Collaboration

Hamish Gibbs

Collaboration

- Most programming isn’t solo.
- In companies / research, you are often working on a small part of a larger project.
- To collaborate - you need to share code!

Version control

- To collaborate, we need:
  - A place to store a shared version of our code.
  - A way to track changes to different parts of the code.
- Solution:
  - A “repository” to store our code
  - A version control system to track changes to the code

Local vs. Remote code

- A repository stores the code for a specific project.
- There are two different types of repository:
    - **Local:**
        - *Think: a file folder on your computer.*
    - **Remote:**
        - *Think: a GitHub repository*

Local vs. Remote code

- With version control, I want my **local** changes to be reflected in the **remote** repository.

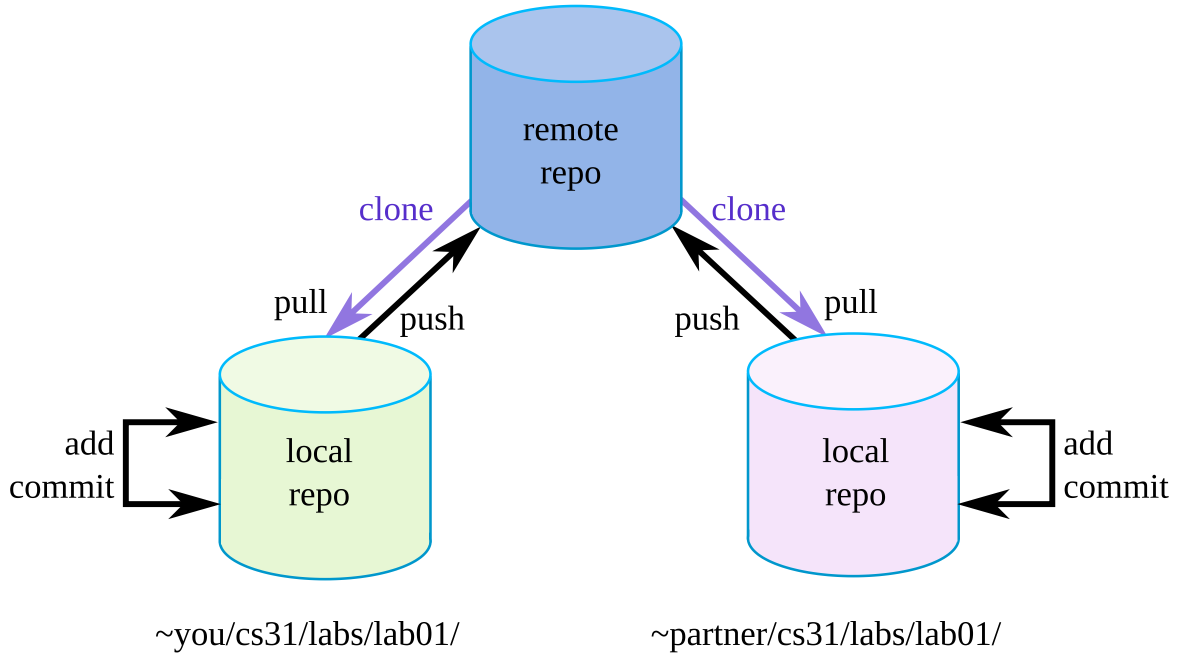

Image credit: Swarthmore Computer Science

git and GitHub

- **git** is an open source version control system.
  - Purpose: recording and reconciling changes to code.
- **GitHub** is a place to store **remote** repositories.
  - Here is the repository for this course!

Google docs

- `git` is kind of like Google Docs.
  - I make a change to a document.
  - You make changes to the same document.
  - Our changes are combined together.
- Except: `git` is *very manual*.

Version control

- With `git` you need to be explicit about:
    - Saving changes (called ‘committing’).
    - ‘pushing’ **local** changes to the **remote** repository.
    - ‘pulling’ changes from the **remote** repository.
    - ‘merging’ changes together.
- ***Good question:*** *Why does this have to be explicit?*

Aims: tomorrow

- After the challenge, we will each:
  - Push our code to a shared repository.
  - Compare solutions.
- Shared repository URL:
  - soda_python_foundations_challenge_2024
- Please give me your GitHub usernames today!

Diving deeper into `git`

- This is a very basic introduction to `git`
- Advanced uses of `git` / GitHub:
  - Collaboration on large projects, automated testing, issue / version tracking
- Introductory uses of `git` / GitHub:
  - Storing code / not losing what you have done
  - **Building a ‘portfolio’ of coding projects**

Tip

- Github’s Education Benefits give you access to a lot of free stuff!
  - GitHub Copilot
  - GitHub Copilot Chat
  - Free web hosting

Exercise #1

1. Setup a GitHub account.
2. Create a GitHub repository.
3. Commit your **local** code (Add, Commit).
4. Push your code to your **remote** repository (Remote).

Extra

- If you breeze through the GitHub setup / repository creation:
  - Try editing files in the **remote**, then `pull` the changes to your **local** repository.
  - Try creating a **local** `branch`, then `push` the branch to the **remote** repository.
  - Try setting up a repository with someone else, making changes to the same file.