

Check if you're ready

If you're **stuck** on a step, enter the step **letter** in the **comments** on Zoom

✓ 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)
- The email should be the one you used to sign up for **GitHub**

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

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

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 **xwməθkwəyəm** (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/>

Introduction to Git and GitHub

Part 1

Kendra Oudyk, PhD (she/her)
Graduate Academic Assistant for Digital Scholarship
Research Commons, UBC Koerner Library

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/>.

Goals

Part 1

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

Part 2 (next week)

- Share your work on GitHub (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating GitHub
- 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**
 - **1st time 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

Check if you're ready

If you're **stuck** on a step, enter the step **letter** in the **comments** on Zoom

✓ 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)
- The email should be the one you used to sign up for **GitHub**

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

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

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

Goals

Part 1

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

Part 2 (next week)

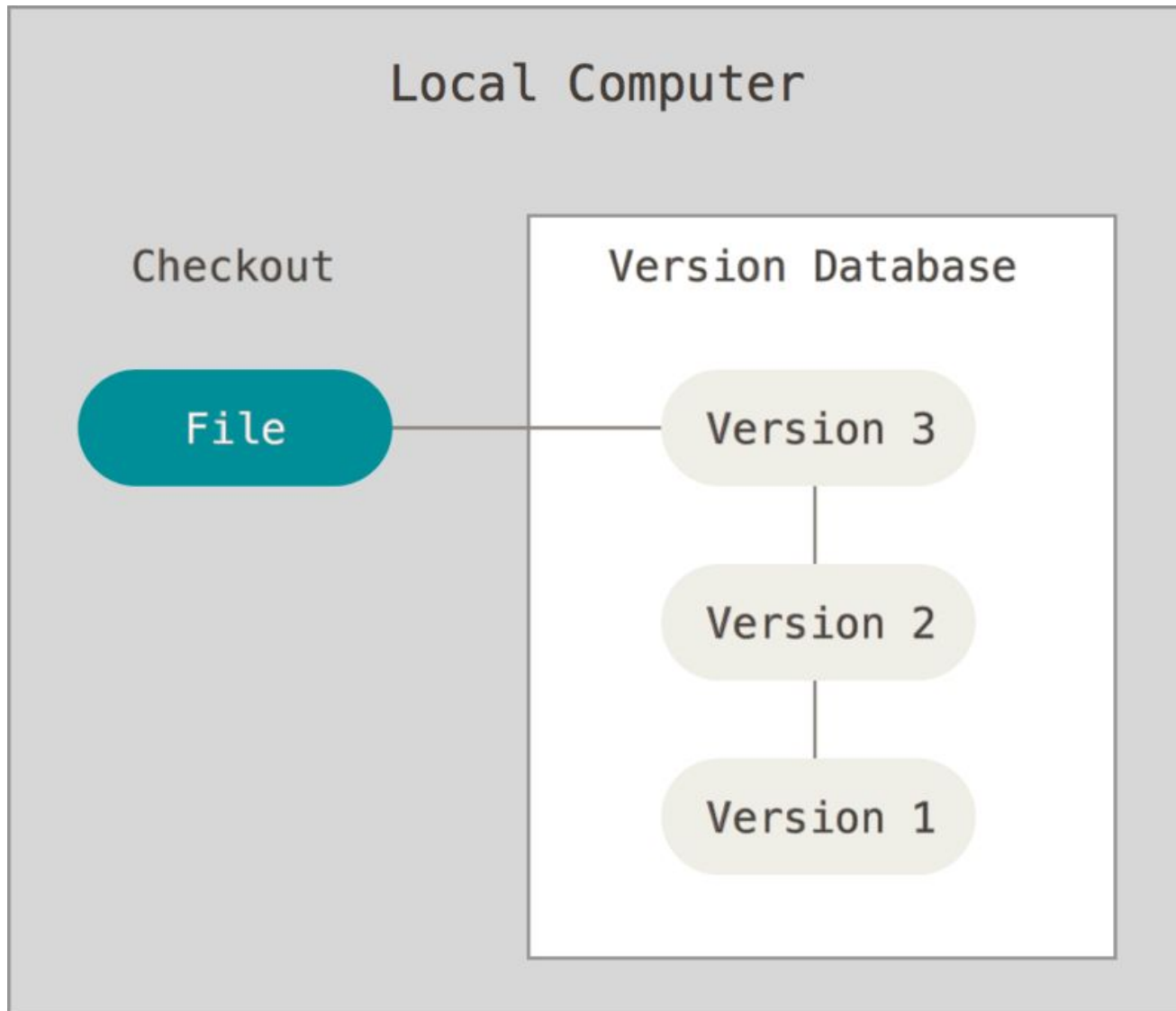
- Share your work on GitHub (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating GitHub
- Try it out!

Version control

- Tracks **changes** to files
- Lets you recall different **versions** of files
- Becomes more essential as **collaborative projects** grow
- Can track almost **any type** of file (works best on text-based files)
- There are different **types** of version control

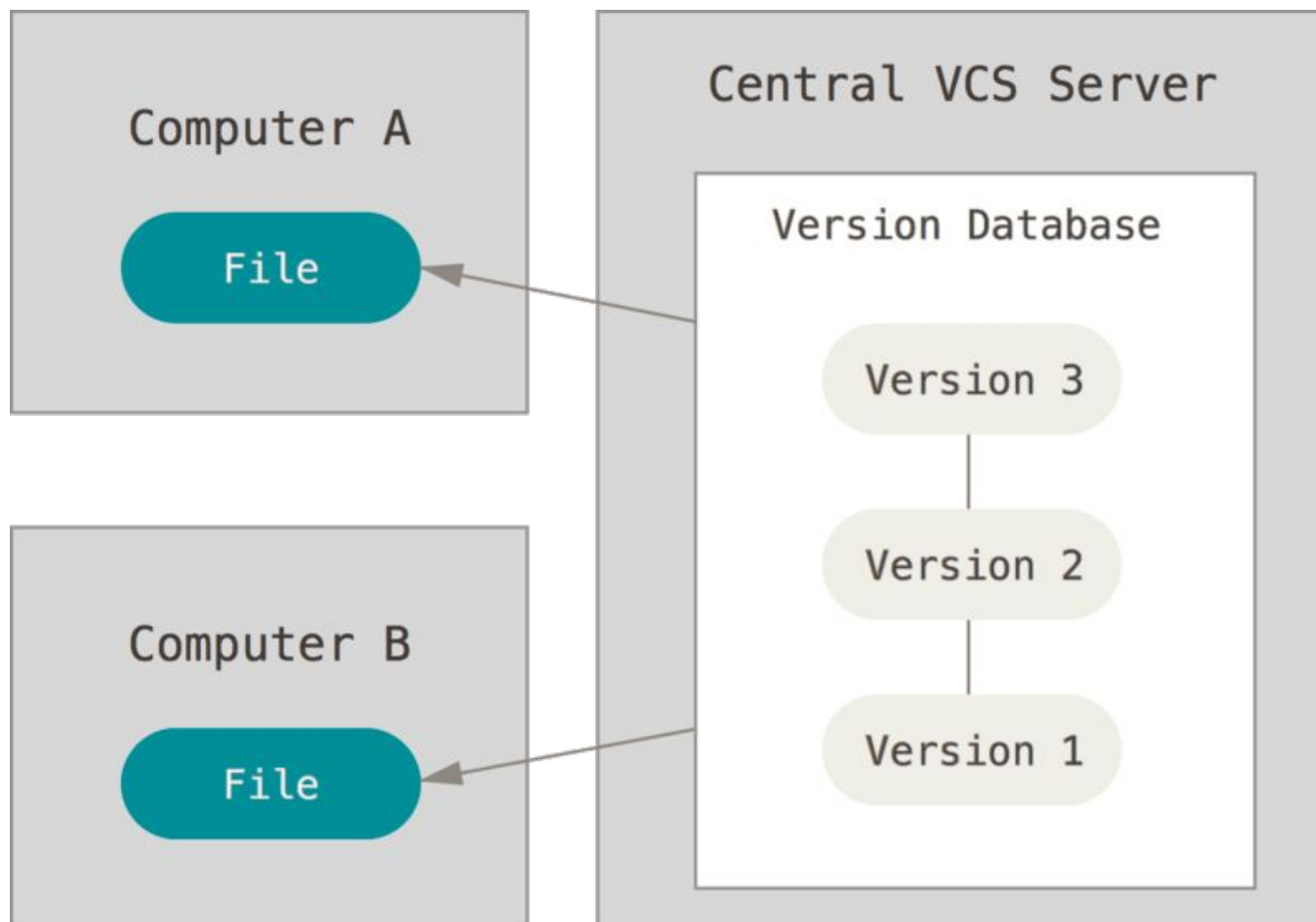


Local version control



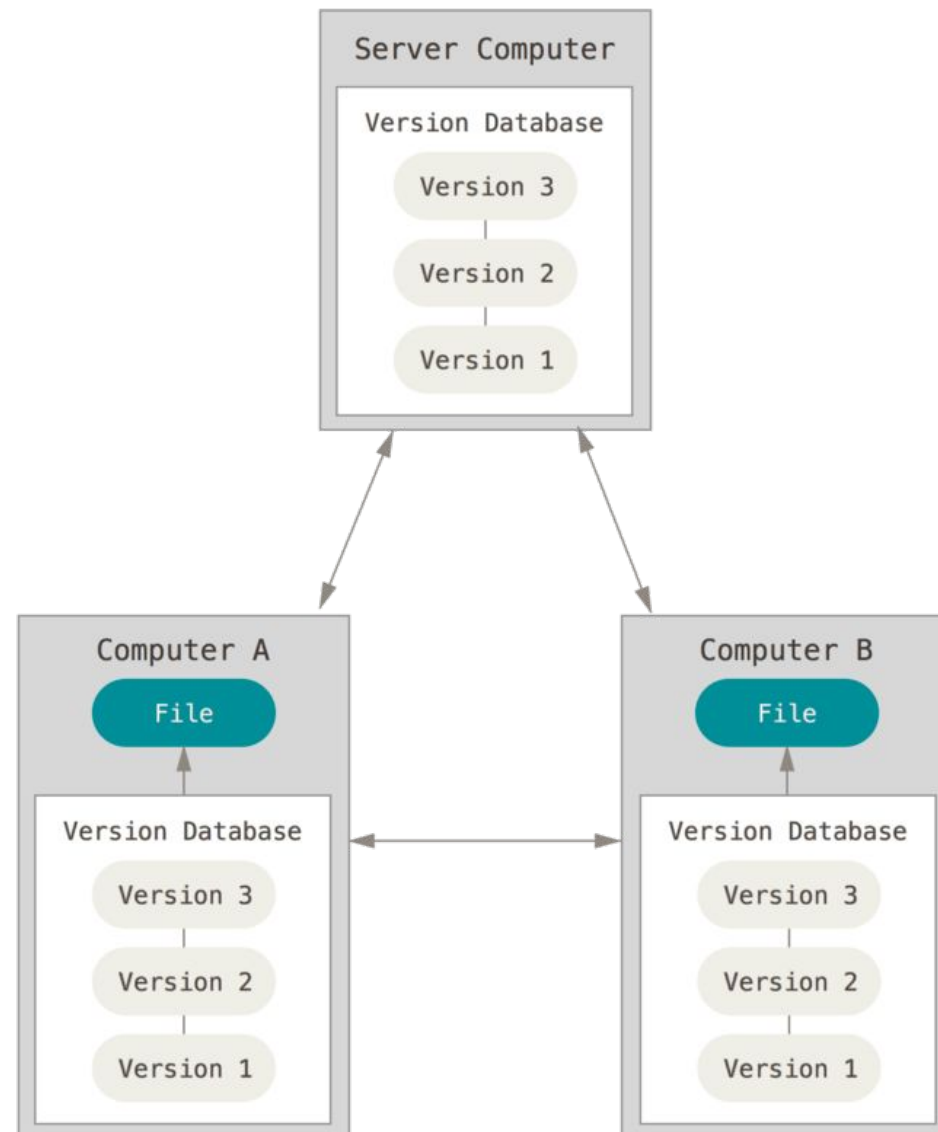
But how do we collaborate?

