Git, GitHub and version control

*** - Instructor notes
** - Important

* - Consider this

Note: I provide more information than you will need in this course for your own studies outside of class, think of it as reference. Be curious and ask questions.

1. Introduction

Git and GitHub – two essential tools in modern software development.
Git and GitHub are used for version control.

Version control is like keeping track of changes in your code and being able to go back in time to a previous version of it.

Recall the editing history function on google docs. You are able to revert back to a different version of that file
Git is the tool that tracks changes to all files and GitHub is an online tool that applies git (it's like google drive with git as the magic behind it).

2. Git?

Git was created by Linus Torvalds in 2005 to manage Linux kernel development and keep track of changes to be able to rollback to working versions.
It's a distributed version control system, meaning every person with access to a repository has full access of its history.
Terms to learn:
- Repository: the project folder that is tracked by git. (Structured folder where your code lives)
- Working directory: The "folder" where you are working.
- Staging area: Area where you prepare "stage" files to commit.
- Commit: creating a snapshot of the current staged files.

3. Basic Git Workflow

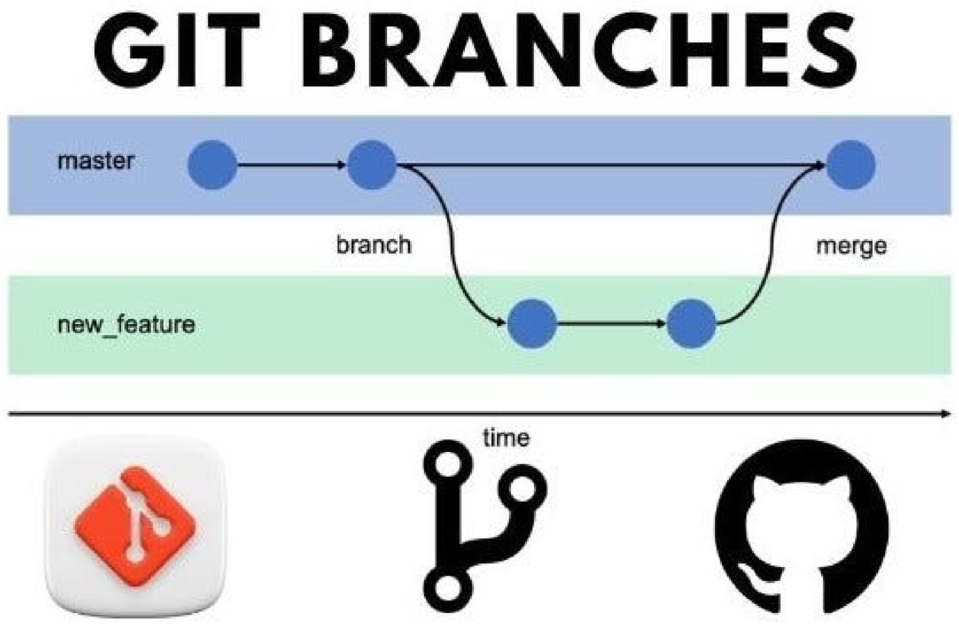
These are the basic commands (actions) in git that you should know:
- `git init`: Begins tracking of the folder in git.
- `git status`: Shows you information regarding what files have changed what is staged (both added and removed files will show details).
- `git add`: Stage changes (add changes to a commit)
- `git commit`: Save your snapshot.
- `git log`: Show commit history.

Demo these in a code space terminal or use screenshots. Emphasize how this helps developers avoid losing work.

4. Branching and Merging

Branches let you work on different sections of code to reduce the chances of having “conflicts” or edits of the same section of code of which git does not know which one to keep.

Think about it this way: git allows you to make a copy of main (master) so that you can work and that is called a branch. You can merge branches back to root so that the code edits are reflected in the main (master) branch.



Key commands:
- `git branch` to create and view branches

- `git checkout` to switch to a different branch
- `git merge` to combine them

Note: Sometimes, you'll get merge conflicts – this occurs when despite working on different branches two people change the exact same sections of code and git wants to make sure it keeps the right version whether its one or the other or a combination of both.

5. What is GitHub?

GitHub is a website that hosts Git repositories and adds even more collaboration tools.
It's like Google drive but for developers and their code.

Other tools like GitHub exist, GitLab and BitBucket, all of which use git as the version control utility. If you understand git, switching between tools is simple.

Show GitHub, sample repo and branches

6. GitHub Basics

GitHub allows to to connect a local (on your pc only) repository and upload it to an online version. You can also create a repository on GitHub itself and clone it (similar to downloading).

To create repo and download: (might do this in camp)
- Create a repo on GitHub.
- `git clone` to download it.

To create local git repo and then push to GitHub: (you will not be doing this in summer camp but it is good to know)

- `git init` - Initialize or make a regular folder a git folder

- `git remote add origin https://github.com/yourusername/yourrepo.git` - Location of remote git repository

- `git push -u origin main` - Push to the remote git.

7. Collaboration Workflow

GitHub lets teams collaborate efficiently, something you will be doing in the third week the following is the general steps taken when collaborating in a group:
- Fork: Copy someone's repo to your own account. (if you want to have your own copy)

or you can

- Branch: have your own branch you work on that you can later merge (combine) with the master branch.

Or you can work directly on the Main branch (dangerous) with a lot of care to prevent conflicts.
- Clone: Download the repo (only do this once if you do not have the repo available to work with on your computer. **This will not apply here, codespaces automatically clones for you**).

Then:
- Make changes and push them:

  - Pull updates (good practice to make sure you have latest code) (git pull)
  - Stage changes (git add)
  - Commit changes (git commit) remember to add comments to explain changes.
  - Push changes (git push)

- Pull Request: Ask the original repo to accept your changes. (when merging branches)

8. Ways to work with git and GitHub effectively

Habits that make Git/GitHub work better for you:
- Write short and clear commit messages.
- Name branches after features or issues that are being resolved.
- Use `.gitignore` to exclude unnecessary files. **you can learn this on your own**
- Always write a README and in some cases include a LICENSE

** README.md Is a way to describe why your repo matters, what it contains and describe it in detail. In GitHub it is also a way to Initialize the repo, not adding a README will require you to initialize (git init) locally after you clone.**

9. Real-World Applications

Git and GitHub are used extensively by major companies and open-source organizations.
Show GitHub repo examples

It simplifies and makes collaboration of very intricate source code. Imagine having 1,000 employees trying to edit a code file not knowing who is changing what. What a headache!

10. Some repo examples

All of these have many collaborators and many branches, each of which is a different feature or issue being resolved.

https://github.com/BetterDiscord/BetterDiscord – Good Discord mod platform

https://github.com/rust-lang/rustup – Rust toolchain installer

https://github.com/hashcat/hashcat – Password recovery tool