

Introduction to Git and GitHub

Part 2

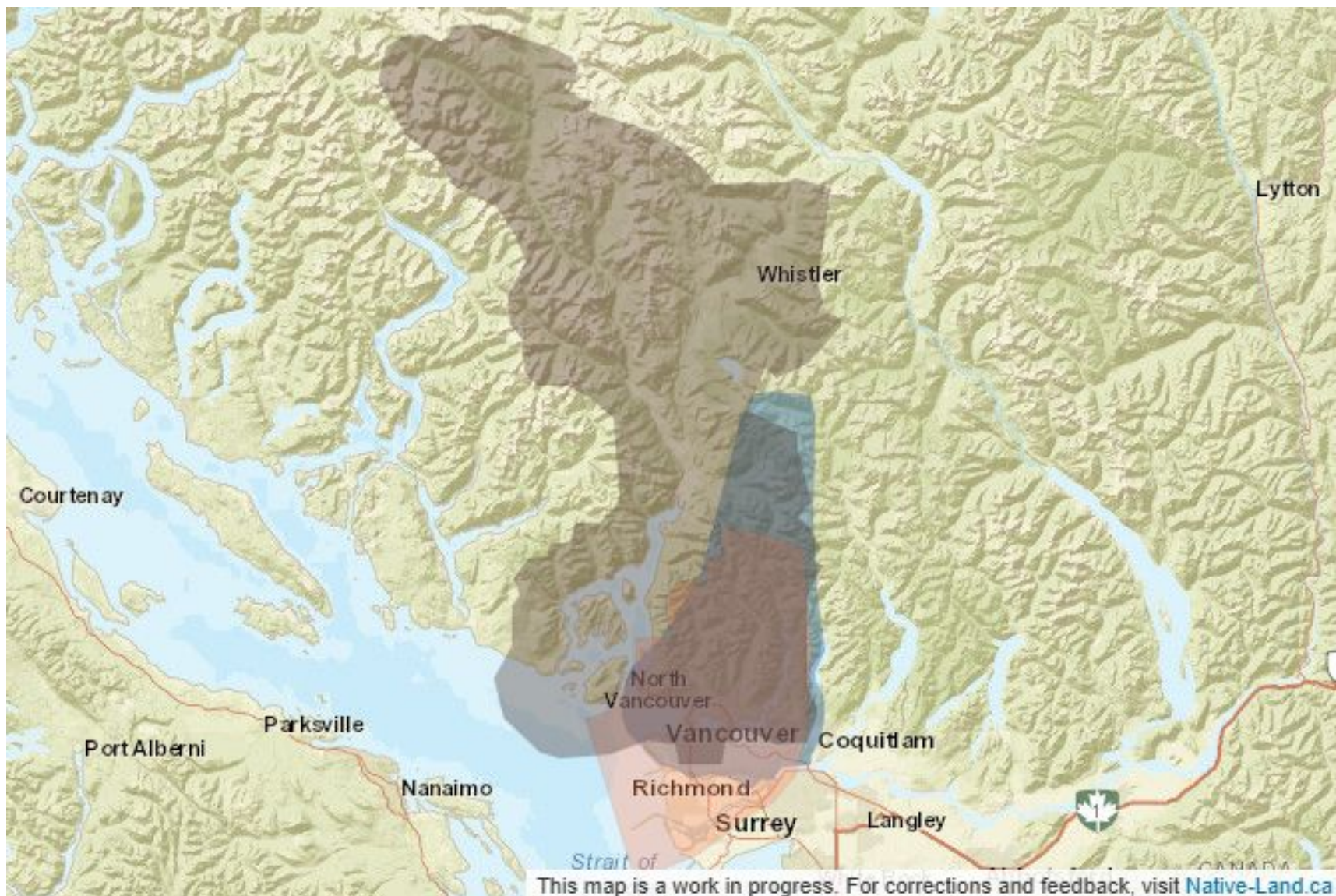
Kendra Oudyk (she/her)

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 x^wməθk^wəy̓əm (Musqueam), səlilwətał (Tsleil-Waututh), and Skwxwú7mesh-ulh Temíxw (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/>

Introduction to Git and GitHub

🔍 Search 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

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**
 - **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

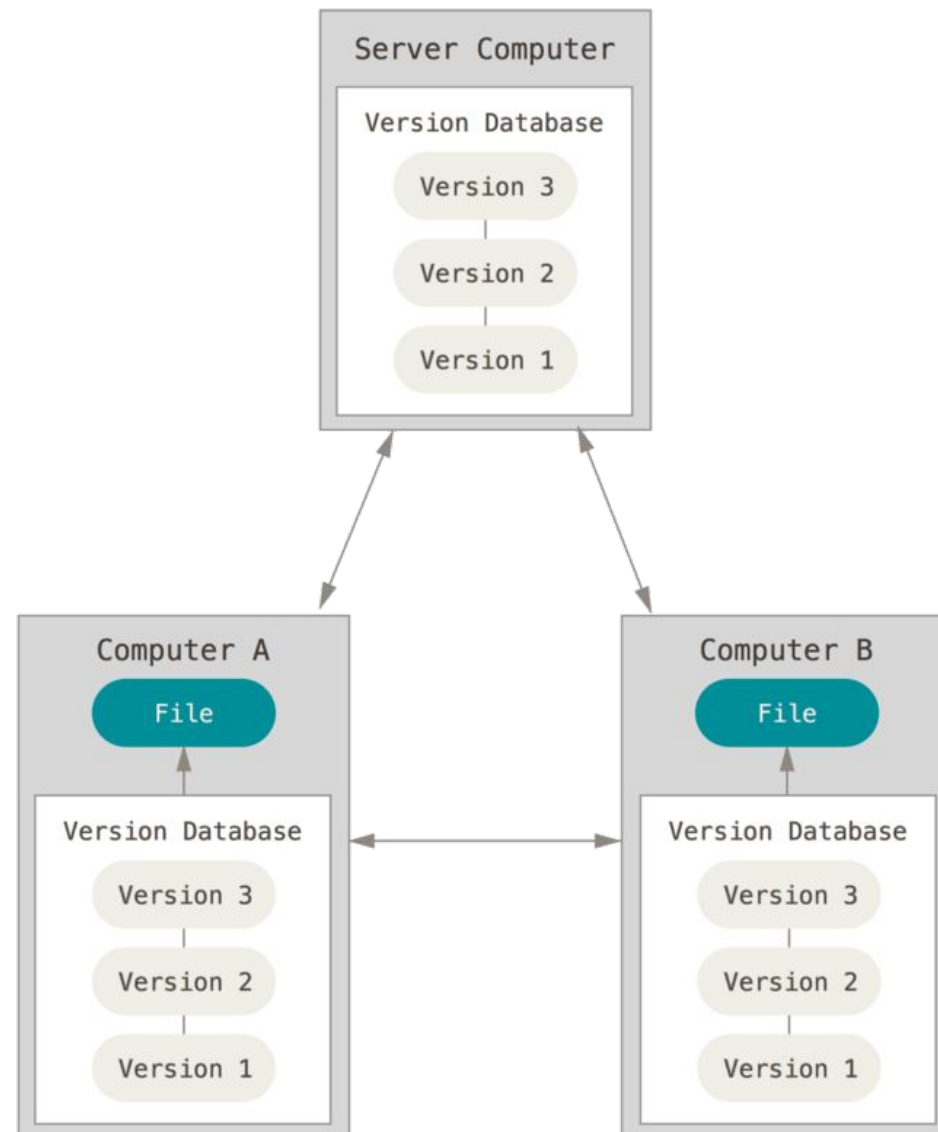
Part 1 (last week)

