Lab 3: (basics of) Git

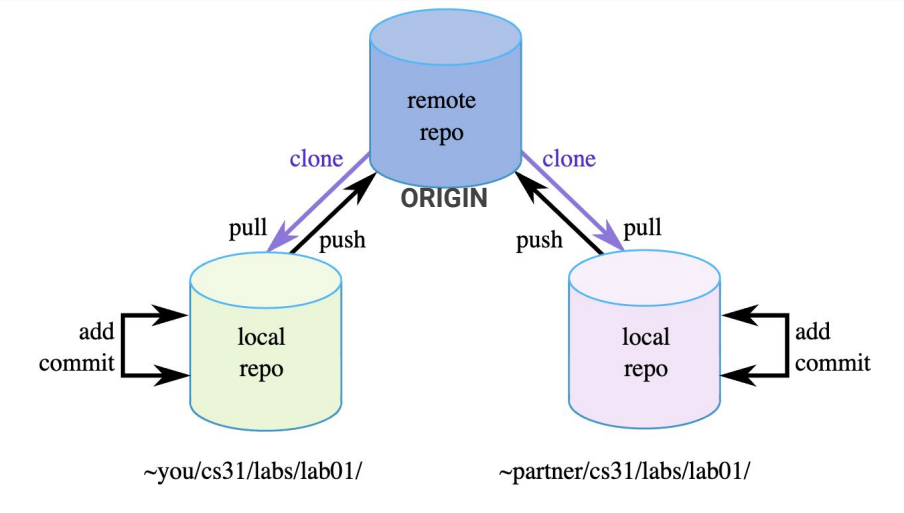
What is Git and Version Control?

- **Git - Version Control Systems (VCS)**
  - tool that stores different versions of project files
  - helps track changes, enables reverting to earlier versions
  - makes collaboration easy

- **GitHub** is a website that hosts git repositories

Setup

- Remote repo - central copy of project, hosted on a remote server (like GitHub)
- Local repo - developer's 'working copy' of repo



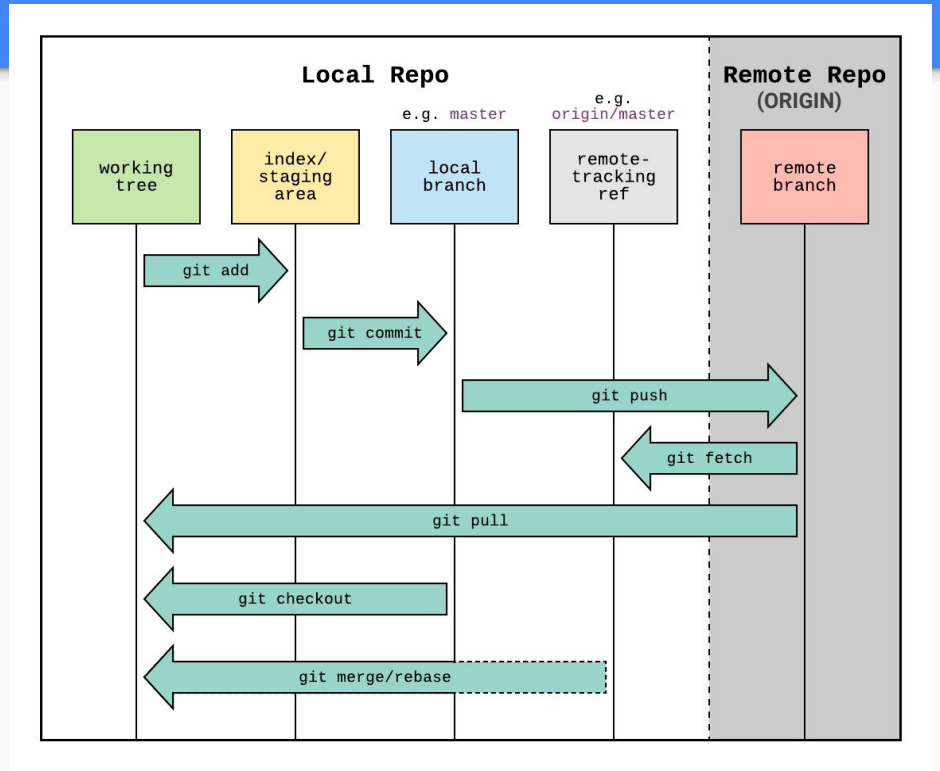
Setting up git

- Git Local Setup
  - Mac and Linux users generally have git already installed on their system
  - install git on Windows: https://help.github.com/articles/set-up-git/#setting-up-git

- GitHub Account Creation
  - Go to https://github.com/
  - Sign Up/Create new account if you don't already already have one.
  - Sign Up for GitHub student! (gives you some additional features)

Cloning a repo(sitory)

- In order to work with a repository, you first need to make a copy that runs on your computer.
- Making this copy is called “cloning”.



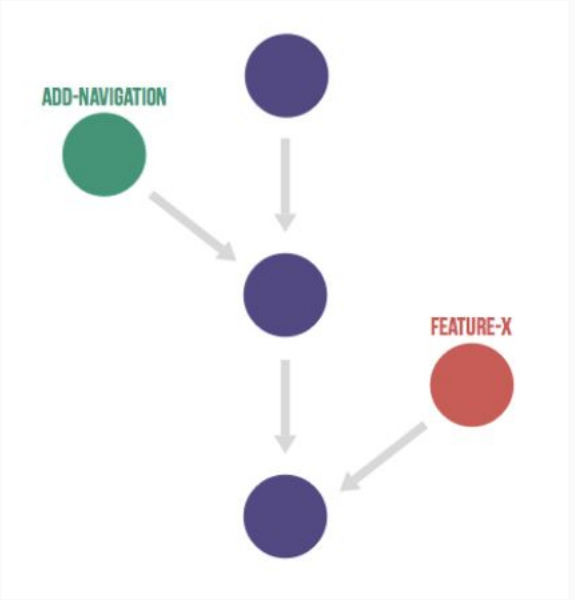
DEMO: Git

Basics (Making changes to a repo)

- **Pull**: Use **git pull origin main** to pull any latest changes from the remote repo to your local repo.
- **Status**: Use **git status** command to see the staged (shown in green) and un-staged (shown in red) files in your local repository.
- **Staging**: Use **git add <filename>** to stage a changed file for commit
- **Commit**: Use **git commit -m “<your message here>”** to commit the staged files.
  - Keep your message short, descriptive and specific.
- **Push**: Use **git push origin main** to push all the changes made locally to the origin.

Basics (Branching & Merging)

- **Branching**:
  - **git switch –c <branch name>** to create a new branch
  - **git switch <branch name>** to switch to a different branch
- **Push**:
  - **git push origin <branch name>** to push any changes made on this branch.
- **Merging**:
  - **git merge <branch name>** to merge changes in <branch name> to your current branch.



Basics (logging)

- **Log**: Use **git log <options>** to view the history of changes

- Different options, e.g.:
  - **git log --help**
  - **git log --decorate --all**

More references available: https://swcarpentry.github.io/git-novice/

Potential for confusion (empty folders)

- Git is *not* just like the folder structure on your computer, but it looks like it, at first glance, which is problematic.
- In Git, you cannot just create a new, empty folder.
- The file is the basic unit of organization in Git, as its primary purpose is content tracking.
- So everything starts with a readMe file (even if it is empty).

Any questions?

- Anything at all!
- Don’t be shy!