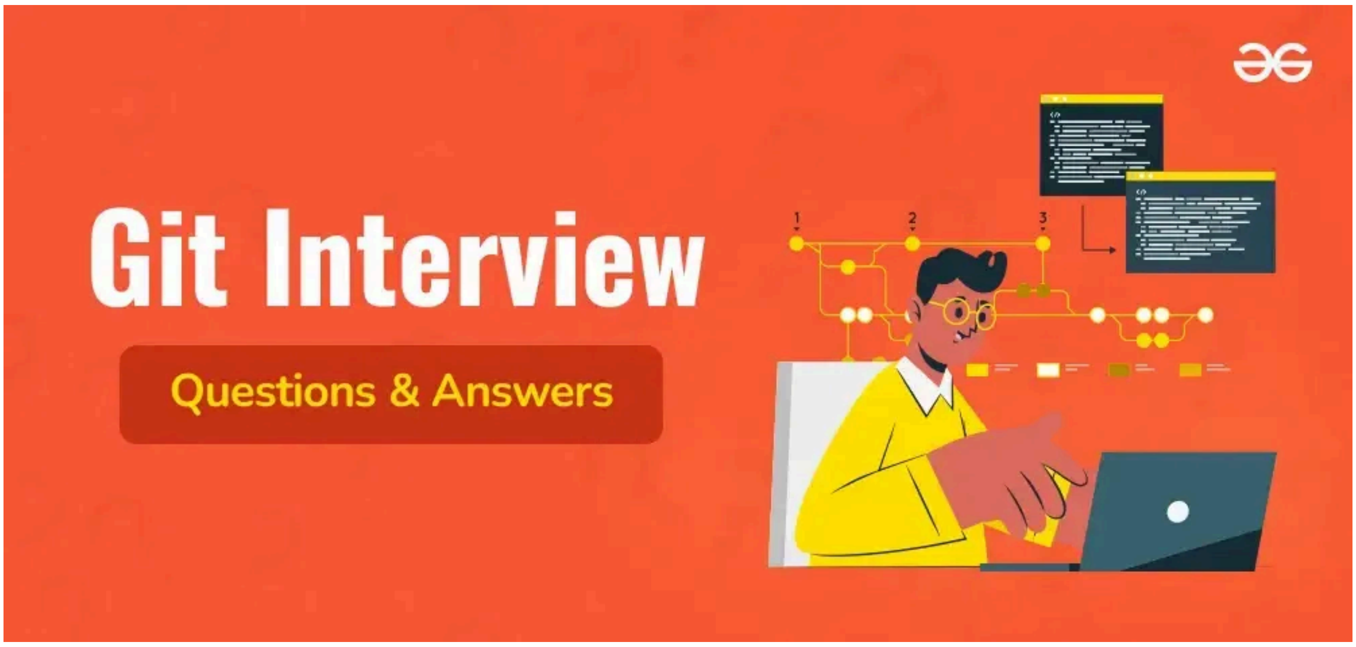
Top 70+ Git Interview Questions and Answers - 2025

GeeksforGeeks July 2, 2024



Top 70+ Git Interview Questions and Answers - 2025

Git is a distributed version control system (DVCS) designed to track changes in source code during software development. In 2005, Linus Torvalds, who is also the creator of the Linux kernel, developed it. Today, more than 40 million people use Git universally. If you are a developer, DevOps engineer, or any professional in the tech industry, knowing Git is vital.

In this Git interview questions article, we will provide Top Git and GitHub interview questions and answers. To help you get ready for interviews, we have gathered the **top 70+ Git interview questions for 2025** for freshers, intermediate, and experienced candidates. Going through these questions will boost your chances of getting a job at top MNCs.

Table of Content

Basic Git Interview Questions for Freshers

1. What is Git?

Git is a distributed version control system (DVCS) that is used to track changes in source code during software development. It permits multiple developers to work on a project together without interrupting each other's changes. Git is especially popular for its speed, and ability to manage both small and large projects capably.

