Understanding Git

Our Repository Link: https://github.com/VanshikaGuptaa/AIHT

What is Git?

At its core, **Git** is a version control system that allows you to track changes to files. It provides a structured way to coordinate work on those files, making it possible for multiple people to collaborate seamlessly. While it's commonly used in software development, Git is versatile and can be applied to track changes to any set of files.

With Git, we can:

- Keep a detailed record of who made changes to a file.
- Easily revert back to earlier versions of files when needed.
- Collaborate with others by sharing and merging changes into a single version of a file.

In essence, Git acts as a meticulous historian for your project, documenting every change made by every contributor.

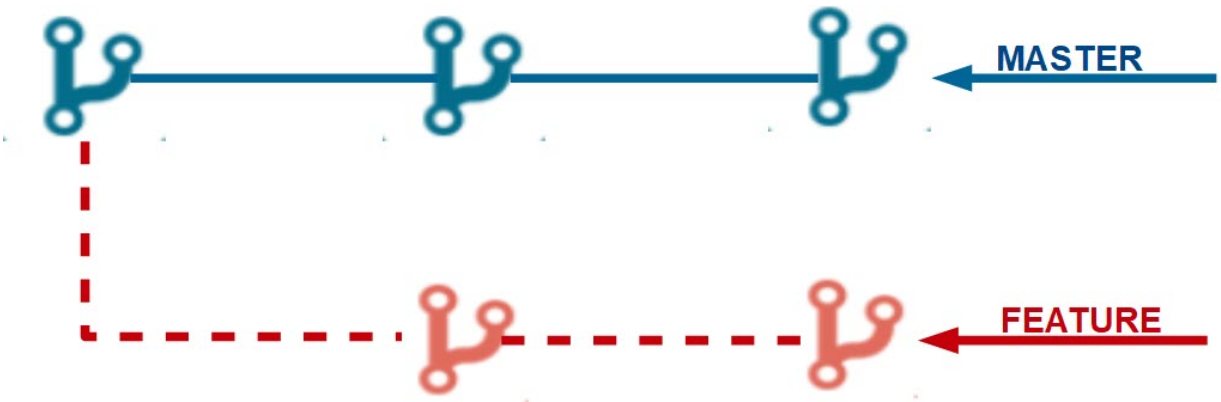
Goals of Git:

- Speed
- Support for non-linear development
- Fully distributed
- Able to handle large projects efficiently

Each update in the code is recorded in Git’s history, which is stored in a data structure called a Git repository. This repository is the core of Git.

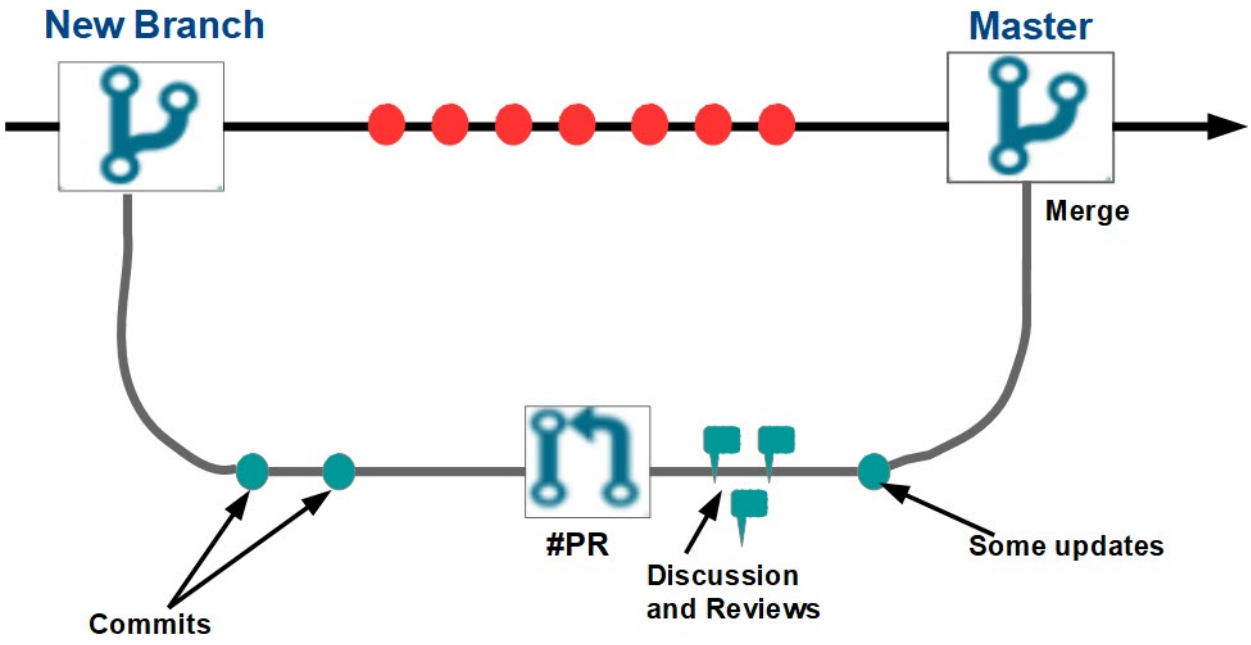
Why Git?