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

Part 2

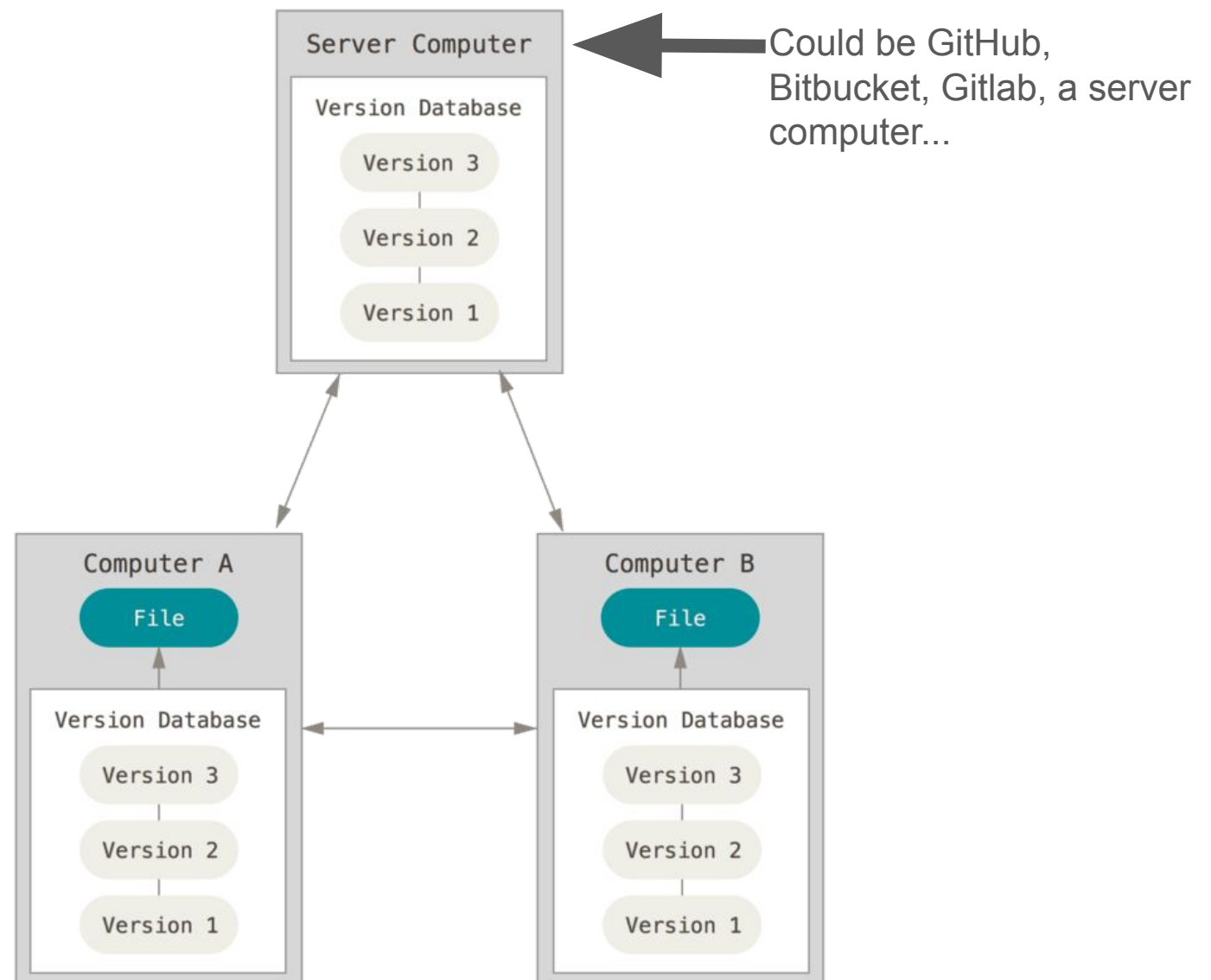
- Share your work on GitHub
- Collaborate on GitHub
- GitHub features
- Try it out!

Distributed version control

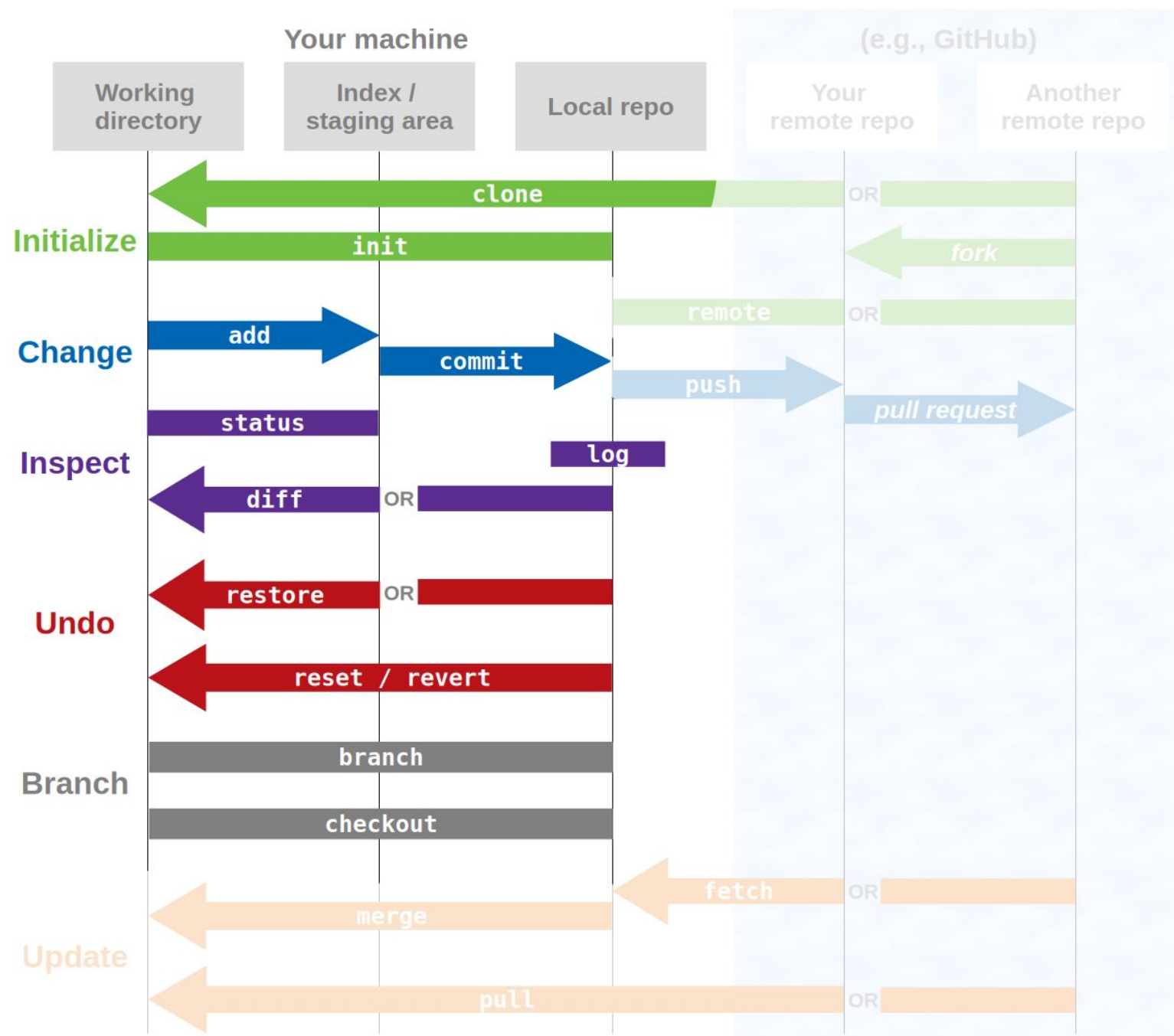


Git vs. GitHub

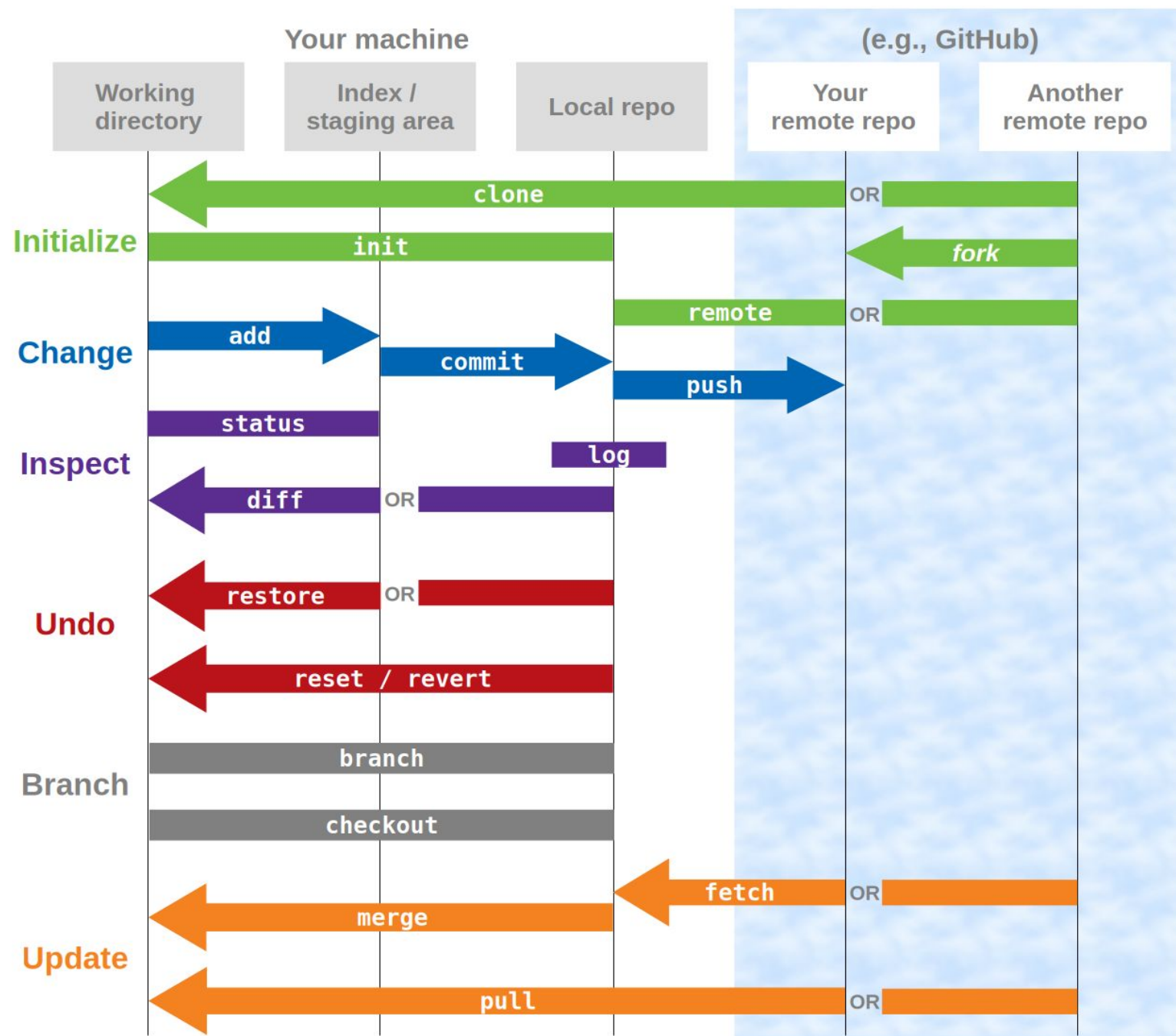
- Git is the "**language**" we use to do version control.
- GitHub **hosts** git repositories online.



