**✅ Git Interview Questions & Answers for Freshers (0–1 Year Experience) – 10 Q&A**

**Q1. What is Git?**

**A**: Git is a distributed version control system (DVCS) used to track changes in source code during software development. It allows multiple developers to work on a project simultaneously, manage versions, and collaborate efficiently.

**Q2. What is the difference between Git and GitHub?**

**A:**

Git is a version control tool that runs locally on your machine.

GitHub is a cloud-based hosting service for Git repositories. It provides collaboration features like pull requests, issue tracking, and code review.

**Q3. How do you initialize a Git repository?**

**A:** Use the command:

git init

This creates a new .git directory in the current folder, initializing it as a Git repository.

**Q4. What is a commit in Git?**

**A:** A commit is a snapshot of your code at a specific point in time. It records changes made to files with a unique hash, author, timestamp, and commit message.

Command:

git commit -m "Your message here"

**Q5. How do you check the current status of your repository?**

**A:** Use:

git status

It shows modified, staged, and untracked files.

**Q6. What is the staging area in Git?**

**A:** The staging area (or index) is where you prepare changes before committing. You add files to the staging area using:

git add <filename>

or

git add .

**Q7. How do you clone a remote repository?**

**A:** Use:

git clone <repository-URL>

This creates a local copy of the remote repository.

**Q8. What is a branch in Git?**

**A:** A branch is an independent line of development. The default branch is usually main or master. You can create branches to work on features or fixes without affecting the main code.

**Q9. How do you create and switch to a new branch?**

**A:**

git checkout -b feature/new-ui

or (newer syntax):

git switch -c feature/new-ui

**Q10. How do you push changes to a remote repository?**

**A:** Use:

git push origin main

This pushes your local commits to the main branch on the remote (origin).

**✅** Git Interview Questions & Answers for 2 Years Experience – 20 Q&A

**Q1. Explain the Git workflow.**

**A:** The typical Git workflow involves:

git clone – get the repo

git checkout -b feature/xyz – create a new branch

Make changes, git add, git commit

git push origin feature/xyz

Create a Pull Request (PR) on GitHub/GitLab

Code review → merge → delete branch

**Q2. What is the difference between git pull and git fetch?**

**A:**

git fetch downloads changes from remote but doesn’t merge them.

git pull = git fetch + git merge. It downloads and merges changes automatically.

**Q3. How do you revert a commit that has already been pushed?**

**A:** Use:

git revert <commit-hash>

This creates a new commit that undoes the changes. Safe for shared branches.

**Q4. What is a merge conflict and how do you resolve it?**

**A**: A merge conflict occurs when Git can’t automatically merge changes (e.g., same lines modified in both branches).  
To resolve:

Open the conflicting file.

Look for <<<<<<<, =======, >>>>>>>.

Edit the file to keep desired changes.

git add <file> and git commit.

**Q5. What is the purpose of .gitignore?**

**A:** .gitignore lists files/folders Git should not track (e.g., node\_modules/, .env, logs). Prevents unnecessary or sensitive files from being committed.

**Q6. How do you view commit history?**

**A:** Use:

git log --oneline

Shows a compact list of commits with hashes and messages.

**Q7. How do you delete a branch locally and remotely?**

**A:**

Local: git branch -d <branch-name>

Remote: git push origin --delete <branch-name>

**Q8. What is git stash?**

**A:** git stash saves your uncommitted changes temporarily so you can switch branches. Later, use git stash pop to restore them.

Example:

bash

1

2

git stash # save changes

git stash pop # restore latest stash

**Q9. How do you rename a local branch?**

**A:**

git branch -m old-name new-name

**Q10. What is the difference between git merge and git rebase?**

**A:**

git merge combines branches by creating a new merge commit.

git rebase moves the entire branch to begin on the tip of another, resulting in a linear history.

Use merge for public branches, rebase for cleaning up local commits.

**Q11. How do you squash multiple commits into one?**

**A:** Use interactive rebase:

git rebase -i HEAD~3

Then mark commits as squash or s to combine them.

**Q12. What is a detached HEAD state?**

**A:** It occurs when you're not on any branch, usually after checking out a specific commit. To fix, create a branch:

git checkout -b temp-branch

**Q13. How do you discard local changes in a file?**

**A:**

git checkout -- <filename>

or (in newer Git versions):

git restore <filename>

**Q14. What does git reset do?**

**A:** git reset moves the current branch pointer backward.

git reset --soft: Keeps changes staged.

git reset --mixed (default): Unstages changes, keeps in working directory.

git reset --hard: Discards all changes (dangerous!).

**Q15. How do you find which commit introduced a bug?**

**A:** Use git bisect:

git bisect start

git bisect bad HEAD

git bisect good <known-good-commit>

# Test until Git finds the bad commit

**Q16. What is a pull request (PR)?**

**A:** A pull request is a mechanism (on GitHub/GitLab) to propose changes from one branch to another, usually for code review before merging.

