**Git & Github**

**1. Add**

git add stages changes you’ve made to your working directory for the next commit. It moves changes from the working directory to the staging area.

**Commands:**

git add <filename> # Stage a specific file for the next commit

git add . # Stage all modified and new files at once

git add \*.html # Stage all files that match a specific pattern (e.g., all .html files)

**2. Commit**

git commit records a snapshot of the staged changes. It captures the state of the project and creates a new point in the version history.

**Commands:**

git commit -m "Your commit message here" # Commit the staged changes with a descriptive message

git commit -a -m "Your commit message" # Commit all modified files (except untracked ones) directly without `git add`

git commit --amend # Edit the previous commit message or add changes to the last commit

**3. Push**

git push sends your committed changes to a remote repository like GitHub. This is how you share your changes with others.

**Commands:**

git push origin <branch-name> # Push your local commits to the remote repository on a specific branch

git push -u origin <branch-name> # Push for the first time and set up tracking (only needed the first time for a branch)

git push --force # Force push (use with caution, can overwrite changes on the remote)

**4. Pull**

git pull fetches and integrates changes from a remote repository to your local repository. It’s a combination of git fetch and git merge.

**Commands:**

git pull origin <branch-name> # Pull changes from the remote branch into your current branch

git pull --rebase # Pull changes and rebase to avoid merge conflicts

git pull --ff-only # Pull changes only if fast-forward is possible (no merge commits)

**5. Creating user profile**

Before making commits, you need to set up your name and email so that Git knows who is making the changes. These details are tied to each commit.

**Commands:**

git config --global user.name "Your Name" # Set your GitHub username

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

git config --global --list # Check your current Git configuration

**6. Pull remote repo to local**

You can use git clone to copy a remote repository to your local machine, allowing you to work on it locally.

**Commands:**

git clone <repository-url> # Clone the entire repository to your local machine

git clone <repository-url> <directory> # Clone the repository into a specific directory

git clone --branch <branch-name> <repo-url> # Clone only a specific branch

**7. Version control system in Git and GitHub**

Git is a distributed version control system that helps you track changes, collaborate on code, and maintain a history of your project. GitHub hosts your Git repositories online.

**Common Commands:**

git init # Initialize a local repository

git status # Check the current state of the working directory and staging area

git log # View the commit history

git diff # View changes not yet staged or committed

**8. Merge**

git merge integrates changes from one branch into another. This is usually done to bring feature branches back into the main branch.

**Commands:**

git checkout <target-branch> # Switch to the branch where you want to merge changes

git merge <source-branch> # Merge changes from the source branch to the target branch

git merge --no-ff <branch-name> # Create a merge commit even if fast-forward is possible

**9. Fork**

Forking is done on GitHub to create a copy of someone else’s repository into your own GitHub account. After forking, you can clone it locally to make changes.

**Commands after forking:**

git clone <forked-repo-url> # Clone your forked repository to your local machine

git remote add upstream <original-repo-url> # Link the original repo to your fork

git fetch upstream # Fetch updates from the original repo (upstream)