Centralized version control



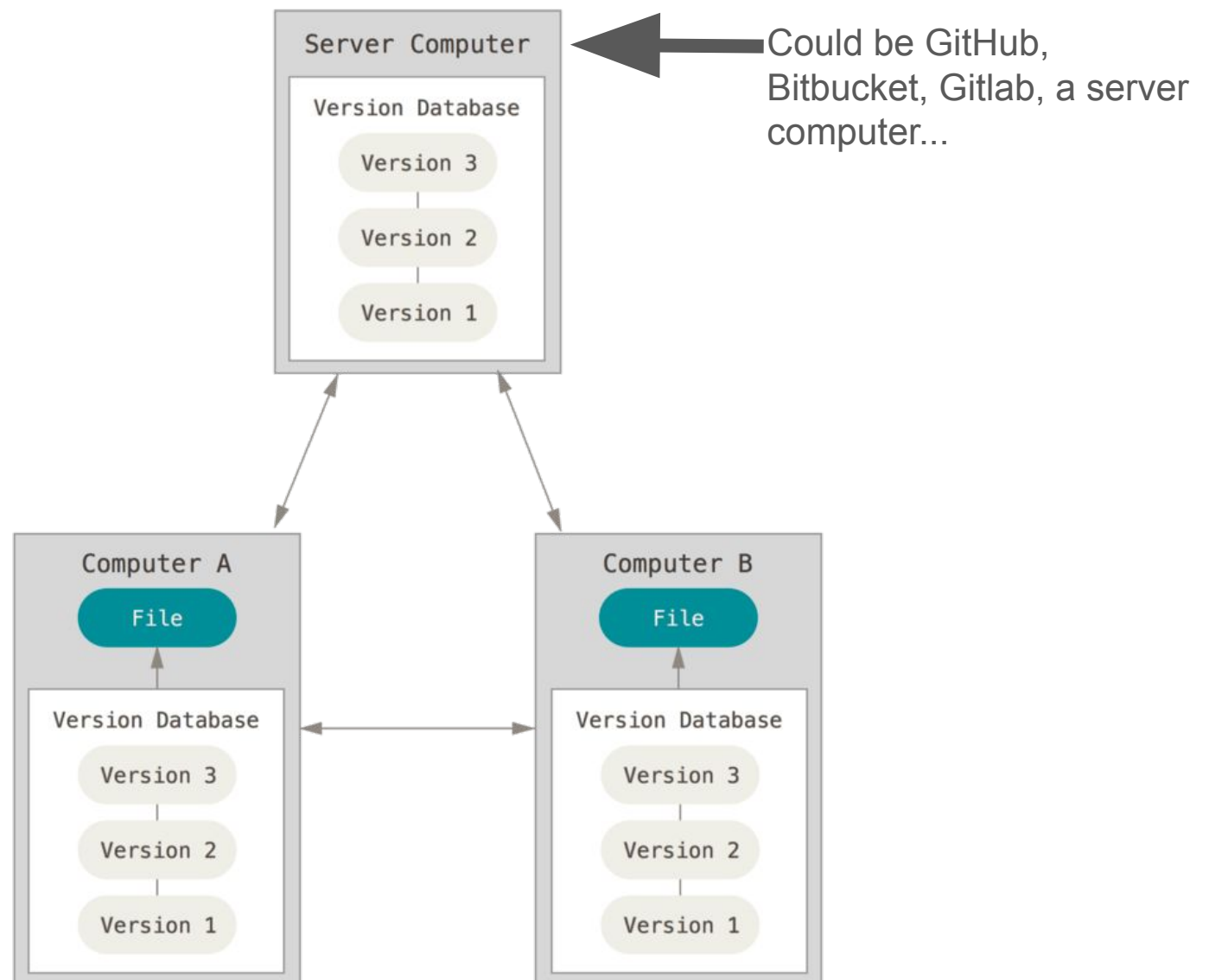
But what if the server crashes?

Distributed version control



Git vs. GitHub

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



Goals

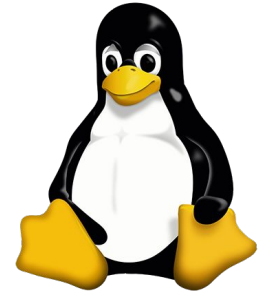
Part 1

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

Part 2 (next week)

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

git was made for Linux



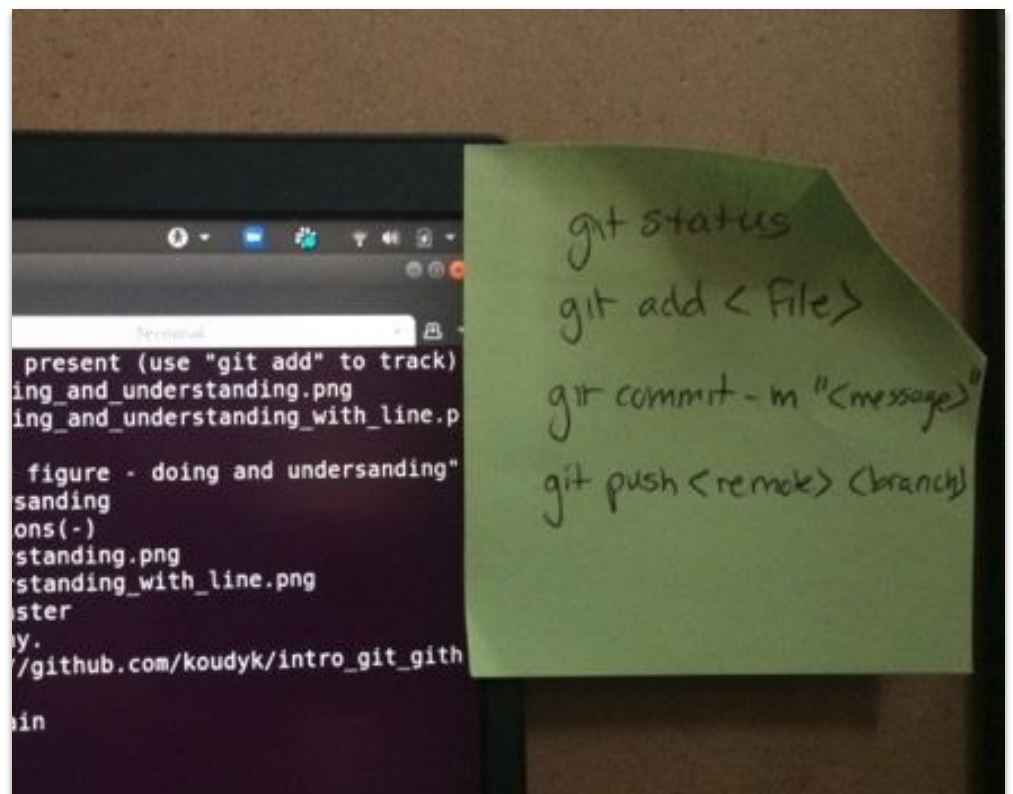
i.e., a huge open-source project.

- In the last month, “1067 authors have pushed 5,329 commits...”
[<https://github.com/torvalds/linux/pulse/monthly>, as of 2021-07-14]

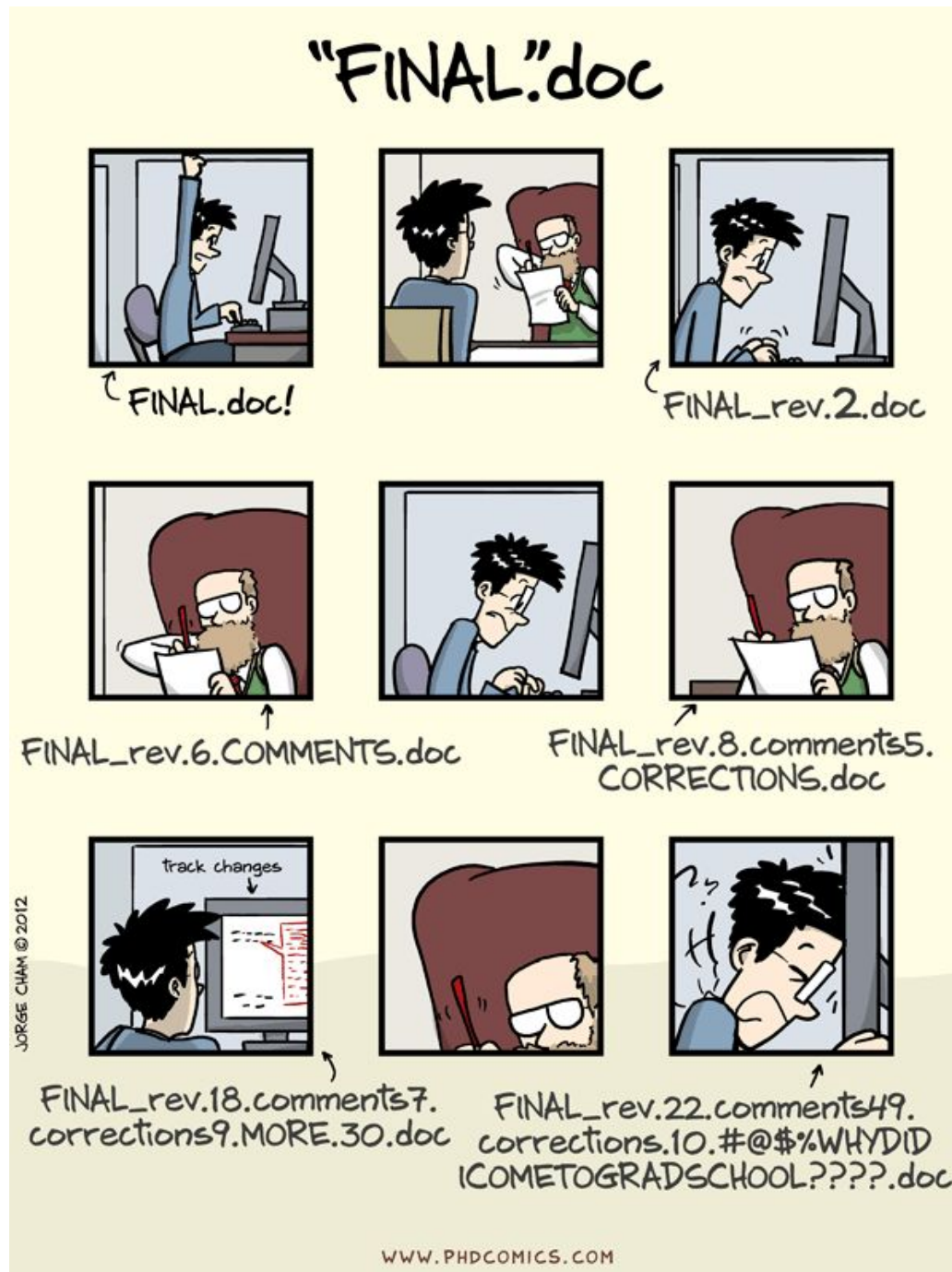
It is designed to be

- Simple and fast
- Fully distributed
- Able to support non-linear development
- Scalable

