GIT

What is Git?

Git is a **distributed version control system** that helps track changes in your source code during software development. It allows multiple developers to work on a project simultaneously without interfering with each other's work.

- Created by **Linus Torvalds** in 2005 (creator of Linux).
- It's **local**, **lightweight**, and **fast**.
- Used primarily for **source code management (SCM)**

Why Git?

- Tracks changes
- Allows multiple developers to collaborate
- Supports branching and merging
- Works offline
- Open-source and fast

How Git Works (Internally)

Git stores data as a series of **snapshots** of a filesystem.

- **Working Directory**: Where you edit files.
- **Staging Area (Index)**: Where you prepare files to be committed.
- **Repository**: Where Git stores committed snapshots.

Installing Git

Windows: Install from https://git-scm.com

Linux: sudo apt install git

Mac: brew install git

To verify:

```
git --version
```

Basic Git Setup

```
git config --global user.name "Your Name"
git config --global user.email "your@example.com"
```

Check Git Config

```
git config --list
```

Git Lifecycle & Important Areas

1. **Working Directory** – You edit files here.
2. **Staging Area (Index)** – You add changes to this area.
3. **Repository** – You commit changes to the local repo.
4. **Remote Repository** – You push changes here (e.g., GitHub).

CORE GIT COMMANDS

Initialize a Git repository

```
git init
```

Creates a new Git repository in your project directory.

Clone a Repository

```
git clone <repository_url>
```

Downloads a project and its entire version history.

Check Current Status

```
git status
```

Displays files changed, untracked files, etc.

Add Files to Staging Area

```
git add filename
git add .        # Adds all changed files
```

Commit Changes

```
git commit -m "Your commit message"
```

Commits staged changes to the local repo.

View Commit History

```
git log
```

Shows list of previous commits with hash, author, and message.

Branching

```
git branch              # List branches
git branch new-feature    # Create new branch
git checkout new-feature  # Switch to new branch
git switch new-feature    # Alternative (newer way)
```

Merge Branches

```
git merge branch-name
```

Merges the given branch into the current one.

Delete a Branch

```
git branch -d branch-name
```

Stash Changes Temporarily

```
git stash
git stash pop      # Bring back stashed changes
```

Show Difference Between Files

```
git diff              # Shows unstaged differences
git diff --staged       # Differences in staging area
```

Reset Files

```
git reset filename       # Unstages file
git reset --hard         # DANGER: Deletes changes
```

Revert a Commit

```
git revert <commit_id>
```

Safely undoes a commit without deleting history.

Delete File from Git

```
git rm filename
```

Common Git Commands

Command	Description
git init	Initializes a new Git repository
git clone <repo>	Clones an existing repo from GitHub or other source
git status	Shows the status of files (staged, unstaged, untracked)
git add <file>	Stages a file
git add .	Stages all files
git commit -m "message"	Commits staged files with a message
git log	Shows commit history
git diff	Shows differences between changes
git branch	Lists branches
git branch <name>	Creates a new branch
git checkout <branch>	Switches to a branch
git merge <branch>	Merges a branch into current branch
git remote add origin <url>	Connects local repo to remote
git push -u origin <branch>	Pushes branch to remote
git pull	Pulls latest changes from remote
git stash	Temporarily saves uncommitted changes
git reset --hard	Resets repo to last commit (DANGEROUS)
git rm <file>	Removes a file from the repo

Git Branching (Core Power)

Creating a branch:

```
git branch feature-login
```

Switching to the branch:

```
git checkout feature-login
```

Create + switch:

```
git checkout -b feature-login
```

Merging:

```
git checkout main
git merge feature-login
```

Git Clean-up Commands

Command	Description
git stash	Save uncommitted changes
git stash pop	Reapply saved changes
git reset HEAD <file>	Unstage a file
git revert <commit>	Revert a commit safely
git rebase <branch>	Apply changes on top of another base
git cherry-pick <commit>	Apply a specific commit

Git Ignore File (.gitignore)

Used to ignore files from being tracked.

Example:

```
node_modules/
*.log
.env
```