Introduction to Git

Git is a **version control system (vcs)** designed to track changes in Source code and collaborate efficiently with others it is widely used for software development due to its speed, flexibility and support for non-linear workflows (eg.branching and merging) this what a typical git workflow in a startup or enterprise application code.

What is VCS

VCS stands for version control system is a tool that helps manage and track changes to source code or other files over time vsc is essential for software development and other projects where maintaining a history of change collaboration and versioning is critical.

Popular VCS tools

- **Git :-** The most widely used DVCS: support branching and distributed workflows
- **Subversion :-** A CVCS used in enterprise application
- **Mercurial :-** Another DVCS simper than git is some aspect
- **Perforce :-** A CVCS often used for larger than -scale enterprise projects

Why do we need VSC

 Tracking and changing :- it records changes to file over time enabling developer to see who change what and when

Collaboration:- Multiple people can work on the same project simultaneously without overwriting each other's work

Branching and merging:- Developers can create separate branch for different feature or experiment and later merge into the main project

Version History :- It keep a history of all changes making it easy to revert to prevent version if needed

Conflict resolution:- Helps manage and resolve conflicts when multiple developer make change to the same file

Backup and recovery:- Acts as a backup for the project

Basic Git commands

command	Description
`git inti`	Initialize a new Git repository
`git clone <url>`	Clone repository from url
`git status`	Show the status of working directory
`git add <file>`	Stage a file for commit
`git add .`	Stage all changes in the current directory
`git commit -m "message"`	Commit stage change with a message
`git push`	Push commit is remote repository
`git pull`	Fetch and merge changes from a remote repo
`Git branch`	List branches
`Git branch <name>`	Create a new branch
`Git checkout <branch>`	Switch to a specific branch

Git merge <branch>	Merge a branch into the current branch
Git log	View commit history
Git diff	Show difference between working files
Git reset <file>	Unstage a file
Git stash	Save changes without committing
Git stash pop	Reapply stashed changes
Git remote add <name> <url>	Add a remote repository
Git fetch	Download object and refs from another repo
Git rebase <branch>	Reapply commit on top another branch

How git works

Git manages projects by tracking changes to files and directories within a working directory . when a git repository is initialized ,a hidden **.git** directory is created , which contains all metadata and object databases for the project. This directory is where Git stores information about commit branch tags and other version control data

Here's A breakdown of how git uses the .git directory :

- **Initialization**
  when you run **git init** in directory Git creates a .git directory within that directory
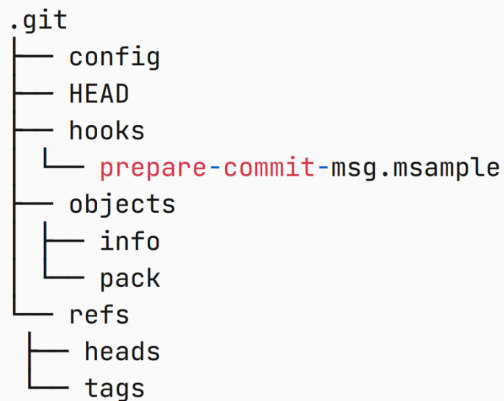 This directory serves as the repository for the project.
- **Tracking changes:**
  Git track changes to file to find within the working directory

What is `.git/` Directory ?

As soon as we run **git init** command ,we get following message in the console:

Initialized empty Git repository in/home/coding/shubham-dev/projects

```
.git                                                    ⎘
├── config
├── HEAD
├── hooks
│   └── prepare-commit-msg.msample
├── objects
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

- **Config** is a text file that contains your git configuration for the current repo.
- **Head** contain the current head of the repo
- **hooks** contain any scripts that can be run before/after git does anything.
- **Objects** contain the git object,ie the data about the file ,commit etc in your repo. we will go in depth into this in a blog.
- **Refs** as we previously mentioned, stores references (pointers)