Working Of Git Repository

A *Git repository* is a collection of files and their revision history, managed by the Git version control system. The working of a Git repository involves several key components and processes:

1. *Initialization*: To create a new Git repository, you run the command *git init* in a directory. This initializes the directory as a Git repository, setting up the necessary data structures.
2. *Staging:* Git uses a staging area (also called the "index") to prepare files for committing. You can selectively stage changes by using the command *git add <file>* to add specific files or *git add* **.** to add all modified files.
3. *Committing:* Once the changes are staged, you create a commit with a message describing the changes. The command *git commit -m "Commit message"* records a snapshot of the current state of the files in the repository.
4. *Branching and Merging:* Git allows you to create branches, which are independent lines of development. Branching enables you to work on different features or versions simultaneously. Merging combines the changes from one branch into another, incorporating the changes into the main branch.
5. *Remote Repositories:* Git supports remote repositories hosted on platforms like GitHub, GitLab, or Bitbucket. You can clone a remote repository using *git clone <repository_url>*, and then push or pull changes to synchronize with the remote repository.
6. *Version Control:* Git tracks changes at a granular level. Each commit represents a specific version of the files. You can view the commit history, compare changes between versions, and revert to previous states if needed.
7. *Collaboration:* Git facilitates collaboration among multiple developers. Developers can work on different branches, merge their changes, and handle conflicts. Git also provides features for code review and collaboration workflows.

Git Architecture

Git follows a distributed architecture rather than a traditional three-tier architecture. However, we can draw an analogy between the components of Git and a three-tier architecture to understand their respective roles. Here's a simplified representation:

1. Local Repository (Presentation Tier):

- *Working Directory:* Represents the local copy of the repository where files are modified.
- *Staging Area (Index):* Acts as a middle ground between the working directory and the repository. Changes are selectively staged here before committing.

2. Repository (Application Tier):

- *Local Repository:* Contains the entire history and metadata of the project. It resides on the local machine and is accessed using Git commands.
- *Remote Repository:* Represents a separate Git repository hosted on a remote server (e.g., GitHub, GitLab). It serves as a centralized location for collaboration and sharing changes.

3. Remote Server (Data Tier):

- *Hosting Service:* The remote server provided by hosting services like GitHub or GitLab. It stores the remote repository and enables collaboration and access control.
- *Remote Repository:* The remote copy of the repository stored on the hosting service. It can be cloned, pulled, or pushed to/from the local repository.

In this analogy, the local repository and its components (working directory, staging area) represent the presentation tier, where developers interact with the project files. The repository, both local and remote, serves as the application tier, managing the version control and history of the project. The remote server and its remote repository form the data tier, hosting the project and enabling collaboration.

Basic Git Commands

1. git init:

Initializes a new Git repository in the current directory.

2. git clone <repository>:

Creates a local copy of a remote repository on your machine.

3. git add <file>:

Adds a file to the staging area, preparing it for a commit.

4. git commit -m "message":

Commits the changes in the staging area to the repository with a descriptive message.

5. git status:

Displays the current status of the repository, including modified, staged, and untracked files.

6. git log:

Shows a history of commits in the repository, including commit IDs, authors, dates, and commit messages.

7. git pull:

Fetches the latest changes from a remote repository and merges them into the current branch.

8. git push:

Pushes the local commits to a remote repository, updating it with the latest changes.

9. git branch:

Lists all branches in the repository and highlights the current branch.

10. git checkout <branch>:

Switches to a different branch in the repository.

11. git merge <branch>:

Merges changes from a specified branch into the current branch.

12. git remote:

Lists the remote repositories associated with the current repository.

13. git diff:

Shows the differences between the working directory and the staging area or the repository.

14. git reset <file>:

Removes a file from the staging area, preserving its changes in the working directory.

15. git stash:

Temporarily saves the modified and staged changes, allowing you to switch branches without committing.