**Q17. How do you list all branches (local and remote)?**

**A:**

git branch -a

**Q18. What is the difference between a shallow and deep clone?**

**A:**

Deep clone: Full history (default).

Shallow clone: Limited history (e.g., git clone --depth 1) – faster but limited for rebasing.

**Q19. How do you check who changed a specific line in a file?**

**A**: Use git blame:

git blame <filename>

Shows who last modified each line.

**Q20. How do you set up a Git alias?**

**A:** Example:

git config --global alias.co checkout

Now git co works like git checkout.

**✅ Git Interview Questions & Answers for 5 Years Experience – 20 Q&A**

**Q1. Explain Git internals: what is a commit object?**

**A:** A commit object contains:

Tree hash (root directory structure)

Parent commit hash(es)

Author and committer info

Timestamp

Commit message  
All stored as SHA-1 (or SHA-256 in newer Git) hash.

**Q2. What is the Git object model?**

**A:** Git stores data as objects:

Blob: File content

Tree: Directory structure (points to blobs/subtrees)

Commit: Points to a tree and parent(s)

Tag: Named reference to a commit

All stored in .git/objects.

**Q3. How does Git ensure data integrity?**

**A:** Every object is checksummed using SHA-1 (or SHA-256). The hash becomes its ID. Any change alters the hash, making tampering detectable.

**Q4. What is a signed commit, and how do you create one?**

**A:** A signed commit uses GPG to verify authorship.  
Create with:

git commit -S -m "Signed commit"

Requires GPG setup and user.signingkey configured.

**Q5. Explain the difference between rebasing a feature branch vs. merging it into main.**

**A:**

Merge: Preserves history as-is, creates a merge commit. Safe for shared branches.

Rebase: Rewrites history, makes linear timeline. Best for local cleanup before PR.

Rebasing public branches can disrupt collaborators.

**Q6. How do you recover a deleted branch?**

**A:** If the branch was recently deleted:

Find the commit hash via git reflog

Recreate the branch:

git checkout -b branch-name <commit-hash>

**Q7. What is git reflog and when would you use it?**

**A:** git reflog logs all reference changes (e.g., branch switches, resets). Useful to recover lost commits or branches.

**Q8. How do you rewrite the last commit message?**

**A:**

git commit --amend -m "New message"

If already pushed, use git push --force-with-lease cautiously.

**Q9. What is git cherry-pick and when is it used?**

**A:** Applies a specific commit from one branch to another.

git cherry-pick <commit-hash>

Useful to backport hotfixes or move isolated changes.

**Q10. How do you handle large files in Git?**

**A:** Use Git LFS (Large File Storage) to store big files (videos, datasets) outside the repo, with pointers in Git.

**Q11. What are Git hooks? Give examples.**

**A:** Scripts that run automatically on Git events.  
Examples:

pre-commit: Run tests/lint before commit

pre-push: Validate code

post-merge: Install dependencies

Stored in .git/hooks/.

**Q12. What is a submodule?**

**A:** A submodule is a Git repo embedded inside another Git repo. Useful for including external libraries or shared components.

Commands:

bash

1

2

git submodule add <repo-url>

git submodule update --init --recursive

**Q13. How do you configure Git for multiple accounts (e.g., work and personal)?**

**A**: Use conditional includes in ~/.gitconfig:

ini

# ~/.gitconfig

**[includeIf "gitdir:~/work/"]**

path = ~/work/.gitconfig

**[includeIf "gitdir:~/personal/"]**

path = ~/personal/.gitconfig

Each sub-config sets user.name and user.email.

**Q14. What is the difference between git reset --hard and git clean?**

**A:**

git reset --hard: Removes uncommitted changes in tracked files.

git clean: Removes untracked files (use -f to force, -fd for directories).

**Q15. How do you list all commits that changed a specific file?**

**A:**

git log --follow <filename>

--follow tracks file across renames.

**Q16. What is a bare repository?**

**A:** A bare repo has no working directory – only .git contents. Used as central repos (e.g., on servers). Created with:

git init --bare

**Q17. How do you debug a broken rebase or merge?**

**A:** Use:

git status to see conflicts

git rebase --abort or git merge --abort to cancel

git reflog to go back to a safe state

**Q18. What is git worktree and why use it?**

**A:** git worktree allows multiple working trees for the same repo. Useful to work on multiple branches simultaneously without stashing.

Example:

git worktree add ../feature-branch feature/new-ui

**Q19. How do you enforce commit message conventions in a team?**

**A:** Use:

Commit hooks (e.g., commit-msg hook)

Tools like Husky + Commitlint

CI checks to reject invalid messages

**Q20. Explain Git’s distributed nature and its advantages.**

**A:** Every developer has a full copy of the repo with complete history.  
Advantages:

Work offline

Faster operations

Redundancy (no single point of failure)

Flexible workflows (feature branches, forks)