GIT

**1. What is a GIT**

Git is a **distributed version control system** used to track changes in source code during development. It allows multiple developers to work on the same project without interrupting each other’s changes, manage versions, and collaborate effectively.

Git helps with branching, merging, rollback, and maintaining a complete history of the codebase.

**2. What happens when we enter git init?**

When we run git init, Git initializes a new empty repository in the current directory by creating a hidden .git folder. This folder contains all the metadata and configuration files required to track versions, like objects, refs, HEAD, and config. After initialization, we can start tracking files, staging changes, and committing files in that directory.

**3. What is git clone?**

git clone is used to **create a local copy of a remote Git repository**. It downloads all the files, branches, and commit history so we can start working on the project locally. It also sets up the origin remote, so we can pull and push changes to the original repository.

**4. What is the difference between git init and git clone?**  
git init initializes a **new local Git repository** in the current directory. On the other hand, git clone is used to **download a complete copy of an existing remote repository** including all files, commit history, and branches onto your local machine. It also sets up the remote (origin) automatically.

**5. What is a Git strategy in your current company?**  
  
In my company, we follow the **GitFlow strategy**. It provides a structured workflow with long-lived branches like main and develop, and feature branches for new developments.

* Developers create **feature branches** from develop and merge them back after completion.
* For every release, we create a **release branch** from develop, test it, and merge it into main and develop.
* **Hotfix branches** are created from main to fix production issues and are merged back into both main and develop.

This strategy helps maintain a clean history and supports parallel development, CI/CD, and stable production releases.

**6. What are key features of GIT?**  
  
Git has several key features that make it a powerful version control system:

1. **Distributed version control** – Every developer has a full copy of the repo with its history.
2. **Branching and merging** – Lightweight branches allow safe, parallel development.
3. **Fast performance** – Most operations are local and very fast.
4. **Data integrity** – all changes are tracked with unique hashes (SHA-1).
5. **Staging area** – lets us review and prepare changes before committing.
6. **Collaboration** – Works well with platforms like GitHub, GitLab, and Bitbucket.
7. **Undo capabilities** – Easy to revert commits, reset changes, or go back to earlier versions.

**7. What is a fork?**  
  
A **fork** is a copy of a repository made on platforms like GitHub or GitLab. It allows a user to **independently work on a project** without affecting the original repository. Developers usually fork a repo to contribute changes, and then create a pull request to suggest updates to the original project.

**8. What is the difference between GIT and SVN?**  
  
Git and SVN are both version control systems, but they work differently:

* **Git** is **distributed**, meaning every developer has a full copy of the repository. SVN is **centralized**, where there’s a single central repository.
* In Git, you can work offline and commit locally. In SVN, most operations require internet access.
* Git handles **branching and merging** more efficiently than SVN.
* Git is faster because it performs most actions locally, while SVN depends on the server.
* Git ensures better **data integrity** using SHA-1 hashes.

In short, Git is more flexible and modern, while SVN is older and better suited to centralized workflows.

**9. What is the difference between git merge and git rebase?**

Both git merge and git rebase are used to integrate changes from one branch into another, but they do it differently:

* **git merge** combines the histories of two branches and creates a **new merge commit**. It keeps the complete history and is useful for collaboration.
* **git rebase** moves your branch’s commits and **reapplies them on top** of another branch. It creates a **linear, cleaner history**, but rewrites commit hashes.

### ****10. What is the difference between**** git pull ****and**** git fetch****?”****

git pull and git fetch are both used to get updates from a remote repository, but they behave differently:

* **git pull** does **both fetching and merging** in one step. It downloads changes from the remote and automatically merges them into my current branch.
* **git fetch** only **downloads the changes** from the remote repo into your local .git folder. It doesn’t change your working directory or merge anything.

So, if I want to just **review** changes before applying them, I use **fetch**. If I want to **update my branch directly**, I use pull.