2. What is a repository in Git?

A Git repository (or repo) is like a file structure that stores all the files for a project. It continues track changes made to these files over time, helping teams work together evenly. Git can control both local repositories (on your own machine) and remote repositories (usually hosted on platforms like GitHub, GitLab, or Bitbucket), allowing teamwork and backup.

3. What is the difference between Git and GitHub?

Git	**GitHub**
Git is a version control system used to track changes in files over time	GitHub is a platform where Git repositories can be stored and shared
It runs locally on your computer	It is a cloud-based service
Git can be used offline, as it operates locally on your machine.	GitHub requires an internet connection because it is hosted on the web

4. What is origin in Git?

In Git, "origin" states to the default name offered to the remote repository from which local repository was cloned. Git origin is used as a reference to control fetches, pulls, and pushes.

5. What is the purpose of the .gitignore file?

The '.gitignore' file tells Git which files and folders to ignore when tracking changes. It is used to avoid attaching unneeded files (like logs, temporary files, or compiled code) to your repository. This saves repository clean and targeted on important files only.

6. What is a version control system (VCS)?

A version control system (VCS) records the work of developers coordinating on projects. It keeps the history of code changes, permitting developers to add new code, fix bugs, and run tests securely. If required, they can restore a past working version, verifying project security.

7. What is the git push command?

The 'git push' command is used to share local repository changes to a remote repository. It changes the remote repository with the recent commits from the fixed local branch.

8. What is the git pull command?

The 'git pull' command updates the current local branch with changes from a remote repository and combining it with a local repository.

9. What does git clone do?

The git clone forms a copy of a remote repository upon your local machine. Git clone downloads all files, branches, and history, enabling you to start working on the project or contribute to it. With git clone -b , you can download and work on an individual branch of a repository.

10. What are the advantages of using GIT?

Using Git provides multiple advantages:

- It assists teamwork by supporting multiple developers to work on the same project together.
- Each developer has a local copy of the repository, improving performance and enabling offline work.
- Free and widely supported.
- Git supports work on various types of projects.
- Each repository has only one Git directory.

11. What is the difference between git init and git clone?

The git init develops a new, empty Git repository in the present directory, while 'git clone' copies an existing remote repository, containing all files and history, to a local directory.

12. What is git add?

The git add command marks changes in your project for the next commit. It tells Git which files to involve in the later update, making them ready to be saved in the repository. This is the early step in recording changes in the Git repository.

13. What is git status?

The 'git status' command shows the recent status of your Git repository. It tells you which files have changed, which ones are ready to be committed, and which ones are new and unobserved. This benefits you monitor your work's growth and see what changes want to be set up or committed.

14. What is a commit in Git?

A commit in Git denotes a snapshot of changes made to files in a repository. It grabs all the changes you have made to files—like additions, or deletions of files at a particular moment. Each commit has a unique message explaining what was done. This helps you track your project's history, undo changes if requisite, and work with others on the same project.

15. What is the purpose of the git clean command?

The git clean command is used to erase ignored files from the working directory of Git repository. Its motive is to clean up the workspace by deleting files that are not being saved by Git, checking a clean state with only observed files present.

16. What is a 'conflict' in git?

Git usually manages merges automatically, but conflicts occur when two branches edit the same line or when one branch deletes a file that another edits.

17. What is the meaning of 'Index' in GIT?

In Git, the 'Index' (also called as the "Staging Area") is a place where alterations are temporarily store before committing them to the repository. It permits you to select and prepare specific alterations from your working directory before properly saving them as part of the project's history.

18. How do you change the last commit in git?

To change the preceding commit in Git, use 'git commit --amend' after making changes, stage them with 'git add' , and save with the editor.

19. What is `git checkout`?

The 'git checkout' helps you switch between branches or return files to a previous state in Git. Now, it is suggested to use 'git switch' for changing branches and 'git restore' to return files. These commands are more intent on their particular tasks for better clearness and capability.

20. How do you switch branches in Git?

To switch branches in Git, use 'git checkout ' to move to a present branch. On the other hand, use git switch in newer Git versions for the same objective. This permits you to work on different versions or features of your project stored in separate branches.