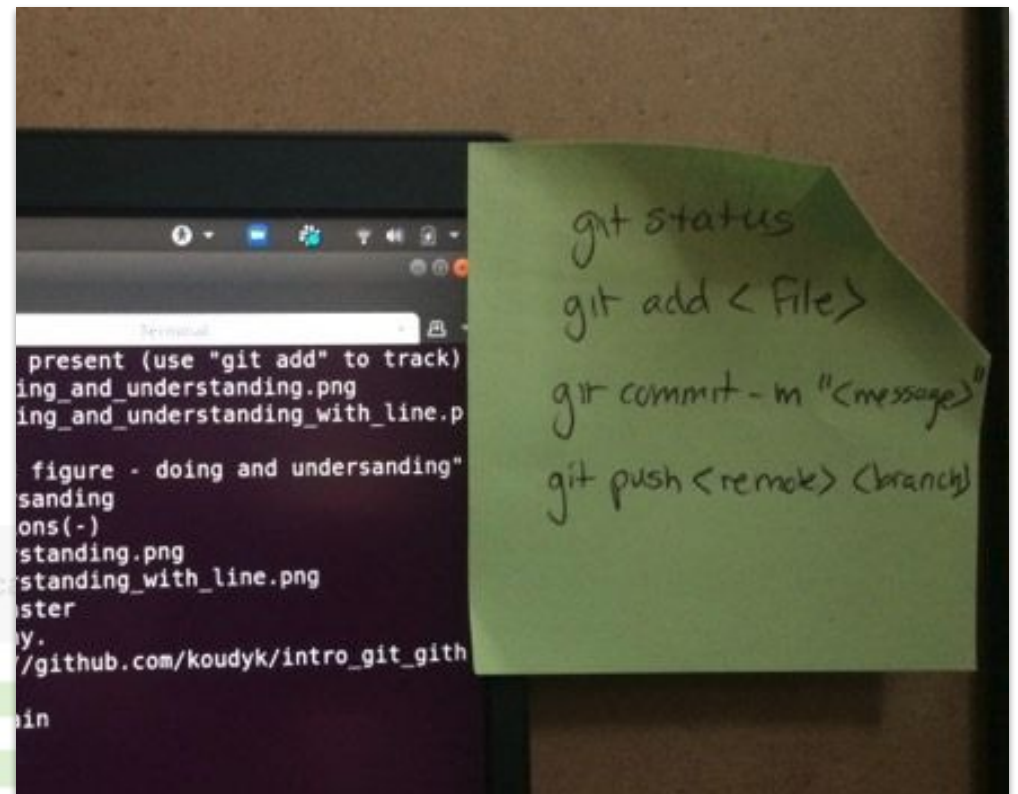
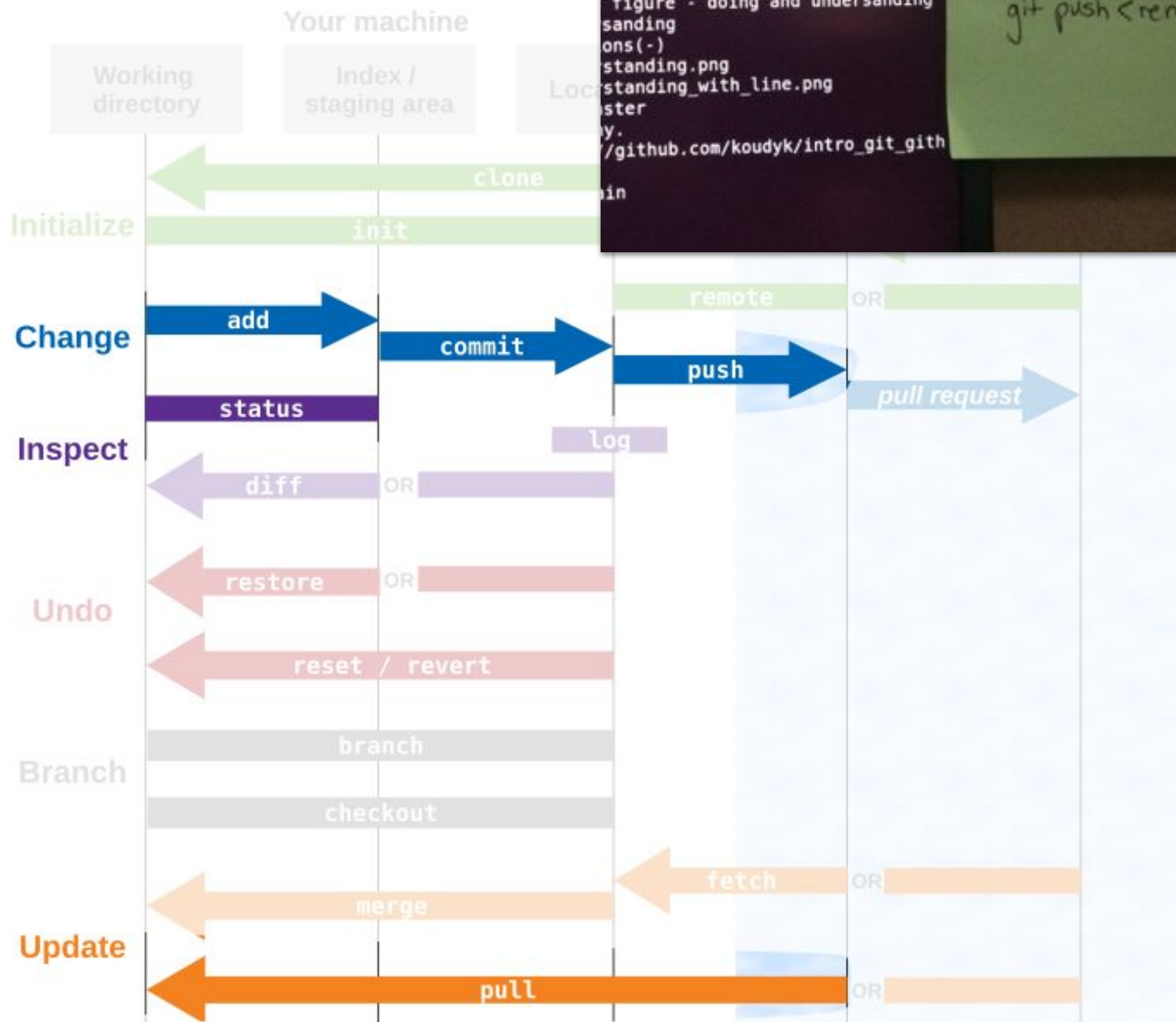
The commands we covered last week



