Introduction to GitHub Workshop

Angelica Tellez
Vice President of Data Science Club
March 28, 2025

Overview

Git vs GitHub

Terminology

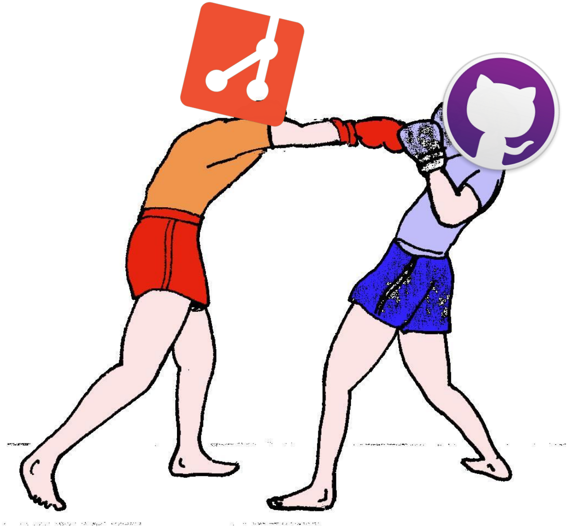
Visual Studio Code

Demo

Resources

Questions

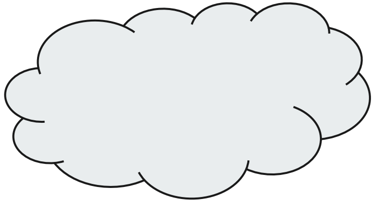
Git vs GitHub

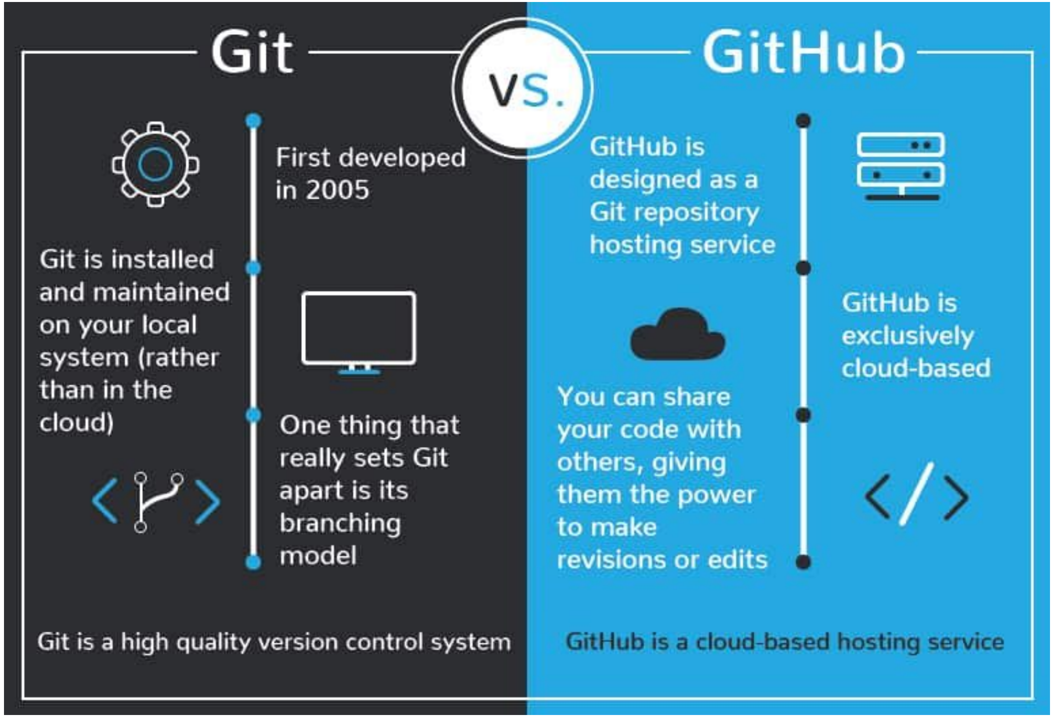


Git vs GitHub

Git is a powerful **version control system** that efficiently tracks changes in files, making it especially useful for collaborative projects where multiple people work on the same files simultaneously.

