1. What is Git? Why Is It Used?

Answer: Git is a distributed version control system that helps track changes in source code

during software development. It is widely used for several reasons:

1. Version Control:

\* Git allows developers to track changes in their codebase over time, providing a detailed history

of modifications.

2. Collaboration:

\* Multiple developers can work on the same project concurrently, with Git managing and merging

their changes seamlessly.

3. Branching and Merging:

\* Git enables easy creation of branches for parallel development, and merging changes back into

the main branch when features are complete.

4. History and Rollback:

\* Developers can review the project's history, understand when and why changes were made, and

easily roll back to previous states if needed.

6. Distributed Development:

\* Git supports distributed development, allowing developers to work independently and merge

their changes when ready.

7. Offline Work:

\* Developers can work offline, committing changes locally, and later syncing with remote repositories.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. What is a Git repository?

Answer: A Git repository, often referred to as a "repo,"

A Git repository is a place where your project's files, history of changes, and different

versions are stored and managed using Git version control.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3. What does git clone do?

Answer: The git clone command is used to create a copy of a Git repository. When you run

git clone, it not only copies the files but also sets up a connection to the remote repository,

allowing you to fetch updates and push changes back to the original repository.

For Exa : git clone https://github.com/example/repository.git

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. Different types of version control systems?

Answer: The two main categories are:

1. Centralized Version Control System (CVCS):

Examples: CVS (Concurrent Versions System), Subversion (SVN).

-- Characteristics:

\* Centralized repository where all team members access and commit changes.

\* Linear workflow, requires a network connection for most operations.

\* Risk of a single point of failure if the central server goes down.

2. Distributed Version Control System (DVCS):

Examples: Git, Mercurial, Bazaar.

-- Characteristics:

\* Every user has a complete copy of the repository, allowing offline work.

\* Branching and merging are efficient and done locally.

\* No single point of failure, as each copy is a full backup of the project.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. What is the difference between Git and GitHub?

Answer: Git :

Definition: Git is a distributed version control system used for tracking changes in source

code during software development.

GitHub :

Definition: GitHub is a web-based platform that provides hosting for software development using Git.

GitHub is a platform built around Git, providing a cloud-based space for collaborative development,

code hosting, and additional tools.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6. How does Git work?

Answer: Git works by tracking changes in a project through a distributed version control system.

Developers create snapshots of the project called commits, forming a timeline of changes. Developers

work in branches, and changes can be merged. It's decentralized for collaboration and efficient version control.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7. What is a commit in Git?

Answer: In Git, a commit is a snapshot of changes made to files in a Git repository. It represents

a specific version of the project and includes a message describing the changes. Commits help

track the project's history and provide a way to revert to or compare different versions.

A commit is a snapshot of changes made to the code at a specific point in time.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8. What is branching in Git?

Answer: Branching in Git is a way to create separate lines of development, allowing you to work

on features or fixes without affecting the main code. It enables isolation, experimentation, and

organized collaboration, with the ability to merge changes back into the main branch when ready.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9. What is a merge in Git?

Answer: In Git, a merge combines changes from different branches. It takes changes made

in a source branch and integrates them into a target branch, creating a unified history.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10. What is a conflict in Git?

Answer: A conflict arises when two separate branches have made edits to the same

line in a file, or when a file has been deleted in one branch but edited in the other.

Conflicts will most likely happen when working in a team environment.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11. What is a pull request?

Answer: A pull request is a proposal to merge a set of changes from one branch into

another. In a pull request, collaborators can review and discuss the proposed set of

changes before they integrate the changes into the main codebase.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

12. How do you revert a commit that has already been pushed and made public?

Answer: To undo the modifications introduced by a previous commit while ensuring safety for public

commits, employ `git revert <commit\_hash>`, which generates a fresh commit that reverses the earlier

changes. On the other hand, `git reset` allows reverting to an earlier state; however, exercise caution

when applying it to public commits, as it alters the commit history.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13. How do you clone a repository?

Answer: To clone a repository, use the git clone command followed by the repository URL.

For example: git clone https://github.com/example/repository.git

This command creates a local copy of the repository on your machine.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

14. What is a remote in Git?

Answer: A remote in Git is a common repository that all team members use to exchange their

changes. In most cases, such a remote repository is stored on a code hosting service like GitHub

or on an internal server.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

15. How do you create a new branch?

Answer: Use `git branch <branch\_name>` to create a new branch. Use `git checkout -b <branch\_name>`

to create and switch to it in one step.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

16. What is the difference between `git merge` and `git rebase`?

Answer: The main difference is in how the branch history is presented. `git merge` preserves the history

of a feature branch by creating a new merge commit. `git rebase` rewrites the feature branch's history

to appear as if it was developed from the latest main branch, creating a linear history.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

17. How do you change the last commit?

Answer: Use `git commit --amend` to modify the most recent commit. This can change the

commit's message or include new changes.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18. What is `git push`?

Answer: `git push` is used to upload local repository content to a remote repository. It transfers

commits from your local repo to the remote.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

19. How do you delete a branch?

Answer: Use `git branch -d <branch\_name>` to delete a local branch. If the branch is not fully

merged, you may need to use `-D` instead. To delete a remote branch, use `git push <remote\_name>

--delete <branch\_name>`.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20. What is `git checkout`?

Answer: `git checkout` allows navigating between different branches or reverting working tree

files to a previous state. However, in the latest versions of Git, it is advised to use `git switch`

for changing branches and `git restore` to revert files, each designated for these specific functions.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

21. How do you rename a Git branch?

Answer: To rename the current branch, use `git branch -m <new\_name>`. To rename a different

branch, use `git branch -m <old\_name> <new\_name>`.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

22. Explain the Git branching strategy you use.

Answer: A common strategy is the Git Flow, which involves having a master branch, develop branch, feature

branches, release branches, and hotfix branches, each serving a different purpose in the development cycle.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

23. How do you revert a Git repository to a previous commit?

Answer: Use `git reset --hard <commit-hash>` to revert to a specific commit, discarding all

changes since that commit.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

24. How do you rename a remote branch?

Answer: Rename the local branch, push it to the remote, and then delete the old remote branch.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

25. Explain the Git workflow?

Answer: The basic Git workflow involves creating branches, making changes, committing those

changes, and then merging those changes back into the main branch.