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**

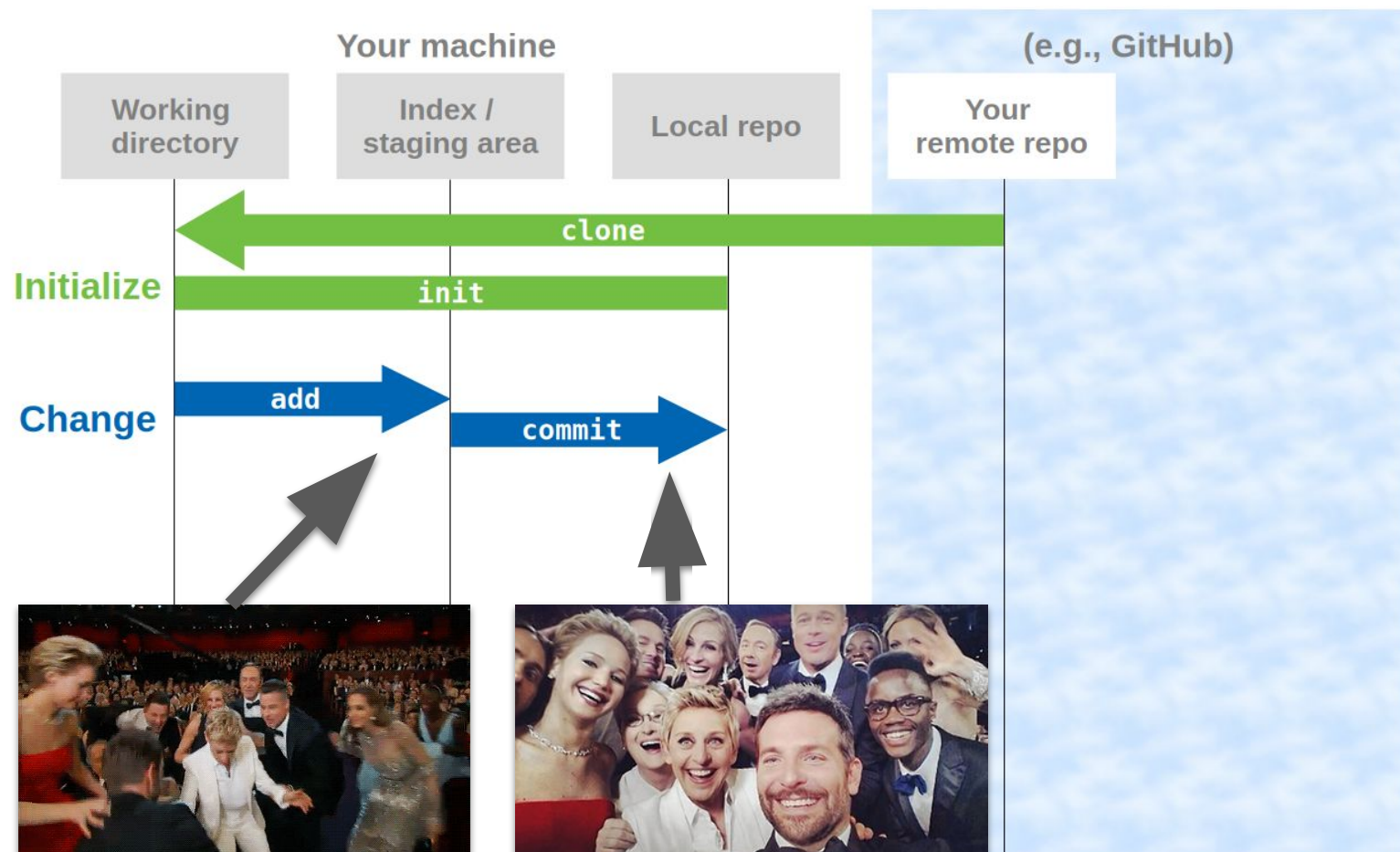
Change a file in your working tree

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

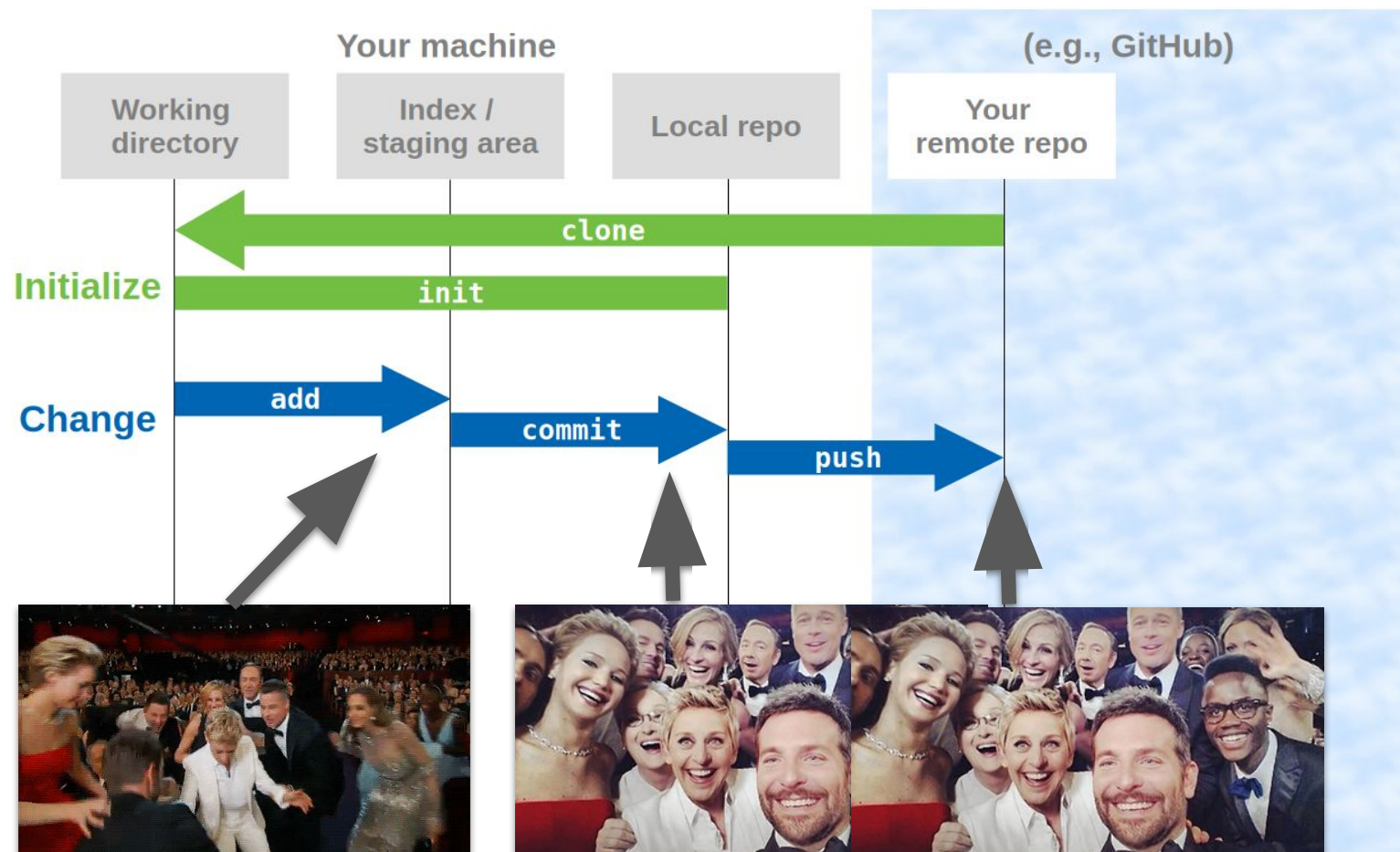
```
git add <filename>
```

3. Commit

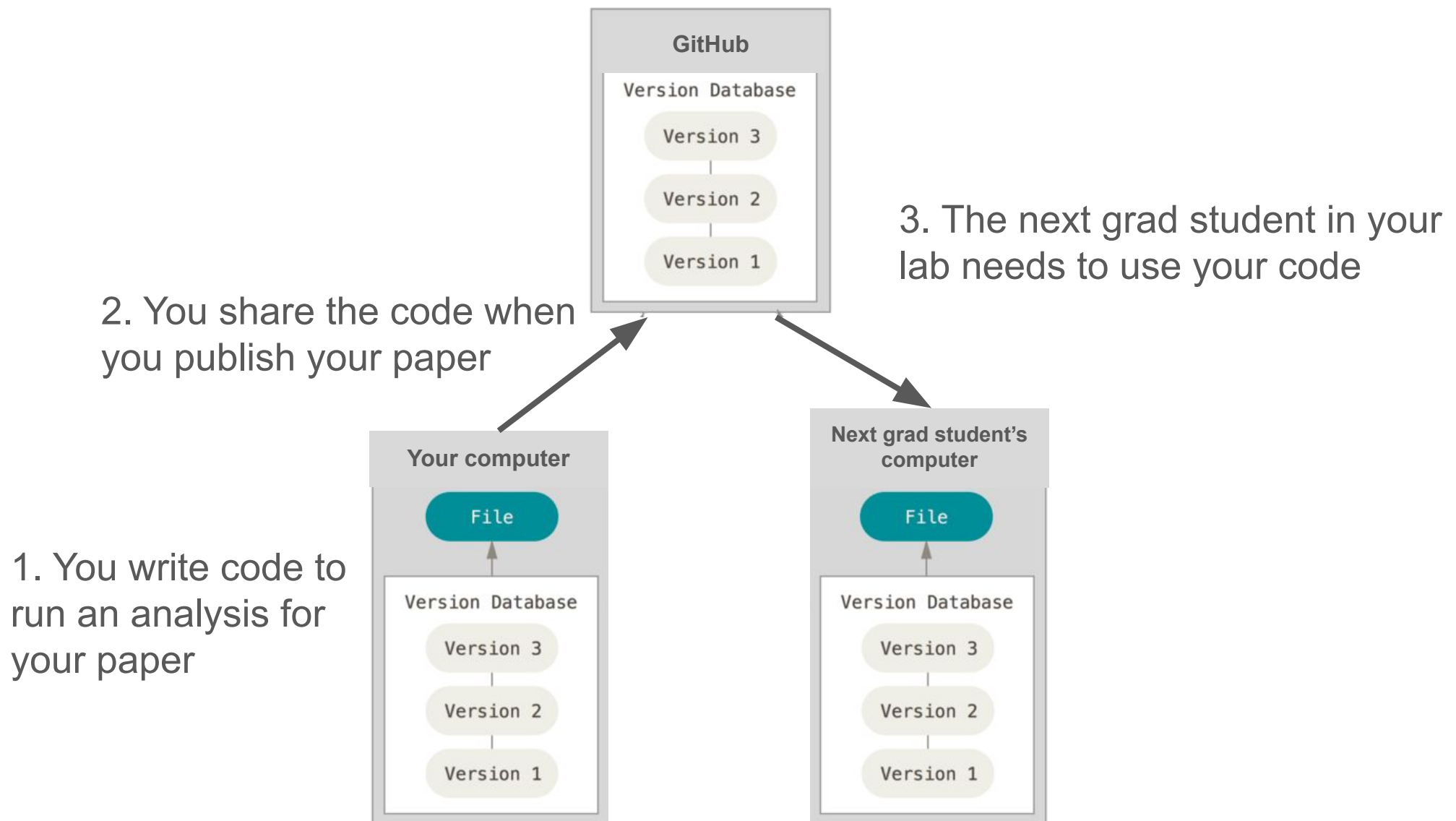
```
git commit -m "<short, informative commit message>"
```

