Introduction to Git and GitHub

Git is a version control system that allows you to track changes in your code over time. GitHub is an online platform that hosts Git repositories, enabling collaboration and code sharing.

Creating a GitHub Repository

A repository (or "repo") is like a folder for your project, where you can store all your code and files.

1. Go to GitHub and create a new repository

2. Choose settings for the repository:
   - Public: Anyone on the internet can see your repository
   - No README file: You can add this later if needed
   - No .gitignore file: You can set this up later

Note: You can also create a repository using command line commands.

Cloning vs. Forking

Git allows you to work with existing code repositories in two main ways: cloning and forking.

Feature	Cloning	Forking
Purpose	Creates a local copy of a repository on your computer	Creates a personal copy of someone else's project in your GitHub account online
Ownership	You don't become the owner of the code	Creates a new repository that you own, based on the original project
Collaboration	Useful when you have permission to directly contribute to a project	Ideal when you want to contribute to a project you don't have direct write access to
Impact of Changes	Can push changes directly to the original project (with permissions)	Make changes in your own copy and propose changes through a "pull request"

When to use which

Clone: When you have write access to a repository or want to experiment locally
Fork: When contributing to a project you don't own or participating in open-source development

Setting Up Your Local Environment

Connecting Your Code Base

After creating a repository, you'll need to connect it to your existing code base.

The .gitignore File
1. Create a file named `.gitignore` in your project directory
2. Use this file to specify which files or directories should be ignored by Git
3. This is useful for excluding temporary files, build outputs, or sensitive information

Revealing Hidden Files:
- On Mac: Press Shift + Command + . (period) in Finder
- On Windows: Change settings in File Explorer to show hidden items

Basic Git Commands

Here are some essential Git commands to get you started:

```
git status    # Shows the current state of your files
git branch    # Lists branches
git log       # Displays a history of changes
git add .     # Stages all changes for the next commit
git commit -m "Your message"    # Saves your changes with a descriptive message
```

Connecting to GitHub

To link your local code to your GitHub repository:
git remote add origin [repository URL]
git push -u origin main

Note: You might need to set up authentication between your computer and GitHub, often using SSH keys.

Branching and Merging

Creating and Managing Branches

git branch branchName # Create a new branch
git checkout branchName # Switch to the branch
git checkout -b newBranch # Create and switch to a new branch

Merging Changes

git checkout main # Switch to the main branch
git merge branchName # Merge changes from branchName into main
git push # Update the remote repository with merged changes

Tagging

Tagging is used to mark specific points in your repository's history:
git tag -a v0.0 -m 'Initial version' # Create an annotated tag
git push origin v0.0 # Push the tag to GitHub

You can view tags on GitHub under the "Tags" section of your repository.

Best Practices

1. Don't delete branches that contain work needed for your project
2. Create new branches starting from the main branch
3. Ensure you're on the correct branch before making changes
4. Regularly check your status and current branch
5. Commit and push changes frequently to save your progress