You won't need all its features

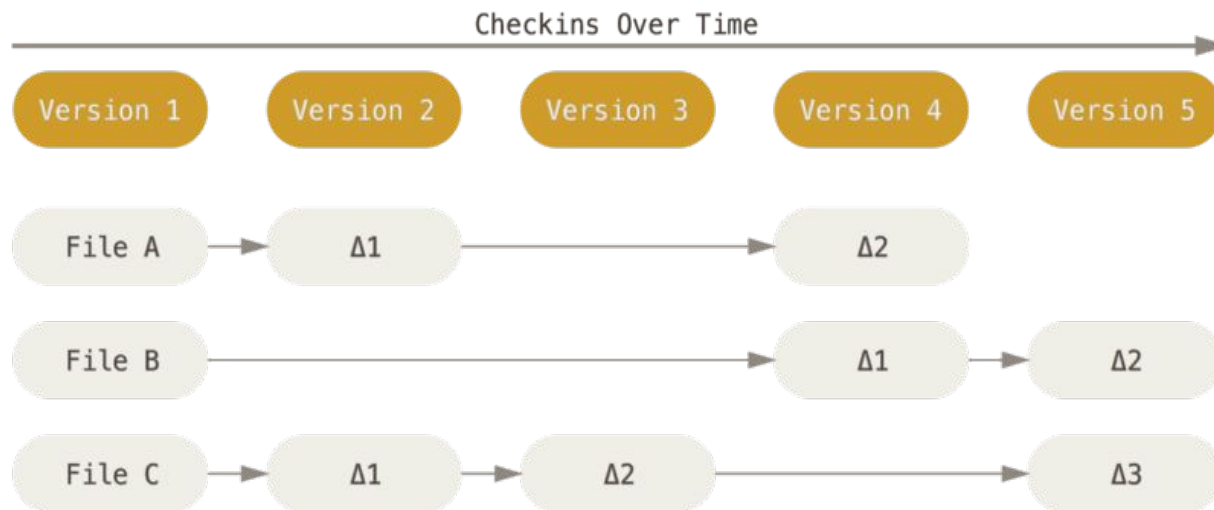


An intuitive user experience

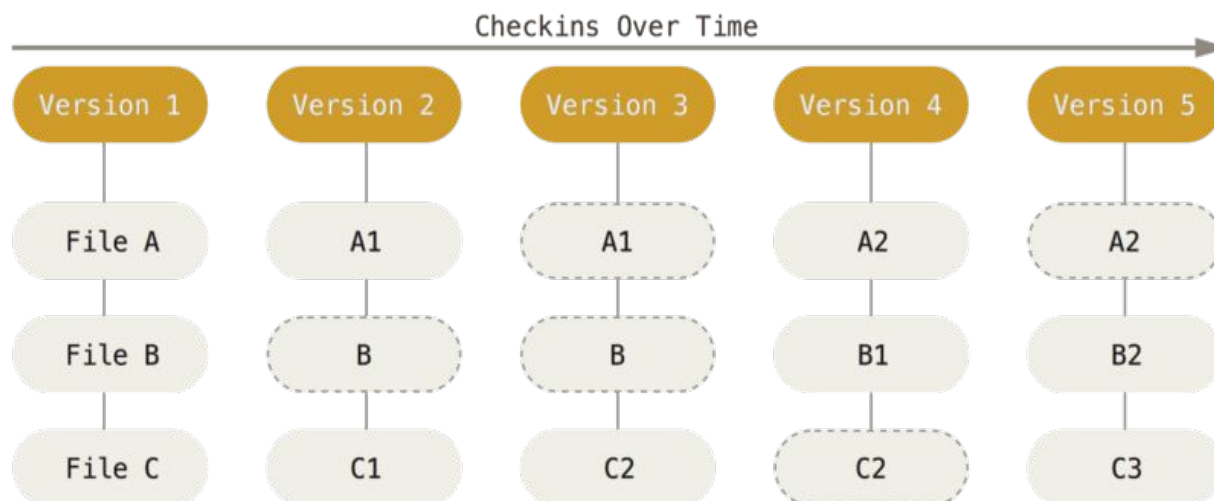


How does git do this?

“Delta-based version control” (other version control systems)

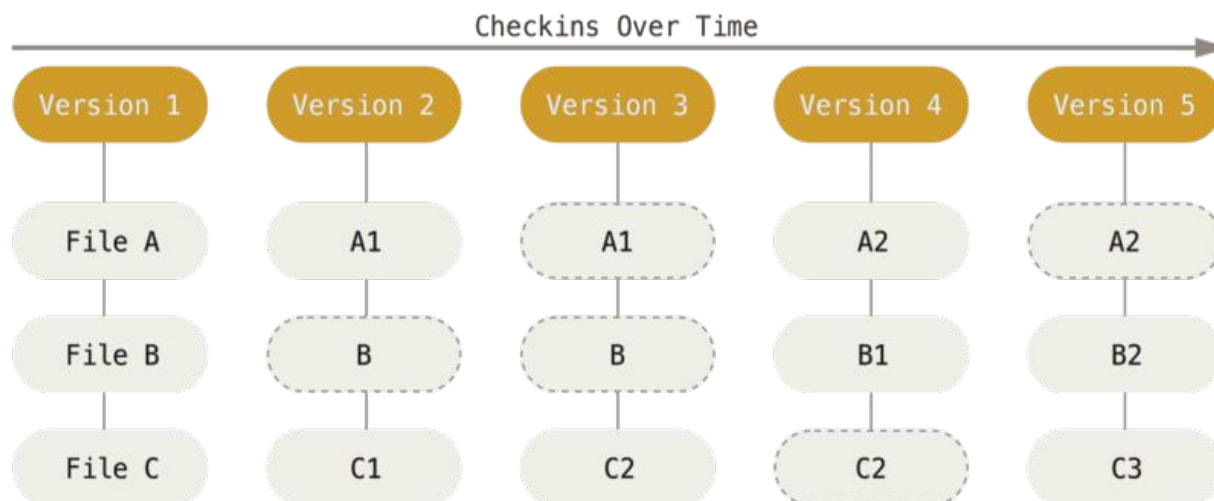


Stream of **snapshots** (git)



Git is reliable

- It doesn't rely on filenames to keep track of files
- It converts the contents of a file/directory → hash
 - E.g., `24b9da6552252987aa493b52f8696cd6d3b00373`
 - It is unique and deterministic (1-way)
 - Changed file content → changed hash



Most developers use git for version control

Stack Overflow developer survey

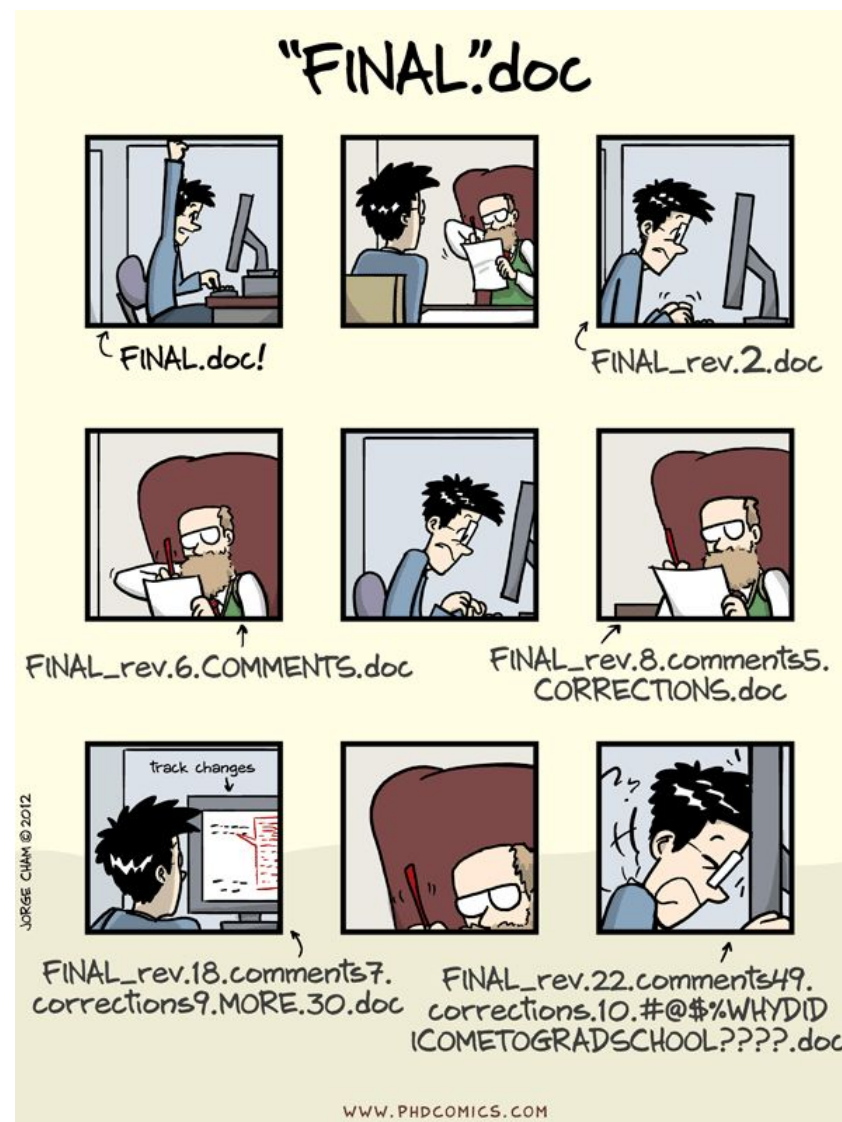
- 2015 (16,694 responses)
- 2017 (30,730 responses)
- 2018 (74,298 responses)

Name	2015	2017	2018
Git	69.3%	69.2%	87.2%
Subversion	36.9%	9.1%	16.1%
TFVC	12.2%	7.3%	10.9%
Mercurial	7.9%	1.9%	3.6%
CVS	4.2%	[i]	[i]
Perforce	3.3%	[i]	[i]
VSS	[i]	0.6%	[i]
ClearCase	[i]	0.4%	[i]
Zip file backups	[i]	2.0%	7.9%
Raw network sharing	[i]	1.7%	7.9%
Other	5.8%	3.0%	[i]
None	9.3%	4.8%	4.8%

But we're not developers

But we're not developers... right?

- We (hopefully) want to do our research in a way that is **open, reproducible, and collaborative**
- Do you have your own system that does this?
(I certainly don't, not for lack of trying)



“Science, after all, aspires to be distributed,
open-source knowledge development.”



McElreath, R. (2020, September 26). *Science as amateur software development* [video].
YouTube. https://www.youtube.com/watch?v=zwRdO9_GGhY

