Using Git/GitHub for Version Control

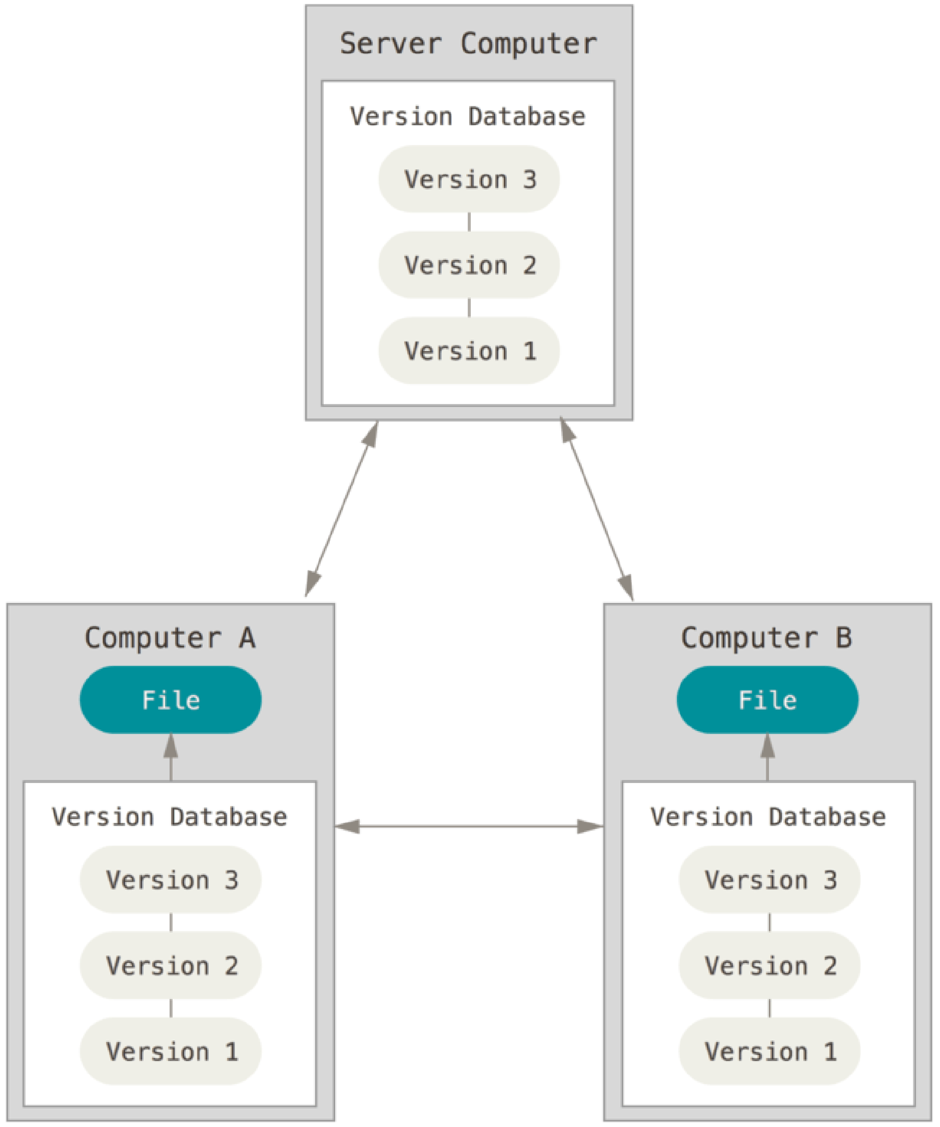
Presenter: Cole Brookson

Date: 2022-02-17

What is Version Control?

