1. What is Git?

Git is a version control system that helps you manage and track changes in your code. It allows multiple developers to collaborate on the same project without overwriting each other's work. Git keeps a complete history of changes, making it easy to revert to earlier versions when needed.

2. What is a Git Repository?

A Git repository (or repo) is a storage space where your project and its entire change history are saved. There are two types:

Local Repository: Stored on your local computer.

Remote Repository: Hosted on platforms like GitHub, GitLab, or Bitbucket.

Git allows synchronization between local and remote repositories.

3.git pull:

The `git pull` command is used to download the latest changes from the remote repository and automatically merge them into your current working branch.

Example:

```
git pull origin main
```

This command pulls the latest changes from the `main` branch of the remote repository.

4. git push

The `git push` command is used to upload your local changes to a remote repository. This is essential for collaboration as it makes your changes visible to others.

Example:

```
git push origin feature-branch
```

This pushes your local branch named `feature-branch` to the remote repository.

5. Git Merging

Merging is the process of combining changes from one branch into another. Typically, you merge a completed feature branch into the `main` branch.

Example:

```
git checkout main
```

```
git merge feature-branch
```

This merges `feature-branch` into the `main` branch.

6. Git Merging Strategies:-

1. Fast-Forward Merge

Used When: The branch being merged has a linear history (no new commits on the target branch since it diverged).

How it Works: Git simply moves the target branch pointer forward to the latest commit on the source branch.

Command:

```
git checkout main
git merge feature-branch
```

Pros: Clean, linear history.

Cons: Doesn't preserve history of feature branches if deleted.

2. Recursive (Three-Way) Merge (Default):-

Used When: There are new commits on both branches since they diverged.

How it Works: Git creates a new commit that merges changes from both branches.

Command:

```
git merge feature-branch
```

Pros: Maintains a clear record of merges.

Cons: History can become cluttered with merge commits.

3. Squash Merge:-

Used When: You want to combine all changes from a feature branch into a single commit.

How it Works: Collapses all commits in a branch into one and merges that into the main branch.

Command:

```
git checkout main
git merge --squash feature-branch
git commit -m "Add feature XYZ"
```

Pros: Clean, single-commit history for features.

Cons: Original commit history is lost.

4. Rebase and Merge:-

Used When: You want to reapply feature branch commits on top of the target branch, avoiding merge commits.

How it Works: Moves your branch to begin on the tip of the target branch and reapplies your changes.

Command:

```
git checkout feature-branch
```

```
git rebase main
```

Pros: Clean, linear history without merge commits.

Cons: Can be risky if rebasing shared branches.

5. **Octopus Merge (Multiple Branches at Once):-**

Used When: Merging more than two branches at once, typically in automation or release branches.

Command:

```
git merge branch1 branch2 branch3
```

Pros: Can combine multiple branches at once.

Cons: Not suitable for complex merge conflicts.

7. Common Git Operations:-

Command	Description
`git init`	Initialize a new Git repository
`git clone`	Clone a remote repository to your local machine
`git status`	Show current branch status and changes
`git add`	Stage files to be committed
`git commit -m "message"`	Commit staged changes with a message
`git push`	Push committed changes to the remote repository
`git pull`	Fetch and merge changes from the remote repository
`git branch`	List, create, or delete branches

Command	Description
`git checkout`	Switch branches or restore files
`git merge`	Merge changes from another branch