Goals

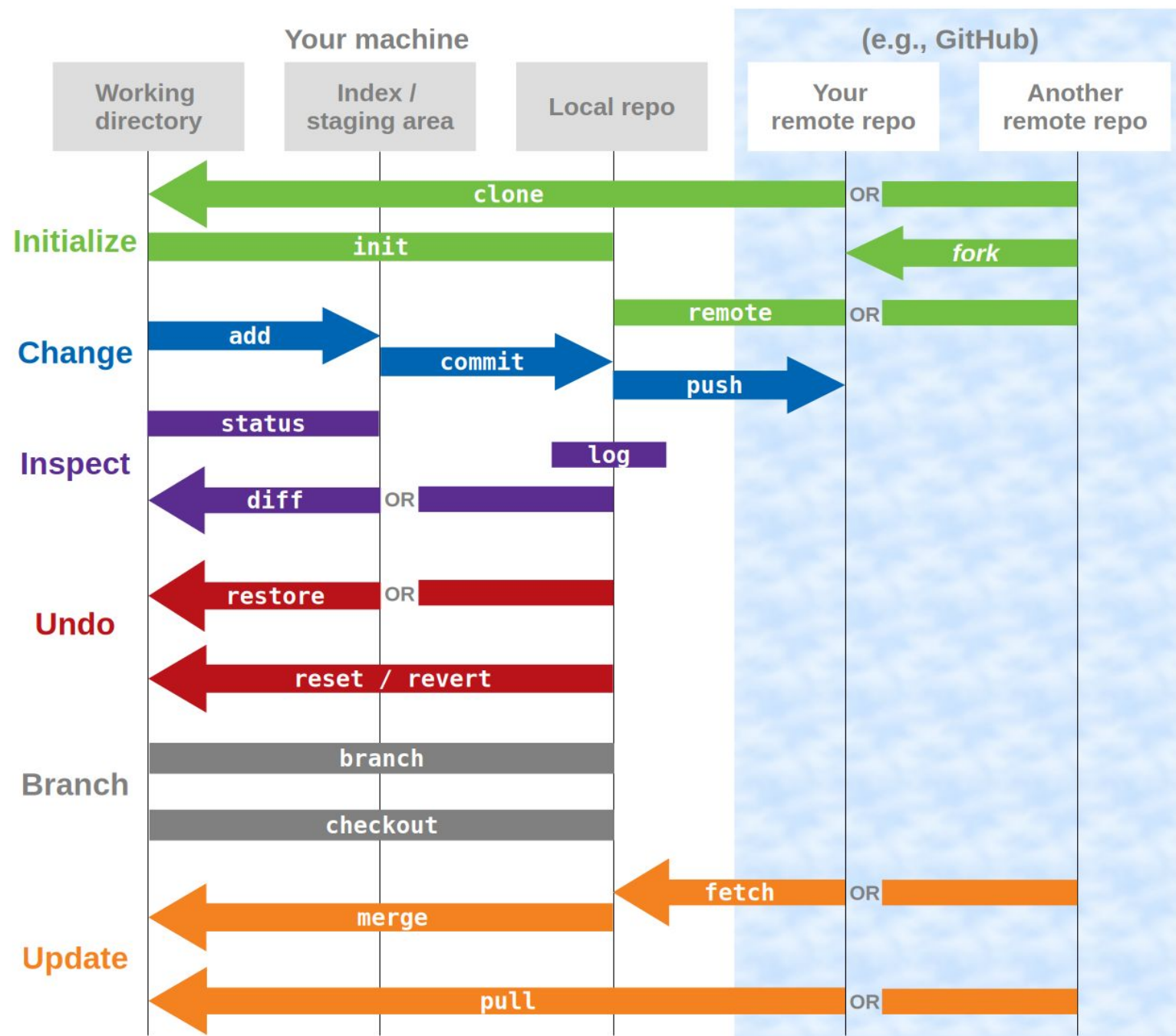
Part 1

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

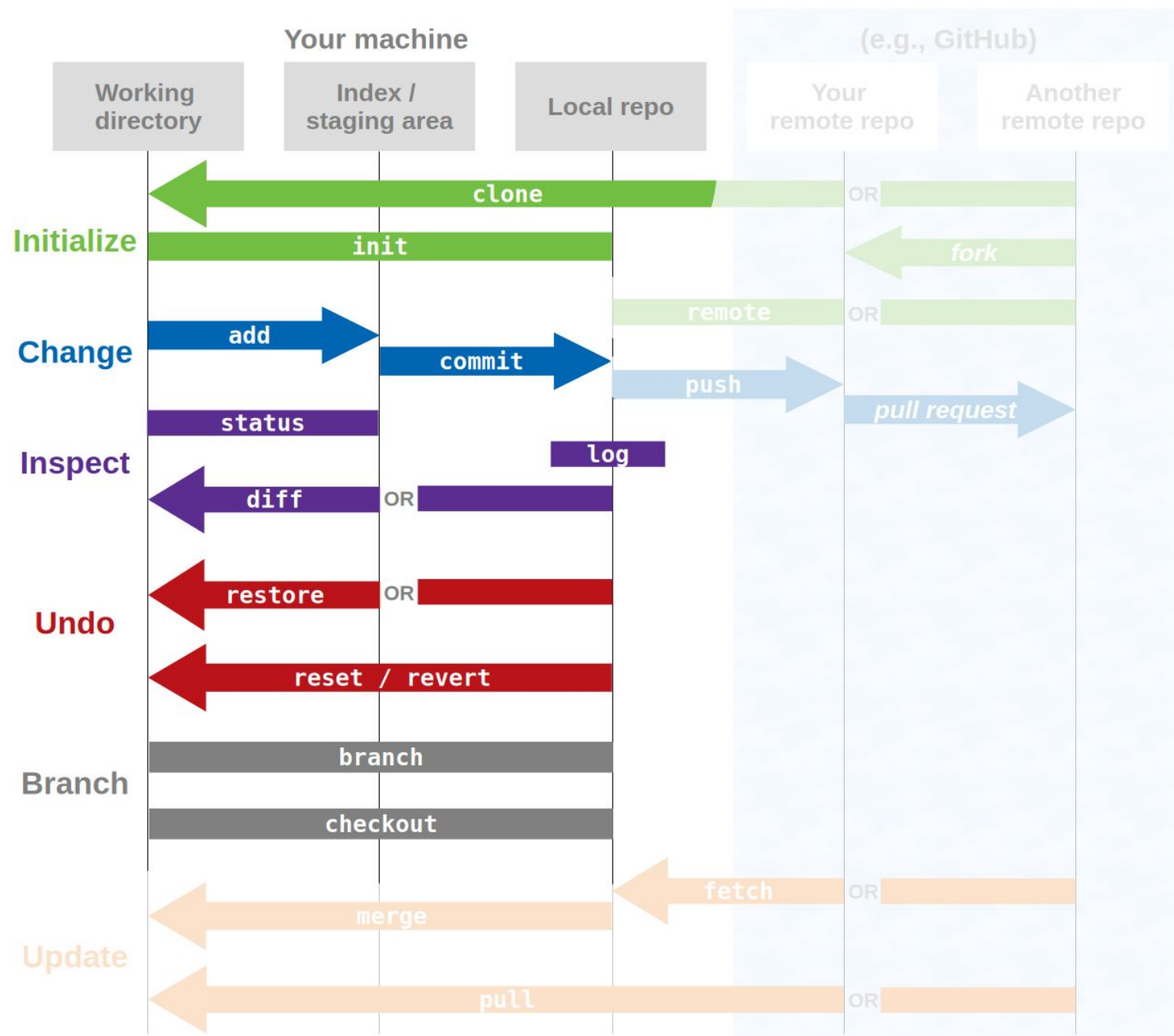
Part 2 (next week)

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

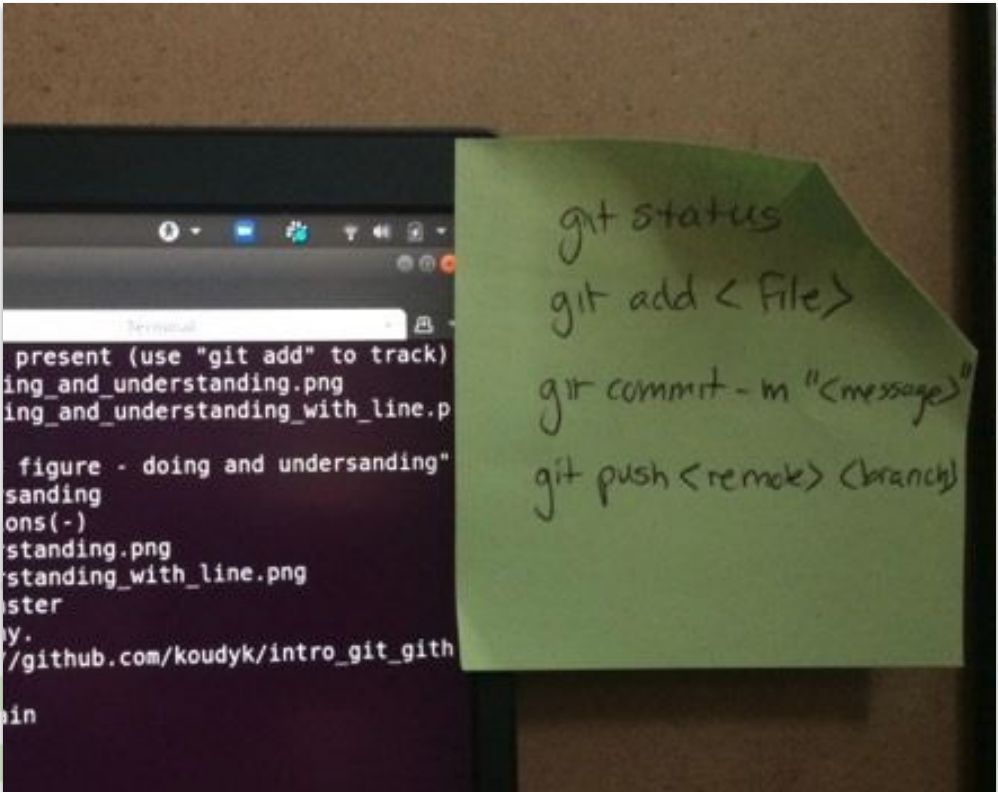
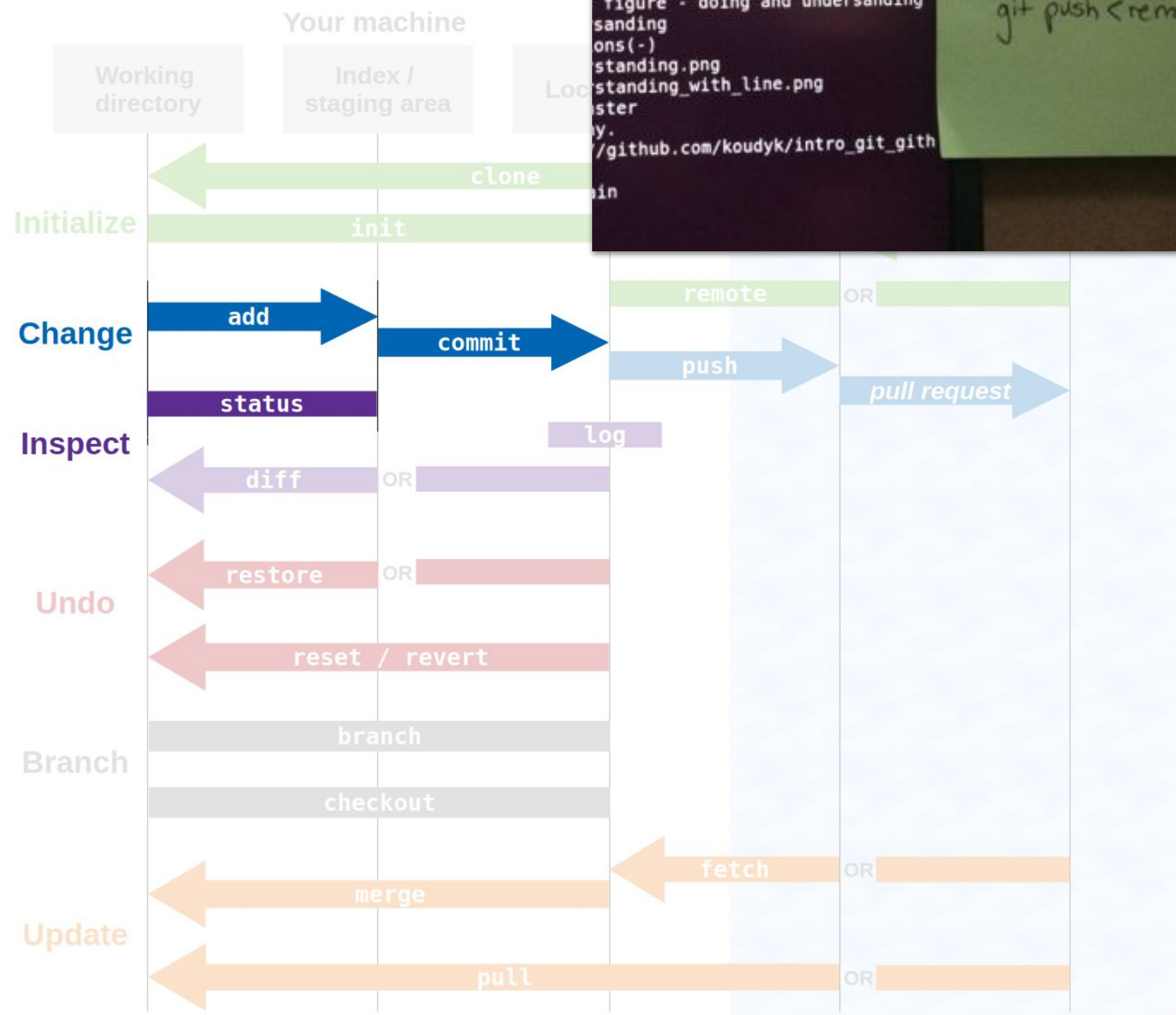
The commands we'll cover today and next week



The commands we'll cover today



The commands you'll need



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

First, how to get help

- In the terminal
(if you know the verb and want to know what it does or what are its options)

```
git help <verb>
```

```
git <verb> --help or git <verb> -h
```

```
man git-<verb>
```

TIP: press 'q' to exit the manual in the terminal

- Look it up



3 parts of a Git project

1. **Working directory**

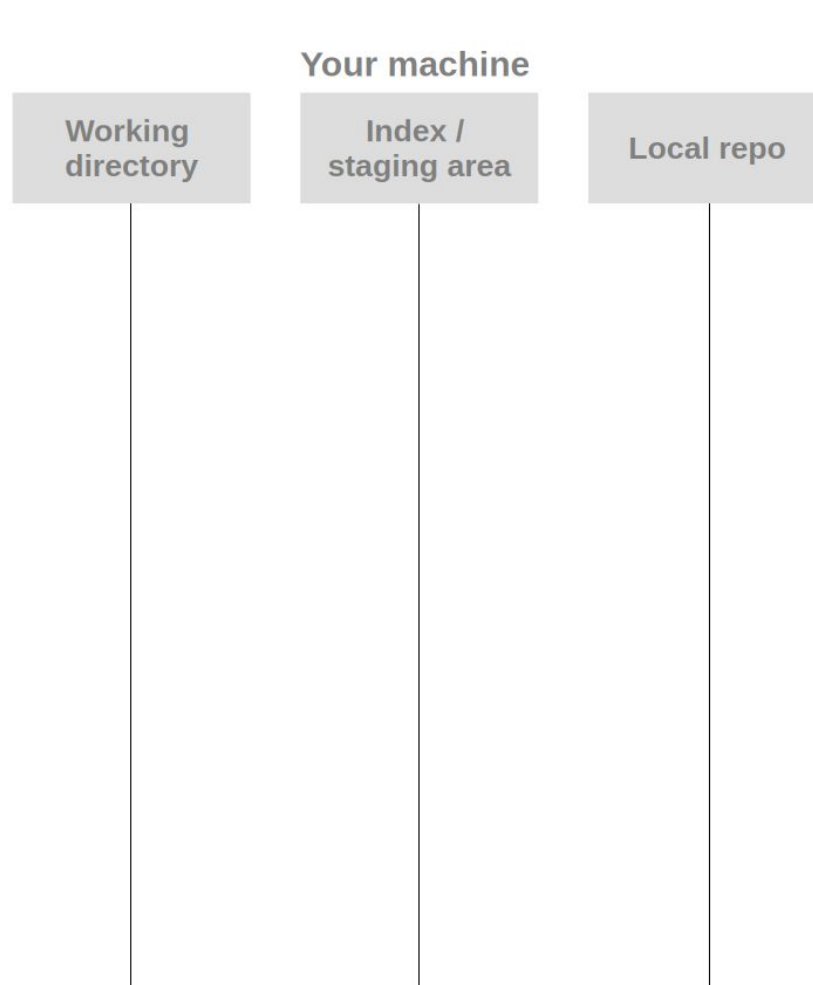
The version of the project that you're working on