### 11. ****What is the difference between**** git reset ****and**** git revert****?****

### ****git reset** is a command to undo changes by moving the head pointer to a specific commit. It can unstage changes or discard them entirely depending on the type of reset (hard, soft and mixed).**

### ****git revert** is a command to undo changes made by a specific commit by creating a new commit that reverses those changes.**

### ****12. What is a git cherry-pick?**** git cherry-pick is a command which is used to apply specific commits from one branch to another without merging the whole branch.

### 13. What is a git webhook?

### A ****Git** **webhook**** is a way to automatically trigger actions or notifications when certain events occur in a Git repository. For example, when a commit is pushed to a repository, a webhook can notify a continuous integration (CI) server to run tests or deploy the code. Webhooks are typically set up in Git hosting services like GitHub, GitLab, or Bitbucket, and they send HTTP requests to a specified URL when an event (like a push, pull request, etc.) happens.

### 14. Commands to delete a branch in local repository and remote repository?

To **delete a branch** in a **local repository**, use the following command:

git branch -d <branch-name>

If you want to **force delete** a branch (even if it hasn’t been merged), use:

git branch -D <branch-name>

To **delete a branch** in a **remote repository**, use:

git push origin --d <branch-name>

### 15. What is git bisect?

### git bisect is a Git command used to **find the commit that introduced a bug** or issue by using a binary search method between good and bad commits.

### 16. What is git reflog?

### git reflog is a command that shows the **history of HEAD and branch references** in your Git repository.

### It helps you track and recover lost commits or changes even if they don’t appear in regular git repository.

### 17. What is SubGit?

### **SubGit** is a tool that enables **bidirectional synchronization** between **Subversion (SVN)** and **Git** repositories.

It allows teams to use **both Git and SVN** simultaneously, meaning you can have a **Git mirror** of an SVN repository and vice versa. This is useful for teams transitioning from SVN to Git or when some team members prefer using Git, while others continue using SVN.

**18. If one commit got deleted in a branch, how do you know about the details of the deleted commit?**

If a commit got deleted in a branch, you can use **git** **reflog** to check the history of the branch and find details about the deleted commit.  
git reflog tracks the changes to the HEAD and allows you to see past states, even if commits were deleted or overwritten.

**19. How do you solve a merge conflict and are there any additional tools for solving merge conflict problems?**

To resolve a merge conflict, I manually edit the files, stage them, and commit. I also use tools like git mergetool or graphical tools like **Sourcetree** for easier resolution.

**20. How many users can add in git for access management?**

Git itself doesn't limit users; user access is managed by the Git hosting platform (like GitHub or GitLab) or through Git servers (like Gitolite).

**21. Where git history will store in git?**

Git history is stored in the **.git directory**, inside subfolders like .git/objects and .git/refs.

### 22. What is git InstaWeb?

**git instaweb** starts a local web server to view a repository’s commit history in a browser.

**23. What is a git Clean?**

git clean is a command used to **remove untracked files** and directories from your working directory.

**git clean -f**: Removes untracked files.

**git clean -fd**: Removes untracked files and directories.

**git clean -n**: Shows which files would be removed without actually deleting them (dry run).

**24. What is a git stash?**

git stash is used to temporarily store changes in your working directory and that are not yet ready to be committed.(OR) temporarily saves uncommitted changes.

**git stash save “message”**: Saves your changes to a stash.

**git stash list**: Lists all stashes.

**git stash apply**: Reapplies the most recent stash to your working directory.

**git stash pop**: Applies the most recent stash and removes it from the stash list.

**git stash drop**: Removes a specific stash from the list.

**git stash clear**: delete the entire stash list

**25. What is a detached HEAD?**

A **detached HEAD** in Git means that your repository is in a state where you're not on a branch but directly on a specific commit.

OR

A detached HEAD occurs when HEAD points to specific commit instead of latest commit in a branch.

OR

It occurs when you're not on a branch, but on a specific commit

**26. What is the difference between Git and GitHub?**

Git is a version control system, It runs locally on your computer. Git can be used offline as it operates locally on your machine.

GitHub is a web-based platform where git repositories can be stored and shared. It is a cloud-based service. GitHub requires an internet connection because it is hosted on the web.

