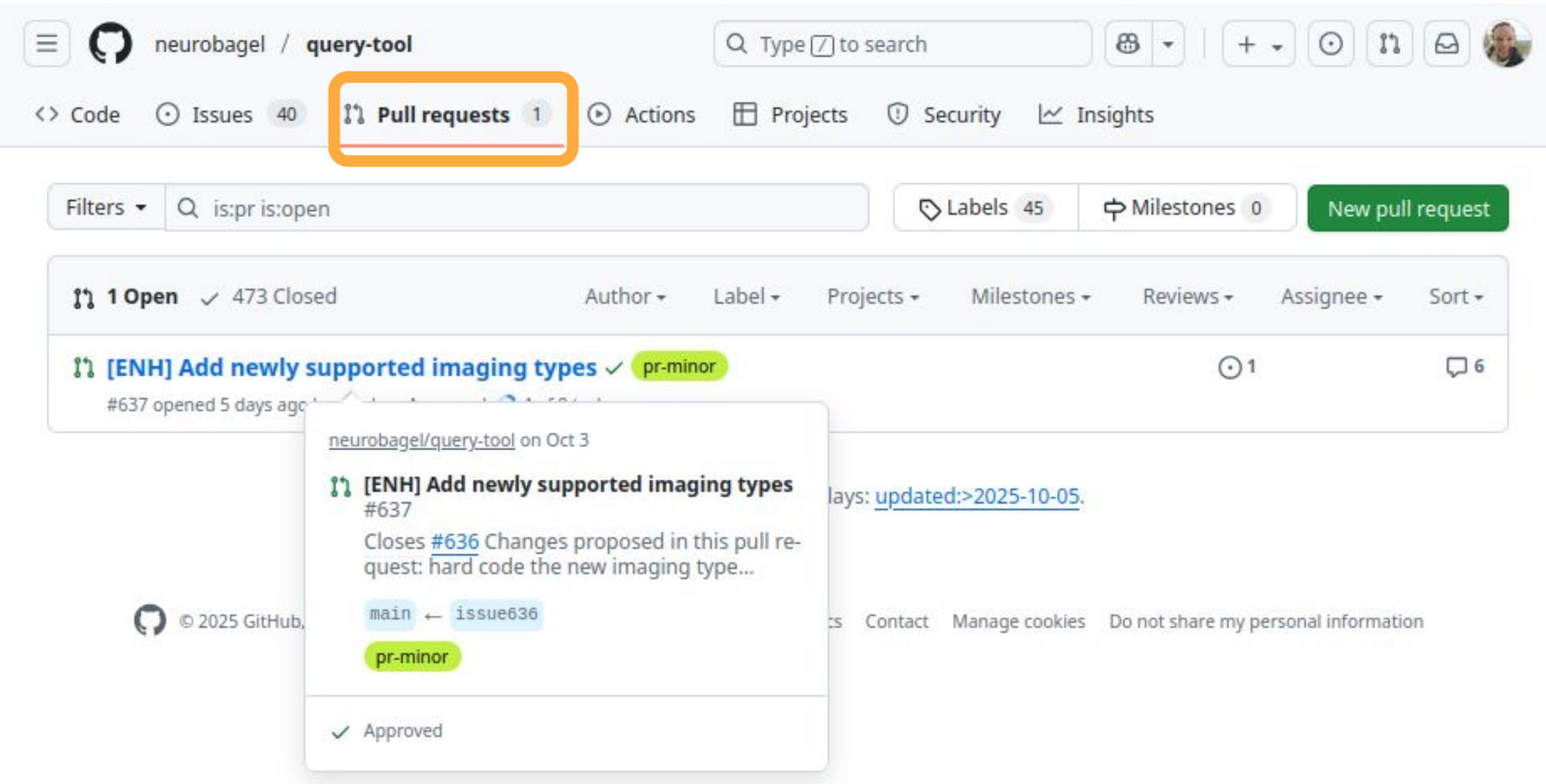
Issues - where you can

- Report a problem
- Propose something new
- Find something to work on



GitHub features

https://github.com/neurobagel/query-tool/pull/637

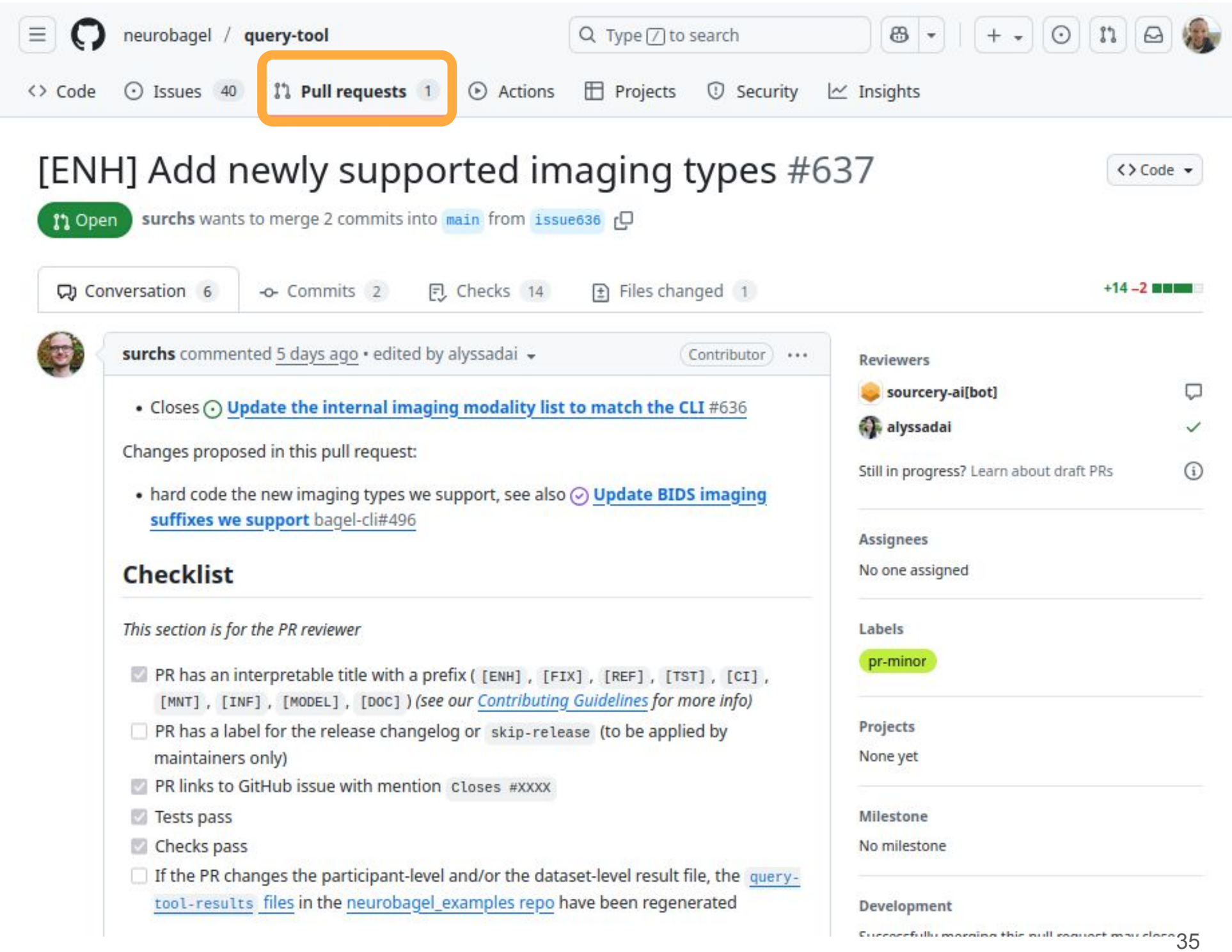


Pull requests

- Push commits from your own brainch to your own fork, then open a pull request
- Then the project maintainers can
  - review your code
  - make suggestions / edits
  - decide whether to merge it into the original repository

GitHub features

https://github.com/neurobagel/query-tool/pull/637



Goals

Part 1 (last week)

- What is distributed version control?
- Why is Git useful?
- Track your own work with Git

Part 2

- Share your work on GitHub
- Collaborate on GitHub
- **Try it out!**

Check if you're ready

☑️ A) Can you open a bash shell?

- Open a terminal, type `echo $SHELL` and press ENTER.
- The output should be `/bin/bash`

☑️ B) Do you have git installed?

- In the bash terminal, `git --version` and press ENTER.
- The output should be `git version X` (where the X is the version number)
- *Don't worry if you don't have the exact same version as I do*

☑️ C) Do you have git configured?

- In the bash terminal, type `git config --list` and press ENTER
- You should see your name and email (and other things that aren't essential to configure)