Version control is the process of tracking and amanging changes to software code 💻

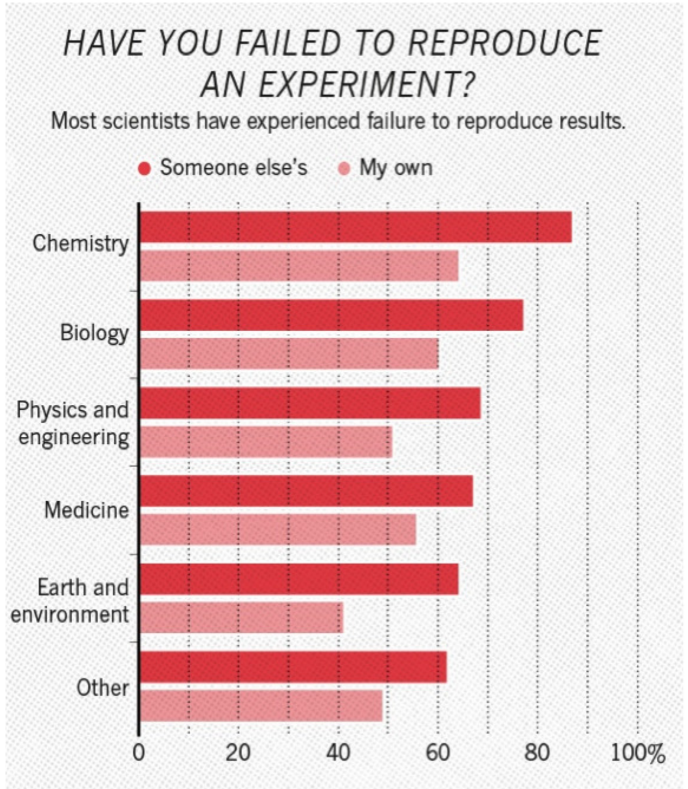
- Distributed Version Control Systems (such as Git) take "snapshots" of the changes made to an entire repository

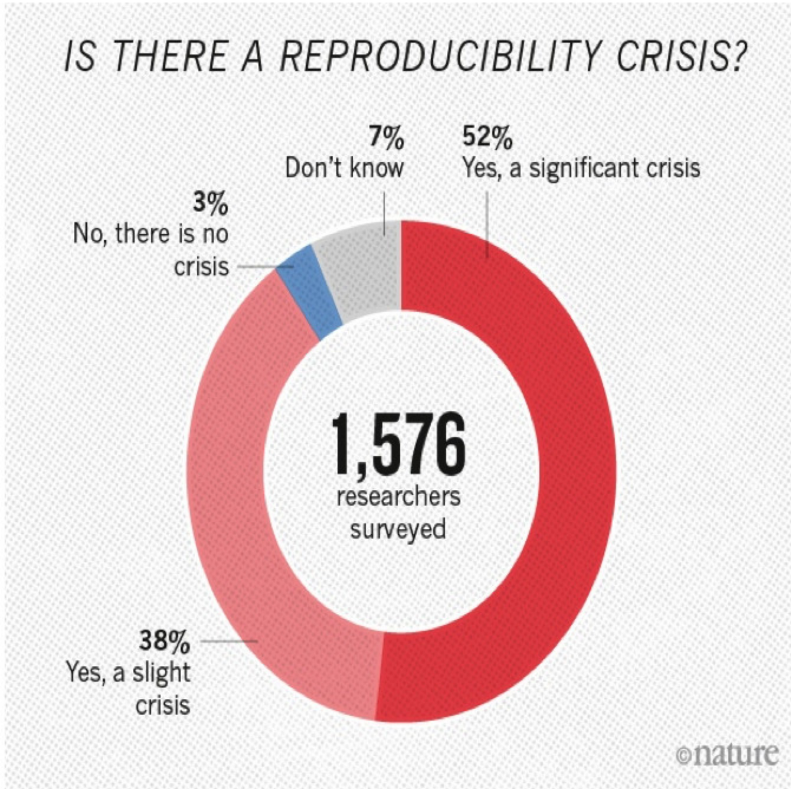


Why Bother?

