**Git Interview Questions with Answer**

**🔰 Beginner Level**

**1. What is Git, and how is it different from other version control systems like SVN?**

**Answer:**  
Git is a **distributed version control system** used for tracking changes in source code during software development. Unlike SVN, which is **centralized**, Git allows each developer to have a full copy of the repository, enabling offline work and faster operations.

**Example:**  
In SVN, if the server goes down, you lose access. In Git, even if the server is down, you can continue working locally and push your changes later.

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

**Answer:**

* **Git**: A version control system.
* **GitHub**: A hosting platform for Git repositories that also provides collaboration features.

**Example:**  
You use Git to commit code changes locally, and GitHub to **push** them to a shared repository so your team can collaborate.

**3. What are the basic Git commands you use daily?**

**Answer:**

* git clone <repo\_url> – Clone a repository
* git add . – Stage changes
* git commit -m "message" – Commit changes
* git push – Push to remote
* git pull – Fetch and merge remote changes

**4. What is the purpose of git init?**

**Answer:**  
git init initializes a new Git repository in your local project folder.

**Example:**  
You just created a new project folder my-app. Run git init to start tracking it with Git.

**5. Explain the difference between git add, git commit, and git push.**

**Answer:**

* git add – Moves changes to staging area.
* git commit – Records changes in local repository.
* git push – Uploads commits to the remote repository.

**Example:**  
Modify index.js → git add index.js → git commit -m "Update index" → git push origin main

**6. What does git status show?**

**Answer:**  
It shows the status of files in the working directory – tracked, untracked, modified, staged.

**7. How do you create a new branch in Git?**

**Answer:**  
git branch feature-login – Creates a branch  
git checkout feature-login – Switches to it  
OR  
git checkout -b feature-login – Both in one step

**8. How do you switch between branches?**

**Answer:**  
git checkout branch-name or git switch branch-name (newer syntax)

**9. What is a merge conflict? How do you resolve it?**

**Answer:**  
Occurs when two branches have changes in the same part of the same file.

**Example:**

* feature1 updates line 10 of index.js
* feature2 also updates line 10  
  When merged, Git cannot decide which change to keep. You resolve manually and commit the fix.

**10. Explain git clone vs git pull.**

**Answer:**

* git clone – Copies a remote repo for the first time
* git pull – Fetches and merges changes from remote into your local branch

**🔄 Intermediate Level**

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

**Answer:**

* git fetch – Downloads updates without merging
* git pull – git fetch + git merge  
  Allows review before merging.

**12. How does git rebase differ from git merge?**

**Answer:**

* git merge – Keeps history of both branches.
* git rebase – Rewrites history to make it linear.

**Example:**  
Feature branch appears like it's based on latest main.

**13. What is the use of .gitignore?**

**Answer:**  
Specifies files Git should ignore.

**Example:**  
Ignore node\_modules/, \*.log, or .env in .gitignore.

**14. What is a detached HEAD in Git?**

**Answer:**  
HEAD is not pointing to any branch – you're checking out a specific commit.

**Example:**  
Running git checkout 3a5b8e1 (a commit hash) puts you in detached HEAD state.

**15. How do you revert a commit?**

**Answer:**

* git revert <commit> – Adds a new commit that undoes the changes.

**16. How do you squash commits?**

**Answer:**  
Use interactive rebase:  
git rebase -i HEAD~3  
Then mark old commits as squash or s.

**17. Explain git stash and how to use it.**

**Answer:**  
Temporarily saves uncommitted changes.  
git stash – Stash current changes  
git stash pop – Apply them later

**18. Difference between git reset, git checkout, and git revert**

**Answer:**

* git reset – Moves HEAD and can unstage or delete commits
* git checkout – Switch branches or restore files
* git revert – Reverses changes with a new commit

**19. How can you view commit history in Git?**

**Answer:**  
Use git log  
Or git log --oneline --graph --all for a graphical view.

**20. What is cherry-picking in Git?**

**Answer:**  
Apply a specific commit from one branch to another.  
git cherry-pick <commit-hash>

**🚀 Advanced Level**

**21. What is Git rebase interactive (git rebase -i) and when do you use it?**

**Answer:**  
Used to edit, squash, reorder commits interactively.  
git rebase -i HEAD~n opens a list of last n commits.

