Git

Git is a distributed version control system for tracking changes in files. It helps manage projects, allowing multiple people to collaborate efficiently by keeping a history of all modifications made to the code or files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

A version control system (VCS) means it records all the changes made to a file or set of data, so a specific version may be called later if needed. This helps ensure that all team members are working on the latest version of the file.

Git manages code versions, and GitHub provides platform for to store, share and collaborate on those versions with others. GitHub is a Git repository hosting service.

Key Concepts:

a. **Repositories:**

- A Git repository is a storage location for a project's files and revision history.
- Repositories can be local (on a developer's machine) or remote (on a server).

b. **Commits:**

- Commits represent a snapshot of the project at a specific point in time.
- Each commit has a unique identifier and includes changes made, the author, and a commit message.

c. **Branching:**

- Branch is a new/separate version of the main repository.
- Git's branching model allows for the creation of isolated branches for different features or bug fixes.
- Branches can be merged back into the main codebase when ready.

d. **Merging:**

- Merging combines changes from different branches.

e. **Remote Repositories:**

- Git supports collaboration through remote repositories hosted on platforms like GitHub, GitLab, or Bitbucket.
- Developers can push and pull changes to and from these remote repositories.

Git Commands:

1. **git init** is a command used to create a new repository. A hidden **.git** directory is added to the folder.
   Most of the git command do not work outside initialized project, so this is the first command you will run in a project.
   Go to project folder>run git init

2. **git add** is a command used to add a file that is in the working directory to the staging area. It is mandatory to stage the code before commit(push to remote) using git add command.
   To stage all files use " . " - **git add .**
   To stage a single file **git add <file-1>**
   To stage a multiple file **git add <file-1> <file-2>**

3. **git commit** is a command saves your changes to your local repository. Everytime you commit you have to add a small message about the changes you made. This will help to keep track of the changes later.
   git commit -m "commit-message

4. **git push** is a command push your changes from local repository to your remote repository. One can only push the committed changes.
   It also creates the repository with the branch name you enter if repository does not exist on remote location.
   If branch is already connected to remote then run **- git push**.

git push <remote> <branch-name>

5. **git pull** is a command used fetches latest changes from remote repository to your local. This is helpful when multiple people are working on same repository. It will help to keep your local repo updated with latest code.
   If branch is already connected to remote then run - **git pull**.
   git pull <remote> <branch-name>

6. **git clone** is a command creates a local copy of a remote repository .
   When you clone a repository the source code gets automatically downloaded to local machine. This local repo will point to remote repository and can PUSH and PULL changes to it.
   git clone <git-repo-url>

7. **git branch** is a command used to create new/separate version of the main repository. Branches allow developers to work on isolated features or bug fixes without affecting the main codebase
   git branch <branch_name>: Creates a new branch from the current HEAD commit.
   git checkout <branch_name>: Switches to the specified branch.
   git merge <branch_name>: Merges the changes from one branch into another.

8. **git status** is a command used to show the status of the current repository including staged, unstaged and untracked files.

9. **git log** is a command shows the commit history of current repository.

10. **git diff**: Displays the difference between the working directory and the staging area (or two commits).

Advantages of Git:

1. Free and Open Source:

- Git is freely available and open source, released under the GPL licence, allowing users to download, modify, and use it without cost.

2. Fast and Small:

- Operations are performed locally, enhancing speed.
- Written in C, Git avoids runtime overheads, and its efficient data compression keeps client-side storage small.

3. Implicit Backup:

- Multiple copies of data on client-side act as implicit backups, reducing the risk of data loss in case of crashes or disk corruption.

4.Security:

- Utilises the secure hash function (SHA1) for identifying and securing objects in its database.

5. No Need for Powerful Hardware:

- Requires less powerful server hardware, as developers interact with the server only for pushing or pulling changes. Heavy lifting occurs on the client side.

6. Easier Branching:

- Branch management is simplified in Git, taking only seconds to create, delete, and merge branches.
- Unlike CVCS, Git's branching is efficient and doesn't involve time-consuming copying of code.