2. **Staging area / Index**

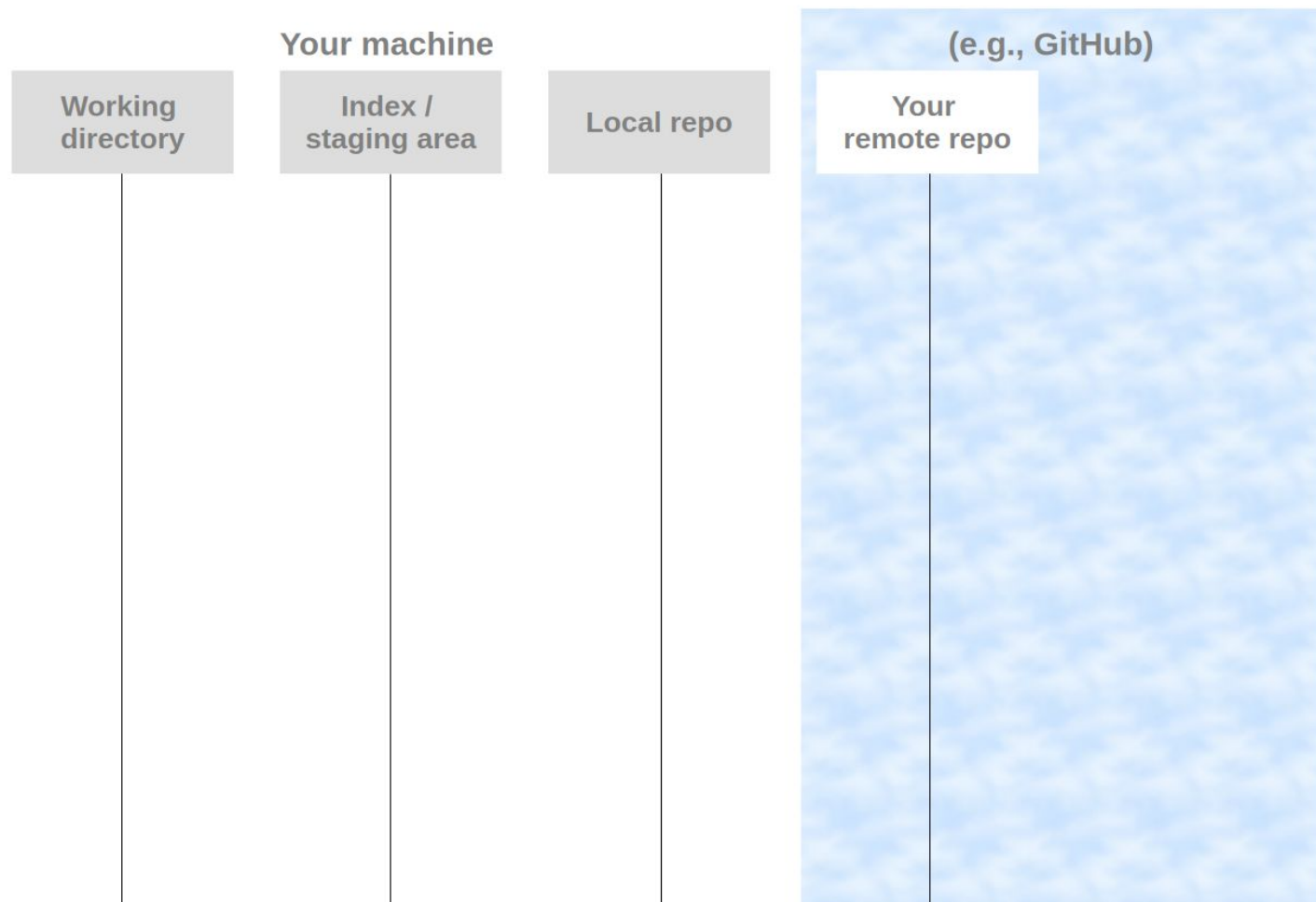
What will be in your next snapshot

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

Metadata and objects that make up the snapshots



Peek forward: Where's GitHub?



Starting a git repo

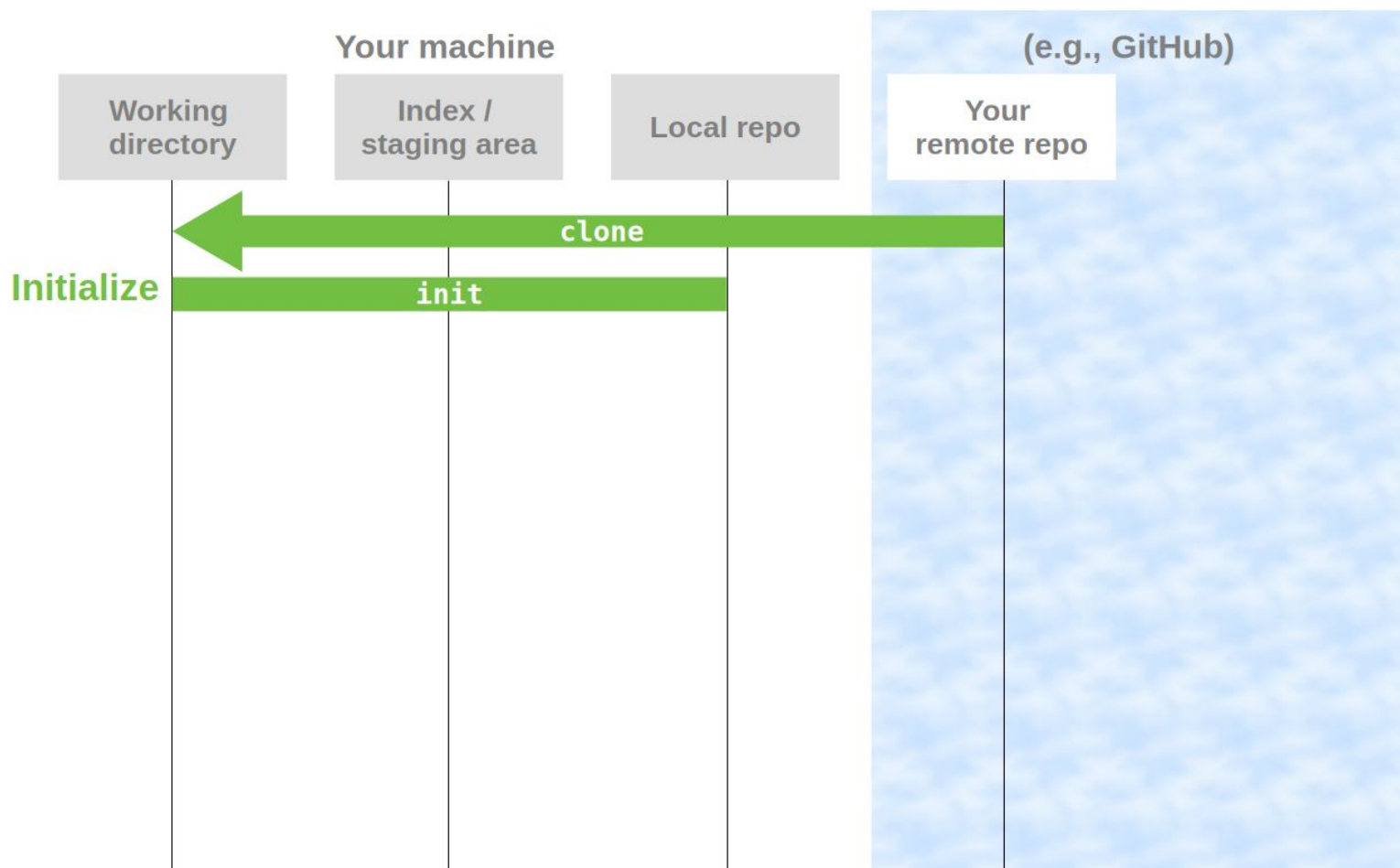
2 options

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 file states

1. **Modified**

You made a change to the file

2. **Staged**

You indicated that you want the modified file in your next snapshot



3. **Committed**

You took the snapshot



3-step basic workflow

1. Modify

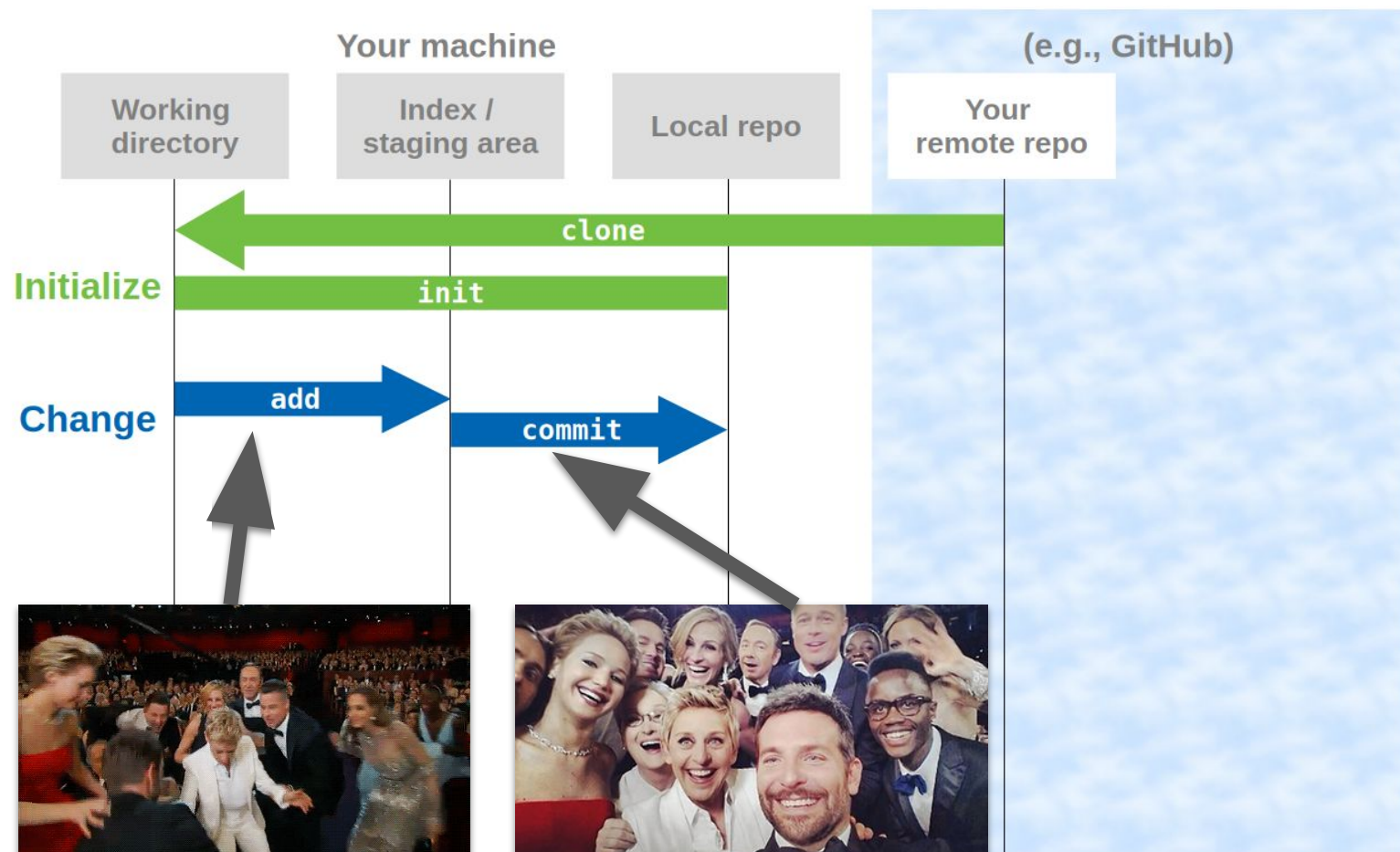
Change a file in your working tree

2. Stage

```
git add <filename>
```

3. Commit

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



Inspecting

(useful commands that don't *do* anything)