21. Name some popular Git hosting services?

- GitHub
- GitLab
- SourceForge.net
- Bitbucket
- Visual Studio Online

22. What are the different types of Git repositories?

Git has two types of repositories

- **Bare Repository:** A repository without a working directory, typically used for sharing projects remotely.
- **Non-Bare Repository:** A repository that includes a working directory where developers can modify and track files.

23. How does Git handle file deletion?

Git provides the git rm command to remove files from the working directory and the staging area. If you only want to remove a file from Git but keep it in the working directory, use:

```
git rm --cached <file_name>
```

24. How can you create an alias in Git?

Aliases can be created to simplify Git commands using:

```
git config --global alias.<alias_name> '<git_command>'
```

25. How do you rename a branch in Git?

To rename the current branch:

```
git branch -m <new_branch_name>
```

To rename a different branch:

```
git branch -m <old_branch_name> <new_branch_name>
```

26. What is the difference between git fetch and git pull?

Command	Description
git fetch	Retrieves updates from a remote repository but does not merge them into the local repository. This allows reviewing changes before integrating them.
git pull	Fetches updates from a remote repository and immediately merges them into the current local branch. Equivalent to git fetch followed by git merge.

27. Explain Git rebase and when do you use it?

The Git rebase is a process to combine alterations from one branch into another. It forms a linear history, avoiding merge commits. Use it to clean up commit history, keep a project history sequential, and make feature branches up-to-date before uniting.

28. How will you create a git repository?

- Download Git on your system if you have not already.
- Create a project folder in the location where you want your repository.
- Open Terminal or Command Prompt and guide to your project folder.
- Run 'git init' in the project folder. This will create a '.git' folder, showing your repository is set.

29. What differentiates between the commands git remote and git clone?

Command	Description
git remote	Manages connections to remote repositories. It allows users to add, remove, and view remote repositories linked to a local project. However, it does not download any files.
git clone	Creates a local copy of an existing remote repository, including all its files, branches, and commit history. This allows developers to start working on a project immediately.

30. What are the benefits of using a pull request in a project?

Teams can together work on distinct parts of the system and later combine their changes using pull requests. This way boosts team capability.

- **Code Review:** Pull requests allow team members to review code before merging, ensuring better code quality.
- **Collaboration:** Multiple developers can work on the same project and discuss changes within the pull request.
- **Version Control:** Helps track proposed changes and keeps the main branch stable.

31. What is a Git bundle?

A Git bundle is a collective file that wraps all data from Git repository, such as commits, branches, and tags. It acts as a handy approach for relocating a repository offline or sharing upgrades when network connection is not available. To form a git bundle, perform the following command:

```
git bundle create <bundle_file> <refs>
```

32. What are the advantages of Git over SVN?

Here the some advantages of Git over SVN:

- **Distributed Version Control:** Git allows every developer to have a complete copy of the repository, whereas SVN relies on a centralized model.
- **Faster Performance:** Git is generally faster due to local operations, while SVN requires network communication for many actions.
- **Branching and Merging:** Git offers lightweight and efficient branching, whereas SVN branches are heavier and more complex.
- **Offline Work:** Since Git is distributed, developers can work offline, committing changes locally before pushing them.
- **Better Conflict Resolution:** Git provides more advanced tools for resolving merge conflicts compared to SVN.

33. What is git stash?

The git stash is a command used to temporarily save changes that are not yet ready to be committed. It allows developers to switch branches or work on something else without losing their progress. Stashed changes can be reapplied later using git stash pop or git stash apply.