GitHub is a **cloud-based platform** designed for storing, sharing, and collaborating on code development.





Git

vs.

GitHub

First developed in 2005

Git is installed and maintained on your local system (rather than in the cloud)

One thing that really sets Git apart is its branching model

Git is a high quality version control system

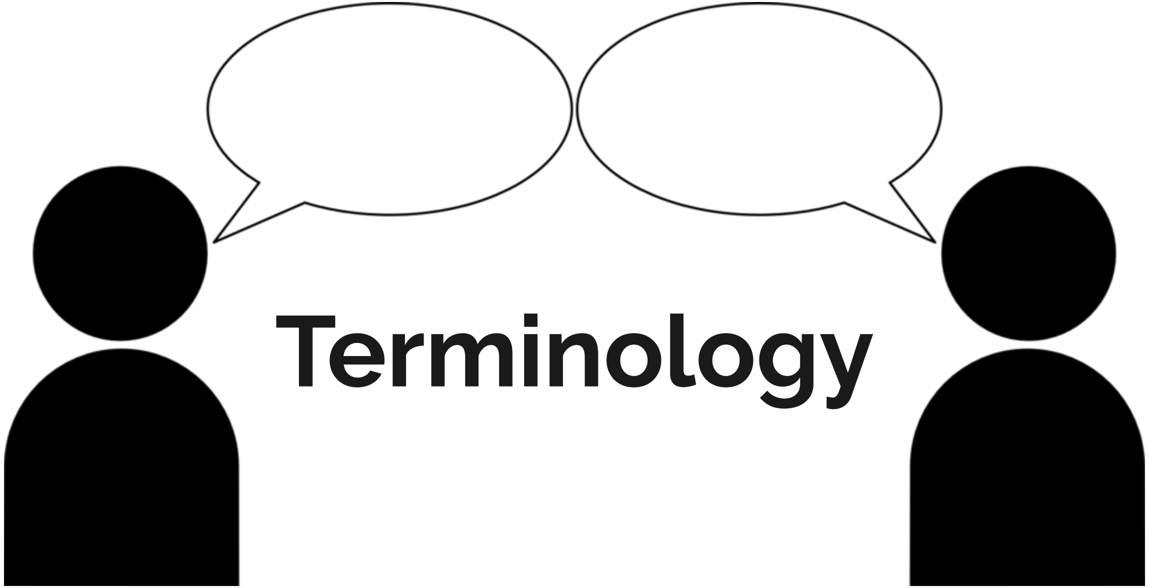
GitHub is designed as a Git repository hosting service

GitHub is exclusively cloud-based

You can share your code with others, giving them the power to make revisions or edits

GitHub is a cloud-based hosting service

Image Source: Devmountain Blog – [Git vs GitHub,]()



Terminology

Repository

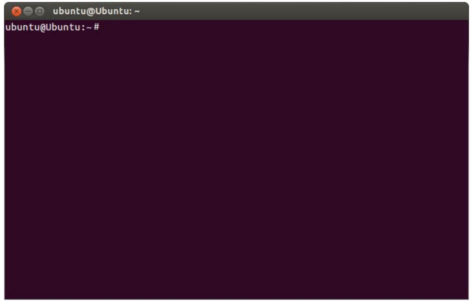
A **repository (repo)** is a collection of source code that tracks changes and enables collaboration.

- **Remote** repository: A version of the repo stored on a remote server (e.g., GitHub), allowing multiple people to work together and sync their changes.
- **Local** repository: A version of the repo saved on your computer. You make edits locally and then push updates to the remote repository when ready.

Command Line

The **command line** is the most widely used and versatile way to interact with Git on any terminal.

- To use Git via the command line, make sure it is **installed** on your local machine.
- You can execute Git commands using **Mac Terminal, Windows PowerShell, or a Linux Terminal**.
- Find a **Git cheat sheet** with common commands on GitHub Education: [GitHub Cheat Sheet].



```
ubuntu@Ubuntu:~#
```

Commits

Every time you modify your code, it's important to track those changes.