- Check the status of the files in your repo

```
git status
```

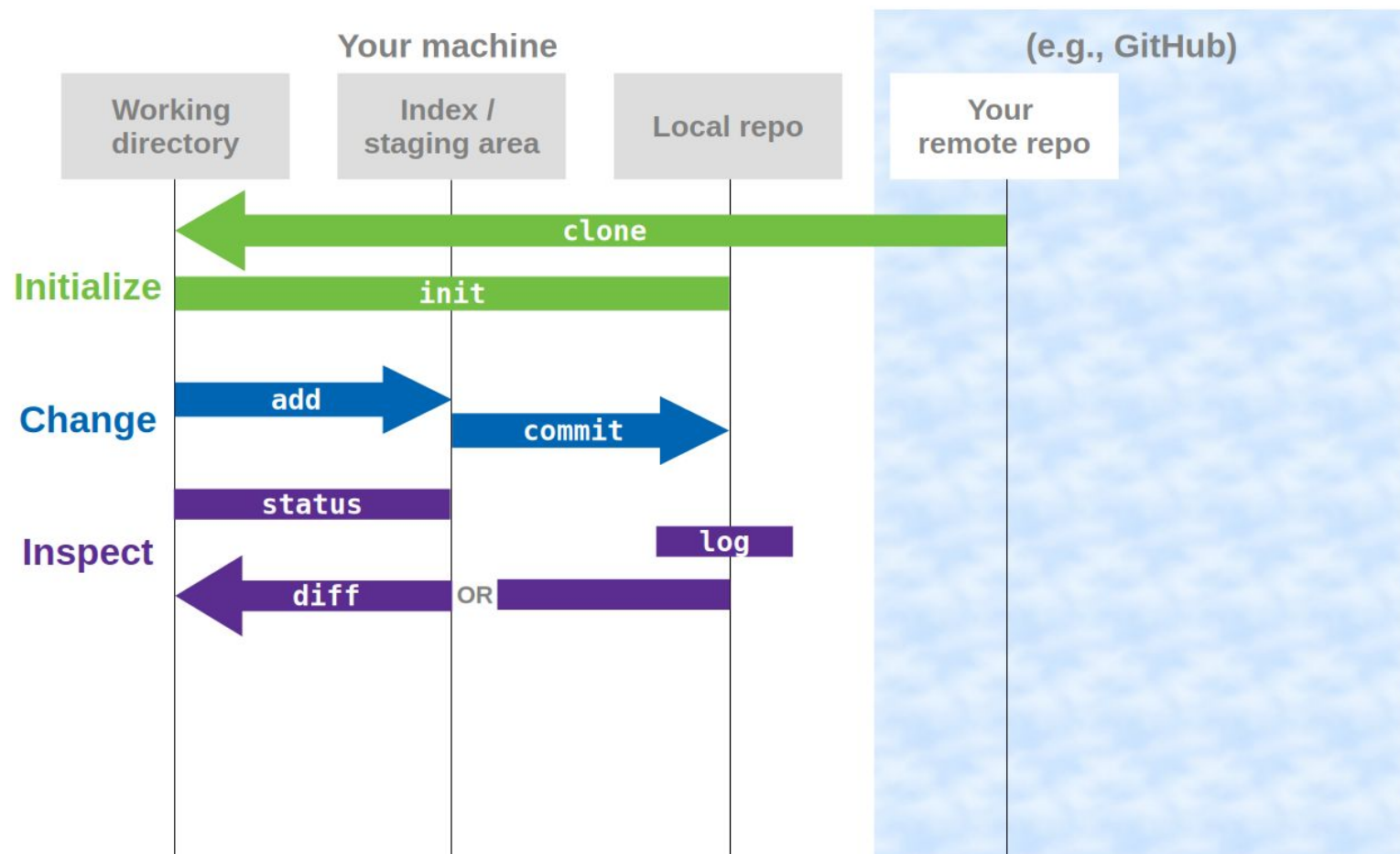
- See what changed

```
git diff
```

- See the history of your repo

```
git log
```

Note: type 'q' to exit the log



Common undoing goals

- **Unmodify a file**

```
git restore <file> or git checkout -- <file>
```

- **Unstage a file**

```
git restore --staged <file> or git reset HEAD <file>
```

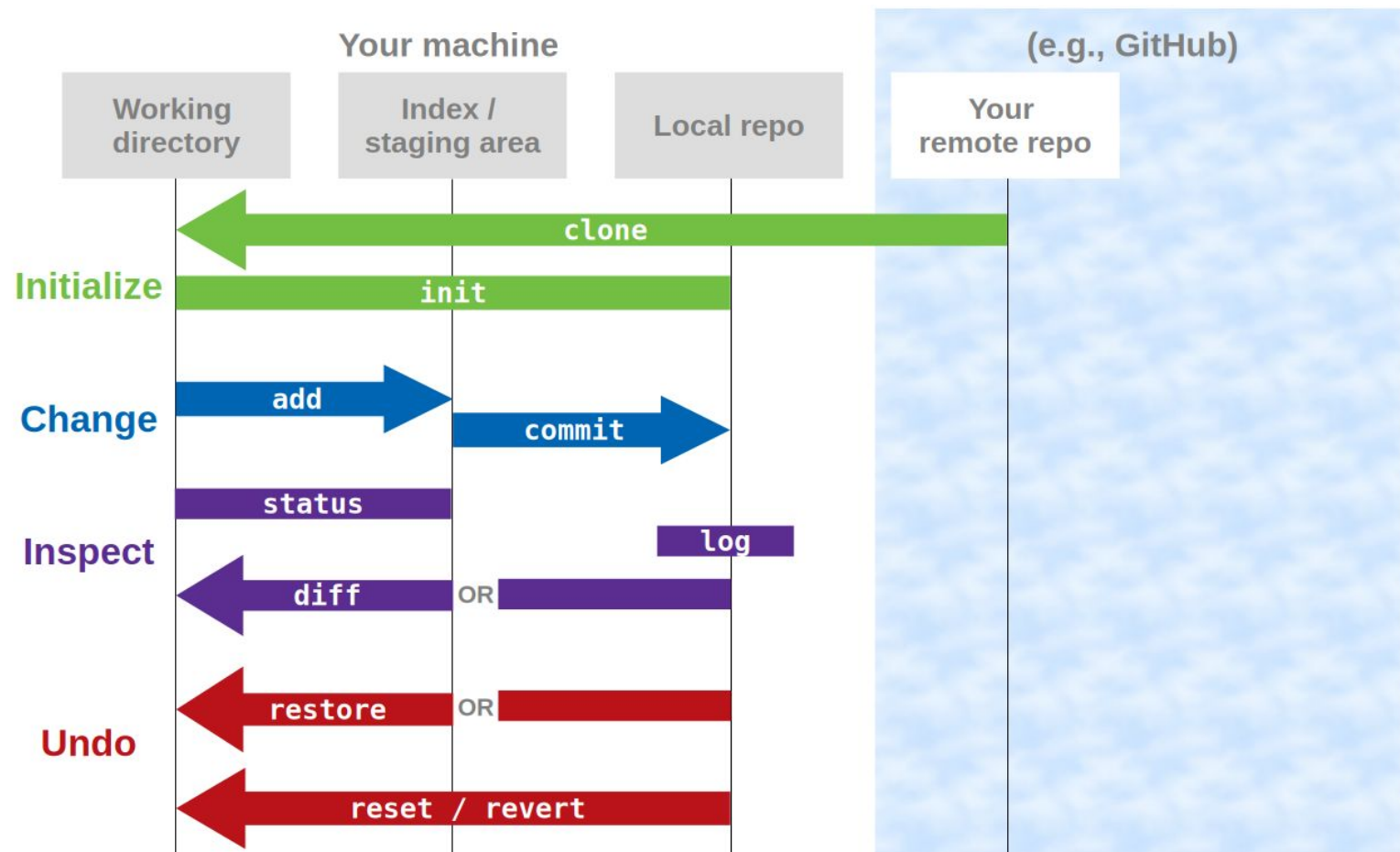
- **Forgot a file in the last commit**

```
git add <file>
```

```
git commit --amend
```

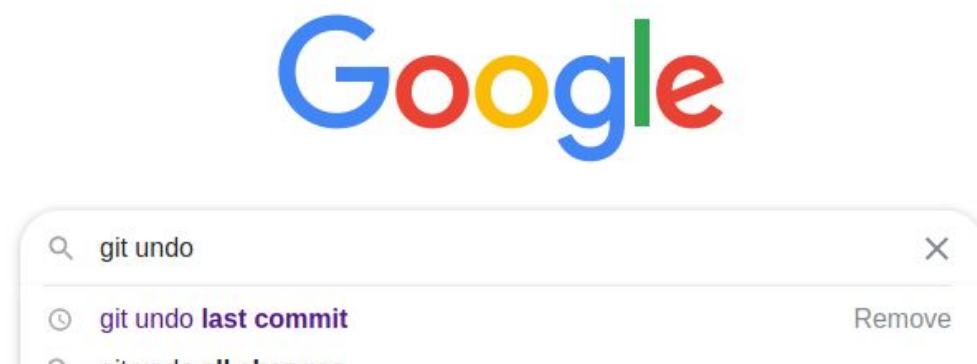
- **Undo the last commit**

```
git reset HEAD~
```



Undoing can feel intimidating

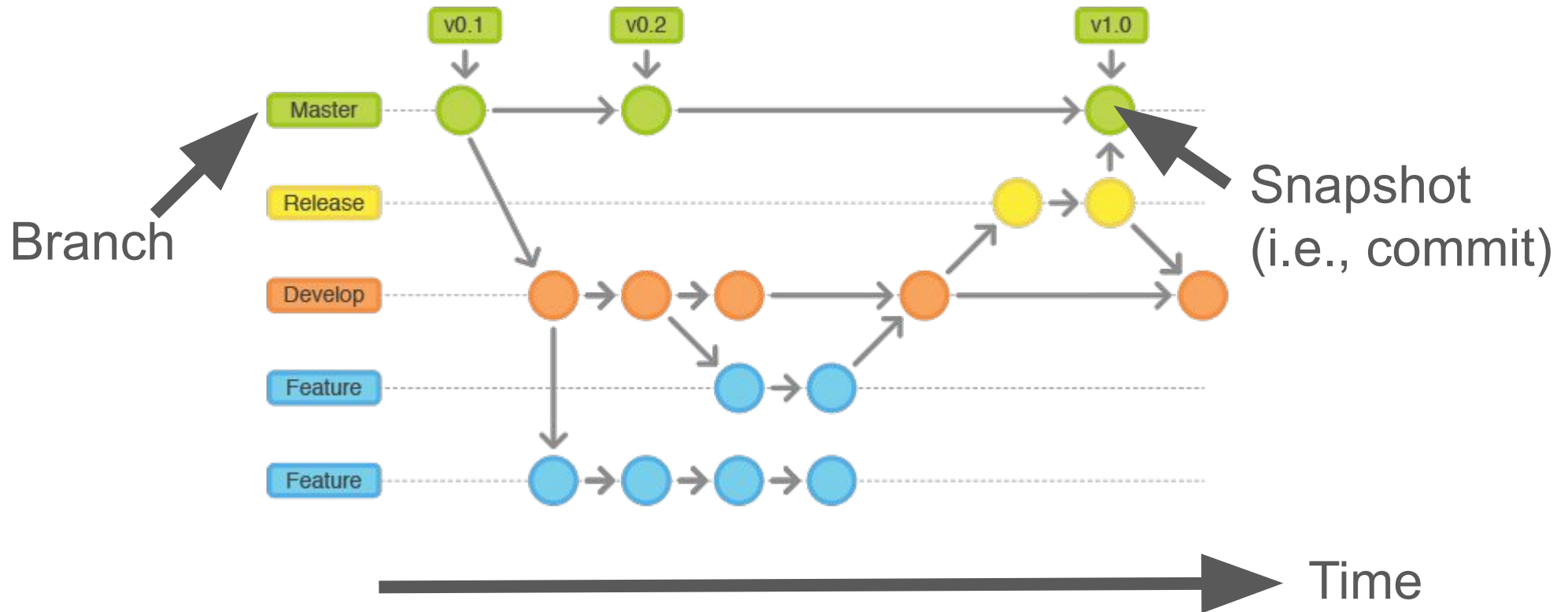
- Don't try to memorize all the commands
- Try it out on a sandbox repo first
- Use `git checkout <commit hash> -b <branch name>` to start a new branch at the point you want to revert to in order to see what it would be like
- For minor recent fixes, try reading the output of `git status`
- Look it up; someone probably had your question before