Key Git Stash Commands:

Save current changes:

```
git stash
```

View stashes:

```
git stash list
```

Reapply the most recent stash and remove it from stash history:

```
git stash pop
```

Reapply a stash without removing it from history:

```
git stash apply
```

Delete a specific stash:

```
git stash drop stash@{n}
```

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

To revert a commit that has been pushed and made public, follow these steps:

Checkout the Branch: Switch to the branch where you want to revert the commit.

```
git checkout <branch-name>
```

Find the Commit to Revert: Use 'git log' to find the commit hash of the commit you want to revert.

```
git log
```

Revert the Commit: Use 'git revert' followed by the commit hash of the commit you want to revert.

```
git revert <commit-hash>
```

- Review Changes: Git will open your default text editor to confirm the revert message. Save and close the editor to proceed.
- Push the Revert: Finally, push the reverted commit to the remote repository.

```
git push origin <branch-name>
```

35. Explain the difference between reverting and resetting?

Operation	Description
git reset	Moves the HEAD pointer to a previous commit, removing changes from the commit history. It can modify staged and working directory changes.
git revert	Creates a new commit that undoes changes from a previous commit without modifying commit history. It is safer when working in a shared repository.

36. What is the difference between git reflog and log?

Feature	git reflog	git log
Purpose	Tracks movements of HEAD and branch changes	Displays commit history of a repository
Visibility	Shows all references, even those not in branch history	Only shows commits in the current branch history
Recoverability	Can help recover lost commits	Does not track changes outside commit history
Use Case	Used to restore lost branches or commits	Used to review past commits and changes

37. What is the HEAD in Git?

- HEAD in Git is a reference to the latest commit on the currently checked-out branch.
- It determines which commit and branch you are working on.
- When switching branches, HEAD updates to point to the latest commit of the new branch.
- If in a detached HEAD state, it means you are working on a specific commit instead of a branch.

38. What is the purpose of 'git tag -a'?

The git tag -a command is used to create an annotated tag in Git. Annotated tags store additional metadata such as the tagger's name, email, date, and a message describing the tag. These tags are useful for marking important points in a repository, such as software releases.

```
git tag -a v1.0 -m "Version 1.0 Release"
```

To push annotated tags to a remote repository:

```
git push origin v1.0
```

39. What is the difference between 'HEAD', 'working tree' and 'index' in Git?

Concept	Description	Role	Location
HEAD	A reference to the current commit or branch you are working on. It points to the tip of the current branch.	Represents the current branch or commit you are on	Located in .git/HEAD.
Working Tree	The directory where your project files reside. It reflects the actual state of files on your filesystem and includes changes made but not yet staged.	Contains the actual files, including any modifications or additions not yet staged.	The local directory in your project.
Index(Staging Area)	A file (also called the staging area) that acts as a buffer between the working tree and the repository. Files staged here are ready to be committed.	Holds changes that you want to commit to the repository.	Located in .git/index.

40. How to resolve a conflict in Git?

To resolve a conflict in Git

- **Identify Conflict:** Git will alert you of a conflict during merge or rebase.
- **Open the Conflicted File:** You'll see conflict markers like <<<<<<< HEAD, =======, and >>>>>>>.
- **Resolve the Conflict:** Edit the file to keep your changes, the incoming changes, or a combination of both.
- **Stage the Resolved File:** Use git add <file> to mark the conflict as resolved.
- **Commit the Changes:** Run git commit to complete the merge.
- **Continue Rebase (if applicable):** Use git rebase --continue if you were rebasing.

41. Explain the difference between 'git merge' and 'git rebase' and when you would use each?

Feature	git merge	git rebase
What it does	Combines two branches and creates a merge commit.	Reapplies commits from one branch on top of another, creating a linear history.

Feature	git merge	git rebase
Commit History	Keeps the commit history of both branches intact, including a merge commit.	Rewrites the commit history to make it linear.
Result	A new merge commit that preserves the branch structure.	A clean, straight history without merge commits.
When to use	When you want to preserve both branches' histories and structure.	When you want a clean, linear commit history without merge commits.
Best for	Collaborative environments and when integrating feature branches.	Personal branches, squashing, or cleaning up commit history before pushing.
Merge Commit	Yes, a merge commit is created.	No, history is rewritten without merge commits.

42. What language is used in GIT?

Git is mainly developed using the C programming language. The core features and commands of Git, containing its data structures and algorithms, are applied in C. This choice of language confirms productivity, speed, and portability across distinct operating systems and platforms.

