**GIT AND GITHUB**

Git is a distributed version control system (VCS) that allows you to track changes in your code or any set of files over time. It's widely used for collaborating on projects, especially in software development.

1. **Repository**: A repository (or repo) is a folder or directory where your project files are stored, along with the entire history of changes made to those files.
2. **Commit**: When you make changes to your files and want to save those changes, you create a commit. A commit is like a snapshot of your files at a particular point in time, with a unique ID and a message describing the changes you made.
3. **Branch**: A branch is a separate line of development within your repository. The main branch is typically called "master" or "main." You can create new branches to work on new features or fixes without affecting the main branch. Once your work is done, you can merge your branch back into the main branch.
4. **Pull and Push**: When working with a remote repository (like on GitHub), you need to "pull" the latest changes from the remote repo to your local machine, and "push" your local commits to the remote repo.

GitHub is a web-based platform that provides hosting for Git repositories. It allows developers to collaborate on projects, track issues, review code changes, and much more.

1. **Remote Repository Hosting**: GitHub allows you to store your Git repositories online, making it easy to collaborate with others and access your code from anywhere.
2. **Forking and Pull Requests**: You can "fork" (create a copy) of someone else's repository, make changes to your fork, and then submit a "pull request" to the original repository, suggesting that your changes be merged.
3. **Issues and Project Management**: GitHub provides tools for tracking issues, assigning tasks, and managing projects effectively.
4. **Code Review**: GitHub allows you to review code changes made by other contributors and provide feedback before merging those changes.

GIT WORKFLOW

**1)Initialize Git Repository**: To start using Git for your project, you need to initialize a Git repository, which track the changes of the project.

**git init**

**2)Create Files:** Once you have initialized the Git repository, you can start creating files or making changes to existing files in your project.

**touch filename.txt**

**3)Stage Changes:** After making changes to your files, you need to stage them for committing. This means telling Git which changes you want to include in your next commit. You can stage individual files or all files using the following commands:

**git add filename.txt (adds single file)**

**git add . (adds all files)**

**4)Commit Changes:** Once you have staged your changes, you can create a new commit. A commit is like a snapshot of your project at a specific point in time, with a unique identifier and a commit message describing the changes you made. To create a commit, use the following command:

**git commit -m “commit message”**

# Create a new directory for your project

mkdir my-project

cd my-project

# Initialize a new Git repository

git init

# Create a new file

touch README.md

# Add the file to the staging area

git add README.md

# Create your first commit

git commit -m "Initial commit"

**6)View Status:** This command shows the current status of the working directory. It displays information about modified, staged, and untracked files.

**git status**

**7)View commit history:** This command displays the commit history for the current branch. It shows a list of all previous commits, their commit messages, authors, dates, and commit IDs.

**git log**

**8)Delete file from git history:**

**Delete the file from your working directory but keep it in Git history (soft delete):**

**git rm filename.ext**

**Permanently delete the file from both your working directory and Git history (hard delete):**

**git rm -f filename.ext**

**9)Hold files:**

**git stash**

This command temporarily saves changes that you don't want to commit immediately. Stashing is useful when you need to switch branches or temporarily set aside your changes.

**# You have some changes in your working directory**

**git status**

**# Stash the changes**

**git stash save "WIP: Feature X"**

**# List the stashes**

**git stash list**

**# Switch to another branch or work on something else**

**git checkout another-branch**

**# ... work on another branch ...**

**# Go back to the original branch**

**git checkout original-branch**

**# Apply the previously stashed changes**

**git stash apply**

**# Optionally, remove the applied stash from the stack**

**git stash drop**

**# git stash pop: Applies the most recent stash and then removes it.**

**# git stash clear: Removes all stashes.**

**9)git remote**

This command is used to manage remote repository connections.

Example: **git remote add origin https://github.com/username/repository.git** (adds a new remote named "origin")

Example**: git remote -v** (lists all configured remote repositories)

Remote repositories are typically used for collaboration and sharing code.

**10)git branch**

This command is used to list, create, or delete branches.

Example: **git branch** (lists all branches)

Example: **git branch new-branch** (creates a new branch named "new-branch")

Branches allow you to work on different features or fixes independently.

**11)git checkout**

This command is used to switch between branches or restore files from a specific commit.

Example: **git checkout branch-name** (switches to the specified branch)

Example: **git checkout -- filename.ext** (restores the file to its state in the current branch)

**12)git merge**

This command merges changes from one branch into the current branch.

Example: **git merge branch-name** (merges the specified branch into the current branch)

Git will attempt to combine the changes from both branches automatically.

If there are conflicts (changes to the same lines), you will need to resolve them manually.

**13)git pull**

This command fetches the latest changes from a remote repository and merges them into the current branch.

Example: **git pull origin main** (pulls changes from the main branch of the origin remote)

This command updates your local repository with the latest changes from the remote repository.

**14)git push**

This command uploads your local commits to a remote repository.

Example**: git push origin main** (pushes local commits to the main branch on the origin remote)

This command is used to share your local changes with the remote repository and other collaborators.