

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

University of British Columbia, Point Grey Campus (**Vancouver**)

We would like to begin by acknowledging that the land on which we gather is the traditional, ancestral, and unceded territory of the **xwməθkwəy̓əm** (Musqueam) People.

University of British Columbia, Okanagan Campus (**Kelowna**)

We would like to begin by acknowledging that the land on which we gather is the unceded territory of the **Syilx** (Okanagan) Peoples.

Find out about the land where you live at <https://native-land.ca/>

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

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

Introduction to Git and GitHub

[Pre-workshop Setup](#)

[Land acknowledgement](#)

[Outline](#)

[Concepts and tools](#)

[Git basics](#)

[Syncing with GitHub](#)

[GitHub features](#)

[Collaborating on GitHub](#)

[Reference](#)

[Extra Material](#)

[Resources and acknowledgements](#)

Search Introduction to Git and GitHub

Introduction to Git and GitHub

Learn the basics of using Git and GitHub for version control and collaboration. Git is a widely used version control software that tracks changes to a group of files, referred to as a repository. GitHub is a popular website for hosting and sharing Git repositories, making it easier to collaborate and share your work. Together, Git and GitHub provide a platform that is increasingly used for collaboration in research and academic environments.

In this beginner workshop, participants will learn key concepts, create their own Git repository, and publish to GitHub. No previous experience with Git is required. Familiarity with the command line interface will be helpful but is not necessary.

Pre-workshop setup

Please make sure to have a Bash Shell and Git installed **before** the workshop.

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
- 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

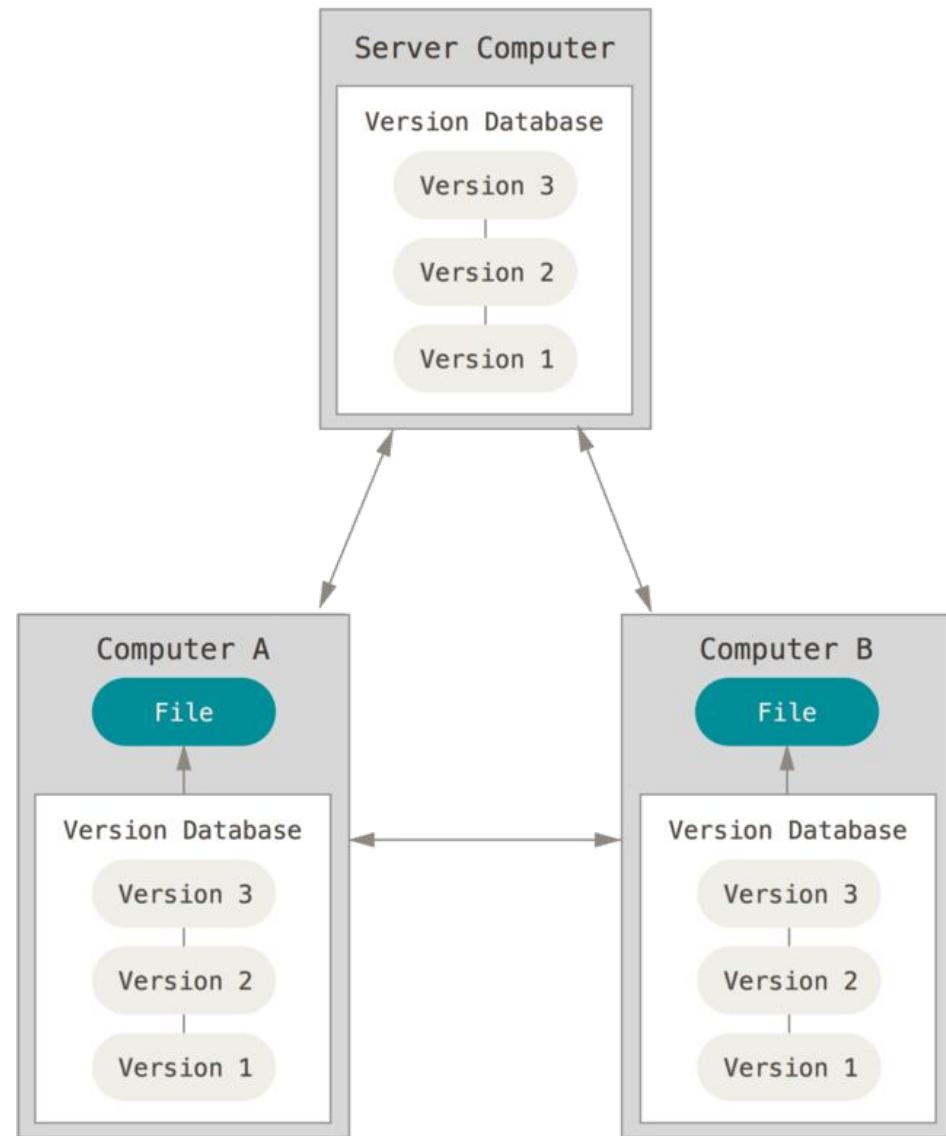
Part 1 (last week)

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

Part 2

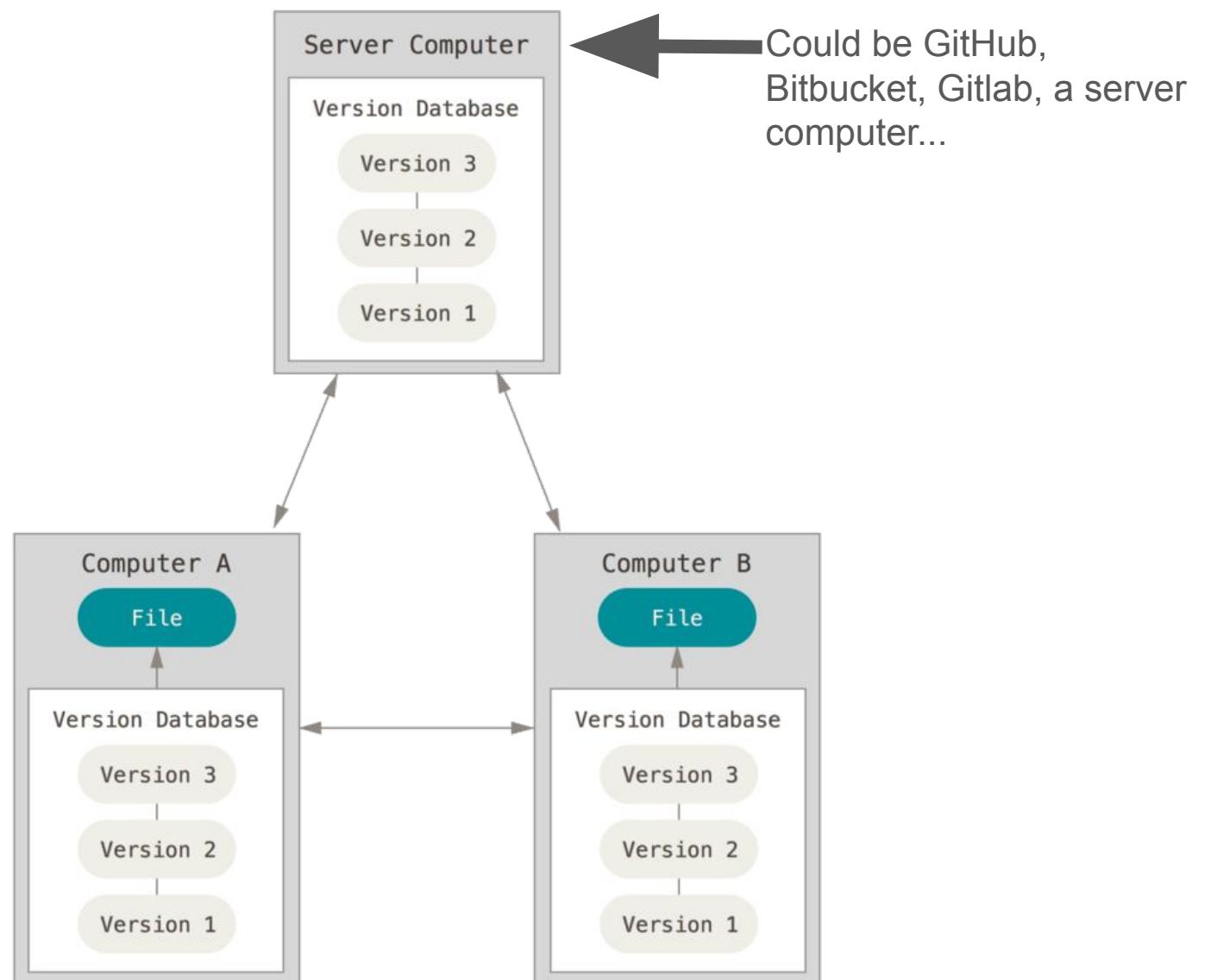
- Share your work on GitHub (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating GitHub
- Try it out!