- A **commit** is a saved version of your project, capturing the state of files and folders at that moment.
- Each commit becomes **a node on the worktree and** creates a new version in the project, forming a history of changes.

Note: Always save your code before staging and committing to make sure changes are being recorded.

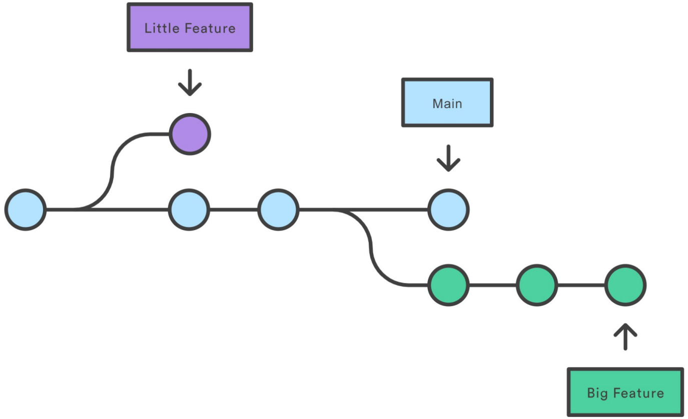


Image Source: Atlassian Tutorial

Branch

A **branch** is a pointer to a commit, allowing you to create separate versions of your code—almost like different timelines or parallel universes!

- The **default branch** is usually "main" or "master", serving as the official version of your project.
- Branches are used for **developing and testing new features** without affecting the main code. They also allow team members to work on different features simultaneously.
- When a feature is complete, branches can be **merged** back into the main branch, integrating changes while preserving project history.

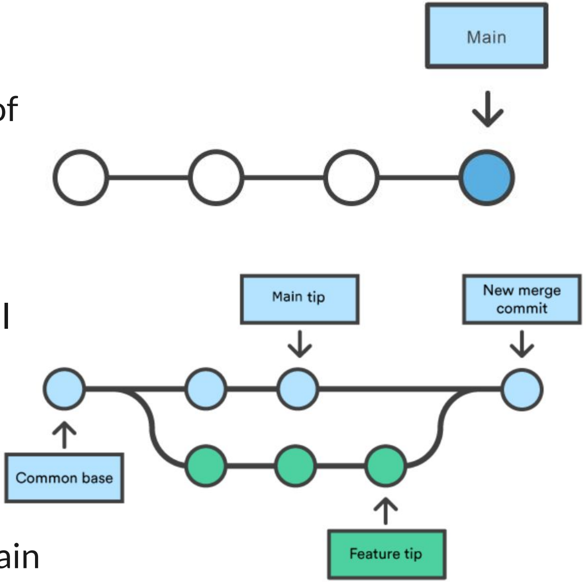


Image Source: Atlassian Tutorial

Remember: If something goes wrong in the main code, you can always revert to a previous commit!

