**Introduction to Git and GitHub**

**Terms**

Directory: Folder

Terminal or Command Line Interface (CLI): Interface for Text Commands

Cd: Change Directory

Code editor: A place to write code

Git: The tool that we use to track code changes over time or what we use to write to github

Github: A website to host your repositories online. Being online helps to work and collaborate with people and can serve as a store where you can house your projects to show people.

Repository (Repo): A project or the folder/place where your project is kept.

Local machine: Your laptop

1. **Introduction to Version Control**

**What is Version Control?**

**Definition:** Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

This can also be the management of changes to documents, computer programs, large websites, and other collections of information.

Version control is basically a way that we as programmers track our code changes. We save an initial version of our code into git and then update and keep updating and this will continue to save.

**Importance:** It allows multiple team members to work on a project simultaneously without overwriting each other's work. It also provides a history of changes, making it easier to track progress, revert to previous versions, and understand the evolution of a project.

**Use Cases:** Software development, documentation, design files, etc.

**2. Why use Git?**

**Git as a Distributed Version Control System:** Unlike centralized version control systems, Git allows each developer to have a complete history of the project on their local machine, enabling offline work and easier branching and merging.

**Key Features**:

**Speed**: Git is fast because most operations are performed locally.

**Branching and Merging:** Git supports lightweight, flexible branches, making it easy to work on different features in parallel.

**Widespread Adoption:** Git is the most widely used version control system, making it essential for developers to learn.

**3. What is Git and Github?**

**Git:** A command-line tool that tracks changes in files and allows collaboration through branches and merges.

It is also a version control system/tool that is free and open source. It is the most widely used version control system in development.

**GitHub:** A web-based platform built on top of Git that hosts repositories online, providing additional tools for collaboration, issue tracking, and project management. It facilitates open-source collaboration and code sharing.

**4. Setting Up Git**

**Installing Git on Your System**

**Windows:** Download and install Git from *git-scm.com*. During installation, select options such as using Git from the command line and using the latest version of Git.

**macOS:** Install Git using Homebrew (brew install git) or Xcode Command Line Tools (xcode-select --install).

**Linux:** Install Git via your package manager *(sudo apt-get install git for Ubuntu/Debian or sudo dnf install git for Fedora)*.

**Basic Configuration (git config)**

**Setting User Information**: Configure your Git username and email, which will be used in your commits:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

**Checking Configuration:** Verify your settings with

git config --list, which will display all your configured settings, including name and email.

**Why It Matters:** This information is attached to your commits and helps identify who made changes to the project, essential for collaboration.

**5. Basic Git Commands**

**Initializing a Repository (git init)**

Definition: Initializing a repository sets up Git tracking in your project folder.

Command: Run git init in the root of your project directory. This command creates a .git directory, which contains all the metadata and history for your project.

Why It Matters: Without initializing a repository, Git won’t track your changes, and you won’t be able to use Git commands.

**Checking the Status (git status)**

Purpose: git status shows the current state of the working directory and staging area, letting you know if there are changes that haven’t been committed.

Output:

Untracked Files: Files that are new and haven’t been added to the repository.

Changes to be Committed: Files that have been staged and are ready to be committed.

Why It Matters: Regularly checking the status helps keep track of your changes and ensures nothing is missed before committing.

**Staging Changes (git add)**

Definition: The staging area is where you prepare changes to be committed. Staging allows you to select specific changes to include in the next commit.

Commands:

**git add <file>**: Adds a specific file to the staging area.

**git add .**: Adds all changes in the current directory to the staging area.

Why It Matters: Staging is an intermediate step that allows you to carefully curate what will be included in your next commit, preventing accidental commits of incomplete work.

**Committing Changes (git commit)**

Definition: A commit is a snapshot of the project’s state at a particular point in time, representing a set of changes.

Command: Use git commit -m "Your commit message" to commit staged changes with a descriptive message.

Why It Matters: Commits are the core units of work in Git. They allow you to roll back to a previous state if something goes wrong and serve as a log of your progress.

**Viewing Commit History (git log)**

Purpose: git log displays a chronological list of commits in the repository.

Output:

Commit Hash: A unique identifier for each commit.

Author: The person who made the commit.

Date: When the commit was made.

Commit Message: A description of what changes were made.

Why It Matters: Reviewing commit history helps you understand the evolution of a project and trace when and why changes were made.

**Other git Commands**

Git clone: Bring a repo that is hosted somewhere like github into a folder on your local machine

Git push

Git pull

Git remote add

Git help config

Git config –help

**6. Working with GitHub**

**Github**

GitHub is a developer platform that allows developers to create, store, manage and share their code. It uses Git software, providing the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

**Creating a GitHub Account**

Process: Guide students through signing up at GitHub.com. Discuss the importance of choosing a username that represents their professional identity.

Why It Matters: A GitHub account is essential for accessing the platform’s features and collaborating with others.

**Creating a Repository on GitHub**

Process:

Click the “New” button on the GitHub dashboard.

Fill in the repository name, description (optional), and choose whether the repository is public (visible to everyone) or private (visible only to collaborators).

Optionally, add a README file, .gitignore, and a license.

Why It Matters: Creating a repository on GitHub is the first step to hosting and sharing your code online, making it accessible for collaboration or portfolio purposes.

**Connecting a Local Repository to GitHub (git remote)**

Definition: A remote repository is a version of your project hosted on the internet or another network. You can link your local repository to it for collaboration.

Command:

git remote add origin https://github.com/yourusername/your-repo.git

Why It Matters: Connecting your local repository to GitHub allows you to push and pull changes, keeping your work synchronized with the online version.

**Pushing Changes to GitHub (git push)**

Command: Use git push -u origin master to upload your commits from the local repository to the GitHub repository.

Explanation:

**-u origin master**: Sets the remote branch master as the default for future pushes.

**origin**: The name of the remote repository.

Why It Matters: Pushing changes ensures that your work is backed up online and accessible to collaborators.

**7. Cloning and Collaborating**

**Cloning a Repository (git clone)**

Definition: Cloning is the process of copying a repository from GitHub (or another remote) to your local machine.

Command:

git clone https://github.com/username/repo-name.git

Why It Matters: Cloning is the first step in collaborating on an existing project, allowing you to work on a copy of the repository.

**Pulling Changes from a Remote Repository (git pull)**

Definition: Pulling combines git fetch and git merge, retrieving the latest changes from the remote repository and integrating them into your local branch.

Command:

git pull origin master

Why It Matters: Regularly pulling changes keeps your local repository up-to-date with the remote repository, preventing conflicts and ensuring you’re working on the latest version.

**Basic Collaboration Workflow (Forking, Pull Requests)**

**Forking**: Forking is creating a personal copy of someone else’s repository on GitHub. It allows you to experiment with changes without affecting the original project.

**Pull Requests**: A pull request is a way of proposing changes to the original repository. After forking and making changes in your copy, you can submit a pull request to the original repository owner for review.

Why It Matters: Understanding this workflow is essential for contributing to open-source projects and collaborating with others on GitHub.

### Create a GitHub Account

**Steps:**

* Go to [https://github.com](%20https://github.com)Click “Sign Up”
* Choose username, email, password
* Verify email address
* Optional: Set up a profile picture

Git Setup After Installation

**Open terminal or Git Bash and run:**

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

And git config –list to verify your settings. This will display all your configured settings, including name and email.

**Basic Git Workflow**

In Terminal:

1. git init
2. git add .
3. git commit -m "First commit"
4. git remote add origin <GitHub-Repo-URL>
5. git branch –M main
6. git push -u origin main