**22. How do you resolve conflicts after a rebase?**

**Answer:**  
Git marks conflicts. You manually edit files, then run:

* git add .
* git rebase --continue

**23. How do you handle large binary files in Git?**

**Answer:**  
Use **Git LFS (Large File Storage)**  
git lfs install, git lfs track "\*.zip"

**24. How do you undo a pushed commit?**

**Answer:**  
If not shared:

* git reset --hard HEAD~1  
  If shared:
* git revert <commit>

**25. What are Git hooks?**

**Answer:**  
Scripts that run on Git events (e.g., pre-commit, pre-push).  
Use for auto-formatting or validation.

**26. How do you configure Git in a CI/CD pipeline?**

**Answer:**  
Use .git to fetch source, clone repo, checkout specific branch. Example using GitHub Actions:

yaml

CopyEdit

- uses: actions/checkout@v2

**27. What’s the difference between local, remote, and upstream branches?**

**Answer:**

* **Local**: Exists on your system
* **Remote**: Copy from central repo
* **Upstream**: Tracking branch you pull from/push to

**28. How do you clean up a Git repository?**

**Answer:**

* Delete old branches: git branch -d old-feature
* Remove large files: bfg-repo-cleaner or Git LFS
* Prune: git gc, git prune

**29. Explain the concept of Git reflog.**

**Answer:**  
git reflog tracks every move of HEAD. Useful to recover lost commits.

**Example:**  
You reset hard and lost work? Use git reflog to find and restore.

**30. What is the difference between origin/main and main?**

**Answer:**

* main – Local branch
* origin/main – Remote tracking branch

**1. You accidentally committed a secret (like a password). How would you remove it from Git history?**

**Answer:**  
Use git filter-branch or BFG Repo-Cleaner to remove the secret from all commits.  
Example using BFG:

bash

CopyEdit

bfg --delete-files ".env"

Then:

bash

CopyEdit

git reflog expire --expire=now --all

git gc --prune=now --aggressive

Force push the cleaned history:

bash

CopyEdit

git push --force

**2. What is git bisect and how do you use it?**

**Answer:**  
git bisect uses binary search to find the commit that introduced a bug.

**Example:**

bash

CopyEdit

git bisect start

git bisect bad # current commit is bad

git bisect good <commit> # known good commit

# Test each checkout and mark good/bad

git bisect good

git bisect bad

...

git bisect reset # return to original state

**3. What’s the difference between git reset, git revert, and git checkout?**

**Answer:**

* git reset – Moves HEAD and optionally changes staging/working directory.
* git revert – Creates a new commit that undoes the changes of an old one.
* git checkout – Switches branches or restores files from a commit.

**4. What is the difference between git pull --rebase and git pull?**

**Answer:**

* git pull = fetch + merge
* git pull --rebase = fetch + rebase your local commits on top of the remote branch.

--rebase keeps history linear and cleaner.

**5. How do you resolve a situation where you committed on the wrong branch?**

**Answer:**

bash

CopyEdit

# Switch to the correct branch

git checkout correct-branch

# Cherry-pick the commits

git cherry-pick <commit1> <commit2>

# Go back and delete from the wrong branch if needed

git checkout wrong-branch

git reset --hard HEAD~2

**6. What is the Git index?**

**Answer:**  
The **index (staging area)** is the middle layer between the working directory and the local repository.  
When you run git add, files are added to the index.  
When you git commit, the index is saved as a new commit.

**7. How would you recover a deleted commit?**

**Answer:**  
Use git reflog to find the commit hash and then check it out or reset to it:

bash

CopyEdit

git reflog

git checkout <commit-hash>

Or:

bash

CopyEdit

git reset --hard <commit-hash>

**8. What’s the difference between annotated and lightweight tags in Git?**

**Answer:**

* **Annotated tag**: Full object stored in Git (with message, author, date).

bash

CopyEdit

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

* **Lightweight tag**: Just a pointer to a commit.

bash

CopyEdit

git tag v1.0

**9. What is git stash, and how do you apply a specific stash?**

**Answer:**  
git stash temporarily shelves your uncommitted changes.

To apply a specific stash:

bash

CopyEdit

git stash list

git stash apply stash@{2}

To apply and delete:

bash

CopyEdit

git stash pop stash@{2}

**10. What is a tracking branch? How do you create one?**

**Answer:**  
A tracking branch follows a remote branch (e.g. origin/main).

**Create one:**

bash

CopyEdit

git checkout -b feature origin/feature

Or set manually:

bash

CopyEdit

git branch --set-upstream-to=origin/main main

**11. How do you find which commit introduced a specific change in a file?**

**Answer:**

bash

CopyEdit

git blame filename

Or for a specific line range:

bash

CopyEdit

git log -L 10,20:filename

**12. How does Git handle file renaming?**

**Answer:**  
Git does not track renames explicitly. Instead, it detects renames heuristically during git diff or git log if enough content is similar.

**13. What does git clean do?**

**Answer:**  
Deletes **untracked** files (not staged or committed).

bash

CopyEdit

git clean -f # Delete untracked files

git clean -fd # Also delete untracked directories

Use git clean -n to preview.

**14. How do you squash the last 3 commits into one?**

**Answer:**

bash

CopyEdit

git rebase -i HEAD~3

# Change "pick" to "squash" for the last 2 commits

**15. What are the contents of a .git folder?**

**Answer:**

* HEAD – current branch ref
* refs/ – branches and tags
* objects/ – blobs, commits, trees
* config – repo config
* index – staging area
* logs/ – reflog history