☑️ D) Can you open a text editor? E.g.,

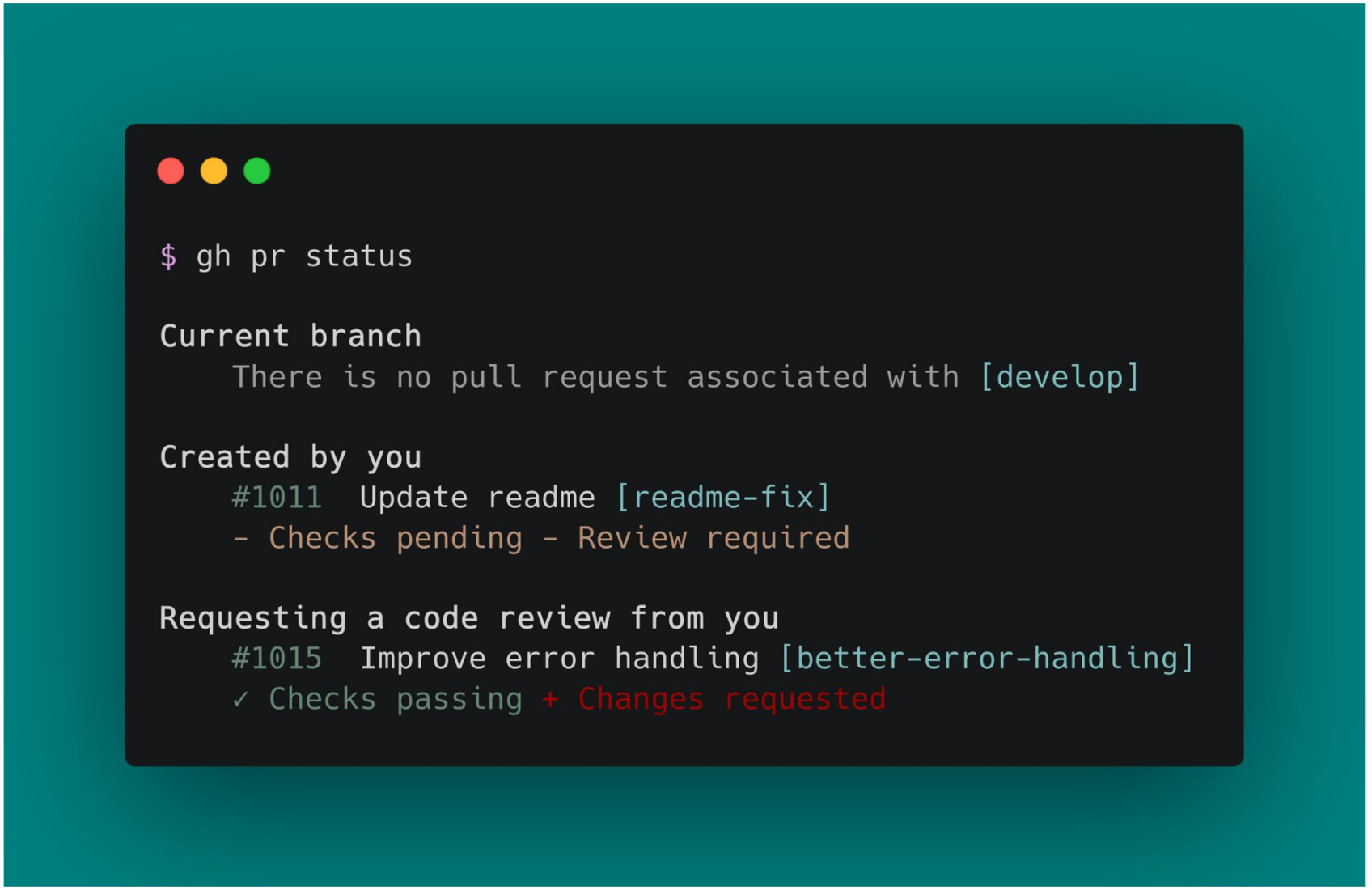
- Linux: gedit, nano
- macOS: textedit
- Windows: notepad

☑️ E) Can you go your GitHub account?

Linking git to GitHub

GitHub CLI (FYI; not recommended here)

- Lets you use GitHub from the terminal
- Makes workflows more efficient for software developers and maintainers
- Probably overkill for many of us



Linking git to GitHub

SSH (FYI; not recommended here)

- Secure Shell Protocol (SSH Protocol)
- “a cryptographic network protocol for operating network services securely over an unsecured network.” (https://en.wikipedia.org/wiki/Secure_Shell)
- Often preferred by programmers
- May be more secure

- May be more challenging for those new to programming

Linking git to GitHub

Git Credit Manager (what we'll use today)

- Manages personal access token and 2-Factor Authentication

- **Windows**
  - If you installed Git for Windows, you already have it!

- **MacOS**
  - Type this in the terminal:
    ```
    brew install --cask git-credential-manager
    ```

- **Linux**
  - More complicated…
  - Install GCM
  - Configure git and GCM

Check if you’re ready, con’t

☑️ F) Have you connected git to your GitHub account?

- This will download the repo with the exercises.
  In the terminal, paste this:

```
git clone https://github.com/ubc-library-rc/git-sandbox.git
```

Acknowledgements

Many parts of this presentation are inspired / based on these great resources:

- Chacon, S., & Straub, B. (2014). Pro git. Springer Nature. Available at https://git-scm.com/book/en/v2
- The Carpentries. (2021). Version Control with Git. https://swcarpentry.github.io/git-nov....
- https://docs.github.com/en/get-started/git-basics/set-up-git

Figure references

Here are the sources of some figures on my slides:

- The Carpentries. (2021). Version Control with Git. https://swcarpentry.github.io/git-nov....
- Chacon, S., & Straub, B. (2014). Pro git. Springer Nature. Available at https://git-scm.com/book/en/v2
- https://github.com/cli/cli#installation

The end

Options

- Commands can become powerful with options

```
git log --pretty=format:"%h - %an, %ar : %s" --graph
```

- “How on earth will I remember that??”

- You can set aliases for commands you use a lot

```
git config --global alias.fancylog 'log
--pretty=format:"%h - %an, %ar : %s" --graph'

git fancylog
```