Feature branch workflow



Each branch function in an isolated environment while updates are made in the codebase. This way, its ensured that the master branch always has the production-quality code

Pull Requests



A developer calls a pull request to ask another developer to merge one of his/her branches into the other's code repositoryMakes it easy for project leaders to monitor and track code changes

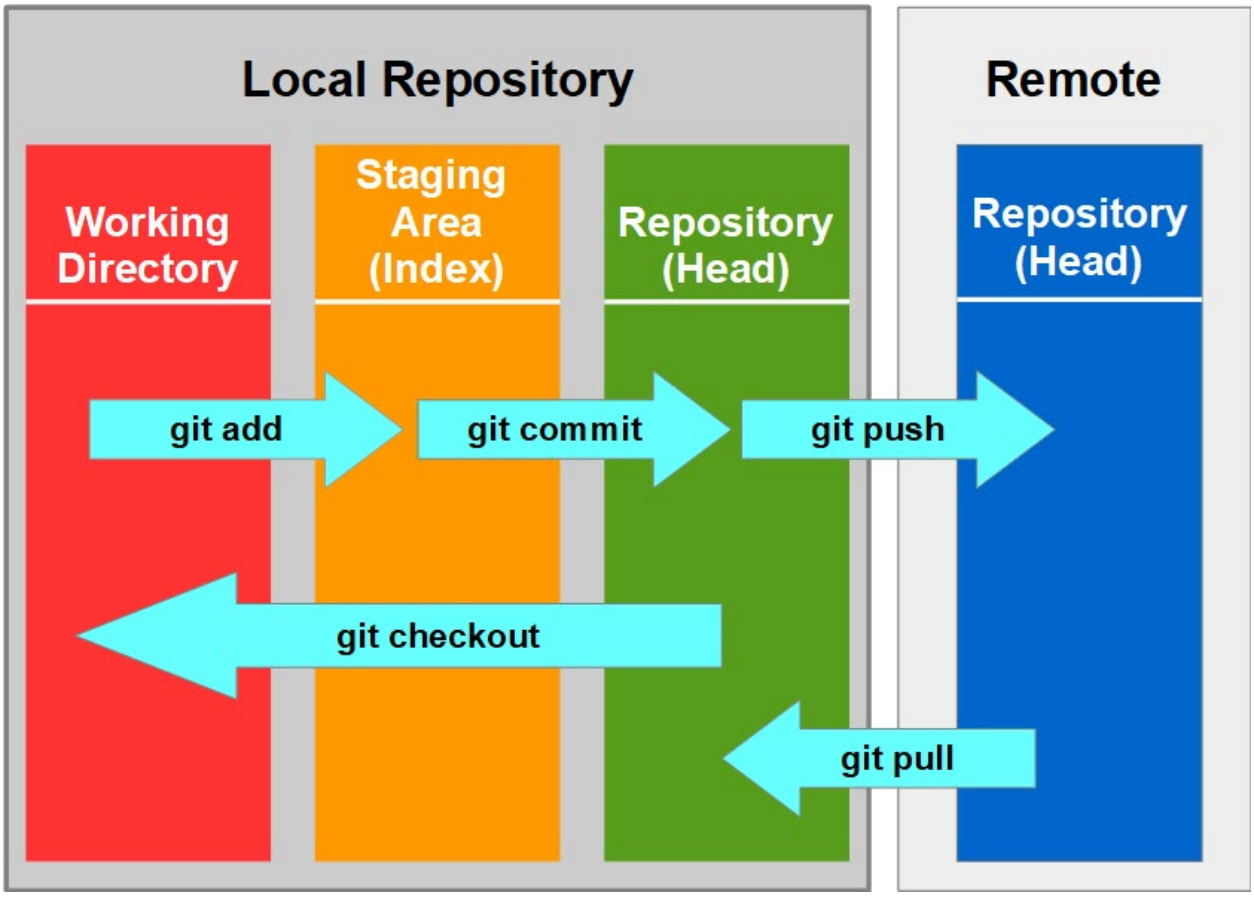
Git Basics

File States