Distributed version control

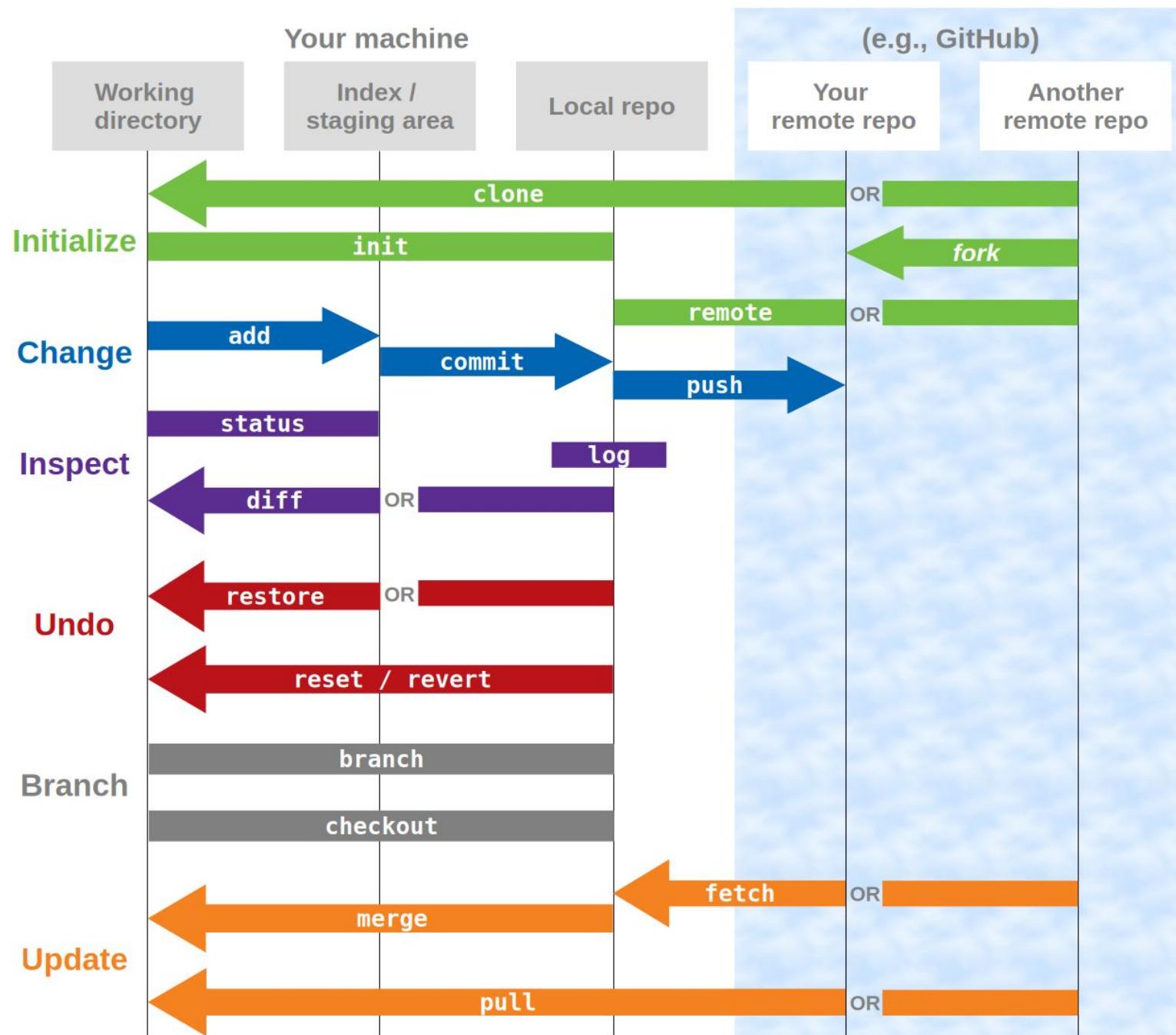


Git vs. GitHub

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

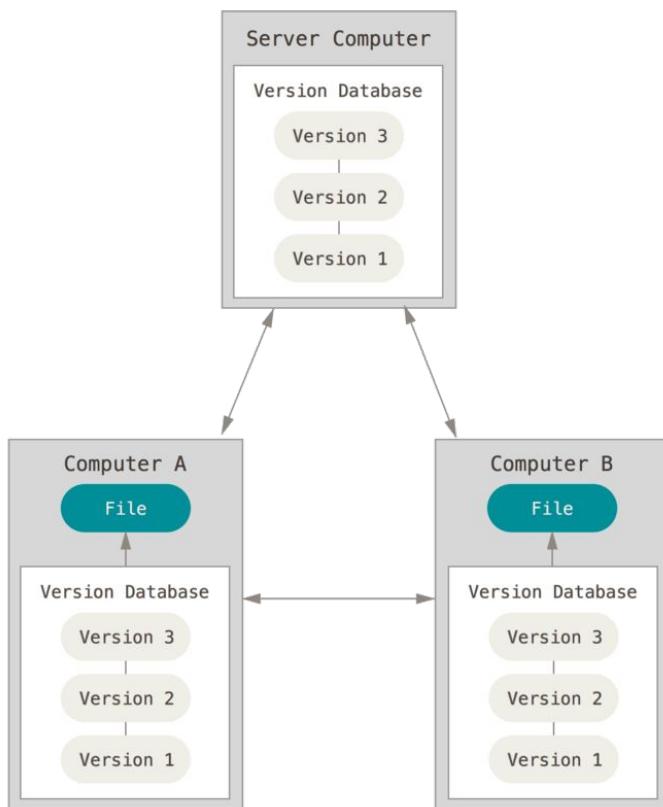


The commands we'll have covered by the end of this lecture

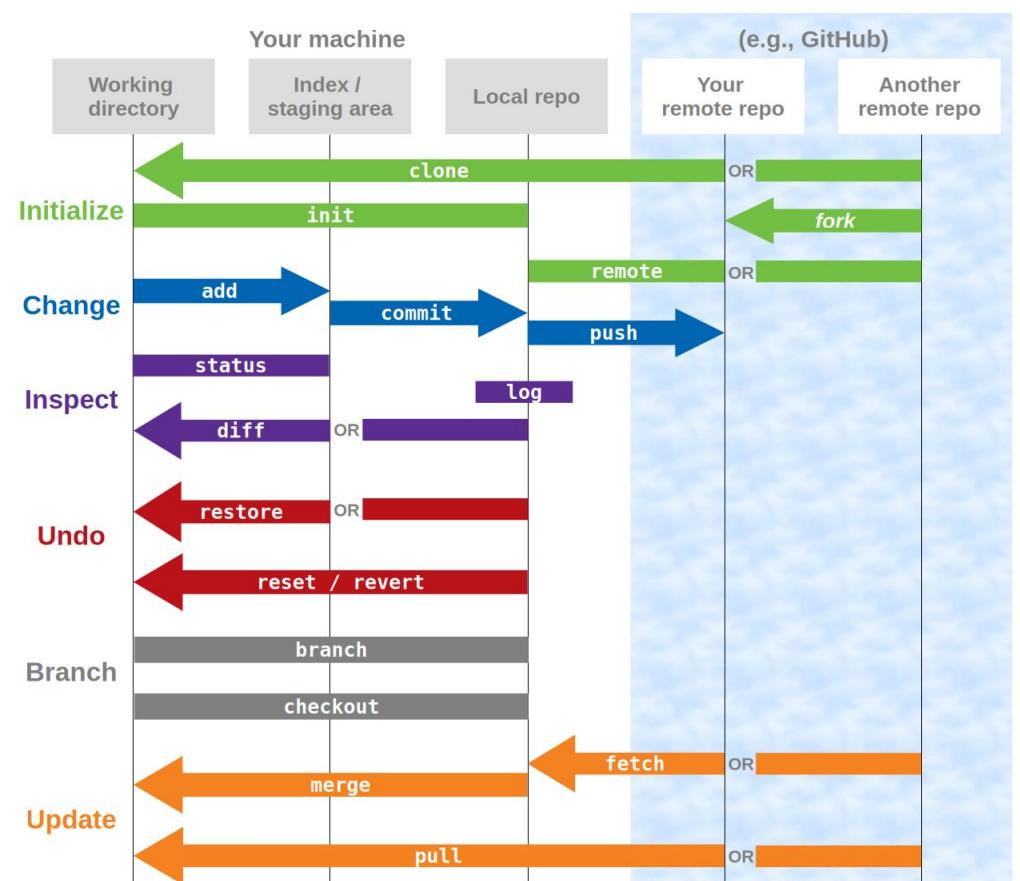


2 kinds of visualizations we'll use

Machines



Commands



3 parts of a Git project

1. Working directory

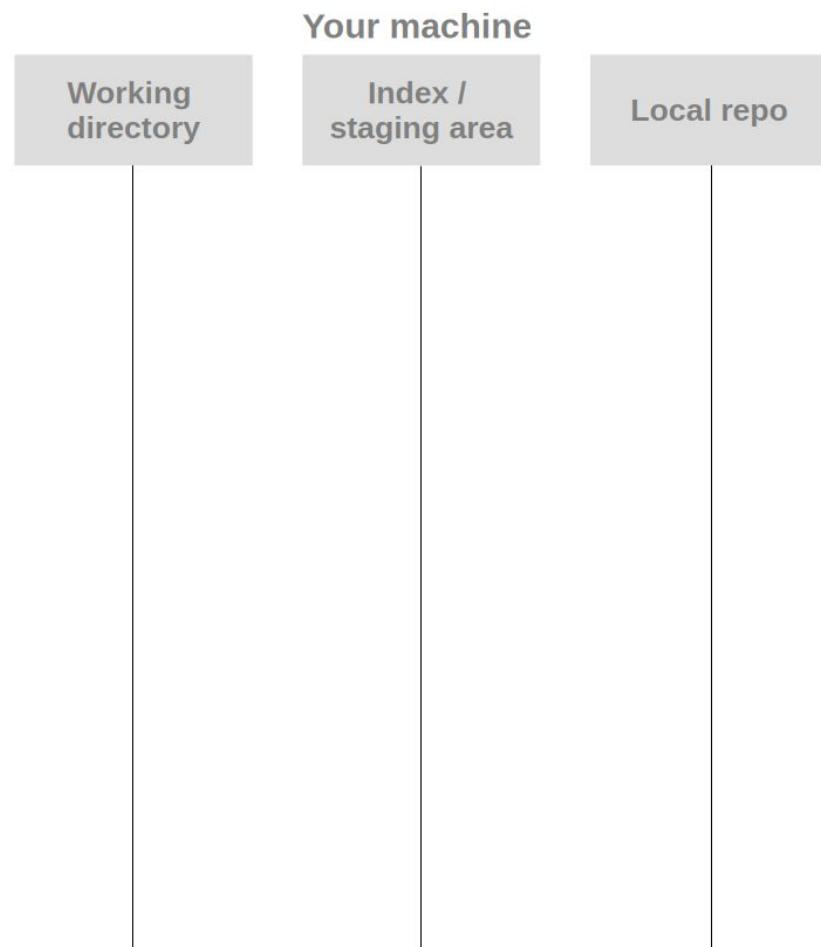
The version of the project that you're working on

2. Staging area / Index

What will be in your next snapshot

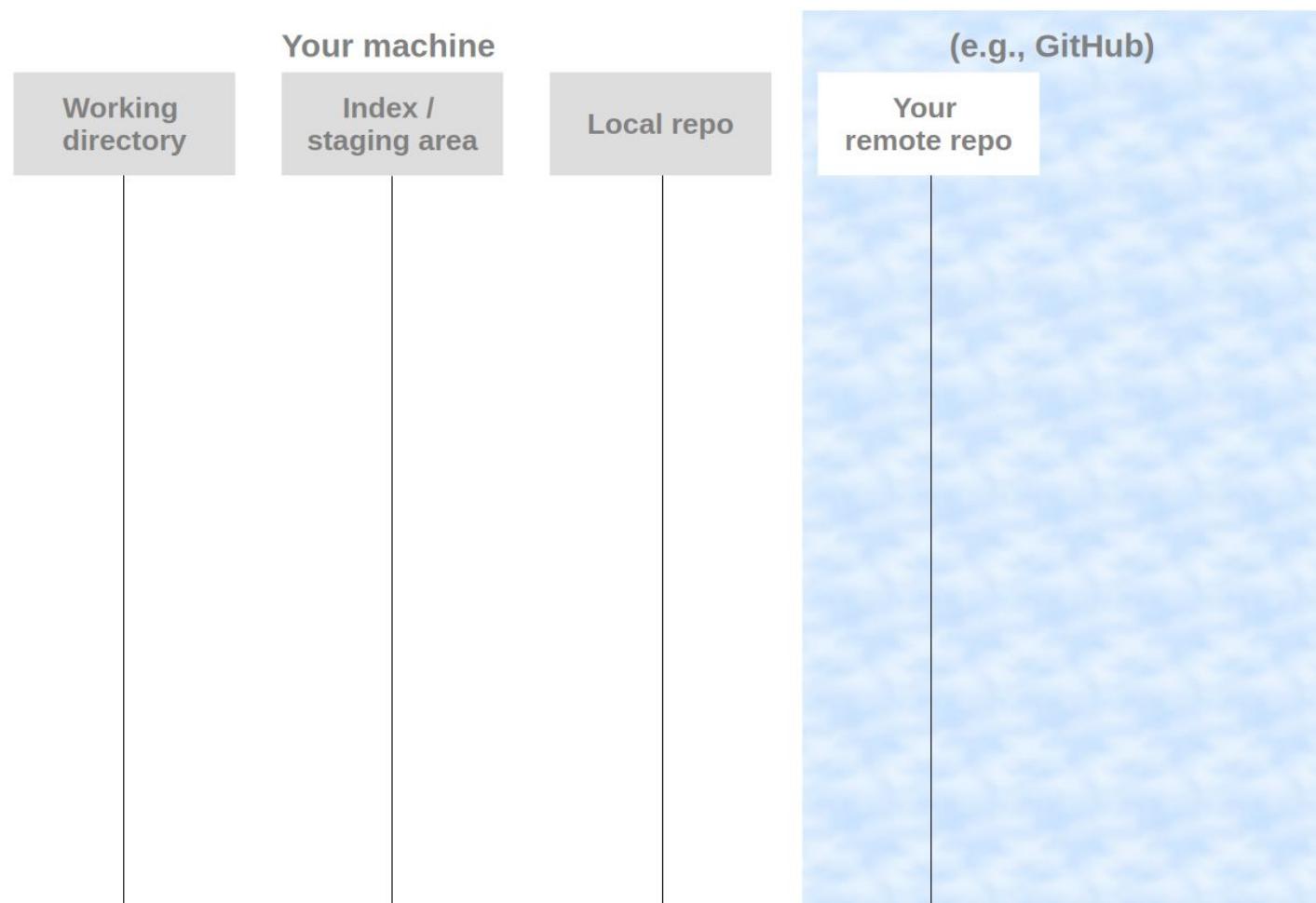
3. Local repository (i.e., .git/ folder)

Metadata and objects that make up the snapshots



Where's GitHub?