Reproducibility for:

1. You!!
2. Your Collaborators
3. Others





Number of respondents from each discipline:
Biology **703**, Chemistry **106**, Earth and environmental **95**,
Medicine **203**, Physics and engineering **236**, Other **233**

©nature

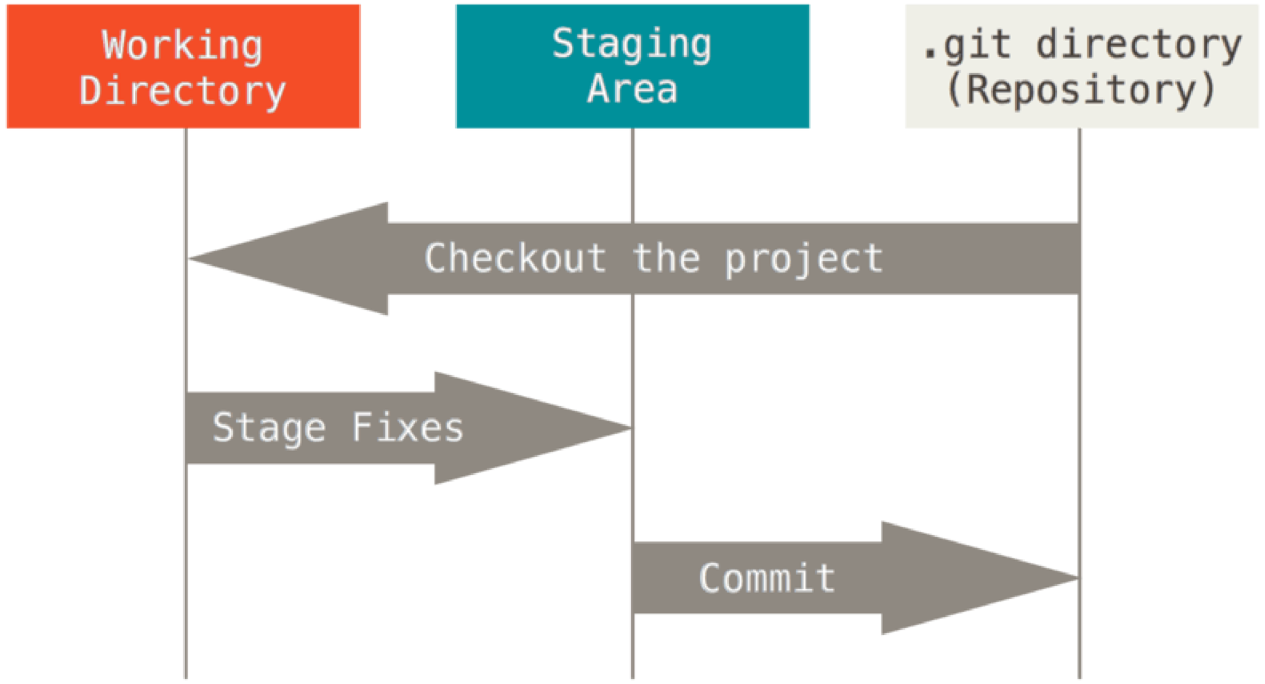
Well, how do we do it?

git

GitHub

Version Control System (Git)

- Git has three main states your files can reside in:
  - `modified`
  - `staged`
  - `committed`
- Your files move through these stages as you make changes



Why Git from the command line?

