Git and GitHub

What is Git?

* Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

* Tracking code changes
* Tracking who made changes
* Coding collaboration

What does Git do?

* Manage projects with Repositories
* Clone a project to work on a local copy
* Control and track changes with Staging and Committing
* Branch and Merge to allow for work on different parts and versions of a project
* Pull the latest version of the project to a local copy
* Push local updates to the main project
* Working with Git

Initialize Git on a folder, making it a Repository

* Git now creates a hidden folder to keep track of changes in that folder
* When a file is changed, added or deleted, it is considered modified
* You select the modified files you want to Stage
* The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
* Git allows you to see the full history of every commit.
* You can revert back to any previous commit.
* Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Why Git?

* Over 70% of developers use Git!
* Developers can work together from anywhere in the world.
* Developers can see the full history of the project.
* Developers can revert to earlier versions of a project.

What is GitHub?

* Git is not the same as GitHub.
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

Git Commands :

These commands are for working locally with Git, the version control system.

**1. Setting up Git:**

* git config --global user.name "Your Name"  
  Set your Git username.
* git config --global user.email "you@example.com"  
  Set your Git email address.
* git config --list  
  Show your Git configuration.

**2. Working with Repositories:**

* git init  
  Initialize a new Git repository in the current directory.
* git clone <repository-url>   
  Clone an existing repository from a remote server (like GitHub).

**3. Tracking and Staging Changes:**

* git status  
  Show the current status of the repository (what files are staged, modified, or untracked).
* git add <file>  
  Stage a file for the next commit.
* git add .  
  Stage all modified and untracked files.
* git diff  
  Show the changes in tracked files since the last commit (useful before committing).
* git diff --staged  
  Show the difference between staged files and the last commit.

**4. Committing Changes:**

* git commit -m "Commit message"  
  Commit staged changes with a descriptive message.
* git commit -a -m "Commit message"  
  Stage and commit all tracked files with one command.

**5. Viewing Commit History:**

* git log  
  Show the commit history for the current branch.
* git log --oneline  
  View a simplified version of the commit history (commit hash and message).
* git log --graph --oneline --all  
  View a visual representation of all branches and commits.

**6. Branches:**

* git branch  
  List all branches and show the current one.
* git branch <branch-name>  
  Create a new branch.
* git checkout <branch-name>  
  Switch to an existing branch.
* git checkout -b <branch-name>  
  Create and switch to a new branch in one command.
* git merge <branch-name>  
  Merge the specified branch into the current branch.
* git branch -d <branch-name>  
  Delete a local branch.

**7. Undoing Changes:**

* git reset <file>  
  Unstage a file (keep the changes in the working directory).
* git reset --hard  
  Reset the working directory and staging area to the last commit (discard all changes).
* git checkout -- <file>  
  Discard changes in a file and reset it to the last committed version.
* git revert <commit-hash>  
  Create a new commit that undoes the changes made in the specified commit.
* git reset --soft <commit-hash>  
  Undo a commit, but keep the changes staged (useful for amending commits).

**8. Syncing with Remote Repositories:**

* git remote add origin <repository-url>  
  Link a remote repository to your local repository.
* git remote -v  
  Verify the remote repository URL.
* git pull origin <branch>  
  Fetch and merge changes from the remote repository into the current branch.
* git push origin <branch>  
  Push the committed changes to the remote repository.
* git fetch  
  Fetch changes from the remote repository without merging.
* git pull --rebase origin <branch>  
  Pull changes from the remote repository and rebase your work on top of them (avoiding merge commits).

**9. Tagging Releases:**

* git tag <tag-name>  
  Create a lightweight tag for the current commit (useful for marking releases).
* git tag -a <tag-name> -m "Tag message"  
  Create an annotated tag with a message.
* git push origin <tag-name>  
  Push the tag to the remote repository.

**10. Stashing Changes:**

* git stash  
  Save uncommitted changes and clean the working directory.
* git stash pop  
  Reapply the most recently stashed changes.
* git stash list  
  List all stashes.
* git stash drop  
  Discard the most recent stash.

**GitHub Commands:**

These commands help you interact with a repository hosted on GitHub, usually in combination with Git. The interaction between Git and GitHub usually involves remote repositories.

**1. Creating SSH Key for GitHub (for authentication):**

* ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"  
  Generate a new SSH key for GitHub.
* ssh-add ~/.ssh/id\_rsa  
  Add the SSH key to your SSH agent.
* Copy the SSH key from ~/.ssh/id\_rsa.pub and add it to your GitHub account in the SSH keys section.

**2. Forking and Cloning a Repository:**

* Go to the repository you want to fork on GitHub and click the **Fork** button to create a copy in your GitHub account.
* git clone <your-forked-repository-url>  
  Clone the forked repository from your GitHub account to your local machine.

**3. Syncing a Forked Repository with the Original Repository:**

* git remote add upstream <original-repository-url>  
  Add the original repository as upstream to your local fork.
* git fetch upstream  
  Fetch changes from the upstream repository.
* git merge upstream/<branch>  
  Merge changes from upstream into your local repository.

**4. Creating a Pull Request (PR):**

After committing your changes and pushing them to your GitHub repository:

* Go to the repository on GitHub and click **New Pull Request**.
* Select the branch with your changes and submit the pull request.

**5. GitHub Issues:**

* **Creating an Issue**:  
  Navigate to the repository on GitHub, click the **Issues** tab, and click **New Issue** to submit a bug report or feature request.
* **Closing an Issue via Commit**:  
  In a commit message, you can reference an issue (e.g., Fixes #123) to automatically close the issue when the commit is merged.

**6. GitHub Actions (CI/CD Automation):**

* **GitHub Actions** are used to automate workflows (like testing and deployment) directly on GitHub.
  + .github/workflows/ - Store your YAML workflow files here.
  + Example: A simple workflow might test your code on different versions of a language or automatically deploy your app after passing tests.