GitHub hosts the remote copy of your repository



Starting a git repo

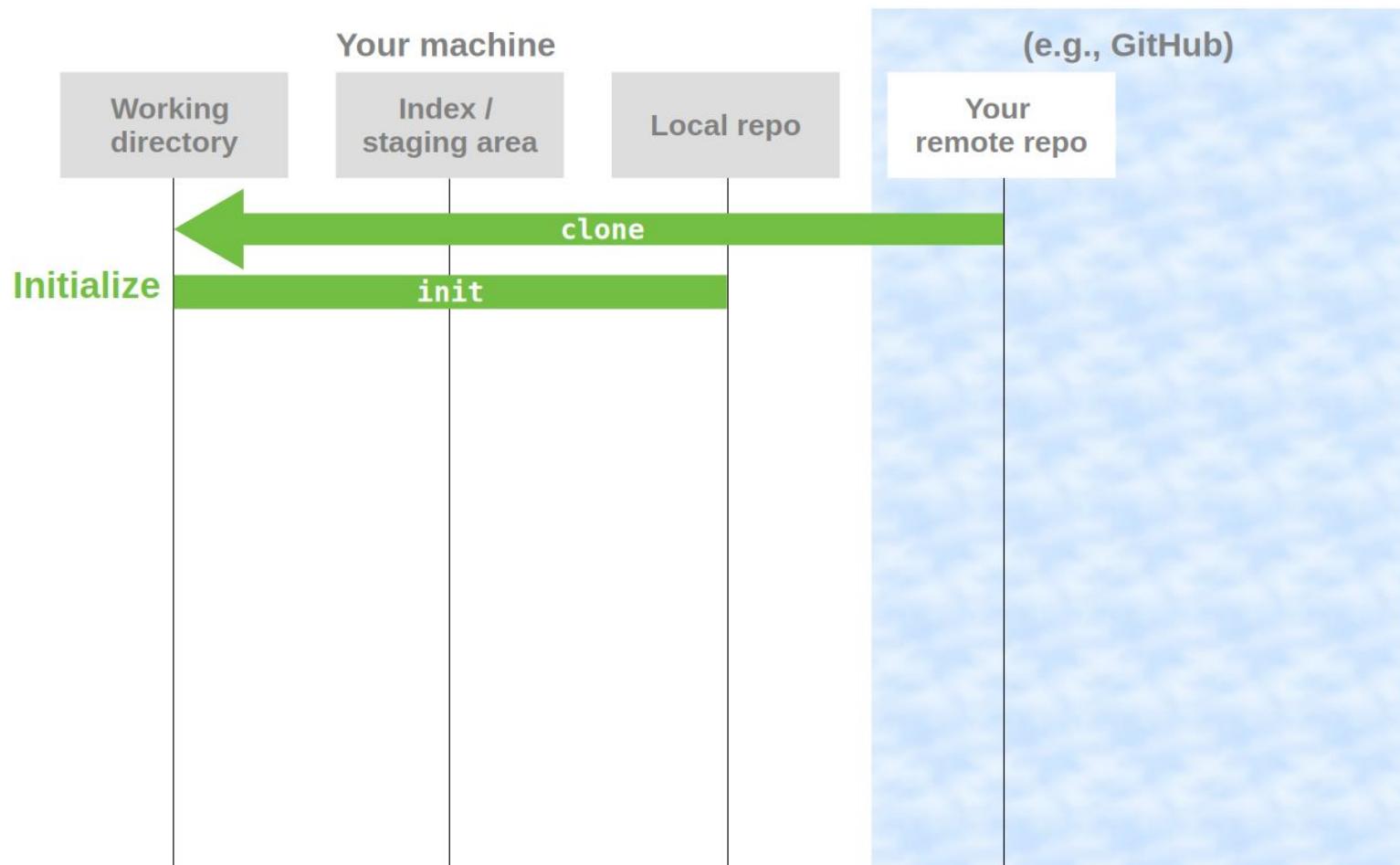
2 options

1. Make an existing folder into a git repo

```
cd <directory>  
git init
```

2. Clone an existing repo (e.g., from GitHub)

```
git clone <repo URL>
```



3-step basic workflow

1. Modify

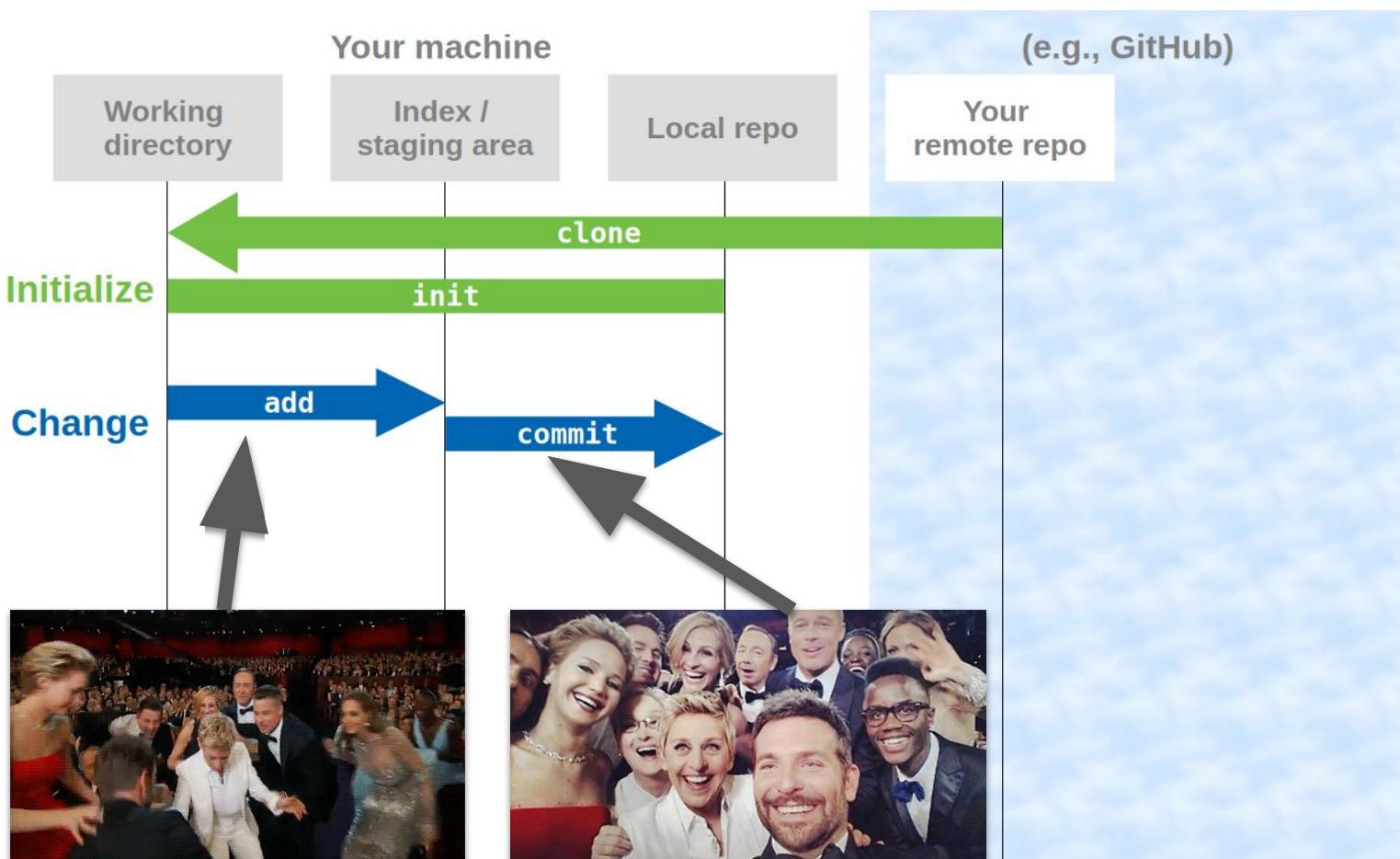
Change a file in your working tree

2. Stage

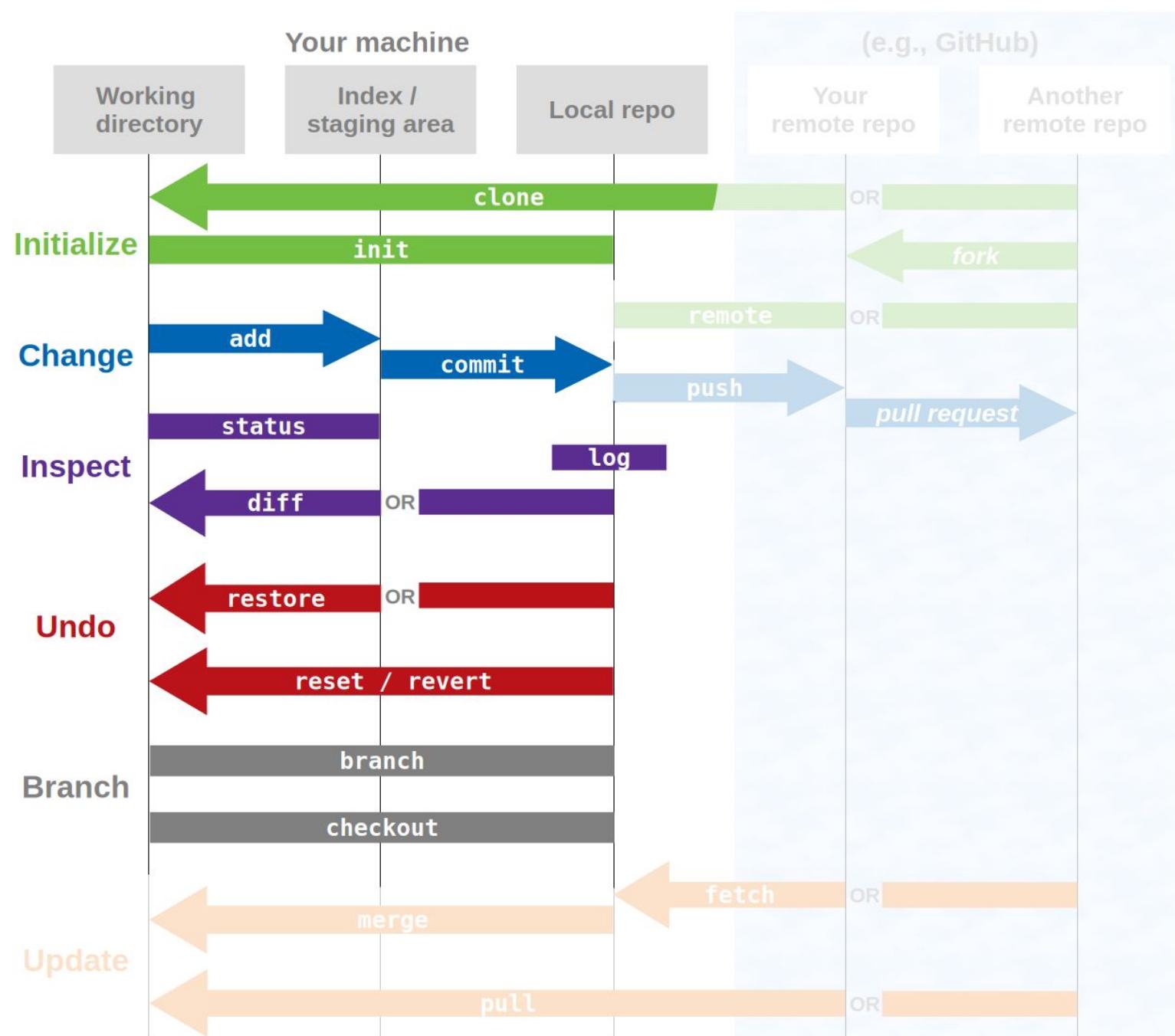
```
git add <filename>
```

3. Commit

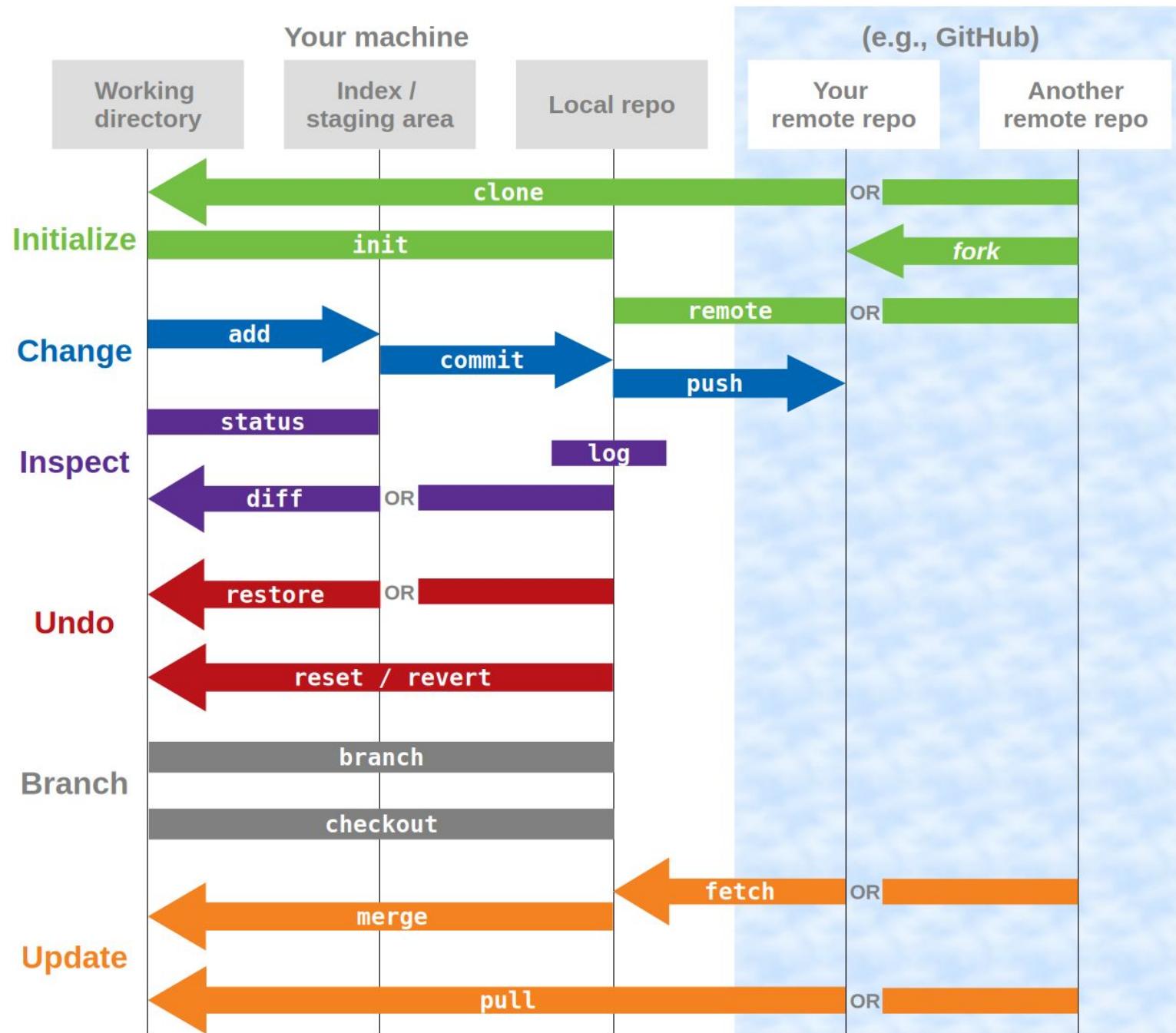
```
git commit -m "<short, informative commit message>"
```