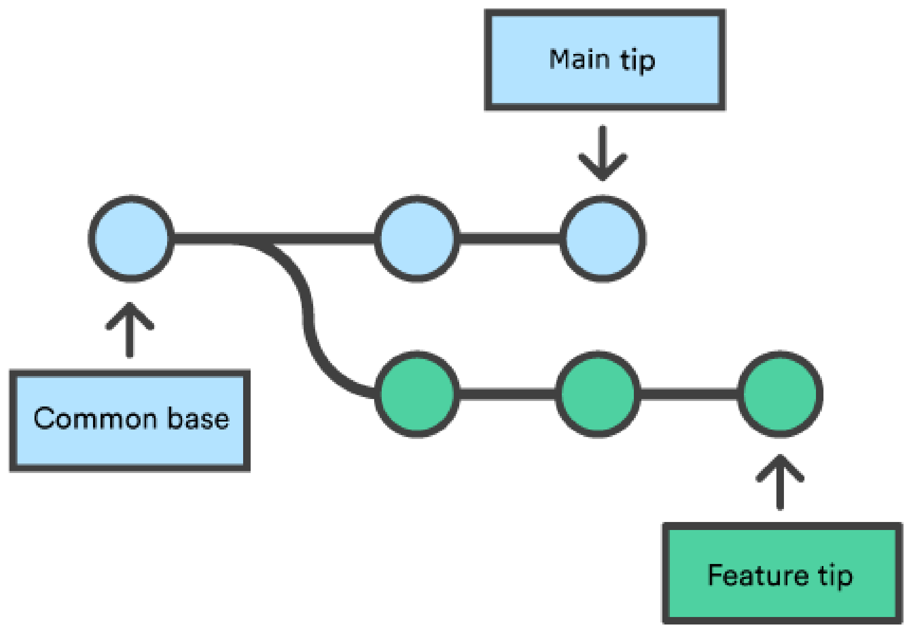
- It's the only place you can run *all* Git commands
- If you know the command line version you can probably figure out a GUI version - the opposite is not necessarily true
- You might have a preference of GUI, but *all* users can use command line tools
- Interacting with servers needs to be done via command line, so you might as well learn how to do it on your own machine
- Language-specific plug-ins (i.e. Git for RStudio) force you to open the IDE for that language every time you need to make a change to a file, even if it's not in that langauge

Cloud-based Git repository hosting service (GitHub)

- A for-profit company that hosts Git repositories
- Free to use for public repositories (makes it *very* popular for open-source projects)
- Provides a nice interface for viewing your repositories contents
- Allows you to publish items with DOIs (links with Zenodo for this)

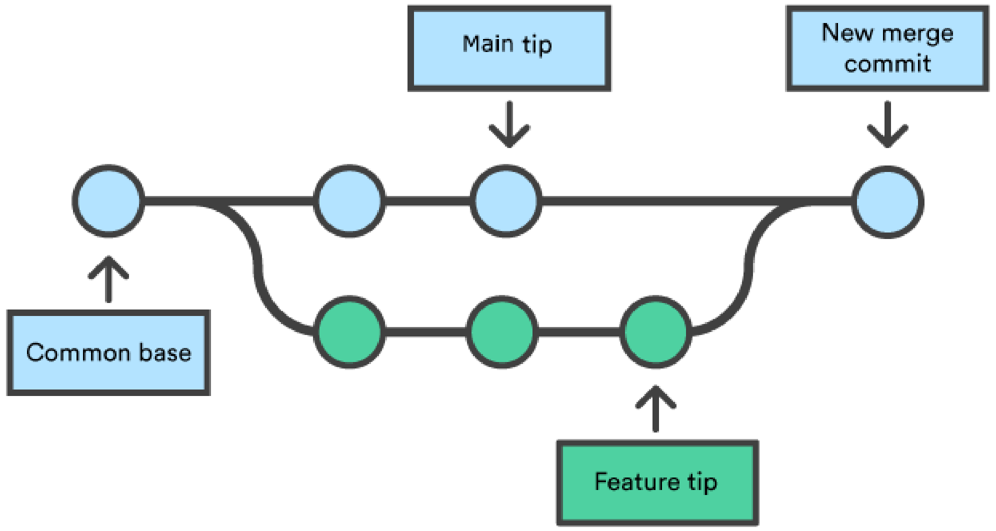
Important Concept: Branches

- When you are making lots of changes, you don't necessarily want to work on the "stable" branch
- This is especially important when collaborating with others who rely on having a working code base



Important Concept: Merging

- Merging is what allows us to make the changes that happened on the "feature branch" present on the main branch, once we're sure we like them
- This can get complicated with large numbers of files, but the great thing about Git is you can **always** go back if you mess up!



Important Concept: Reverting

- We might make mistakes, and it's important to know how to "undo" those mistakes
- There are often two scenarios:
  - You want to keep some of the work you did since the "bad" commit
  - You don't want to keep any of it (usually one or two commits back)
  -