66 Git interview questions to ask developers (with answers)

Questions

1. Can you explain what Git is and why it's useful for developers?
2. How do you handle merge conflicts in Git?
3. What is a 'pull request' and how does it fit into a Git workflow?
4. Can you explain the difference between Git and GitHub?
5. What is a branch in Git, and why would you use one?
6. What is the purpose of a commit message, and what makes a good commit message?
7. How do you revert a commit in Git?
8. Can you explain the difference between git fetch and git pull?
9. How would you undo the last commit while keeping the changes?
10. What is git stash and when would you use it?
11. Can you describe the Git flow branching model?
12. How do you squash multiple commits into one?
13. What's the difference between a fast-forward merge and a three-way merge?
14. How would you find which commit introduced a specific bug?
15. Can you explain what git cherry-pick does?
16. What's the purpose of .gitignore file?
17. How do you remove a file from Git without deleting it from your local filesystem?
18. Can you explain what git rebase does and when you might use it?
19. What's the difference between git merge and git rebase?
20. How would you recover a deleted branch that hasn't been pushed?
21. Can you explain what git bisect does and how you'd use it?
22. What's a detached HEAD state and how might you end up in it?
23. How do you create and delete a tag in Git?
24. Can you explain the difference between git reset and git revert?
25. What's the purpose of git reflog?
26. How would you modify the author of a previous commit?
27. Can you explain what git blame does and when you might use it?
28. How would you collaborate with a team member on the same feature branch?
29. Can you explain what a 'git hook' is and provide an example of how you might use one?
30. How would you handle a situation where you need to undo changes in a file that's already been staged?
31. What's your strategy for keeping a forked repository up to date with the original repository?
32. How would you use Git to find out who last modified a particular line of code?
33. Explain the concept of 'git worktree' and when you might use it.
34. How would you clean up local branches that have been merged into the main branch?
35. What's the difference between 'git rm' and just deleting a file from the filesystem?
36. How would you use Git to find a commit that introduced a bug?
37. Explain the concept of 'git submodules' and when you might use them.
38. How would you resolve a situation where a teammate's changes conflict with yours after a rebase?
39. Can you explain how to use Git hooks for automating tasks in your workflow?
40. What strategies would you employ to manage a large repository with numerous collaborators?
41. How do you handle large binary files in Git, and what tools or techniques would you recommend?
42. Can you describe the process of creating and managing a Git submodule in a project?
43. What is the significance of the 'git gc' command, and when would you use it?
44. How would you implement a workflow that ensures code quality in a shared repository?
45. Can you explain how to use git merge with specific strategies, like 'ours' or 'theirs'?
46. What techniques do you use to keep your Git history clean and meaningful?
47. How would you handle a scenario where you need to apply a series of commits from one branch to another without merging?
48. How do you decide when to create a new branch in Git?
49. What are the key differences between a feature branch and a release branch?
50. How do you handle merging a feature branch back into the main branch?
51. Can you explain the concept of a hotfix branch and when you would use it?
52. What are the benefits and drawbacks of using a long-lived branch strategy?
53. How do you handle branch naming conventions in your projects?
54. What is your approach to keeping a feature branch up-to-date with the main branch?
55. How do you manage multiple remote repositories in a single project?
56. Can you describe a situation where you encountered a complex merge conflict and how you resolved it?
57. How do you prevent merge conflicts from occurring in the first place?
58. What tools or techniques do you use to understand the nature of a merge conflict?
59. How do you approach merging a long-lived feature branch that has diverged significantly from the main branch?
60. What's your approach to resolving a merge conflict when you're unsure about the intent behind certain changes?
61. How do you ensure that resolving a merge conflict doesn't introduce new bugs or break existing functionality?
62. How would you handle a situation where you accidentally committed sensitive information to a public repository?
63. Describe a time when you had to integrate changes from multiple feature branches into the main branch. How did you approach this?
64. You've just realized that the feature you've been working on for the past week conflicts with the project's new direction. How would you handle this situation using Git?
65. How would you use Git to implement a hotfix for a critical bug in production while a major feature is still in development?