Git

- **Git** is a distributed version control system (VCS) designed to handle everything from small to very large projects with speed and efficiency.
- Created by Linus Torvalds in 2005, Git is used for tracking changes in source code during software development, enabling multiple developers to work on a project simultaneously without conflicts.

Purpose of Git:

The primary purpose of Git is to manage the changes to files, particularly source code, in a project over time.

- Collaborate on projects.
- Track and revert to previous versions of files.
- Create separate branches for new features or bug fixes.
- Merge different branches of development.

Repositories:

- A repository is a storage location for a project, containing all files and the history of their changes
- A repository can be local repository or remote repository.

Commits:

- Commit includes a message describing the changes.
- Commits are used to track the history of changes and are the basic units of work in Git.

Branches:

- A **branch** is a parallel version of a repository. It allows developers to work on different features or fixes without affecting the main codebase.
- The **master** or **main** branch is the default branch and usually contains the stable version of the project.
- Developers create new branches to work on specific tasks and later merge them back into the main branch.

Merges

- Merging is the process of integrating changes from one branch into another.
- It involves combining the histories and resolving any conflicts that arise from changes made in different branches.

Git

Tags

- Tags are markers used to denote specific points in the repository's history, often used for releases.
- Unlike branches, tags are immutable and typically used to mark version numbers.

Common Workflows in Git

Git workflows provide a structured way for teams to collaborate on projects. Different workflows can be chosen based on the team size and project complexity.

1) Centralized Workflow :

- All changes are committed to a single central repository, typically the main branch.
- Simple and easy to manage for small teams or projects. .

Steps:

Clone the repository - git clone https://example.com/repo.git

Make changes and commit - git add .

git commit -m "Describe your changes".

Push to central repository - git push origin main

Pull from central repository – git pull origin main

2) Feature Branch Workflow:

- Encourages the use of separate branches for each feature or bug fix.
- Keeps the main branch clean and stable.
- Allows parallel development on multiple features.

Steps:

Create new Branch - git checkout -b feature-branch

Make Changes and commit - git add .

git commit -m "Describe your feature"

Push branh to remote respository - git push origin feature-branch

Git

Merge - git checkout main

git pull origin main

git merge feature-branch

git push origin main

3)Forking workflow:

- Commonly used in open-source projects.
- Each developer works on their own fork of the repository.
- Changes are proposed back to the original repository via pull requests.

Steps:

Clone your fork - git clone https://example.com/your-fork.git

Add the original repo as a remote - git remote add main https://example.com/original-repo.git

Sync with original repo - git fetch upstream

git checkout main

git merge upstream/main

Create new branch - git checkout -b feature-branch

Make changes and commit - git add .

git commit -m "Describe your changes"

Push your branch to workflow - git push origin feature-branch

Best Practices:

- Commit often.
- Keep your main branch clean.
- Use Branches.
- Review your code.
- Commit changes with clear message.

Integration with Remote repositories:

- Platforms like GitHub and GitLab host remote repositories and provide tools for collaboration, issue tracking
- Developers can clone repositories to their local machines, push changes to the remote, and pull updates from the remote.

Git

Basic commands:

- git clone <url>
- git commit –m “message”
- git pull
- git push