43. How do you add a file to the staging area?

To add a file to the staging area in Git, use the command:

```
git add <file>
```

This stages the file for the next commit. To stage all changes, use:

```
git add .
```

44. What is 'git diff?

`git diff` shows the differences between files or commits. It compares changes in the working directory, staging area, or between two commits.

- **Compare working directory with staging area**: `git diff`
- **Compare staged changes with last commit**: `git diff --staged`
- **Compare two commits**: `git diff <commit1> <commit2>`

45. What is a Git commit hash?

A Git commit hash is a unique identifier (SHA-1) for each commit. It helps to reference and track specific commits in the repository. You can use it to check out or reset to a particular commit.

46. What is a detached HEAD state?

A detached HEAD state occurs when you check out a specific commit rather than a branch. In this state, commits are not attached to any branch, meaning they may be lost if not properly managed.

47. How can you delete a remote Git branch?

To delete a remote branch:

```
git push origin --delete <branch-name>
```

This removes the branch from the remote repository.

48. What is the purpose of git cherry-pick?

Here's the purpose of git cherry-pick in points:

- **Selective Commit Application:** Applies changes from a specific commit to the current branch.
- **No Full Branch Merge:** Useful when you want to incorporate changes from a branch without merging the entire branch.
- **Clean Commit History:** Helps maintain a clean history by including only selected changes.
- **Bug Fixes:** Ideal for bringing specific bug fixes or features from one branch to another.
- **New Commit:** The changes are applied as a new commit on the current branch, preserving the original commit's content.

49. What does git ls-files do?

The command git ls-files lists all files that are currently tracked by Git in the repository. It displays the files in both the working directory and the staging area. This command helps identify which files are being tracked and are ready to be committed. It does not show untracked files or files ignored by .gitignore.

50. How do you fetch all remote branches?

Run:

```
git fetch --all
```

This command fetches all branches and their latest changes from all remotes without merging them.

Advanced Git Interview Questions for Experienced

51. What is the Git object model?

The Git object model consists of four major types of objects:

- **Blobs:** Store the file data (the contents of files).
- **Trees:** Represent directory structures and contain references to blobs (files) and other trees (subdirectories).
- **Commits:** Store snapshots of the repository at a particular point in time, along with metadata like author, timestamp, and parent commit references.
- **Tags:** Store references to specific commits, typically used for marking important points in the repository's history, such as releases.

52. Explain 'git rebase' and when you would use each?

git rebase moves or reapplies commits from one branch onto another, creating a linear commit history.

When to use:

- **Clean Commit History:** To avoid merge commits and keep a straight line of history.
- **Sync with Latest Changes:** To update your branch with the latest changes from another branch.
- **Squashing Commits:** To combine multiple commits into one before merging.

53. What is a git hook and how might you use it?

A Git hook is a script that runs automatically at certain points in the Git workflow, like before or after a commit, merge, or push.

Use cases:

- **Pre-commit:** Check code quality or run tests before committing.
- **Post-commit:** Automate tasks like notifications after a commit.
- **Pre-push:** Ensure tests pass before pushing changes.

54. Explain the difference between git reset, git revert, and git checkout?

- **git reset:** Moves HEAD to different commit, potentially changing history.
- **git revert:** Undoes exact commit by making new commit with inverse changes.
- **git checkout:** Switches branches or checks out files from commit, putting you in "detached HEAD" state for direct commits.

55. How do you handle large files with Git?

To handle large files with Git, use Git LFS (Large File Storage). It replaces large files in the repository with pointers, while the actual file content is stored externally.

Steps:

1. Install Git LFS:

```
git lfs install
```

2. Track large files:

```
git lfs track "*.psd"
```

3. Commit and push as usual. Git LFS will handle the large file storage automatically.

56. What is ‘bare repository’ in Git?

A bare repository in Git is a repository that doesn't have a working directory. It only contains the Git data, such as branches, tags, and commits, without the actual project files. Bare repositories are typically used as remote repositories where other developers push and pull changes.