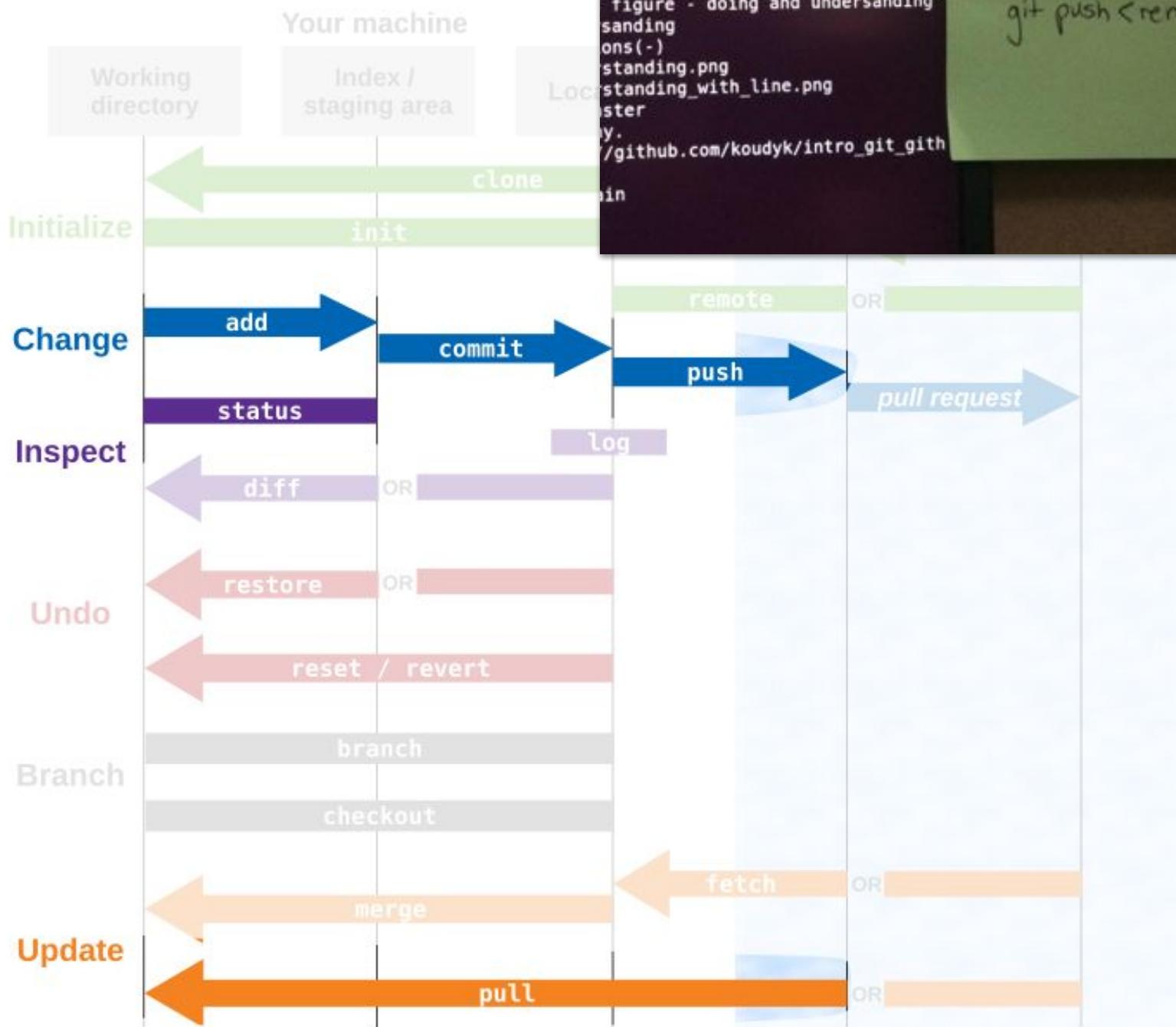
The commands covered in part 1



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



The commands you might want to remember



```
git status  
git add <file>  
git commit -m "<message>"  
git push <remote> <branch>
```

(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 (simple / linear collaboration)**
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating GitHub
- Try it out!

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

1. Modify

Change a file in your working tree

2. Stage

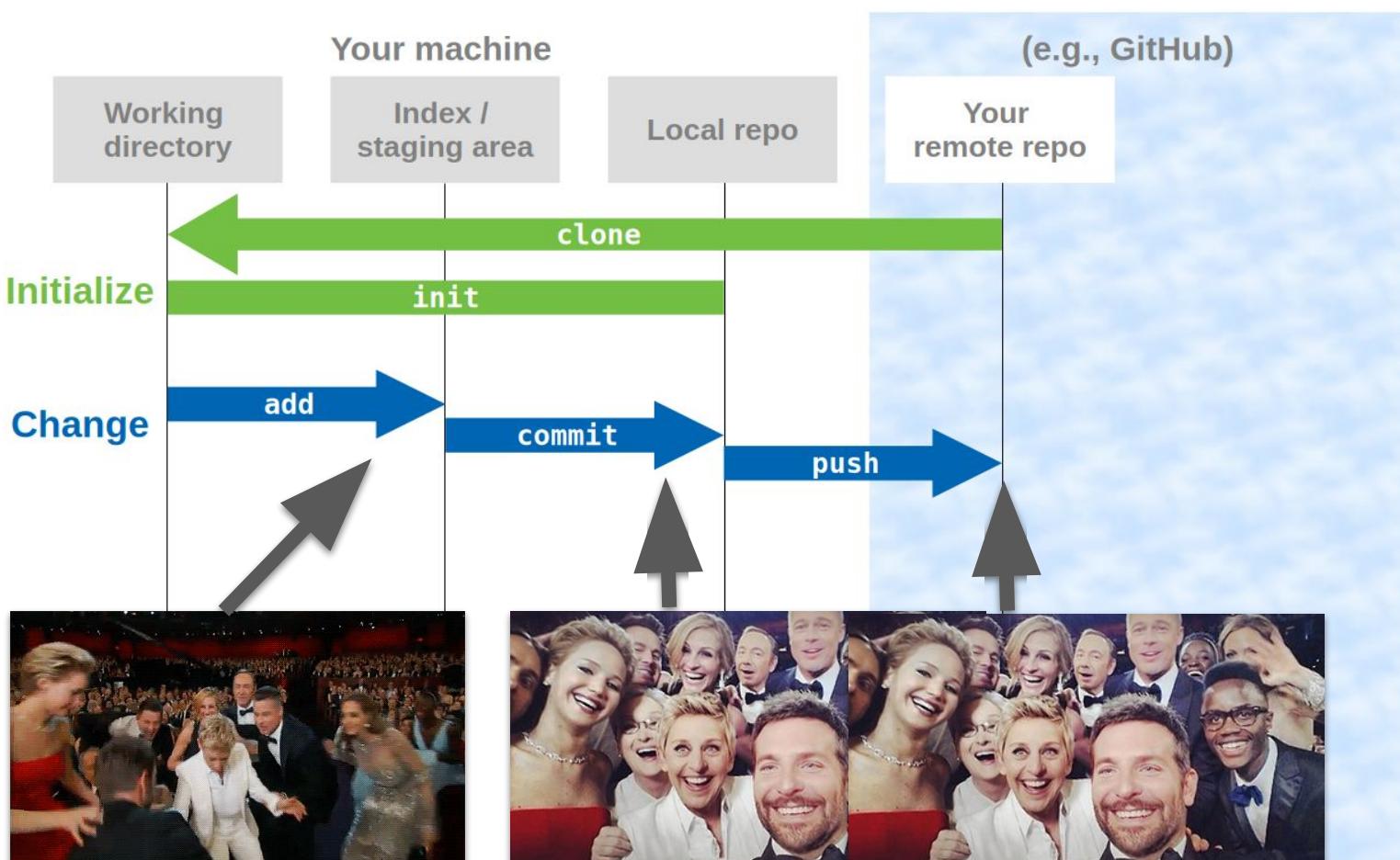
```
git add <filename>
```

3. Commit

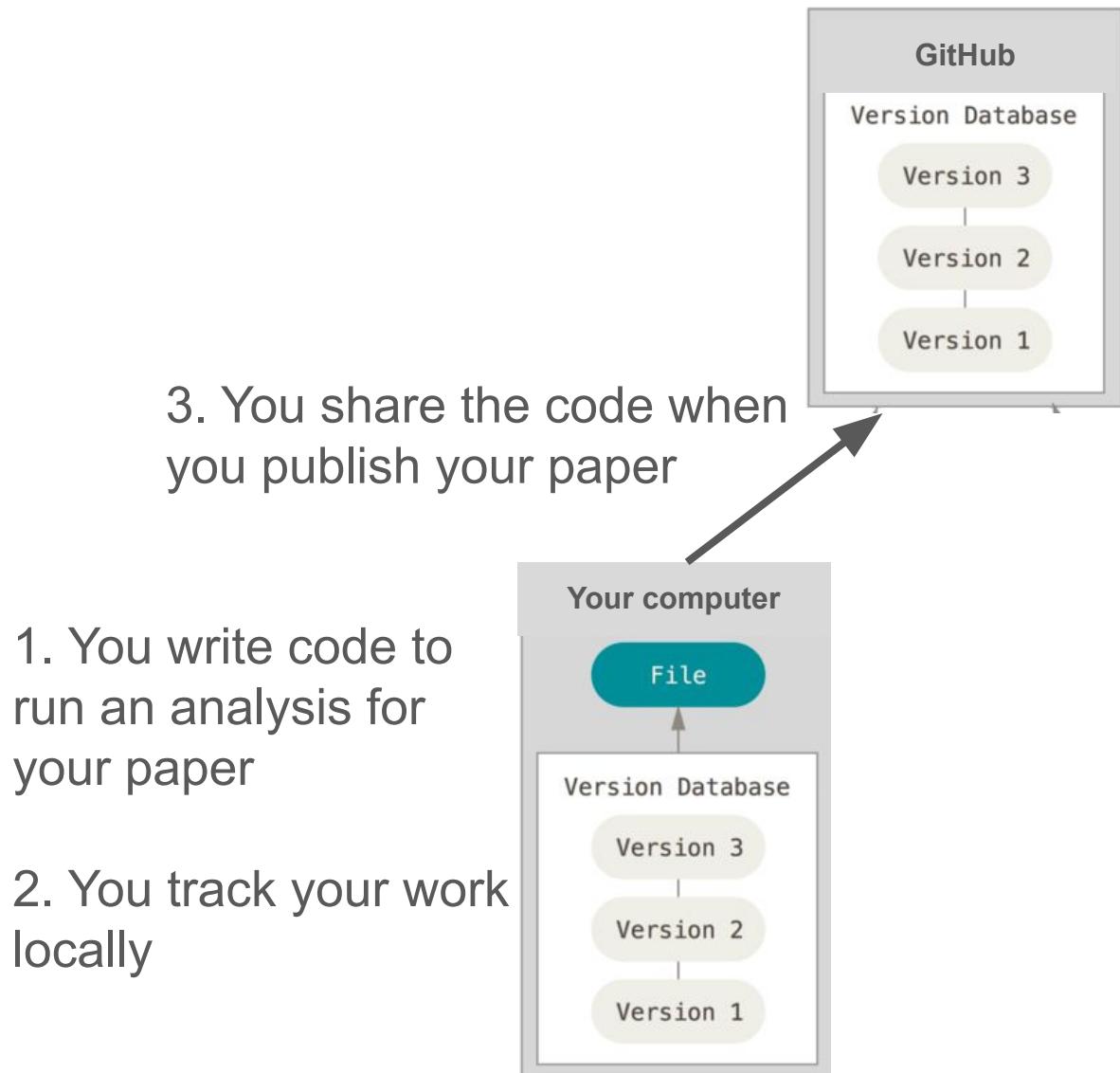
```
git commit -m "<short, informative commit message>"
```

