@coderss_world

50 GIT & GITHUB INTERVIEW QUESTIONS

Basic Git Questions

1. What is Git?

A distributed version control system for tracking changes in source code.

2. What are the differences between Git and GitHub?

Git is a version control system; GitHub is a hosting service for Git repositories.

3. How do you install Git on your system?

Use package managers like apt, yum, or download from git-scm.com.

4. How do you initialize a new Git repository?

Run git init in your project directory.

5. What is a Git repository?

A directory that tracks changes to files with Git.

6. How do you clone a repository?

Use git clone <repository-url>.

7. What is a branch in Git?

A pointer to a specific commit, allowing isolated development.

8. How do you create a new branch?

Use git branch <branch-name> or git checkout -b <branch-name>.

9. How do you switch between branches?

Use git checkout <branch-name>.

10. What is the difference between git merge and git rebase?

git merge combines branches; git rebase moves or combines commits from one branch onto another.

11. How do you delete a branch?

Use git branch -d <branch-name> or git branch -D <branch-name>.

12. What is a commit in Git?

A snapshot of changes in the repository.

13. How do you make a commit?

Use git commit -m "commit message".

14. What is a staging area or index in Git?

An intermediate area where changes are kept before committing.

15. How do you stage files for a commit?

Use git add <file-or-directory>.

16. What is the purpose of git add?

To add changes in the working directory to the staging area.

17. How do you view the commit history?

Use git log.

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

git pull fetches and merges changes; git fetch only downloads changes.

19. How do you undo a commit?

Use git revert <commit> or git reset <commit>.

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

When changes from different branches conflict; resolve by editing the files and committing.

Intermediate Git Questions

21. What is git stash and how is it used?

Temporarily stores changes in a dirty working directory; use git stash to save and git stash apply to retrieve.

22. How do you apply a stashed change?

Use git stash apply or git stash pop.

23. What is git cherry-pick and when would you use it?

Applies changes from a specific commit to the current branch; use for applying selective commits.

24. How do you configure a Git repository to ignore certain files?

Create a .gitignore file with the file patterns to ignore.

25. What is git tag and how do you create a tag?

Marks specific points in history; use git tag <tag-name>.

26. How do you view all tags in a repository?

Use git tag.

27. What is git blame and how is it useful?

Shows the author of each line in a file; useful for finding who made changes.

28. What is the purpose of .gitignore?

To specify intentionally untracked files to ignore.

29. How do you remove a file from the staging area?

Use git reset <file>.

30. What is git bisect and how is it used?

Finds the commit that introduced a bug by binary search; use git bisect start.

Advanced Git Questions

41. What is the difference between git stash pop and git stash apply?

git stash pop applies and removes the stash; git stash apply only applies.

42. How do you handle large binary files in Git?

Use Git LFS (Large File Storage).

43. What is the difference between origin and upstream in Git?

origin is the default remote repository; upstream is usually the original repository forked from.

44. How do you handle Git hooks?

Use scripts in the .git/hooks directory.

45. What is git filter-branch used for?

Rewrites Git history by applying filters to the entire branch.

46. How do you set up Git to handle large files (e.g., using Git LFS)?

Install Git LFS and track files with git lfs track <file>.

47. What is git reflog and how is it useful?

Shows a log of all reference updates; useful for recovering lost commits.

48. How do you integrate Git with CI/CD tools?

Configure CI/CD tools to trigger on Git events like pushes and pull requests.

49. How do you enforce commit message guidelines?

Use Git hooks or commit message templates.

50. What strategies can be used for effective branching and merging in a large team?

Use workflows like Git Flow, feature branches, and pull requests for review.

31. How do you rebase a branch?

Use git rebase <branch-name>.

32. What is the difference between git reset and git revert?

git reset moves the branch pointer and working directory; git revert creates a new commit that undoes changes.

33. How do you amend the most recent commit?

Use git commit --amend.

34. How do you set up a remote repository?

Use git remote add <name> <url>.

35. What is git remote and how do you use it?

Manages set of tracked repositories; use commands like git remote add, git remote remove.

36. How do you remove a remote repository?

Use git remote remove <name>.

37. What is git log and how can you customize its output?

Shows commit history; customize with options like --oneline, --graph.

38. How do you use git diff to view changes?

Use git diff to show changes between commits, branches, or the working directory.

39. What is git submodule and how do you manage it?

Manages repositories inside another repository; use git submodule add <repo-url>.

40. How do you squash commits?

Use interactive rebase git rebase -i and mark commits with squash.