1. Explain the workflow of the git with neat diagram

Git is a distributed version control system that helps developers track changes in their code, collaborate with teams, and manage different versions of a project efficiently. The Git workflow typically follows these steps:

**1. Working Directory**

* This is where you modify your files.
* Any new files, modifications, or deletions happen here.

**2. Staging Area (Index)**

* Before committing changes, you need to add them to the staging area using git add.
* This allows you to prepare specific changes for commit.

**3. Local Repository (.git)**

* When you commit, the changes move from the staging area to the local repository.
* This creates a snapshot of your project with a unique commit ID.

**4. Remote Repository (GitHub, GitLab, Bitbucket, etc.)**

* To share your work, you push commits to a remote repository using git push.
* Team members can fetch or pull updates using git pull or git fetch.

Working Directory

↓

(git add)

↓

Staging Area

↓

(git commit)

↓

Local Repository

↓

(git push)

↓

Remote Repository

1. What is git, how to stage a file with git add

Git is a **distributed version control system (VCS)** used to track changes in source code during software development. It allows multiple developers to collaborate, manage different versions of a project, and revert to previous states if needed.

**How to Stage a File with git add**

**Staging a file** means preparing it to be included in the next commit. The **staging area** acts as a middle step between modifying files and permanently saving them in the repository.

**Check the status of your repository**

git status

**Add a specific file to the staging area**

git add <filename>

**Verify staged files**

git status

1. What is git, github, gitlab .Explain the differences

**1. What is Git?**

Git is a **distributed version control system (VCS)** that allows developers to track changes in their code, collaborate with others, and manage different versions of a project. It is **open-source** and primarily used for software development.

**Key Features of Git:**

* Tracks changes to files over time.
* Enables collaboration in teams.
* Supports branching and merging for parallel development.
* Allows reverting to previous versions if needed.

**2. What is GitHub?**

GitHub is a **web-based platform** that provides Git repository hosting services. It allows developers to store, manage, and share their Git repositories online.

**Key Features of GitHub:**

* Cloud-based repository hosting.
* Pull Requests for code review.
* Issue tracking for project management.
* CI/CD integrations (GitHub Actions).
* Free for public repositories; private repositories available.

**3. What is GitLab?**

GitLab is another **web-based Git repository hosting service**, similar to GitHub, but with additional **built-in DevOps** features. It provides Git repository management, CI/CD pipelines, security testing, and deployment automation in one platform.

**Key Features of GitLab:**

* Self-hosting option (for on-premise control).
* Advanced CI/CD pipelines for automation.
* Built-in project management tools.
* Security scanning for vulnerabilities.
* Free and paid plans available.

|  |  |  |
| --- | --- | --- |
| Git | GitHub | GitLab |
| Version Control System | Cloud-based Git hosting | Cloud & Self-hosted Git hosting |
| Tracks code changes | Stores Git repositories online | Stores repositories + DevOps tools |
| Local only | Online collaboration via Pull Requests | Online collaboration with Merge Requests |
| No hosting, only version control | Hosted by GitHub (cloud-based) | Hosted by GitLab (cloud & self-hosted) |

1. How to create a tag, and how can we push it

Tags in Git are used to mark specific points in the project’s history, often to indicate **release versions** (e.g., v1.0, v2.1).

**Creating a Tag**

git tag <tag\_name>

**Creating an Annotated Tag**

git tag -a <tag\_name> -m "Tag message"

**Viewing Tags**

git tag

**View details of a specific tag**

git show <tag\_name>

**Pushing a Tag to a Remote Repository**

**Push a Single Tag**

git push origin <tag\_name>

**Push All Tags**

git push origin –tags

**Deleting a Tag**

**Delete a Local Tag**

git tag -d <tag\_name>

**Delete a Remote Tag**

git push --delete origin <tag\_name>

1. Example the steps which are necessary to send a project from local repo to remote repo

When working with Git, you often start with a **local repository** and later push your project to a **remote repository** (e.g., GitHub, GitLab, Bitbucket). Below are the necessary steps to accomplish this.

**Initialize a Git Repository**

git init

**Add a Remote Repository**

git remote add origin <remote\_repo\_URL>

**Add Files to the Staging Area**

git add .

**Commit the Changes**

git commit -m "Initial commit"

**Push the Project to the Remote Repository**

git push -u origin main

1. What is the pull request, explain its procedure

A **Pull Request (PR)** is a feature in Git-based platforms like **GitHub, GitLab, and Bitbucket** that allows developers to propose changes, review code, and merge updates into the main branch of a project.

It is commonly used in **collaborative development** where multiple contributors work on different features or fixes in separate branches and request the project maintainers to merge their changes into the main branch.

**Fork or Clone the Repository**

**Fork** the repository on GitHub/GitLab.

Clone it to your local machine

git clone <repo\_URL>

**Navigate to the project directory**

cd myproject

If you are working on your own repository, just clone it directly.

**Create a New Branch**

git checkout -b my-feature-branch

**Make Changes and Commit**

git add .

git commit -m "Added new feature"

**Push the Changes to the Remote Repository**

git push origin my-feature-branch

**Merge the Pull Request**

git checkout main

git merge my-feature-branch

git push origin main

**Delete the Branch**

git branch -d my-feature-branch

git push origin --delete my-feature-branch

1. What is fork, Why it is necessary

A **fork** is a personal copy of a repository that exists in your own GitHub, GitLab, or Bitbucket account. It allows you to freely experiment with changes without affecting the original repository.

**Forking is commonly used in open-source projects** where contributors don’t have direct access to the main repository but still want to contribute.

* Creates a copy of a repo in your account
* Independent copy, changes don’t affect the original
* Used to contribute to projects without write access