4. Push

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



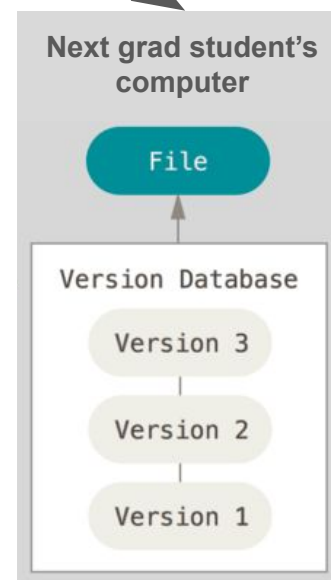
Sharing your work



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

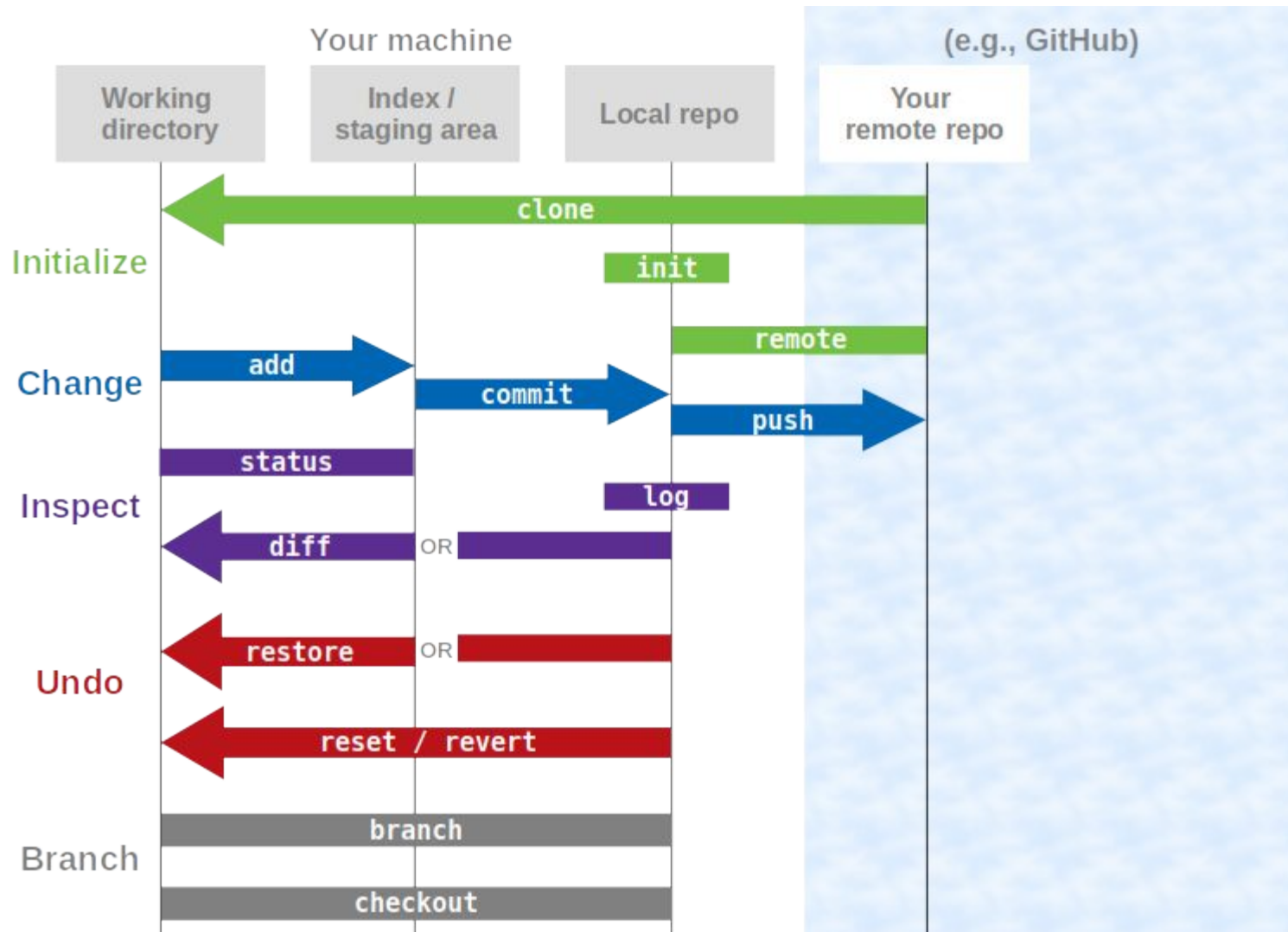
- Show your remote repos

```
git remote -v
```

- Push commits to a remote repo

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


Remotes - sharing your work



Goals

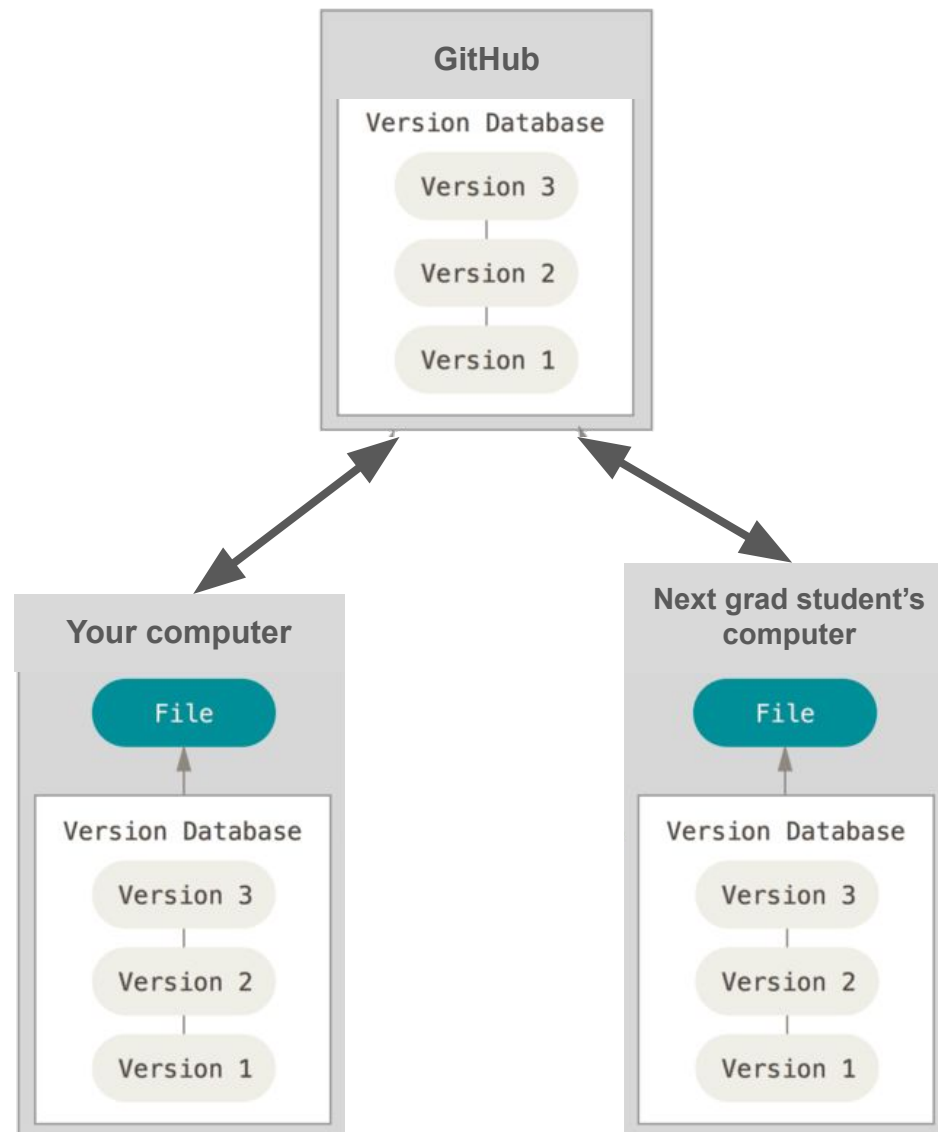
Part 1 (last week)