4. Push

```
git push <remote> <branch>
```

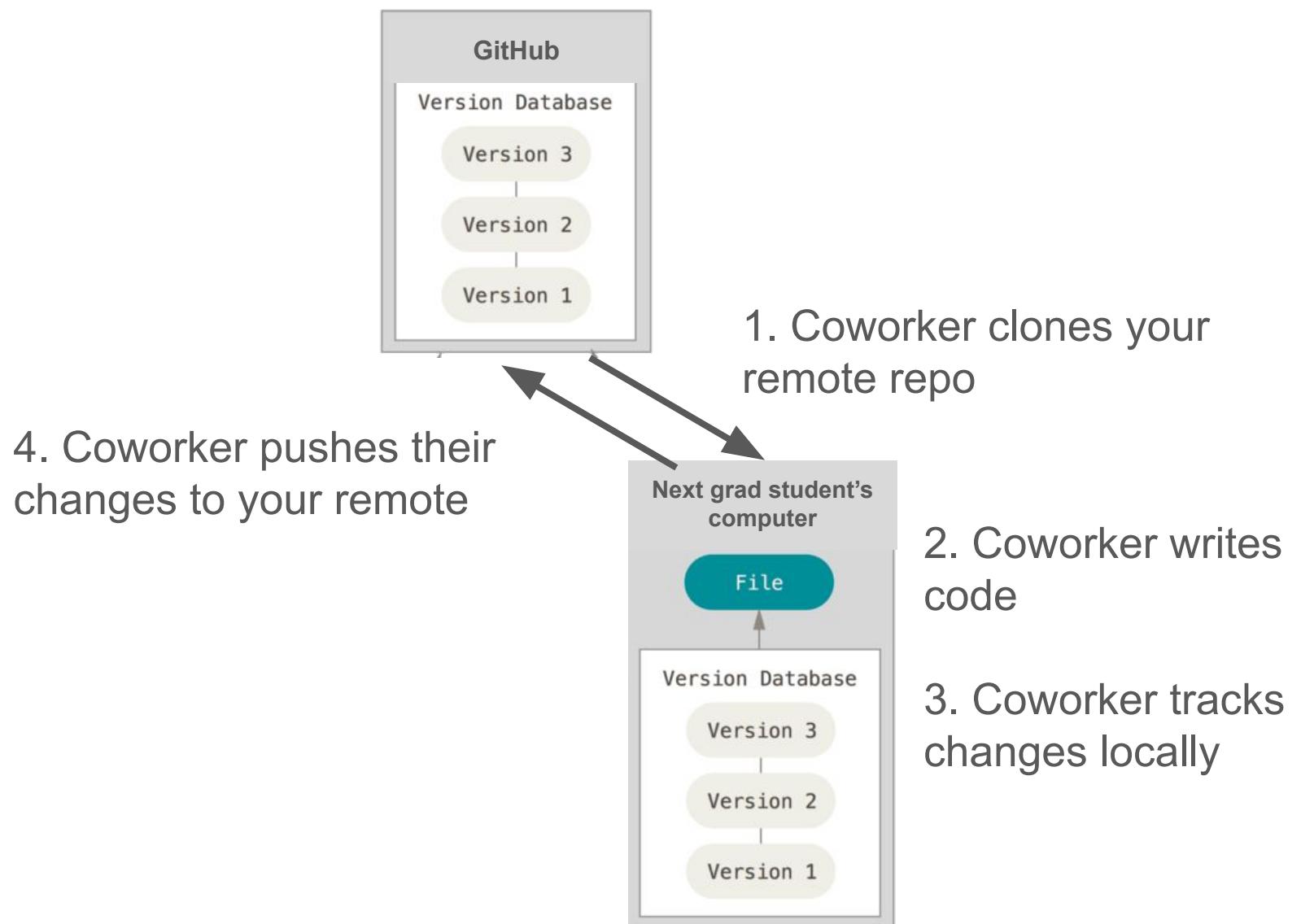


Sharing your work

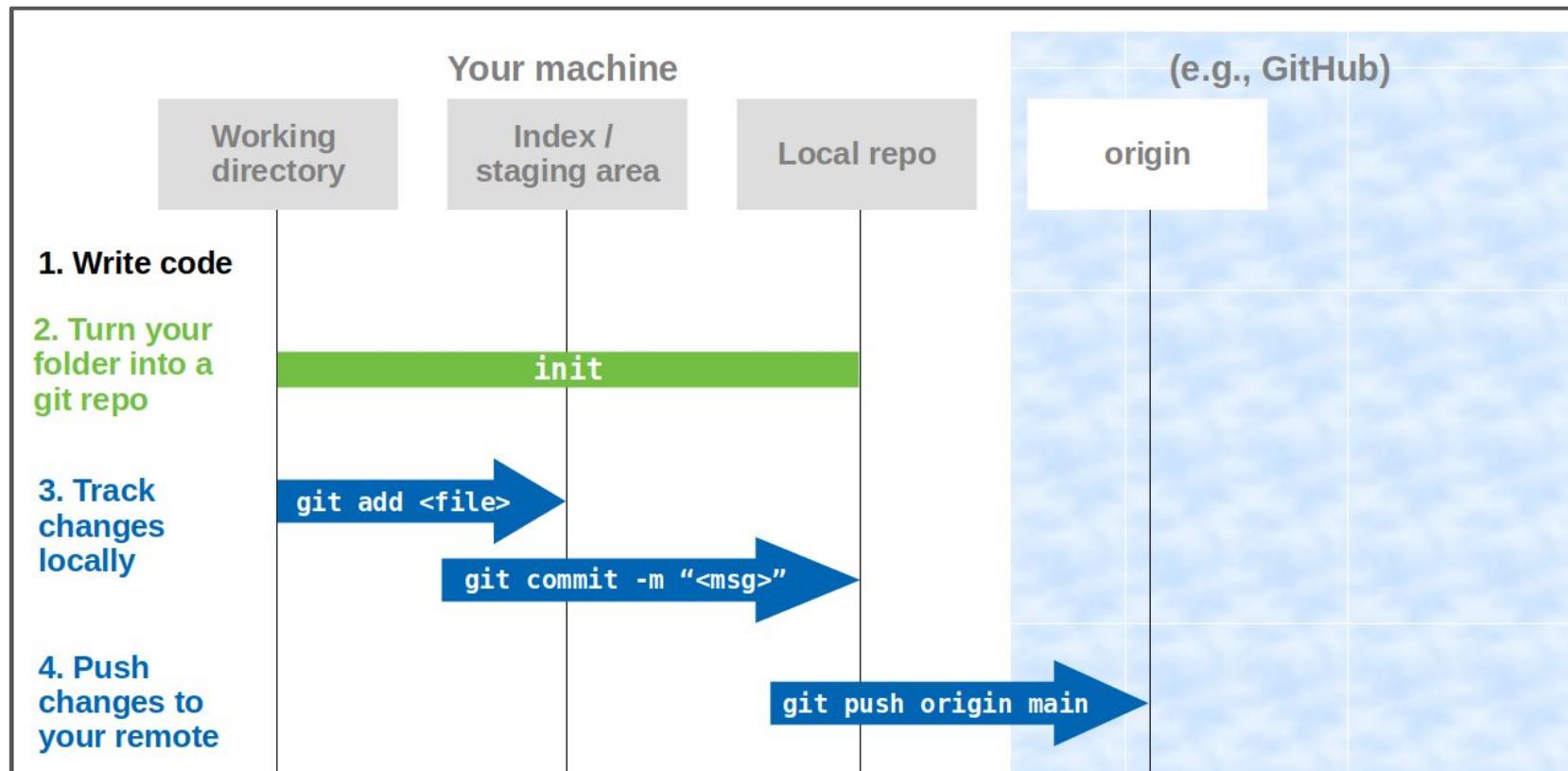
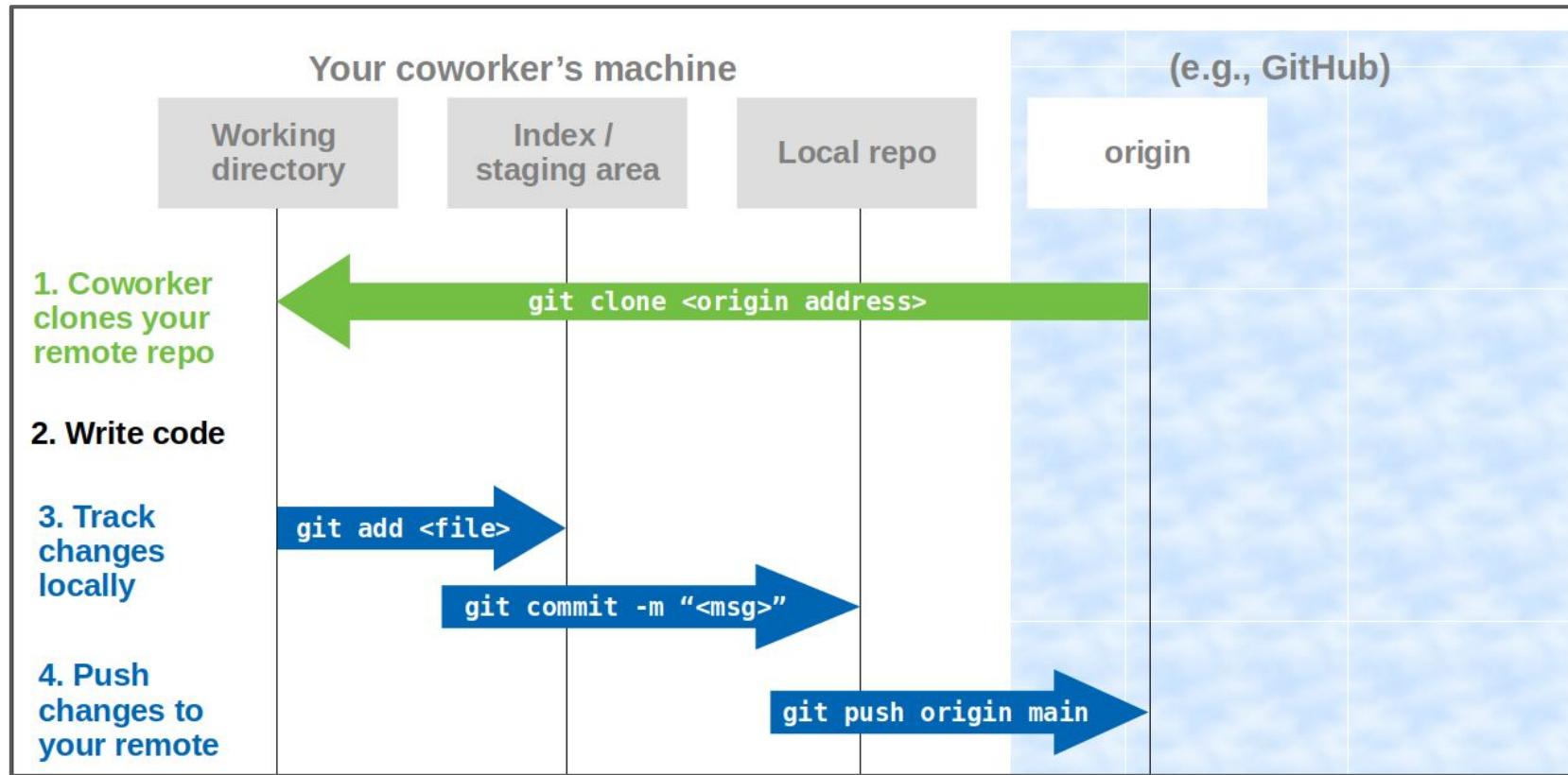