There are three main states that your Git files can reside in:

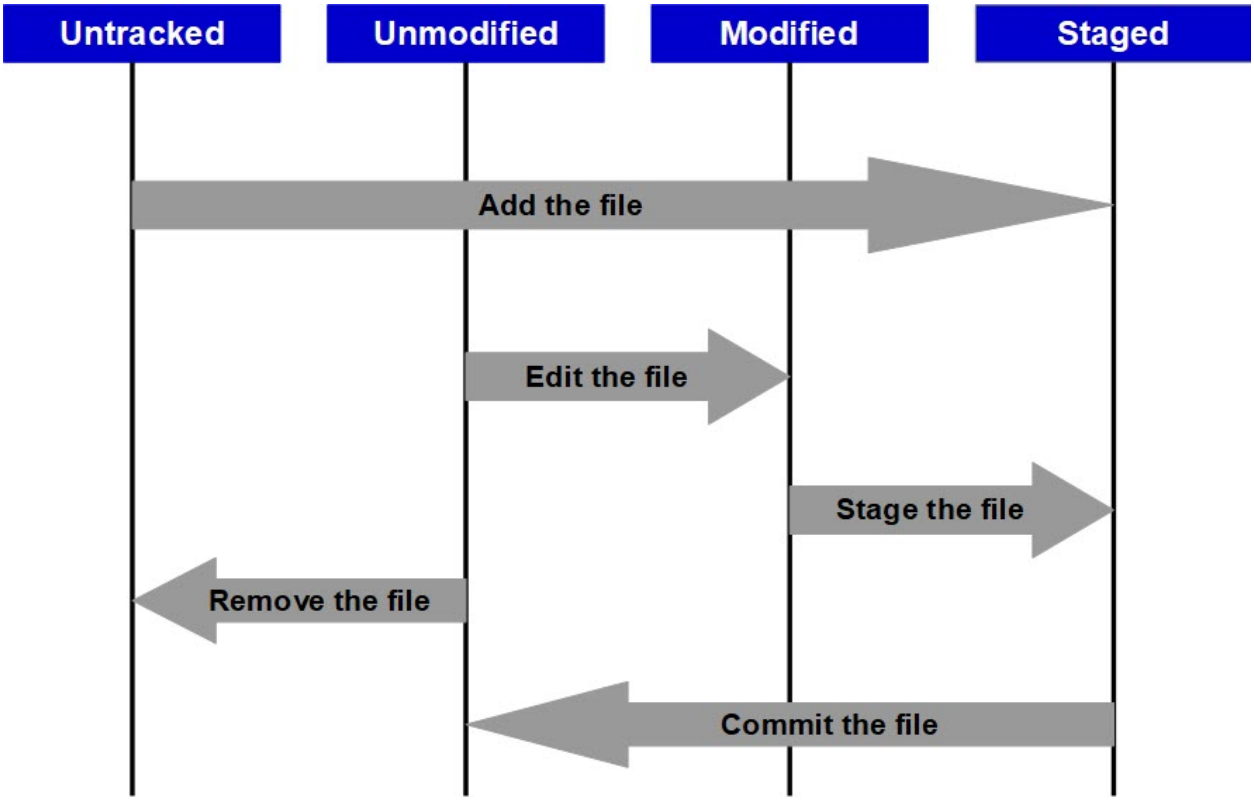
- In your local repo (Committed)
- HEAD – last commit snapshot, next parent
- Checked-out and modified but not yet committed (working directory)
- Sandbox – contents unpacked into actual files
- Or, in-between, in a “staging” area
- Staged files are ready to be committed
- A commit saves a snapshot of the current state of staging area
- Index – proposed next commit snapshot