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

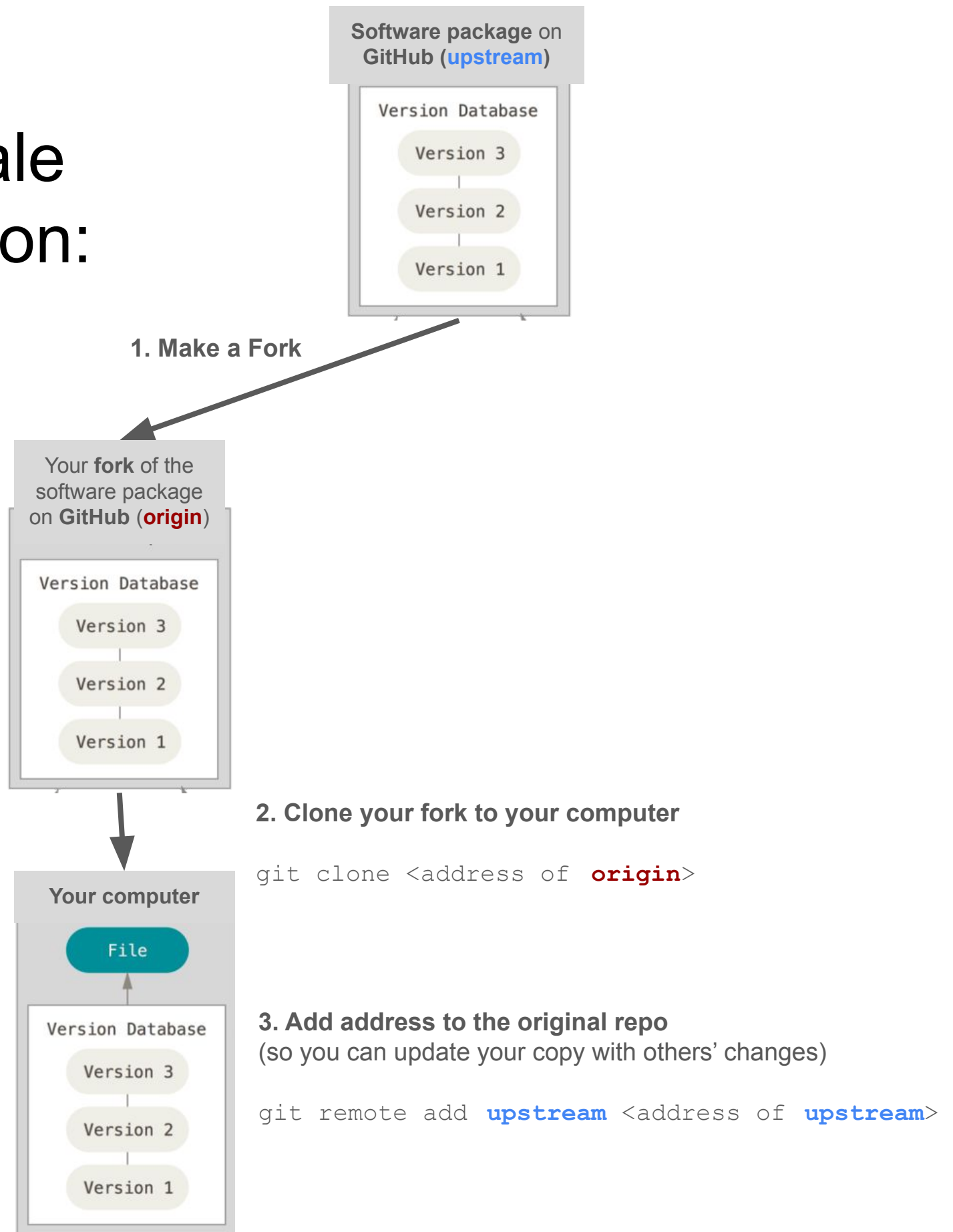
Part 2

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

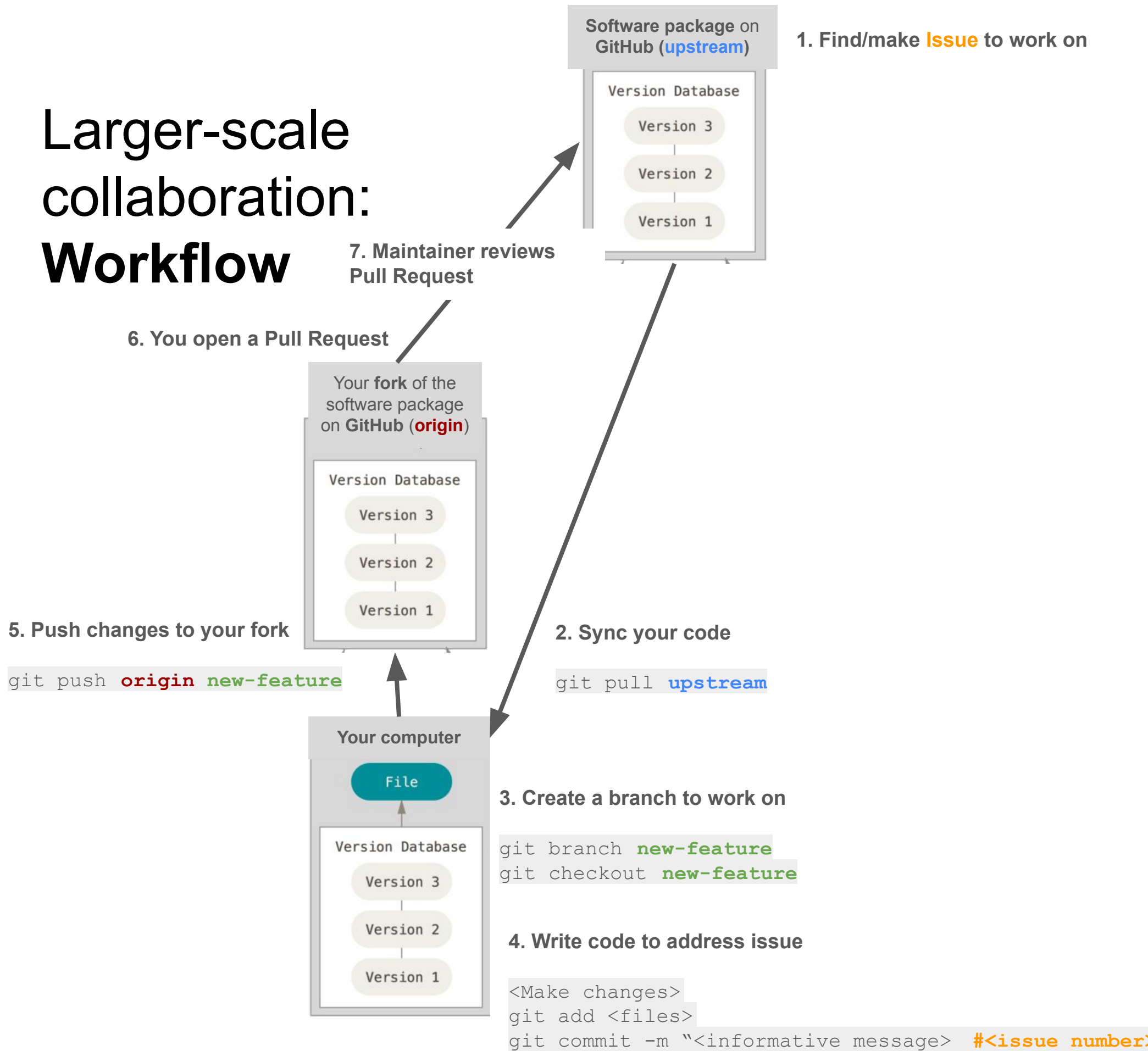
Small-scale collaboration



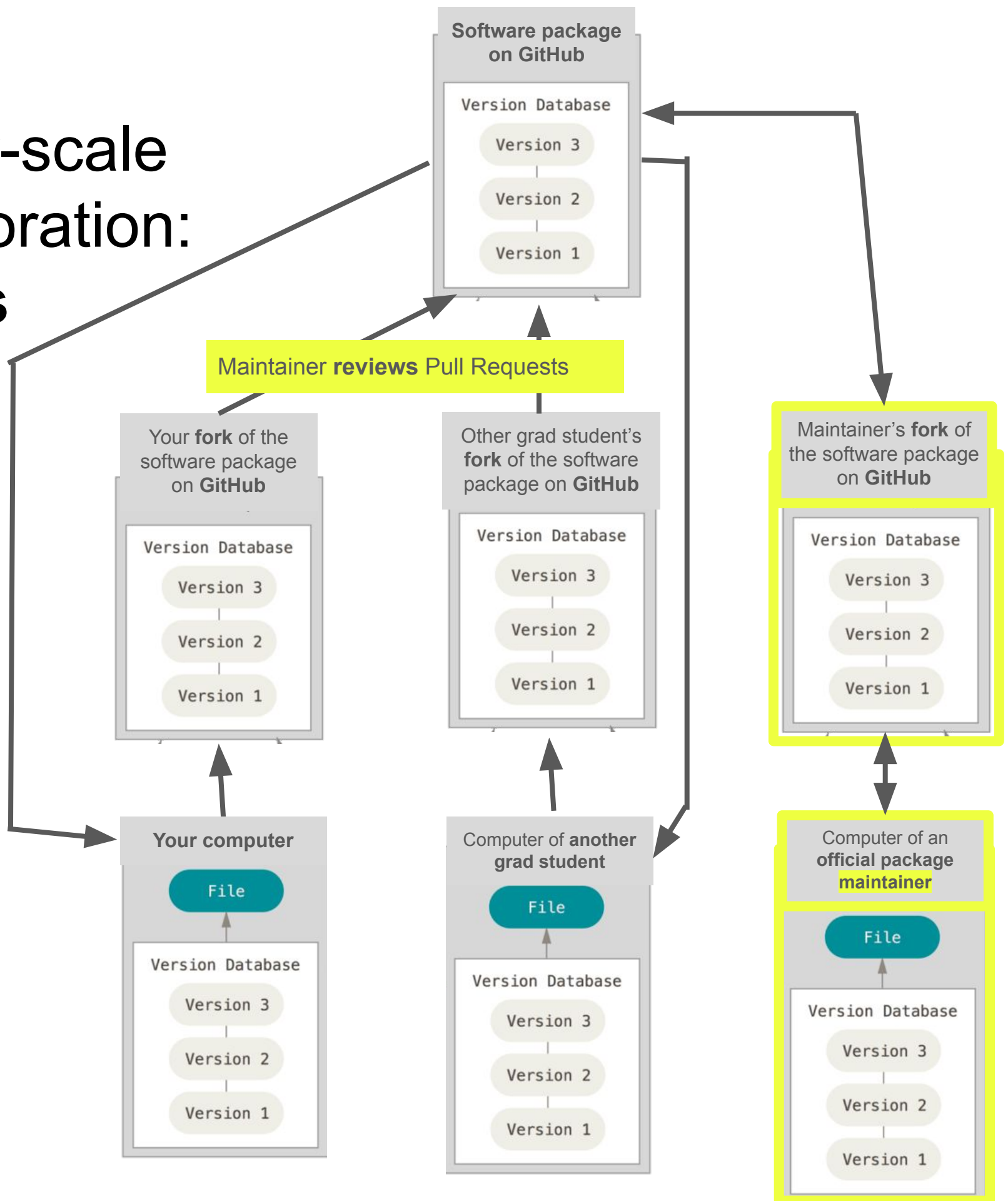
Larger-scale collaboration: Setup