Sharing your work

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



Sharing your work



Git commands for sharing your work

- Show your remote repos

```
git remote -v
```

- Push commits to a remote repo

```
git push <remote name> <branch>
```

Goals

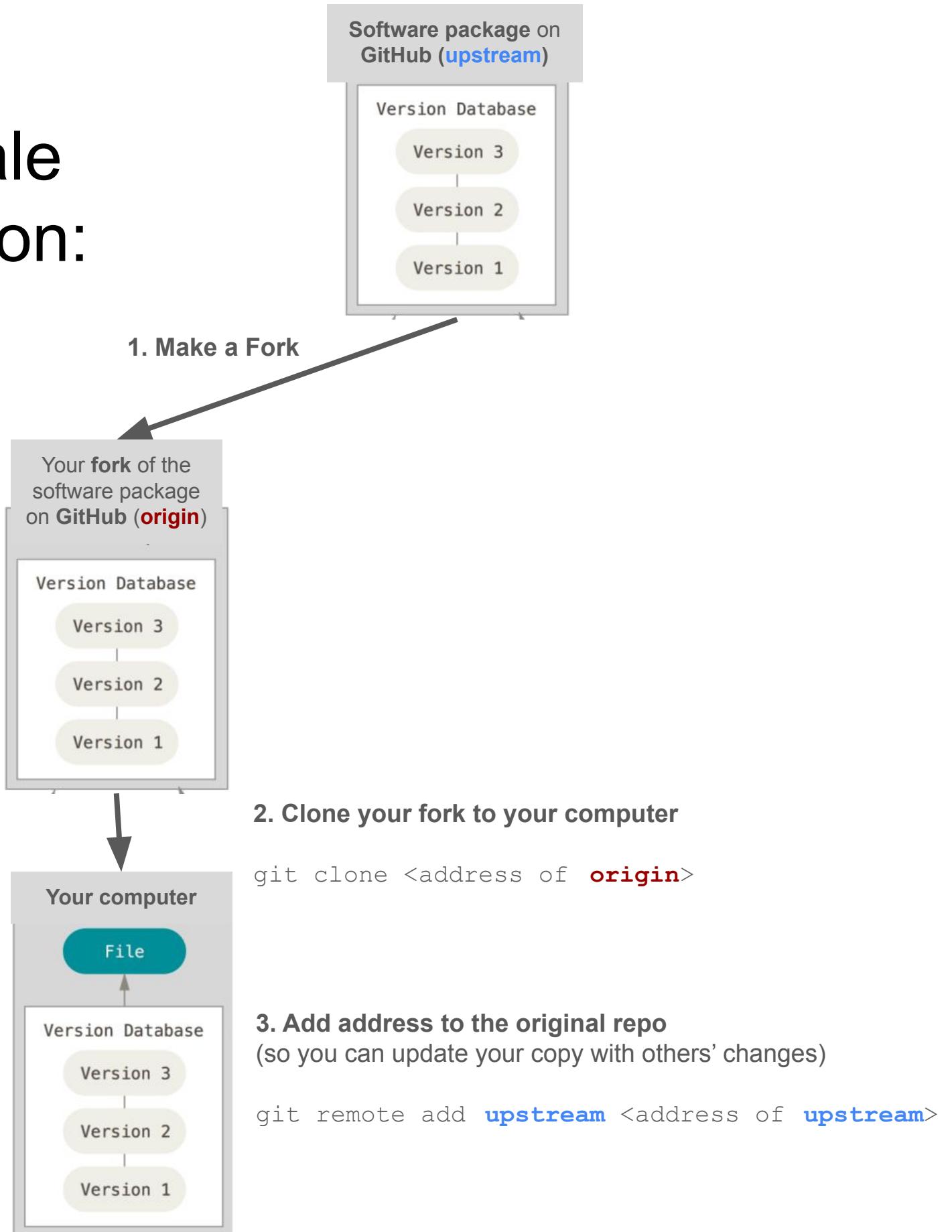
Part 1 (last week)

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

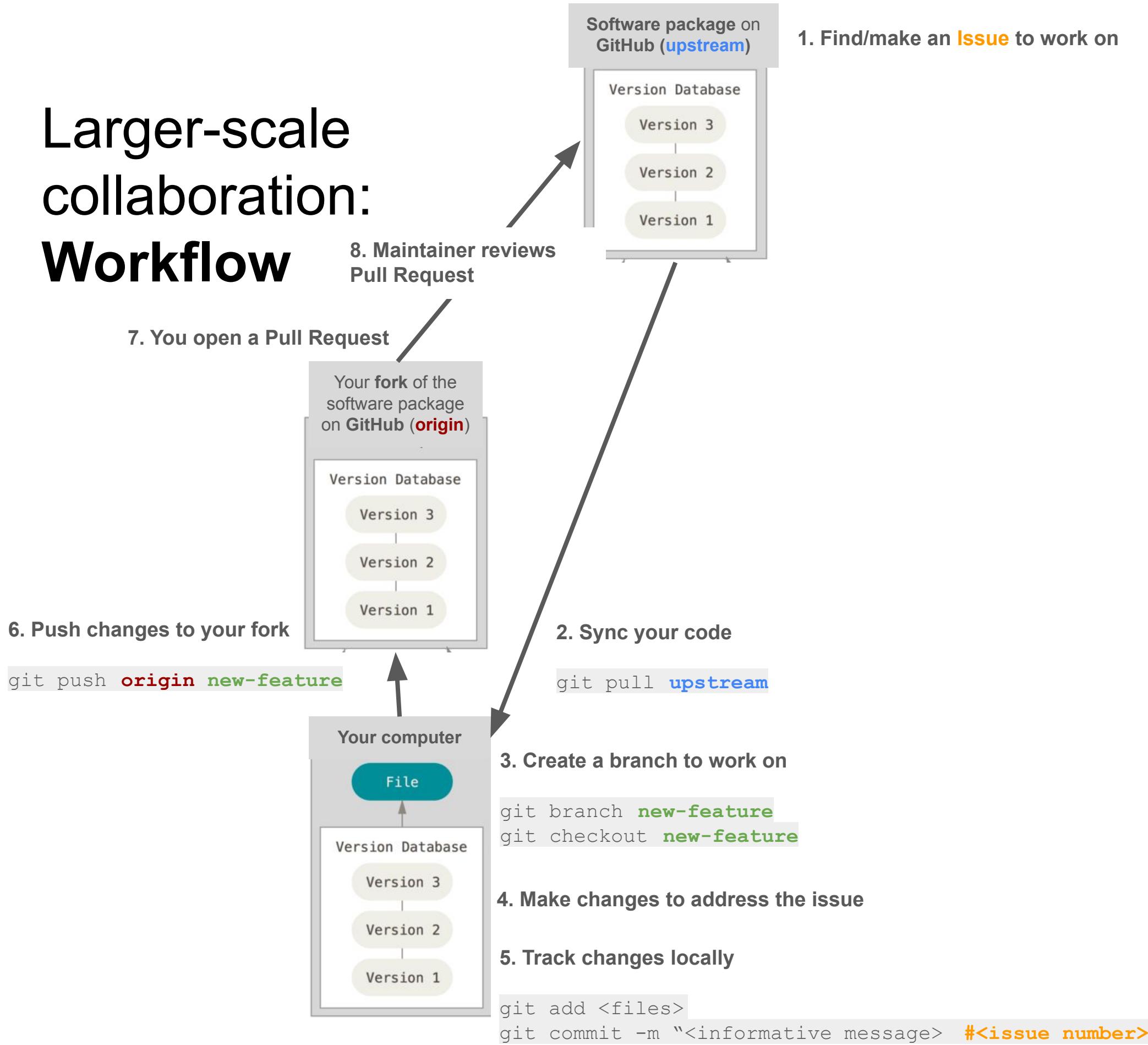
Part 2

- Share your work on GitHub (simple / linear collaboration)
- **Collaborate through GitHub (complex / nonlinear collaboration)**
- Navigating GitHub
- Try it out!

