Git Interview Questions (Basic to Advanced)

Basic Git Questions

1. What is Git? Why do we use Git?

2. What is the difference between Git and GitHub?

3. What are Git repositories?

4. What is a commit in Git?

5. What is the command to initialize a Git repository?

6. How do you check the status of your Git repository?

7. What does 'git add' do?

8. What does 'git commit' do?

9. What is a remote repository?

10. What is the difference between 'git fetch' and 'git pull'?

11. How do you clone a repository?

12. What is a branch in Git? Why do we need branches?

13. What is HEAD in Git?

14. What is the default branch in Git called?

15. How do you view all branches in a repository?

16. How to create a new branch?

17. How to switch to another branch?

18. What does 'git push' do?

19. What is '.gitignore' file used for?

20. How do you see your commit history?

Medium Git Questions

21. Explain the lifecycle of a file in Git. (Untracked, Staged, Committed)

22. What is the difference between a local branch and a remote branch?

23. How do you merge two branches?

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

25. What is 'git stash'? When would you use it?

26. What is 'git rebase' and how is it different from 'git merge'?

27. What is 'git cherry-pick'?

28. What is a detached HEAD state in Git?

29. What is 'git reset'? What are the differences between '--soft', '--mixed', and '--hard' reset?

30. What is the use of 'git revert'? How is it different from 'git reset'?

31. What is fast-forward merge?

32. How do you delete a branch locally and remotely?

33. How do you track a remote branch locally?

34. What are Git tags? Difference between lightweight and annotated tags?

35. How do you compare the difference between two branches?

36. What is 'git log --oneline --graph'? Why use it?

37. What are Git hooks?

38. Explain the three stages of Git (Working Directory, Staging Area, Repository).

39. What is '.git/config' and '~/.gitconfig'?

40. What is the difference between 'git pull --rebase' and 'git pull'?

Advanced Git Questions

41. Explain how Git works internally (object storage model - blob, tree, commit, tag).

42. What is the difference between 'origin/master' and 'master'?

43. How can you recover a deleted branch in Git?

44. What happens if you delete '.git' folder inside a repository?

45. How can you squash multiple commits into one?

46. How to undo the last commit but keep the changes in the working directory?

47. Explain 'git reflog' and when you use it.

48. How do you find a specific commit if you don't know the commit hash?

49. How to move a commit from one branch to another?

50. How would you handle multiple people working on the same file and frequent merge conflicts?

Scenario-Based Git Questions

1. You and your teammate both edited the same line of a file and pushed changes. How will you resolve th

2. You accidentally did a 'git reset --hard' and lost changes. Can you recover them? How?

3. You committed to the wrong branch by mistake. How will you move that commit to the correct branch?

4. You pushed a wrong commit to the shared remote branch. How will you fix it without breaking others' wo

5. How would you clean up messy commit history before pushing code to production (using rebase/squash

6. You started working on a new feature but suddenly got another urgent task. How will you save your curr

7. You cloned a repo but only main branch is visible. How would you fetch all branches and switch?

8. You forked a project, made changes, and want to merge it back to the original repo. How will you raise a

One-liner Explanations for Git Concepts

1. 'git init' - Initializes a new Git repository in the current directory.

2. 'git clone <repo-url>' - Clones a remote repository to your local machine.

3. 'git status' - Displays the state of the working directory and staging area.

4. 'git add <file>' - Adds changes in the specified file to the staging area.

5. 'git commit -m "<message>"' - Commits the staged changes with a descriptive message.

6. 'git log' - Shows the commit history for the current branch.

7. 'git pull' - Fetches changes from the remote repository and merges them with the local repository.

8. 'git push' - Pushes local commits to the remote repository.

9. 'git branch' - Lists, creates, or deletes branches in the repository.

10. 'git merge <branch>' - Combines changes from the specified branch into the current branch.

11. 'git rebase <branch>' - Moves or combines commits from one branch onto another, rewriting history.

12. 'git reset --hard <commit>' - Resets the working directory, staging area, and commit history to a specifi

13. 'git revert <commit>' - Reverts a commit by creating a new commit that undoes the changes.

14. 'git stash' - Temporarily saves changes that are not yet committed and reverts the working directory.

15. 'git fetch' - Downloads updates from a remote repository but does not merge them.

16. 'git log --oneline --graph' - Shows a compact, graphical representation of the commit history.