The remote repository (git directory) is the server where all the collaborators upload changes made to the file, it is where the work from all developers gets integrated.



Basic Git Workflow

- Modify the files in your working directory.
- Move the files to the staging area by staging them – adding snapshots of them to your staging area.
- Commit, which takes the files in the staging area and stores that snapshot permanently to your Git directory.
- Git generates a unique ID which is a 40 character string of hex digits for each commit and refers to the commits by this ID rather than the version number



Branching & Merging

A branch in Git is lightweight and the pointer always points to one of the commits.The default branch name in Git is “master”. Each time a commit is made, the master branch pointer moves forward automatically.

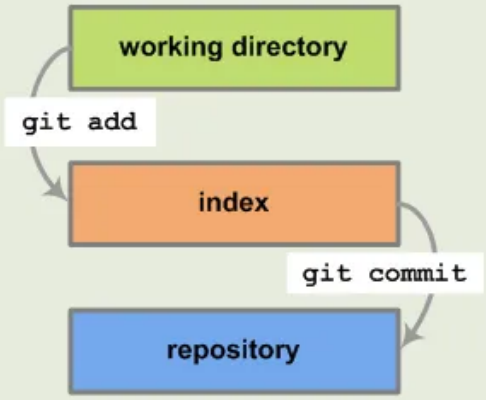
Role of GIT in DevOps

As a DevOps tool, Git empowers teams to work more efficiently and effectively, reducing errors and improving the overall quality of the software being developed. This plays a crucial role in enabling faster release cycles, which aligns with the principles of DevOps.

Git is crucial when working in large organizations where multiple teams collaborate on the same project, and each team needs to track changes made. That's why anyone who wishes to have a career in DevOps should start by understanding Git.

Basic Git command

Below are some of the basic Git commands:



Edit the files using vim / emacs / etc.,

Stage the changes using `git add`

Review the changes using `repo status .`

Commit the changes using `git commit`

Common Commands

`repo init`	initializes a new client
`repo sync`	syncs client to repositories
`repo start`	starts a new branch
`git add`	stages files
`repo status`	shows status of current branch
`git commit`	commits staged files
`git branch`	shows current branches
`git branch [branch]`	creates new topic branch
`git checkout [branch]`	switches HEAD to specified branch
`git merge [branch]`	merges [branch] into current branch
`git diff`	shows diff of unstaged changes
`git diff --cached`	shows diff of staged changes
`git log`	shows history of current branch
`git log m/[codeline]..`	shows commits that are not pushed
`repo upload`	uploads changes to review server