Git Interview-Questions-Answers

1. What is Git?

Git is a distributed version control system used for tracking changes in source code during software development.

2.What makes Git a Distributed Version Control System?

Git allows developers to work locally with full version history, then push changes to a remote repository like GitHub.

3.Compare Git and SVN.

Git is distributed and allows offline work; SVN is centralized and requires internet connectivity.

4.What is a Git repository?

It's a storage space where Git tracks and saves all changes made to project files.

5.How to initialize a Git repository?

git init initializes a new Git repository.

6.What is a bare repository?

A Git repo without a working directory, typically used as a central shared repository.

```
git init --bare myproject.git
```

- This way, the bare repository acts like a GitHub or GitLab repository, but hosted in-house.
- Since the repo is not meant for editing or working in, having a working directory is unnecessary.
- It avoids confusion or merge conflicts caused by someone accidentally editing files in a shared repo.

7.How do you configure Git user details globally?

```
git config --global user.name "Your Name"

git config --global user.email "your@email.com"
```

8.How to create aliases for Git commands?

git config --global alias.co checkout makes git co work as git checkout.

9.What does git clone do?

Downloads an entire remote repository to your local machine.

10.What is the purpose of git add?

Moves changes from the working directory to the staging area.

11.What is the staging area in Git?

An intermediate area to review and format commits before finalizing them.

12.How to commit with both add and message in one step?

git commit -a -m "message"

13.How to edit a commit message?

git commit --amend -m "New message"

14.How to undo a commit but keep the changes staged?

git reset --soft HEAD~1

15.How to remove a file from staging area?

git reset HEAD <filename>

16.Difference between --soft, --mixed, and --hard reset?

- Soft: resets to commit, keeps changes staged
- Mixed: resets and unstages changes
- Hard: resets and discards changes

17.What is HEAD in Git?

A pointer to the current branch reference.

18.What is the purpose of .gitignore?

To exclude files from being tracked by Git.

19.How to view the commit history?

git log

git log --oneline

git log --author="user"

20.How to compare two commits?

git diff commit1 commit2

21.How to recover a deleted file?

git checkout -- <filename>

- Note : It discards local changes in a file

22.How to create and switch to a new branch?

git checkout -b new_branch

23.How to rename a branch?

git branch -m old_name new_name

24.How to delete local and remote branches?

• Local: git branch -d branch_name

• Remote: git push origin --delete branch_name

25.How to see all local and remote branches?

git branch -a

26.Difference between git fetch and git pull?

git fetch

- Downloads commits, files, and references from the remote repository into your local repository, but does not merge them into your working directory.
- Safe to use because it doesn't change your working files.
- Ideal when you want to see what others have changed before integrating it into your code.

git pull

- It is essentially: git fetch + git merge

- It fetches changes from the remote and then automatically merges them into your current branch.
- Can cause merge conflicts if your local changes conflict with remote ones.

27.How to fetch a remote branch without merging?

git fetch origin branch_name

28.How to merge a branch?

Be in the target branch, then: git merge source_branch

29.What causes a Git merge conflict?

When changes in the same line/file differ across merged branches.

30.How to resolve merge conflicts?

Edit the file manually, then git add and git commit.

31.How to abort a merge conflict?

git merge --abort

32.What does git rm do?

Deletes files from working directory and stages the deletion.

33. How to check which branches have been merged into master?

git branch --merged

34.Describe Git branching strategy you've used.

Feature, Task, and Release branching (explain with real project use).

35.What is Git stash and when do you use it?

Temporarily stores changes not ready to commit. Useful when switching branches mid-work.

36.Commands related to Git stash:

- Save: git stash save "msg"
- Apply: git stash apply stash@{0}
- Pop: git stash pop

- List: git stash list

37.How is git rebase different from git merge?

- Merge:
  - Preserves full history with a merge commit
  - Combine two branches, keeping both histories
  - Collaborative work, preserves all contributions
- Rebase:
  - Move (replay /rebase) your changes onto another branch
  - re-applies commits in a linear fashion; cleaner history
  - Rewrites history – dangerous if used on public/shared branches

38. How to revert a commit?

git revert <commit_id>

39.How to change the remote URL?

git remote set-url origin <new_url>

40.What is a pull request (PR)?

A request to merge changes from one branch to another, usually with a code review.

41.What is a fast-forward merge?

A merge where the target branch pointer is simply moved forward to the latest commit.

42.How to force push changes?

git push origin branch_name --force

43.How to tag a specific commit?

git tag v1.0 <commit_id>

Push tags: git push origin v1.0

44.Difference between annotated and lightweight tags?

Annotated has metadata and is stored in Git history. Lightweight is just a pointer.

45.How to list and delete tags?

• List: git tag

• Delete: git tag -d tagname

46.What is a detached HEAD state?

When HEAD points to a specific commit instead of a branch.

47.How to cherry-pick a commit from another branch?

git cherry-pick <commit_id>

48.How to rebase interactively?

git rebase -i HEAD~n to squash/reword commits.

49.How do you lock a branch in GitHub?

In repository settings → Branches → Add branch protection rules.

50.How to grant access to a GitHub repository?

Go to Settings → Collaborators → Invite user by GitHub username.

51.what is the difference between git revert and git reset?

git revert

- Creates a new commit that undoes the changes made by a previous commit.
- Does NOT remove any commit from history.
- Safe to use in shared/public branches (like main or develop), because it preserves history.

git reset

- Moves the HEAD and optionally updates the working directory and staging area.
- Can be destructive (especially with --hard), because it removes commits from history.
- Typically used in private branches or local development.

Common options:

git reset --soft : Keeps changes staged

git reset --mixed (default): Keeps changes in working directory git reset --hard : Discards all changes

52.What are the steps to push your code to a central repo (e.g., GitHub)?

Answer: git init #### Initialize local repo

git add . #### Stage changes

git commit -m "message" #### Commit changes

git remote add origin #### Link to central repo

git push -u origin main #### Push code

53. git push vs git pull ?

git push

- You use it to share your code with others.
- Only works if your local branch is ahead of the remote.

Example:

git push origin main – pushes your local main to the remote origin.

git pull

- Combines git fetch + git merge.
- It brings in changes from the remote repository into your current branch.

Example:

git pull origin main – fetches and merges changes from origin/main.

54. GitHub vs Bitbucket ?

GitHub

- A popular platform for hosting Git repositories, especially used for open-source projects.
- Owned by Microsoft and widely adopted by individual developers and the open-source community.
- Strong features like GitHub Actions, Copilot, and seamless integration with VS Code.

Bitbucket

- A Git repository hosting service designed for teams and enterprises.
- Owned by Atlassian, and integrates deeply with Jira, Trello, and Confluence.
- Offers Bitbucket Pipelines for built-in CI/CD, great for managing private team projects.

55. You have code on your local machine and have pushed it to a remote repository. Now you want to push the same code to a different (new) remote repository. How can you change the remote URL?

To change the remote repository URL

Check current remote URL:

git remote -v

Change the remote URL to the new repository:

git remote set-url origin <new-repo-url>

Replace with the URL of the new Git repository (e.g., from GitHub or Bitbucket).

git push origin main

(Use main or whatever your branch name is.)

git remote set-url origin https://github.com/username/new-repo.git

git push origin main

For More information:

https://github.com/devopstraininghub/Interview-Question-Answers/tree/main/GIT