Larger-scale collaboration: Setup

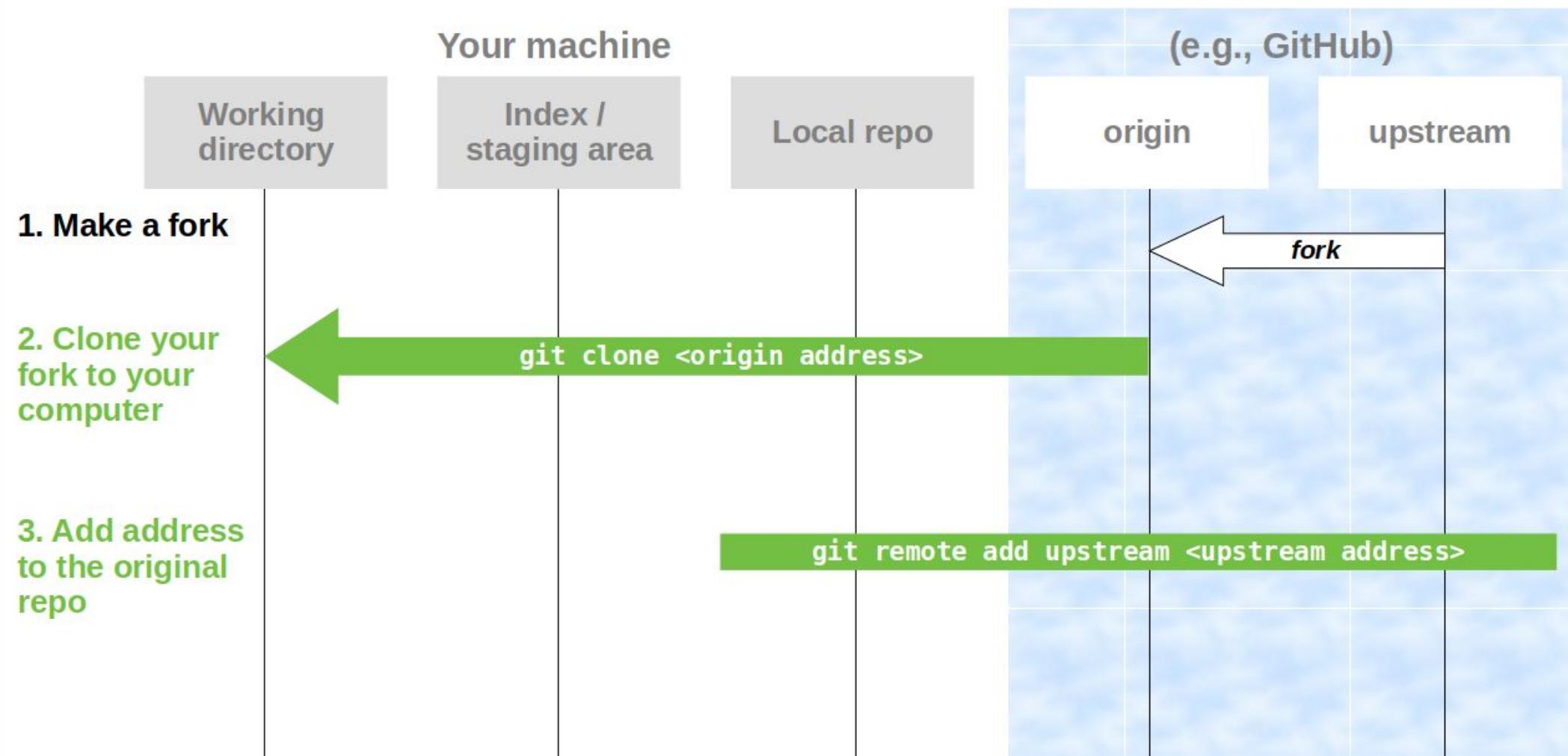


Larger-scale collaboration: Workflow



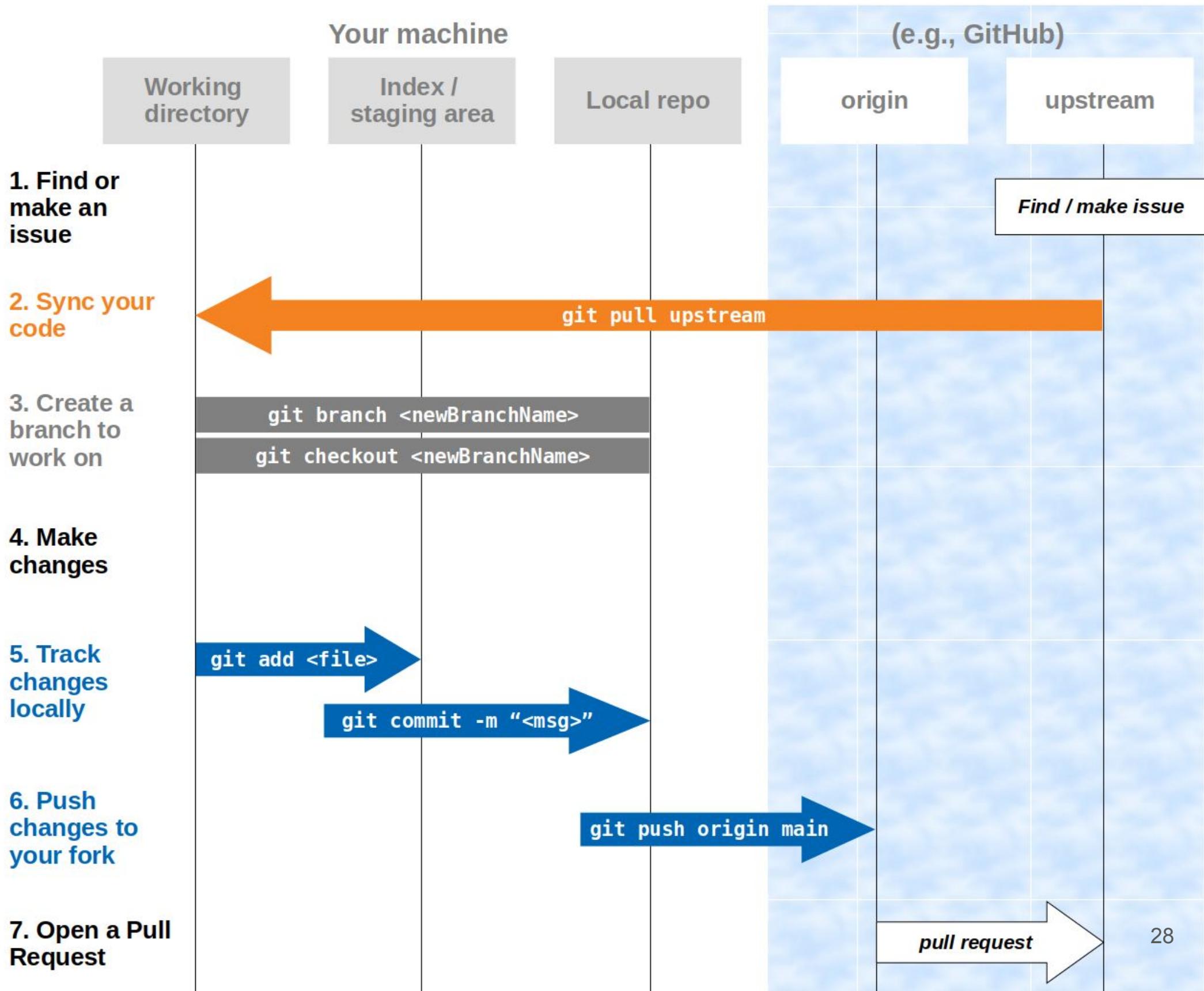
Another way to visualize this

Larger-scale collaboration: Setup

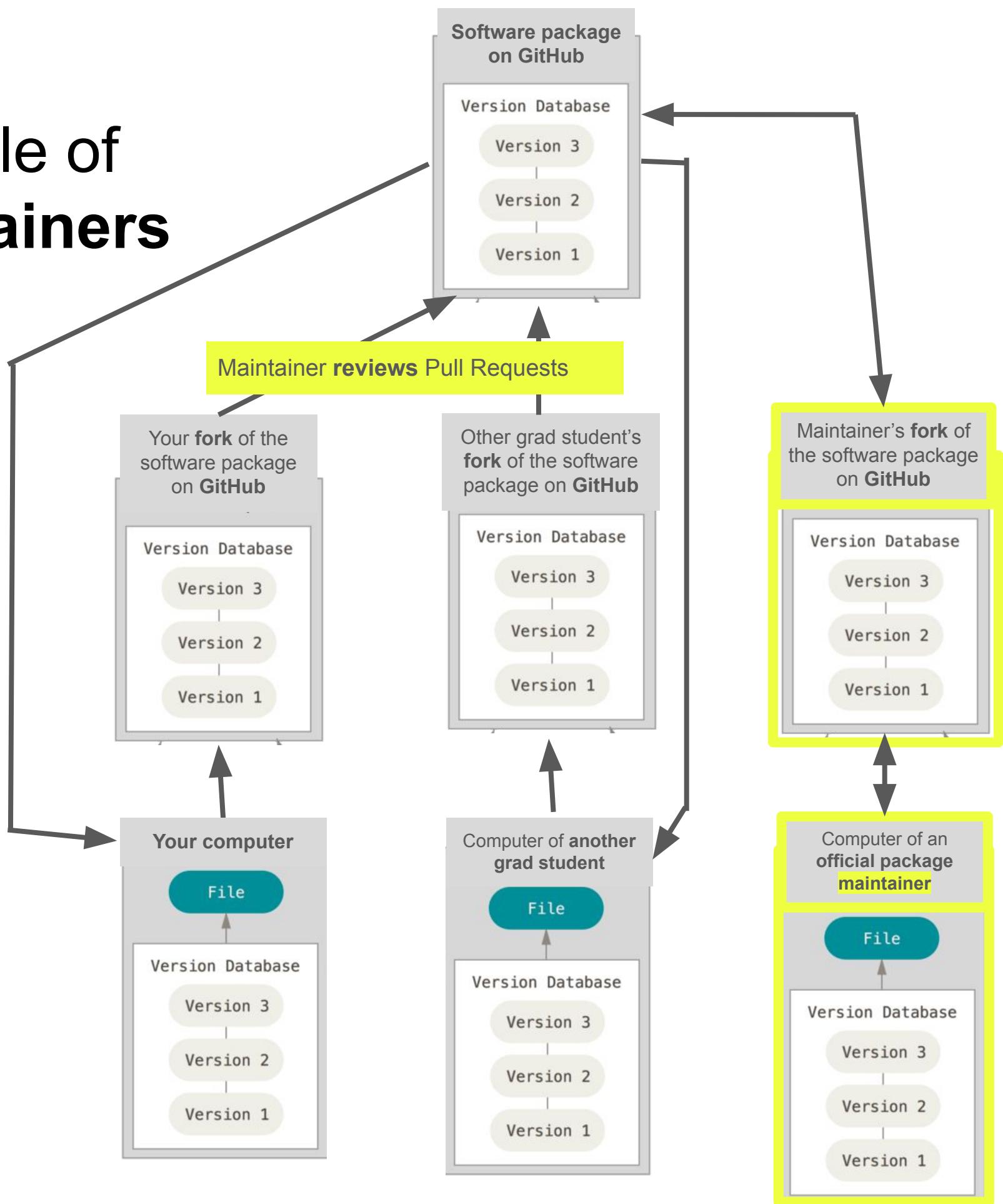


Another way to visualize this

Larger-scale collaboration: Workflow



The role of maintainers



Git commands for collaboration

- Show your remote repos

```
git remote -v
```

- Add a remote repo

```
git remote add <remote name> <remote address>
```

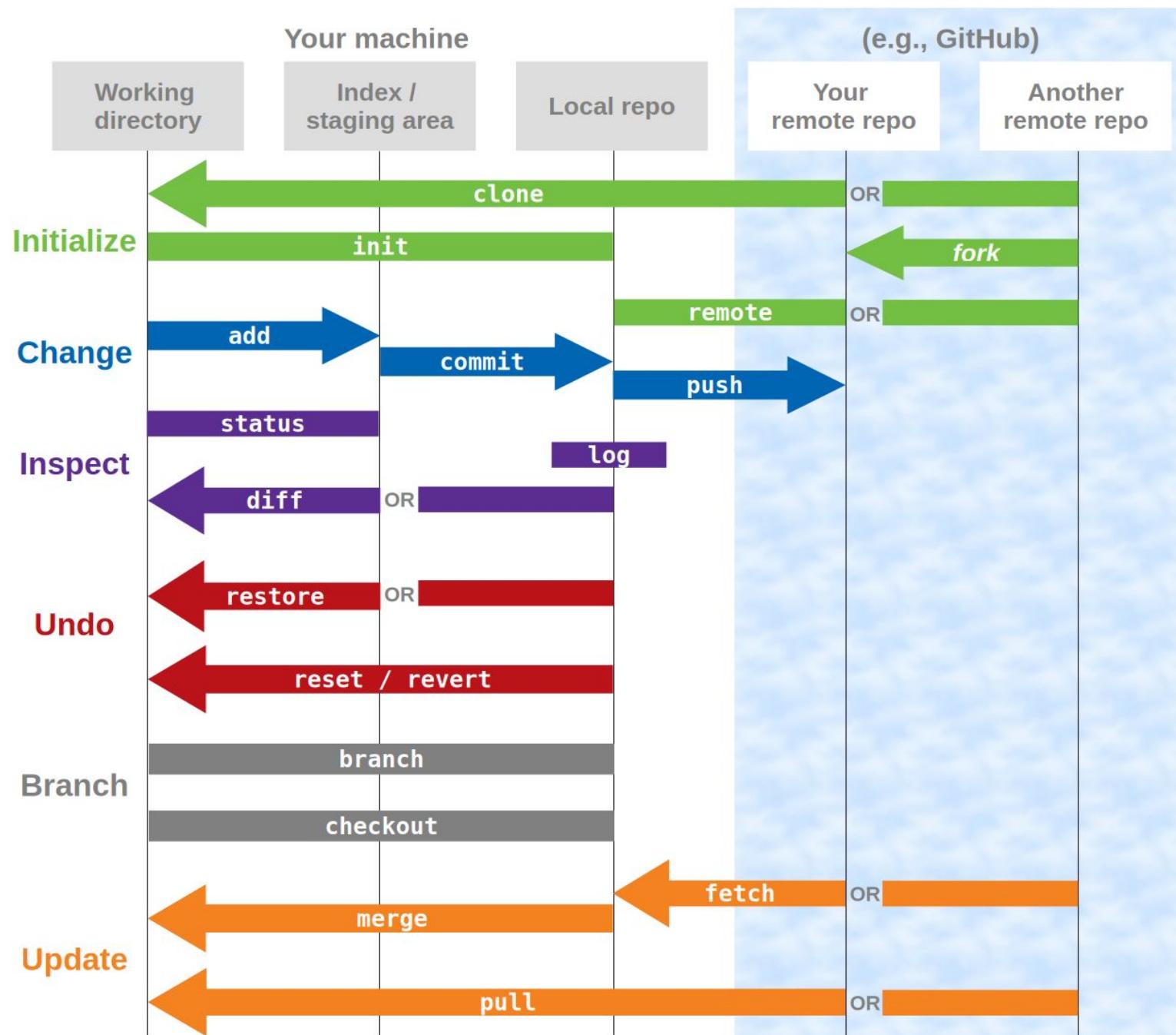
- Push commits to a remote repo

```
git push <remote name> <branch>
```