No working directory (no actual files).

Used for sharing changes between collaborators (remote).

Stored typically in the .git folder format.

57. What is branching in Git?

Branching in Git permits forming separate lines of development. It allows users to work on features or fixes separately from the main codebase, helping parallel development and simpler integration of changes.

58. What is a Git submodule?

A Git submodule is a repository embedded inside another Git repository. It allows you to keep track of an external repository as part of your project. Submodules are useful when you want to include external libraries or dependencies while keeping their history separate.

59.How can you undo a commit that hasn’t been pushed to the remote repository?

We can use git reset to undo a commit:

To keep changes in the working directory:

```
git reset --soft HEAD~1
```

To remove changes from both the commit and working directory:

```
git reset --hard HEAD~1
```

60. How do you squash multiple commits into one using Git rebase?

To squash multiple commits into one:

1. Start an interactive rebase:

```
git rebase -i HEAD~<number-of-commits>
```

2. In the editor, change pick to squash (or s) for the commits you want to combine.

3. Save and close the editor to squash the commits.

4. Git will open another editor to combine commit messages. Edit the message and save.

61. How do you reset a commit to a previous commit without losing changes in the working directory?

To reset to a previous commit without losing changes:

```
git reset --soft <commit-hash>
```

62. What is the difference between git reset --hard and git clean -fd?

- **git reset --hard:** Resets the commit history, the staging area, and the working directory to a specific commit, discarding any changes.
- **git clean -fd:** Removes untracked files and directories from the working directory but does not affect the commit history or staged changes.

63. How do you apply a patch from a remote Git repository?

To apply a patch:

1. Fetch the patch from the remote repository:

```
git fetch <remote> <branch>
```

2. Apply the patch:

```
git apply <patch-file>
```

64. How do you configure a Git repository to use a specific user for a particular project?

To set a specific user for a project:

```
git config user.name "Your Name"
git config user.email "your.email@example.com"
```

65. What is a Git reflog, and how is it useful?

The Git reflog tracks the history of changes to the HEAD and branches, including changes that are not part of the commit history (e.g., resets, rebase). It allows you to recover lost commits by providing references to previous states.

Use:

```
git reflog
```

66. How do you manage multiple remotes in a Git repository?

We can manage multiple remotes using:

Add a new remote:

```
git remote add <remote-name> <remote-url>
```

View remotes:

```
git remote -v
```

Remove a remote:

```
git remote remove <remote-name>
```

67. How do you configure and use Git hooks for pre-commit checks?

Git hooks allow you to run custom scripts at various points in the Git workflow. For pre-commit checks:

- Go to the .git/hooks directory.
- Rename or copy pre-commit.sample to pre-commit.
- Write your script to check code style, run tests, etc.
- Make the hook script executable:

```
chmod +x .git/hooks/pre-commit
```

68. How do you manage large binary files in Git without using Git LFS?

One way to manage large binary files without using Git LFS is by using Git Large File Storage (LFS), but if you can't use LFS, you can store large files in a separate repository and reference them using Git submodules.

69.How do you fetch changes from multiple remotes in Git?

To fetch changes from all remotes:

```
git fetch --all
```

70. How do you perform a Git bisect to find the commit that introduced a bug?

To perform a Git bisect:

Start the bisect process:

```
git bisect start
```

Mark the current commit as bad (where the bug is present):

```
git bisect bad
```

Mark a known good commit (where the bug is not present):

```
git bisect good <commit-hash>
```

- Git will check out a commit between the "good" and "bad" ones. Test the code to see if the bug is present.
- Repeat the process by marking the commit as bad or good based on whether the bug is present.
- Once the bad commit is found, stop the bisect process:

```
git bisect reset
```

Conclusion

In conclusion, **Git** is a crucial tool for developers, especially when it comes to managing projects and collaborating on code. If you're preparing for an interview, understanding **GitHub interview** questions and knowing your way around common **Git commands interview questions** can greatly improve your chances of success.

Top 70+ Git Interview Questions and Answers - 2025