### 27. What is the purpose of .git directory?

### It stores all the information about the repository like configuration, history, branches, and objects.

### 28. What is a merge conflict?

### A merge conflict happens when two people change the same part of a file, and Git doesn’t know what changes to keep. Git can’t automatically merge changes, it creates a conflict. Resolve by manually editing the file, staging it, and committing.

**29. What is git tag?**

Used to mark specific points (like releases) in Git history. Example: git tag v1.0.

### ****30. How do you undo the last commit?****

**Answer:**

* To remove commit but keep changes: git reset --soft <CID-1>
* To remove commit and changes: git reset --hard <CID-1>

**31. What is a .gitignore?**

A file used to specify which files or directories Git should ignore (not track).

**32. What is Git Submodule and when do you use it?**

Git Submodules allow you to include another Git repository inside your own repo.  
Useful for managing **shared dependencies**

### ****33. How do you integrate Git with CI/CD tools?****

### Git is integrated with tools like **Jenkins, GitHub Actions, GitLab CI** to trigger builds, tests, and deployments automatically on:

* Push events
* Pull requests
* Tag creation

We use webhooks or native Git integrations.

### 34. ****How do you secure Git repositories****

* Use SSH or HTTPS for secure access.
* Set proper **branch protection rules** (e.g., require PR reviews, disallow force pushes).
* Use **fine-grained access controls** (e.g., via GitHub teams or IAM roles in AWS CodeCommit).

**35. What is a git alias?**

A **Git alias** is a shortcut command that you can define to simplify longer Git commands. It helps improve productivity by allowing you to create custom, easy-to-remember Git commands.

**Example**: git config --global alias.co checkout

### 36. How do you fix a detached HEAD?

### Create a new branch from the detached HEAD state with ‘ git branch <new br.name>’ and check it out with ‘git checkout <branchname>’to move back to a non-detached state.

### 37. How do you combine multiple commits into one without merging?

### Use ‘git rebase –interactive’ to squash commits into a single commit without creating a merge commit.

### 38. How do you list all the remote branches?

### Use ‘git branch –r’ to list all remote branches.

### 39. What is the purpose of `git log --graph`?

### ‘git log –graph’ displays the commit history in a graphical representation.

### 40. What is a fast-forward merge?

### A fast-forward merge happens when the target branch's (master) head is behind the merged branch's head, allowing the target branch to fast-forward to the tip of the merged branch.

### 41. How do you list all the remote connections for a repository?

### Use `git remote -v` to list all remote connections.

### 42. How do you list all the branches that are merged into the current branch?

### Use `git branch --merged` to lists all **branches that have been fully merged** into your current branch. ‘git branch -no-merged’ It lists the branches that have not been merged.

### 43. What is a ‘git commit --amend’?

### In Git, git commit --amend is used to ****modify the most recent commit****. I usually use it when I forget to include a file or want to correct the commit message. It replaces the pervious commit with a new one.

### For example, if I forgot to add a file, I stage it and run ‘git commit --amend --no-edit’ to include it in the last commit without changing the message.

### 44. What is ‘git commit –am “<message>”?

### In Git, when we use git commit -am "message", it takes changes directly from the workspace to the local repository but only for modified tracked files. Newly created (untracked) files are not included because Git doesn't know about them yet. So, for new files, I first run git add filename to track them, and then commit.

### 45. How to give access to a specific person to repository?

### You can invite users to become collaborators to your personal repository.

### • Under your repository name, click Settings. • In the left sidebar, click Collaborators. • Under "Collaborators", start typing the collaborator's username. • Select the collaborator's username from the drop-down menu. • Click Add collaborator. • The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.

### 46. How to delete Repository in GitHub?

### • On GitHub, navigate to the main page of the repository. • Under your repository name, click Settings. • Scroll to the bottom of the page and you will find Delete this repository button • When you click on that button, another pop up will appear, here you need to type in the name of your repository name and click on the button bellow which says: I understand the consequences, delete the repository.