Branches

- For nonlinear development
- To collaborate
 - Linux: In the last month, “1067 authors have pushed 5,329 commits...”
[<https://github.com/torvalds/linux/pulse/monthly>, as of 2021-07-14]
- To try new things on your own

Branches



Tip: GitHub has a great way of viewing a project's "network"

Branches vs tags



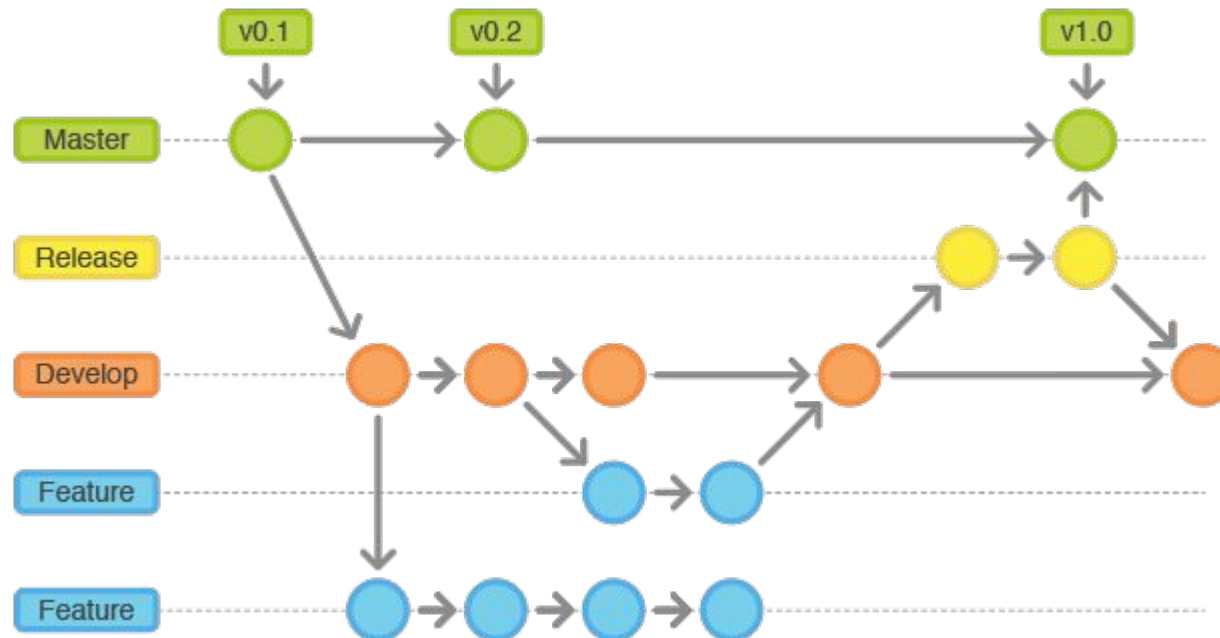
	What is it for?	
Branch	Marks a line of development E..g, a new feature, a collaborator's contribution	
Tag	Marks an important point in history E..g, a version of a software package, a paper publication	

Branches vs tags



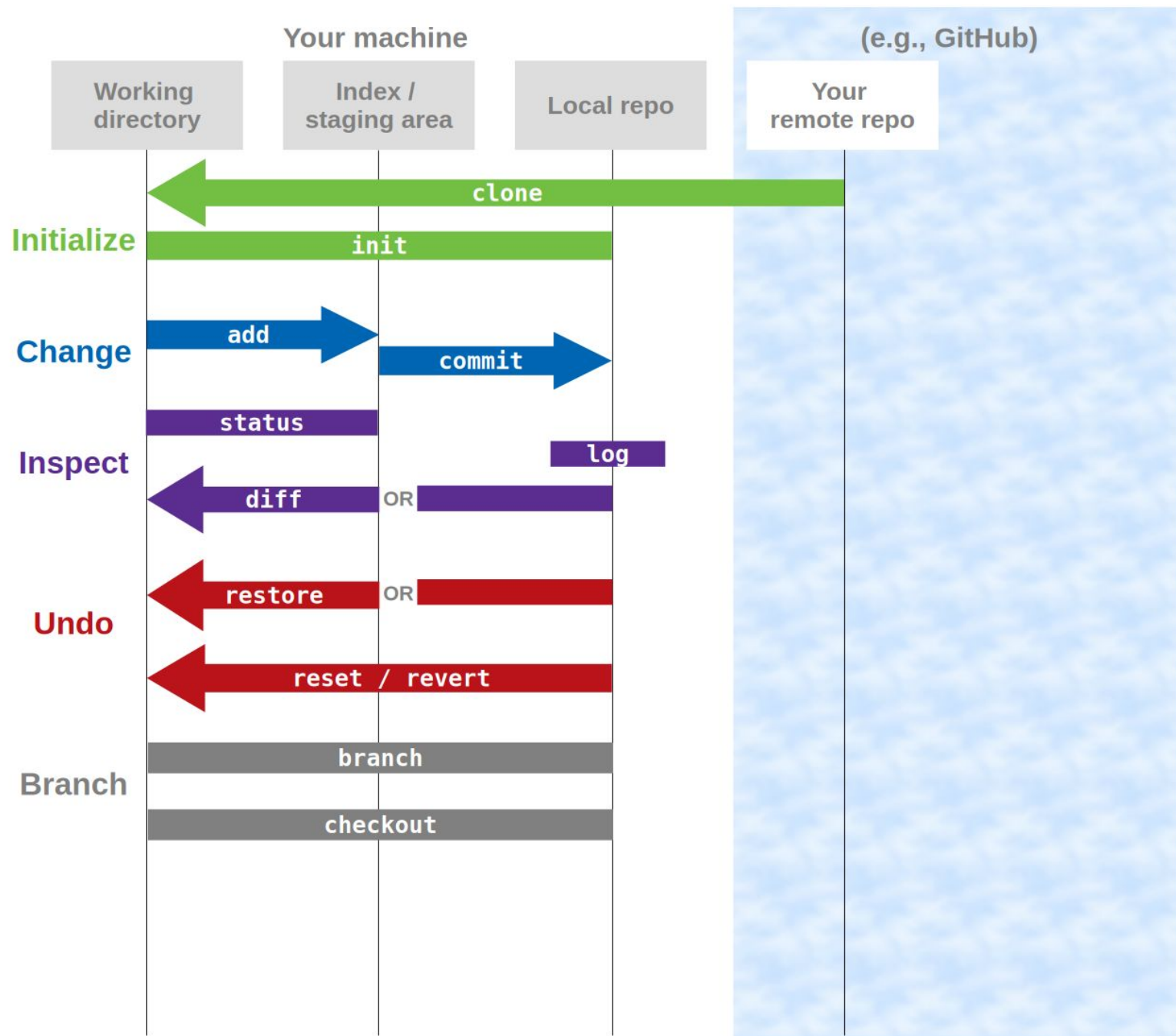
	What is it for?	What <i>exactly</i> is it?
Branch	Marks a line of development E..g, a new feature, a collaborator's contribution	A text file Filename: branch name Contents: commit hash for the latest commit in that branch
Tag	Marks an important point in history E..g, a version of a software package, a paper publication	A text file Filename: tag name Contents: commit hash for the commit when the tag was created

Branches



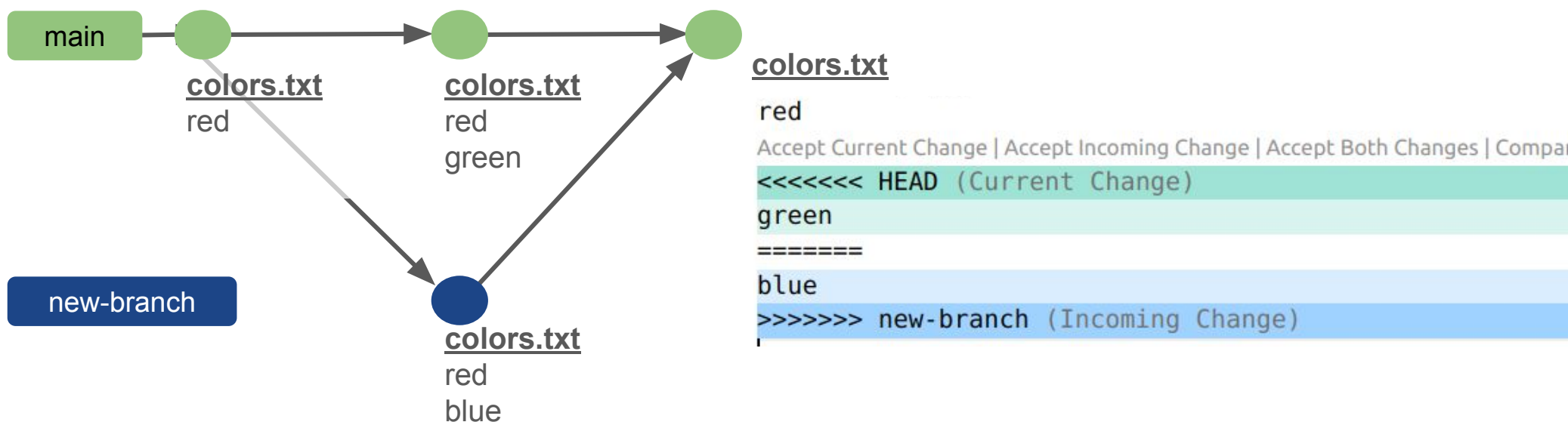
- See which branch you're on
`git branch`
- Start a new branch
`git branch <branch name>`
- Change branches
`git checkout <branch name>`
- Merge a branch into your current branch
`git merge <branch name>`

Branches



Merge conflicts

- What if **several people** edit the same line of code?
(or **you** edit the same line of code on different branches)
→ **merge conflict**
- Someone needs to **manually resolve it**



Goals

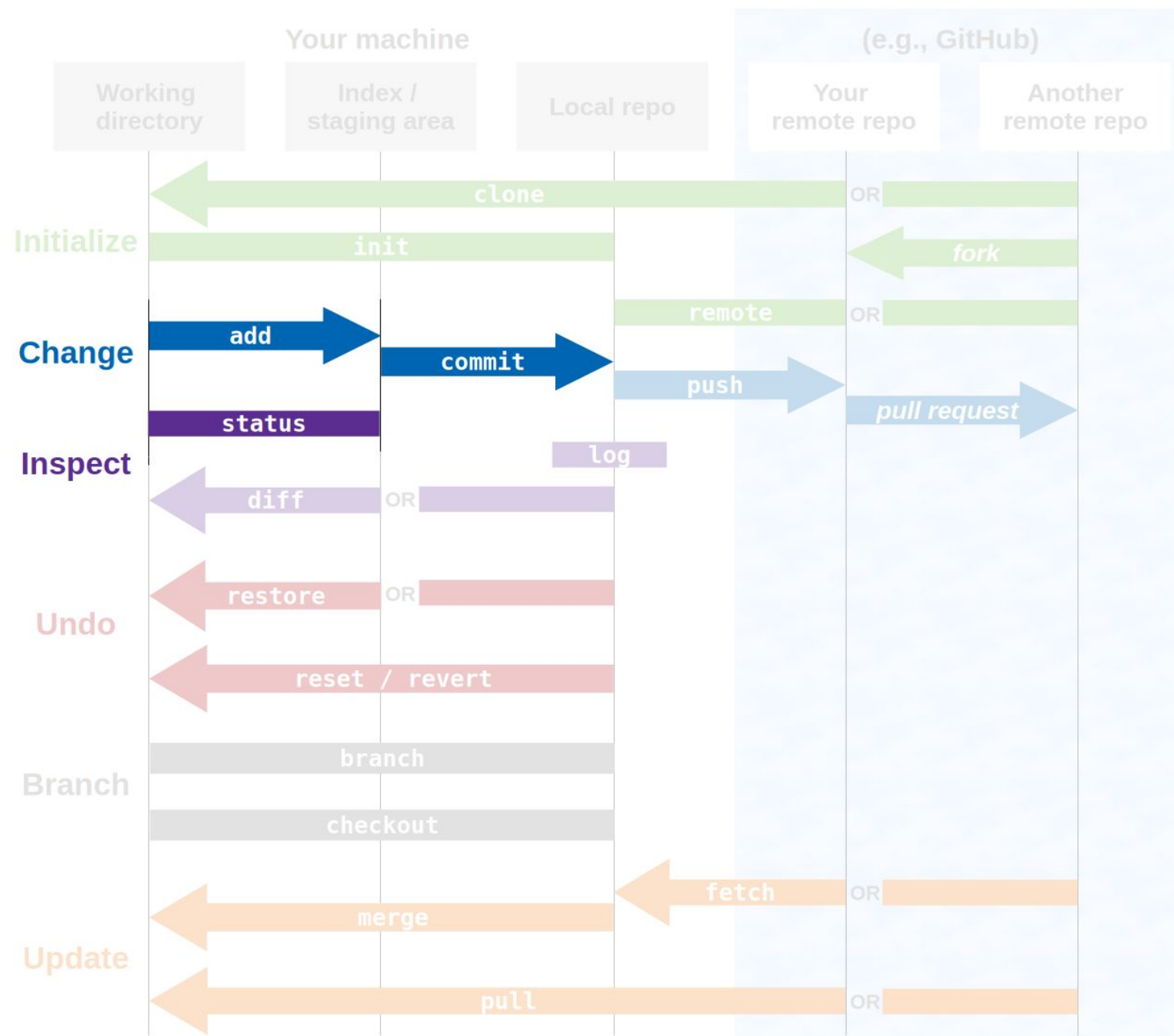
Part 1

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

Part 2 (next week)