Repository Interactions

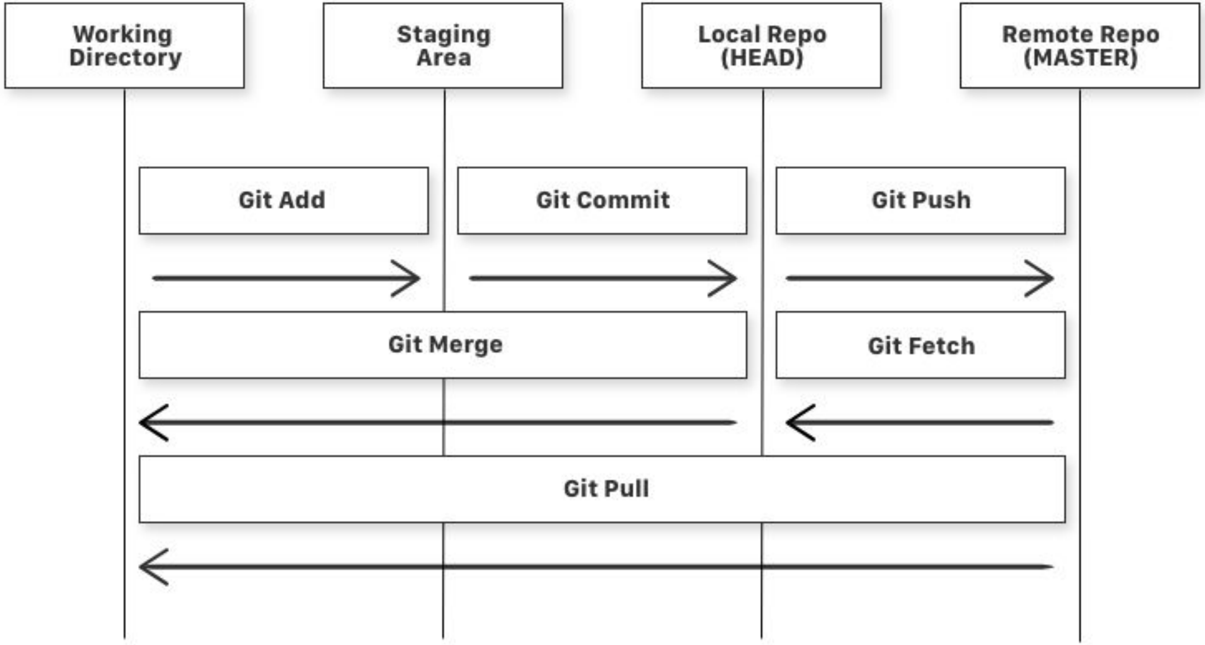
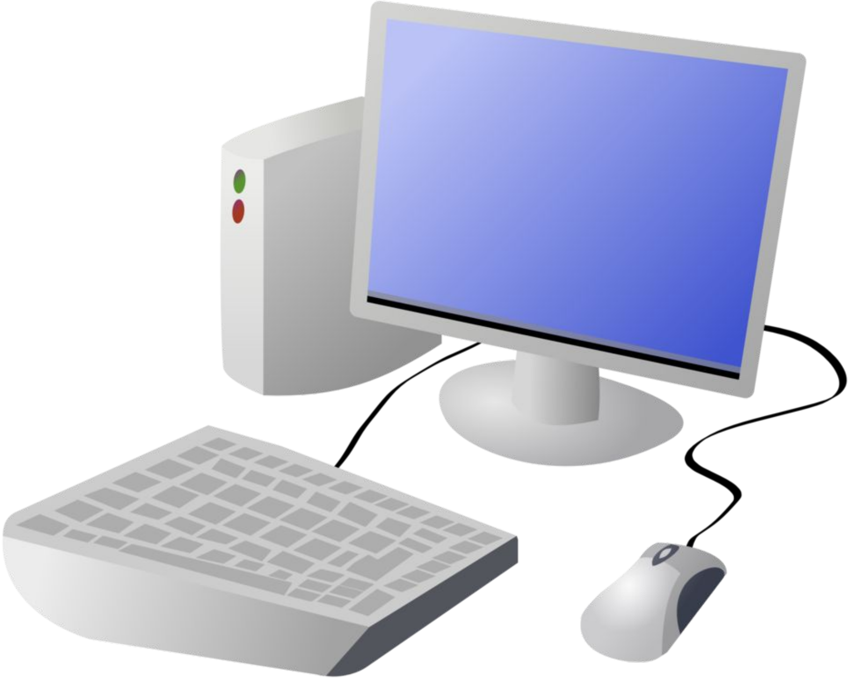


Image Source: TechnoFob

Set Up

1. Sign in or Sign up to GitHub.
2. Download Python or Anaconda , if not already.
3. Open Terminal on your machine.



Demo Time!

Resources

Git Basics:

- Learn the Basics of Git and Version Control
- Using Branches in Git - Atlassian Tutorial
- Git vs. GitHub: What's the Difference?
- Git Cheat Sheet - GitHub Education

Tools:

- GitHub
- Python Downloads
- Conda Package Manager - Anaconda

Questions?