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>
```

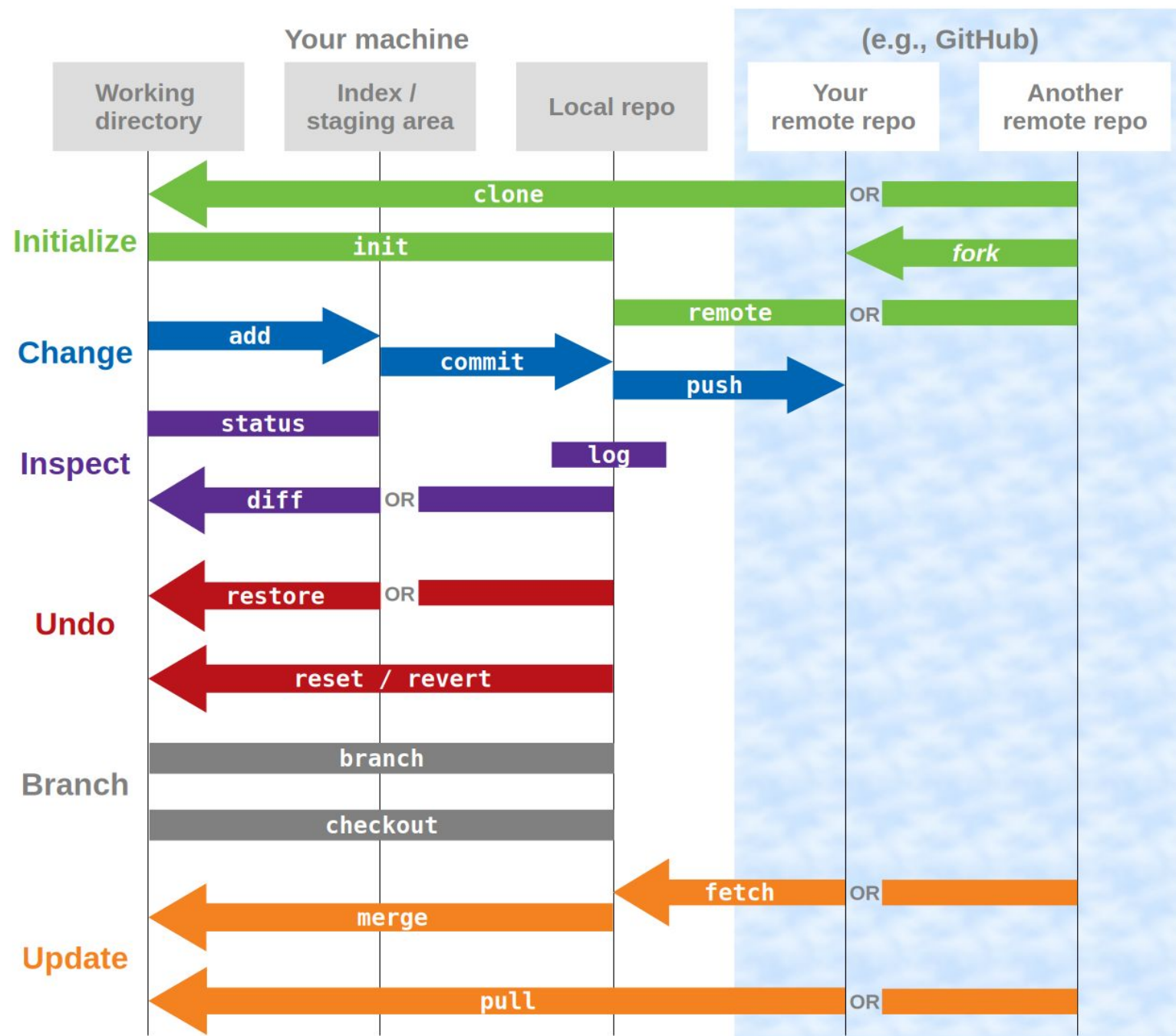
- Push commits to a remote repo

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

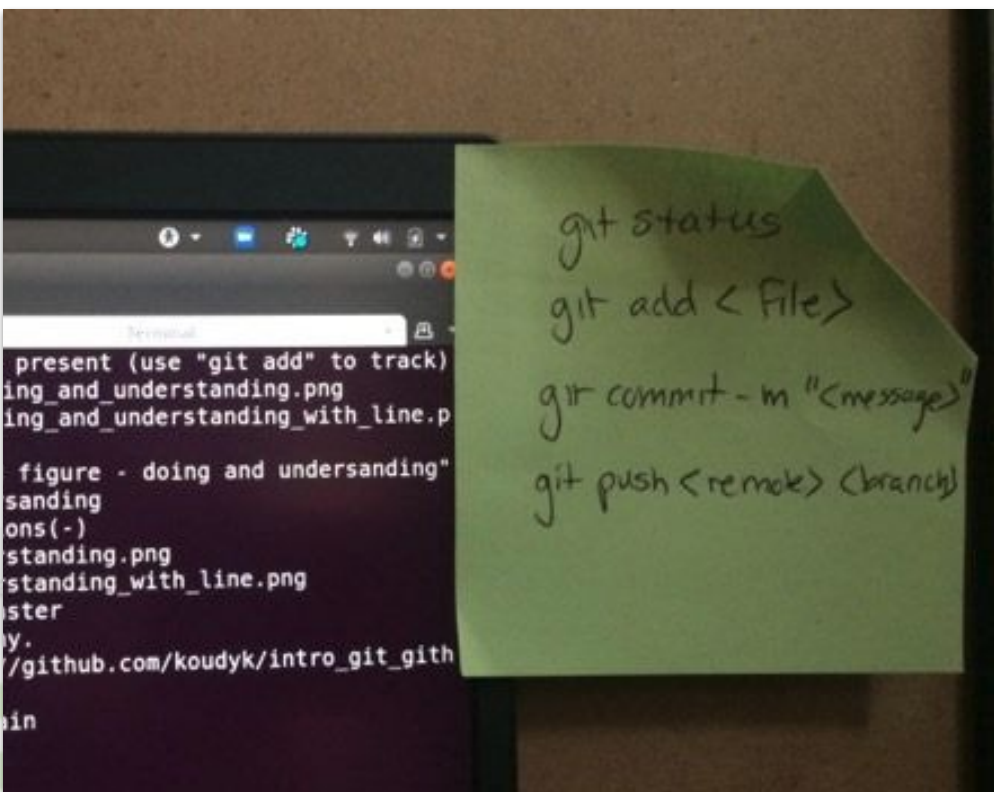
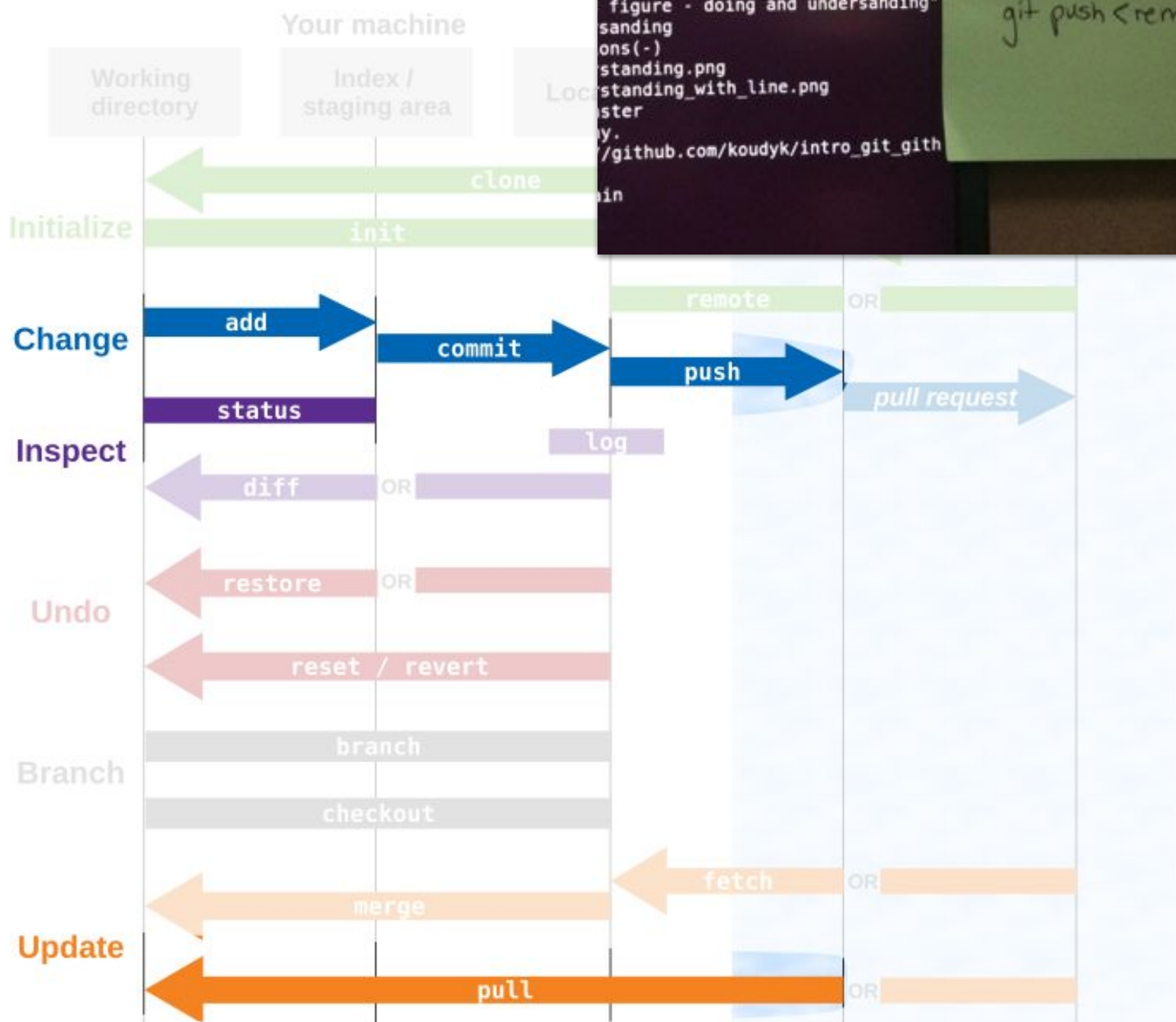
- Pull (fetch and merge) commits from a remote repo