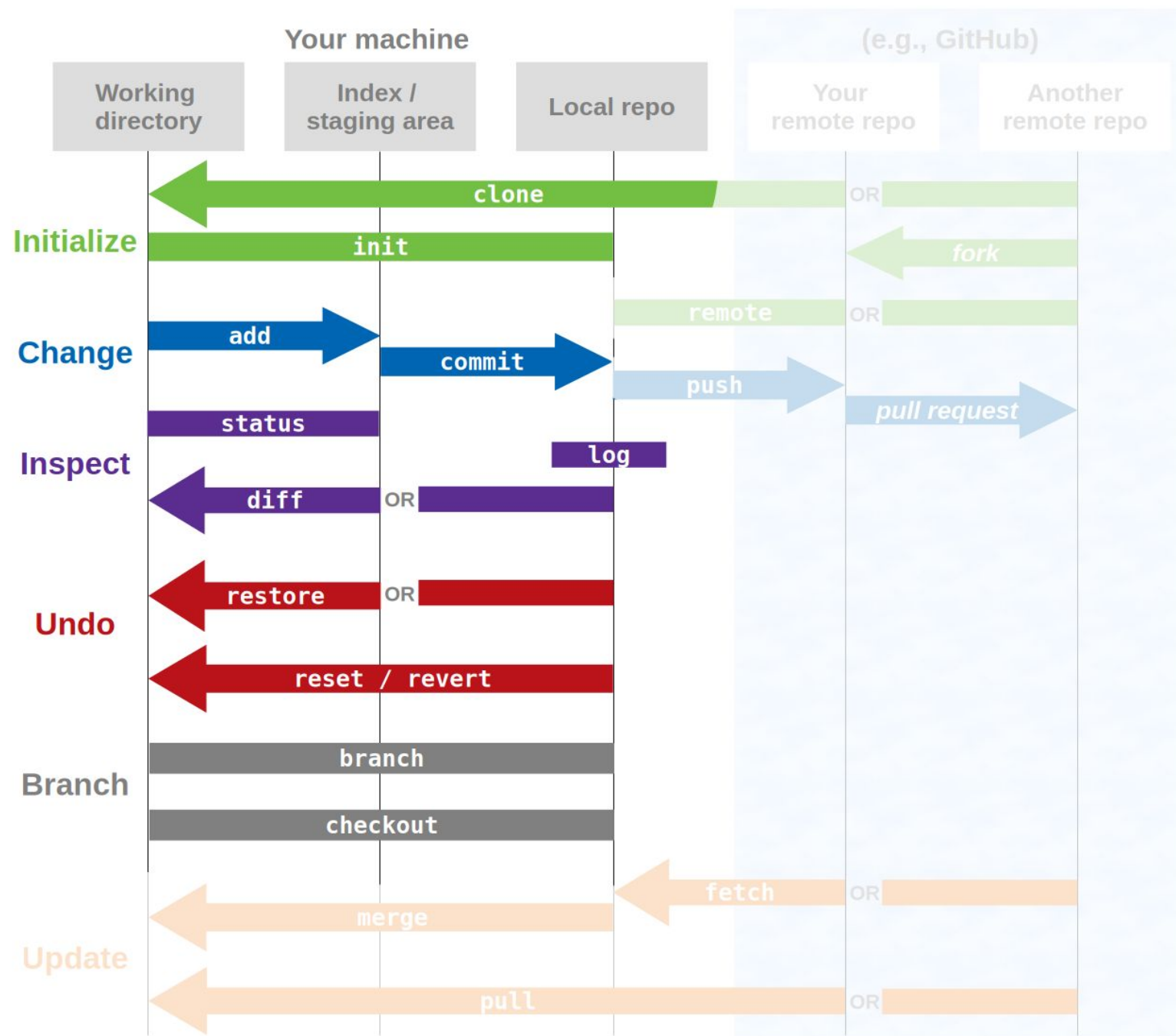
- Share your work on GitHub (simple / linear collaboration)
- Collaborate through GitHub (complex / nonlinear collaboration)
- Navigating GitHub
- Try it out!

The commands you'll need

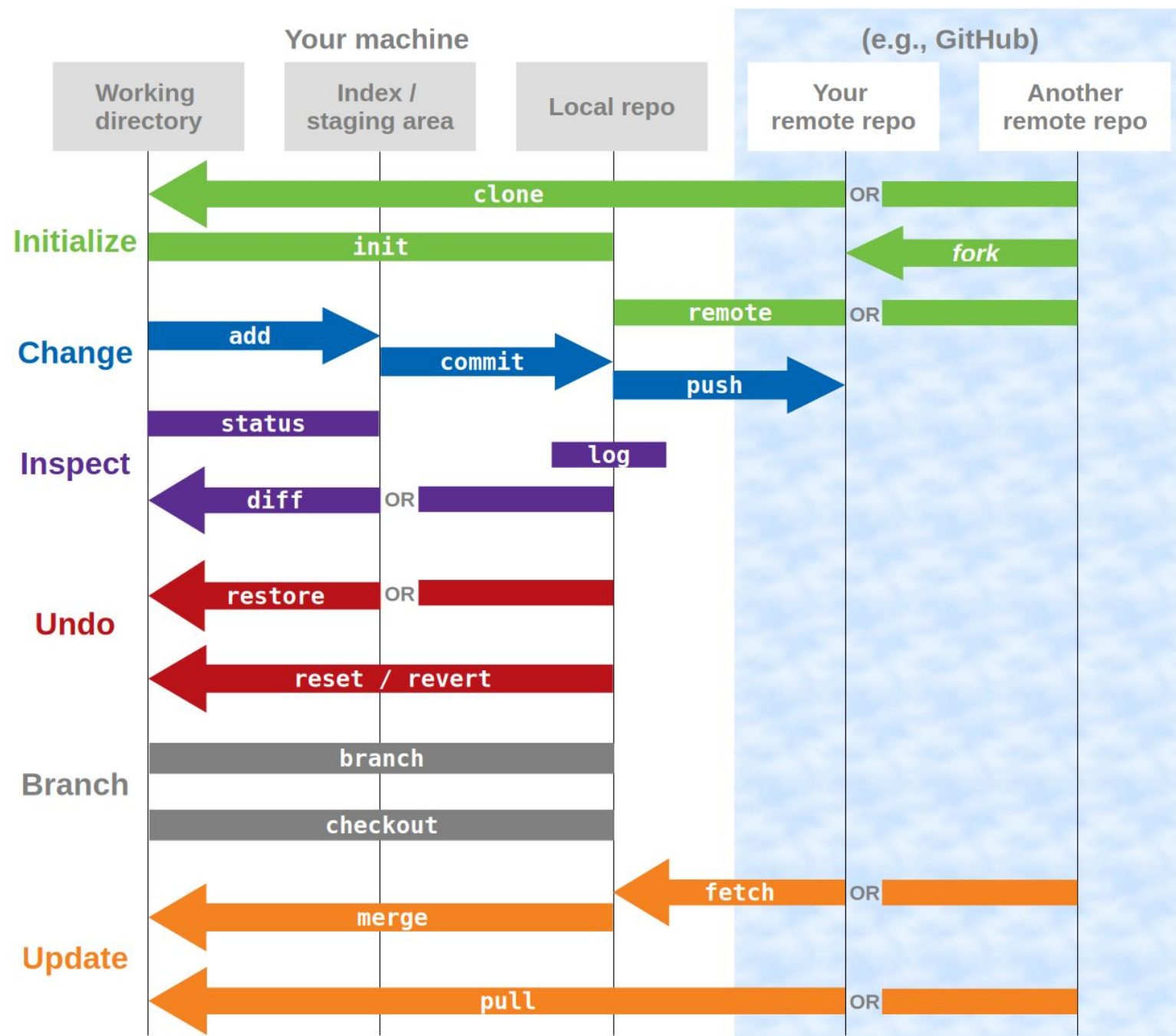


(Everything else you can look up when you need it)

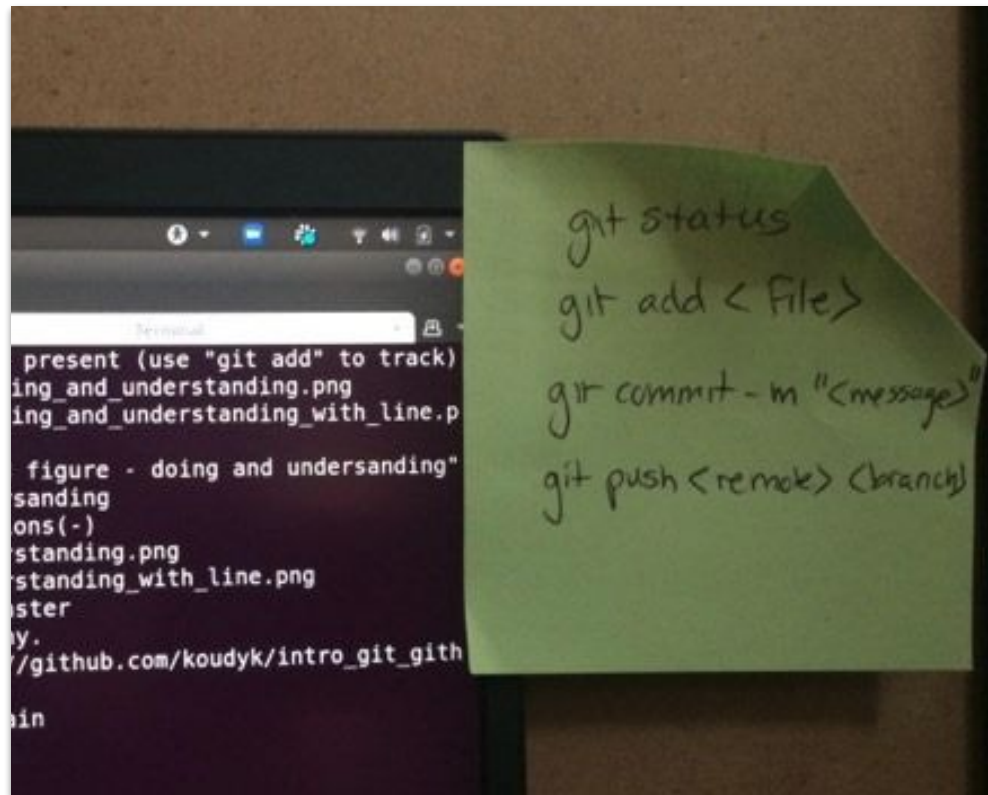
The commands we covered today



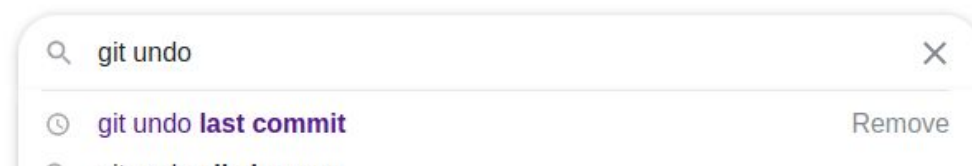
The commands we'll cover today and next week



My advice



Google



Goals

Part 1

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

Part 2 (next week)

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

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....>

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>
- www.phdcomics.com 15 - <https://en.wikipedia.org/wiki/Git>
- McElreath, R. (2020, September 26). Science as amateur software development [video]. YouTube. • Science as Amateur Software Development

The end