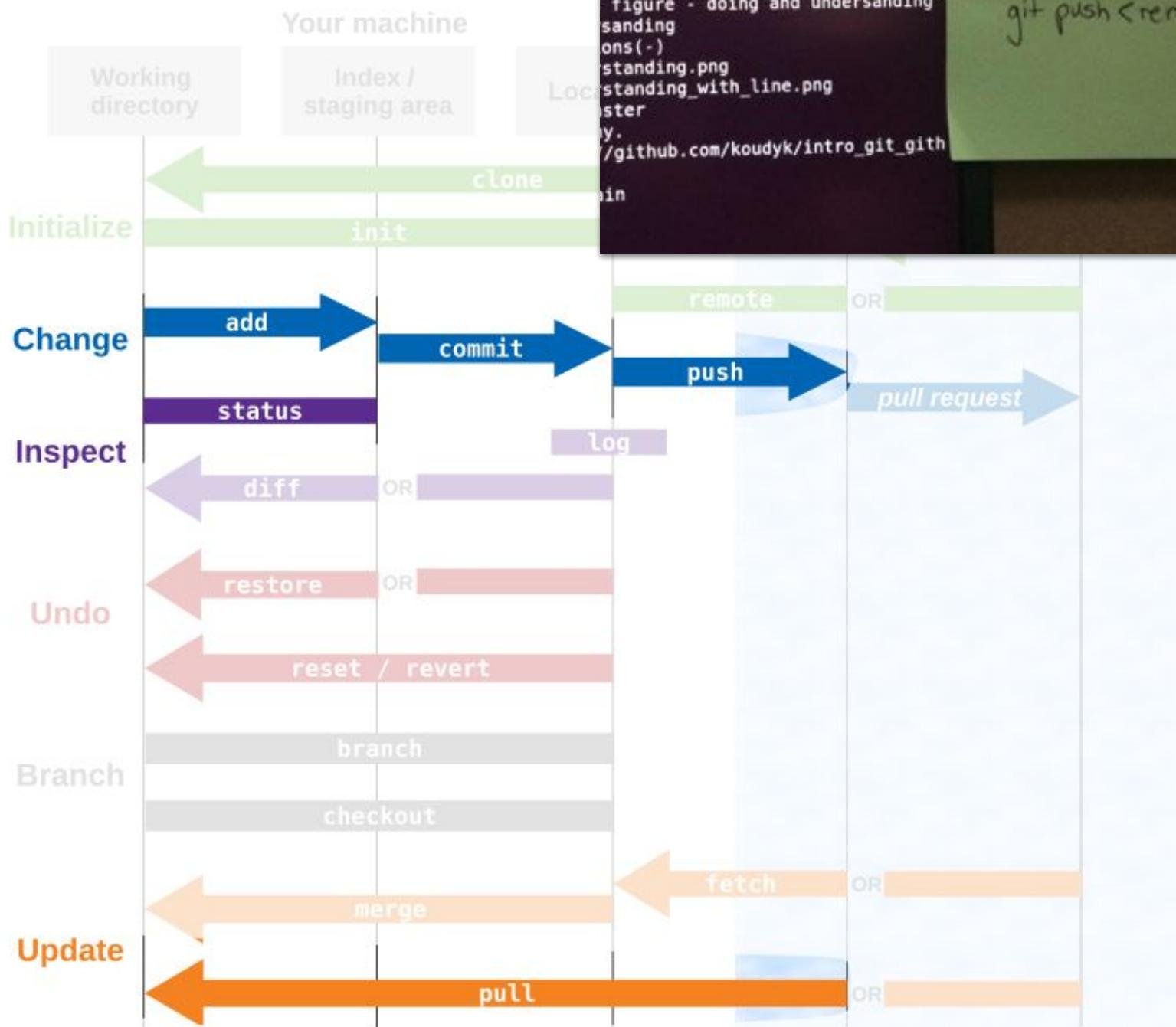
- Pull (fetch and merge) commits from a remote repo

```
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 (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- **Navigating GitHub**
- Try it out!

Navigating GitHub

The screenshot shows a GitHub repository page for 'git-sandbox'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'git-sandbox' is displayed, along with its status as 'Public'. On the right side, there are buttons for Edit Pins, Watch, Fork, and Star. A dropdown menu titled 'Code' is open, showing options for Local and Codespaces. The 'Clone' section is highlighted with orange boxes around the 'HTTPS' tab and the copy icon next to the URL. The URL itself, 'https://github.com/ubc-library-rc/git-sandbox', is also highlighted with an orange box. To the right of the code dropdown, there's an 'About' section with various repository statistics.

No description, website, or topics provided.

Readme
MIT license
Activity
Custom properties
0 stars
0 watching
0 forks

Where to find address for cloning a repo to your computer

Navigating GitHub

The screenshot shows the GitHub interface for a repository named 'git-sandbox'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'git-sandbox' is displayed along with its status as 'Public'. On the right side of the header, there are buttons for Edit Pins, Watch, Fork (which is highlighted with an orange box), Star, and more. The main content area shows a 'Code' tab selected, with a dropdown menu showing 'main'. There are also buttons for Go to file, +, and About.

Forks

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

Issues - where you can

Navigating GitHub

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

The screenshot shows the GitHub Issues page for the repository "neurobagel / query-tool". The "Issues" tab is selected and highlighted with an orange border. The search bar at the top contains the query "is:issue state:open". Below the search bar, there are filters for "Open" (40), "Closed" (116), "Author", "Labels", "Projects", "Milestones", "Assignees", "Types", and sorting by "Newest".

The main list displays ten issues:

- Update the internal imaging modality list to match the CLI (Enhancement, #636, opened 5 days ago)
- Show full name of imaging modality tag on hover (Enhancement, #635, opened 5 days ago)
- Split fetching results into preview and download (Enhancement, #614, opened on Aug 27)
- Consider committing env.dev and env.testing to git (flag:stale, #589, opened on Jul 24)
- version tag update inside package.json on release is not working (flag:stale, flag:detail needed, flag:discuss, flag:schedule, process, Bug, #588, opened on Jul 24)
- Add timestamp or unique ID to filename of each results file (Enhancement, #568, opened on Jun 2)
- Query results download options should be clickable (Enhancement, #567, opened on Jun 2)
- Add info tooltip to explain node selection (flag:stale, documentation, usability, Enhancement, #508, opened on Mar 14)

Navigating GitHub

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

The screenshot shows the GitHub interface for the repository 'neurobagel/query-tool'. The 'Pull requests' tab is active, indicated by an orange box around it. There is 1 open pull request. The first pull request, '#637 [ENH] Add newly supported imaging types', is shown in detail. It was opened 5 days ago and has 1 review. The commit message includes a link to issue #636 and labels 'main' and 'pr-minor'. The pull request is approved.

Pull requests

- Push commits from your own branch 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

Navigating GitHub

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

The screenshot shows a GitHub pull request page for issue #637. The title is "[ENH] Add newly supported imaging types". The pull request has 14 additions and 2 deletions. It has 6 conversations, 2 commits, 14 checks, and 1 file changed. A comment from user surchs is shown, mentioning a closing PR and changes proposed. The checklist section lists several items, some of which are checked. The right sidebar displays review details, assignees, labels, projects, milestones, and development information.

[ENH] Add newly supported imaging types #637

Open surchs wants to merge 2 commits into [main](#) from [issue636](#)

Conversation 6 Commits 2 Checks 14 Files changed 1 +14 -2

Comments

surchs commented 5 days ago • edited by alyssadai

• Closes [Update the internal imaging modality list to match the CLI](#) #636

Changes proposed in this pull request:

- hard code the new imaging types we support, see also [Update BIDS imaging suffixes we support](#) bagel-cli#496

Checklist

This section is for the PR reviewer

- PR has an interpretable title with a prefix ([ENH], [FIX], [REF], [TST], [CI], [MNT], [INF], [MODEL], [DOC]) (see our [Contributing Guidelines](#) for more info)
- PR has a label for the release changelog or skip-release (to be applied by maintainers only)
- PR links to GitHub issue with mention closes #xxxx
- Tests pass
- Checks pass
- If the PR changes the participant-level and/or the dataset-level result file, the [query-tool-results files](#) in the [neurobagel_examples repo](#) have been regenerated

Reviewers

sourcery-ai[bot] alyssadai

Still in progress? Learn about draft PRs

Assignees

No one assigned

Labels

pr-minor

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close #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 (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating 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 include `bash` or `zsh`



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

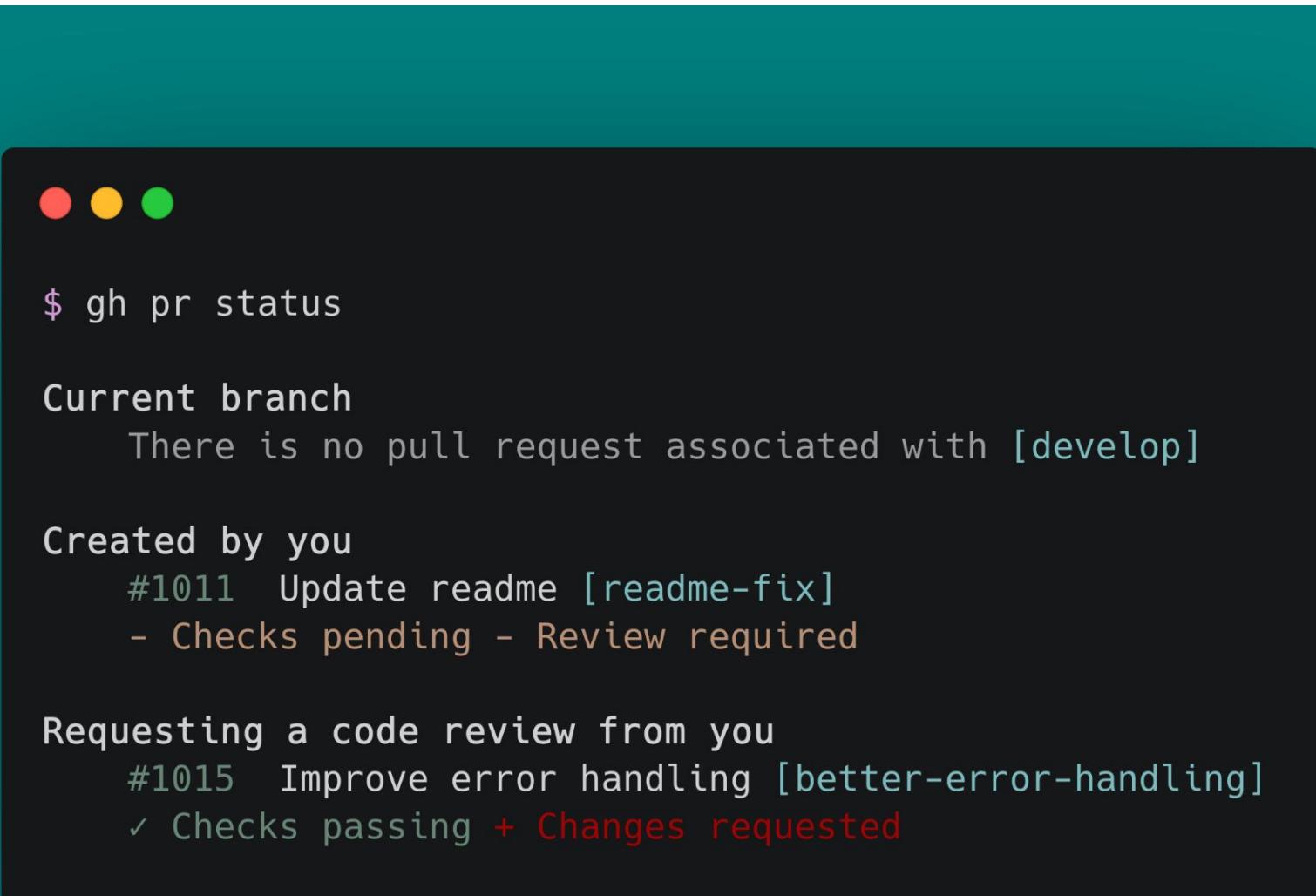


E) Can you go your GitHub account?

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



The screenshot shows a terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top. The terminal output is as follows:

```
$ gh pr status

Current branch
  There is no pull request associated with [develop]

Created by you
  #1011 Update readme [readme-fix]
    - Checks pending - Review required

Requesting a code review from you
  #1015 Improve error handling [better-error-handling]
    ✓ Checks passing + Changes requested
```

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

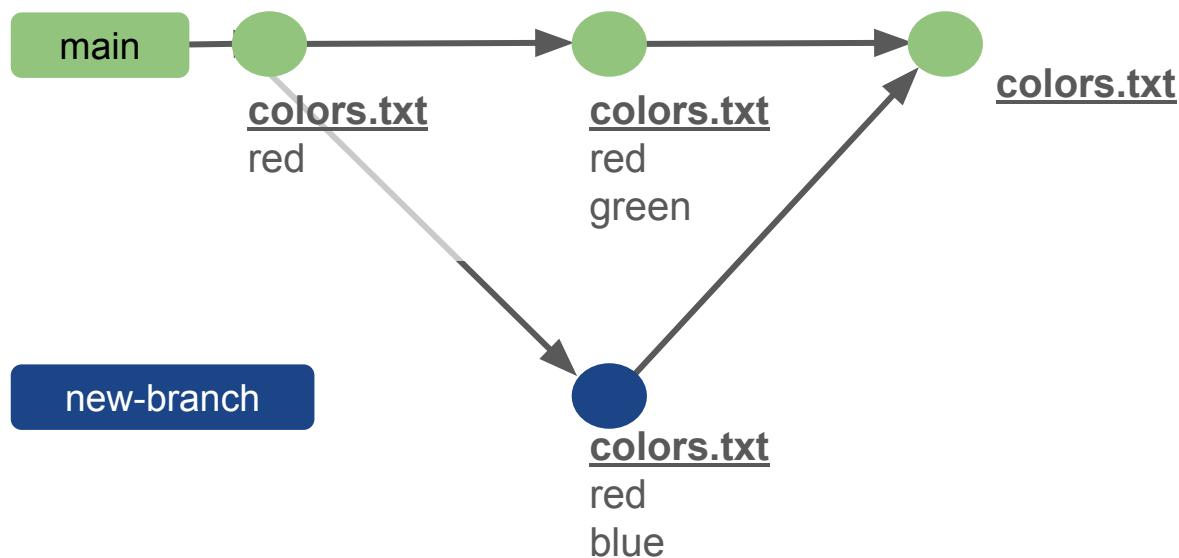
```
git log --pretty=format:"%h - %an, %ar : %s" --graph
```

- “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
```



What do you think you can do now?

(select as many as you want)

- A) Explain why git and GitHub are useful
- B) Track your own work with git
- C) Share your own work on GitHub
- D) Collaborate on GitHub

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