```
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

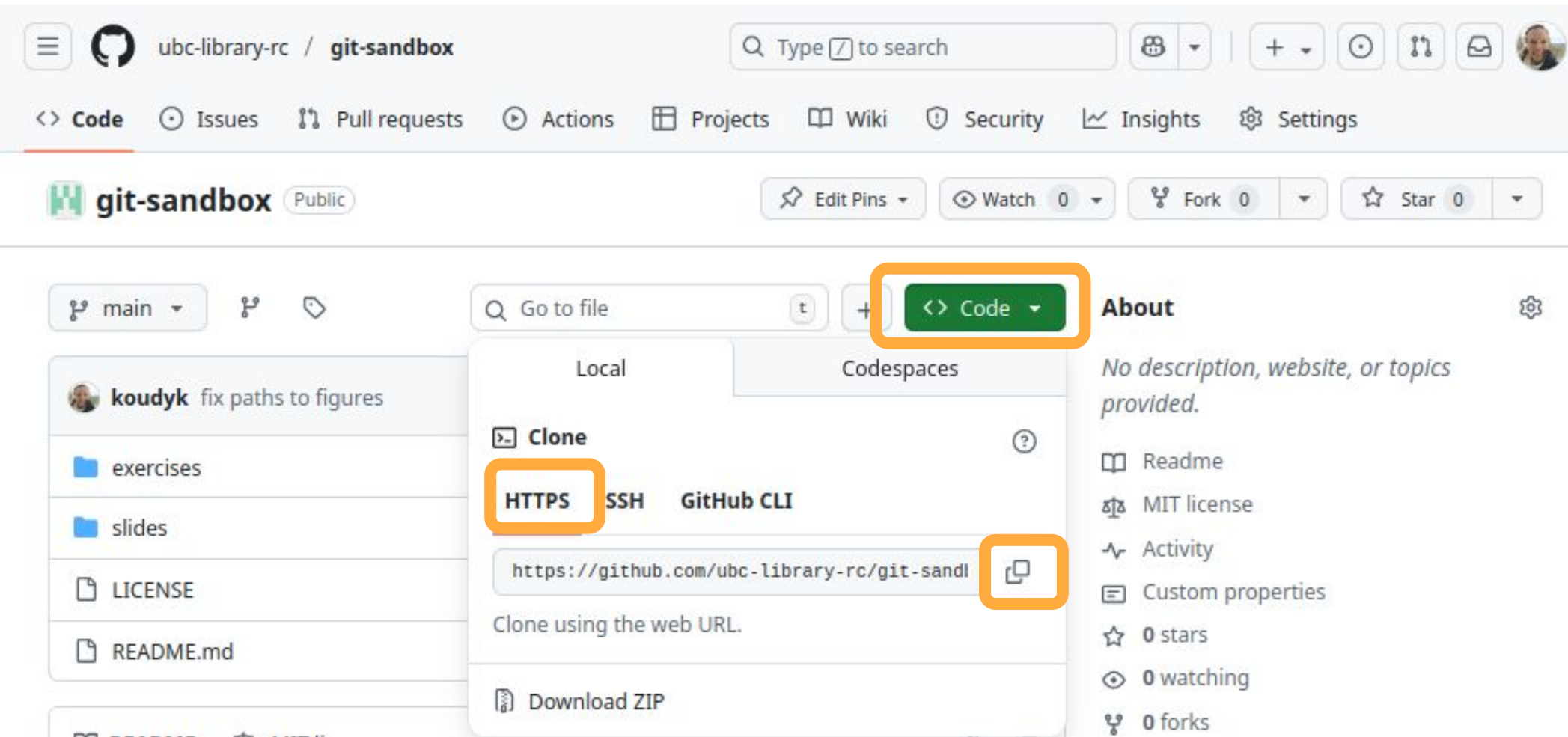
Part 1 (last week)

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

Part 2

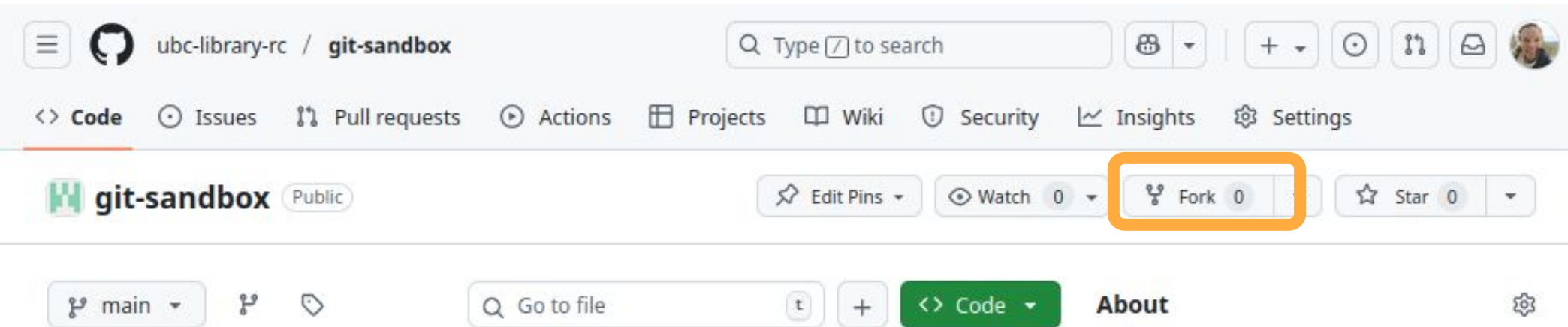
- Share your work on GitHub
- Collaborate on GitHub
- **GitHub features**
- Try it out!

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

Issues - where you can

GitHub features

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

The screenshot shows the GitHub interface for the `neurobagel / query-tool` repository. The `Issues` tab is highlighted with an orange box. The search bar at the top contains `is:issue state:open`. The issues are listed in a table with columns for status, title, labels, and comments. The issues are sorted by 'Newest'.

Status	Title	Labels	Comments
Open 40	Update the internal imaging modality list to match the CLI	Enhancement	1
Closed 116	Show full name of imaging modality tag on hover	Enhancement	
	Split fetching results into preview and download	Enhancement	
	Consider committing <code>env.dev</code> and <code>env.testing</code> to git	flag:stale	1
	version tag update inside <code>package.json</code> on release is not working	flag:stale, flag:detail needed, flag:discuss, flag:schedule, process	2
	Add timestamp or unique ID to filename of each results file	flag:stale, usability	1
	Query results download options should be clickable	flag:stale, usability	1
	Add info tooltip to explain node selection	flag:stale, documentation, usability	1

GitHub features

<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 highlighted with an orange box. Below the navigation bar, there are filters for `is:pr is:open` and buttons for **Labels** (45) and **Milestones** (0). A green button **New pull request** is visible. The list of pull requests shows **1 Open** and **473 Closed**. The first pull request, **[ENH] Add newly supported imaging types** (pr-minor), is selected. A modal window displays details for this pull request, including the title, number (#637), and a description: "Closes #636 Changes proposed in this pull request: hard code the new imaging type...". The modal also shows the branch `main` and `issue636`, and a green **Approved** button.

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

GitHub features

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

The screenshot shows a GitHub pull request page for the repository 'neurobagel / query-tool'. The pull request title is '[ENH] Add newly supported imaging types #637'. The 'Pull requests' tab in the repository navigation bar is highlighted with an orange box. The pull request is open, showing 2 commits merged into the 'main' branch from 'issue637'. The pull request description includes a comment from 'surchs' 5 days ago, which mentions closing a previous pull request and proposes changes to the internal imaging modality list and BIDS imaging suffixes. A checklist for PR reviewers is provided, with most items checked. The right sidebar shows reviewers 'sourcery-ai[bot]' and 'alyssadai', both with checkmarks, and a 'pr-minor' label.

neurobagel / query-tool

Type to search

<> Code Issues 40 **Pull requests 1** Actions Projects Security Insights

[ENH] Add newly supported imaging types #637

Open surchs wants to merge 2 commits into main from issue637

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

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

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



```
$ 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
```

