Git: Introduction and Essential Commands

What is Git and Why Use It?

Git is a distributed version control system that helps track changes in source code during software development. It allows multiple developers to collaborate, revert to previous versions, and manage different versions of a project efficiently.

Basic Git Commands

1. pwd (Print Working Directory)

```
pwd
```

This command prints the current directory path where you are working.

2. mkdir (Make Directory)

```
mkdir project-directory
cd project-directory
```

Creates a new directory and moves into it.

3. git init (Initialize Repository)

```
git init
```

Initializes an empty Git repository in the current directory.

4. echo (Create a File and Add Content)

```
echo "Hello, Git!" > file.txt
```

Creates a file `file.txt` with content "Hello, Git!".

5. git add (Stage Changes)

```
git add file.txt
```

Stages file.txt for the next commit.

6. git commit (Commit Changes)

```
git commit -m "Initial commit"
```

Commits the staged changes with a message.

7. git remote add origin (Set Remote Repository)

```
git remote add origin <repository_url>
```

Links your local repository to a remote GitHub repository.

8. git push (Push Changes to Remote)

```
git push -u origin main
```

Pushes local commits to the remote repository.

9. git reset --hard origin/main (Reset to Remote State)

```
git reset --hard origin/main
```

Resets local changes and syncs with the remote main branch.

10. git pull origin main (Pull Latest Changes)

```
git pull origin main
```

Fetches and integrates the latest changes from the remote repository.

11. git pull --rebase origin main (Rebase Local Changes)

```
git pull --rebase origin main
```

Pulls remote changes and applies local commits on top.

12. git checkout -b main (Create and Switch to a New Branch)

```
git checkout -b new-feature
```

Creates a new branch new-feature and switches to it.

13. git merge (Merge Branches)

```
git checkout main
git merge new-feature
```

Merges new-feature into main.

14. git stash (Save Uncommitted Changes Temporarily)

```
git stash
```

Saves local modifications and resets the working directory.

```
git stash pop
```

Restores the last stashed changes.

Handling Merge Conflicts

Example of Merge Conflict:

1. Assume two developers modify file.txt in different branches.

   When merging, Git detects conflicts:

   ```
   git merge new-feature
   Output:
   CONFLICT (content): Merge conflict in file.txt
   ```

Resolving Merge Conflicts:

Open the conflicting file and manually resolve conflicts:

```
<<<<<<< HEAD
Existing content
=======
New content from new-feature
>>>>>>> new-feature
```

1. Edit to keep the correct version and remove conflict markers.
2. Add and commit the resolved file:
   ```
   git add file.txt
   ```

3.
   ```
   git commit -m "Resolved merge conflict"
   ```