50 Git interview Q&A [Basics to Advanced topics]

1. What is Git?

Git is a distributed version control system that tracks changes to files and allows multiple people to collaborate on projects.

2. What is the difference between Git and GitHub?

Git is the version control system software, while GitHub is a hosting service for Git repositories.

3. Explain the concept of branching in Git.

Branching in Git allows developers to diverge from the main line of development and work on different features or fixes in isolation.

4. What is a commit in Git?

A commit in Git is a snapshot of changes made to files in a repository. It represents a specific point in the project's history.

5. How do you create a new branch in Git?

To create a new branch, you can use the command $ git branch <branch_name>.

6. How do you merge branches in Git?

You can merge branches in Git using the command $ git merge <branch_name>.

7. Explain the difference between merge and rebase in Git.

Merging combines changes from different branches, while rebasing moves the entire branch to a new base commit.

8. What is a pull request in Git?

A pull request is a feature in Git that allows developers to propose changes to a repository and request that they be reviewed and merged into the main branch.

9. How do you resolve merge conflicts in Git?

Merge conflicts can be resolved by manually editing the conflicting files to reconcile the differences and then committing the changes.

10. What is the difference between a fetch and a pull in Git?

A fetch retrieves changes from a remote repository but does not merge them into the local branch, while a pull fetches changes and merges them into the current branch.

11. How do you revert a commit in Git?

You can revert a commit in Git using the command $ git revert <commit_id>.

12. Explain the Git workflow you follow in your projects.

This will depend on the individual's workflow, but common workflows include feature branching, Git flow, and GitHub flow.

13. What are Git hooks?

Git hooks are scripts that run automatically before or after certain Git events, such as committing or merging.

14. How do you squash commits in Git?

You can squash commits in Git using the interactive rebase feature with the command

$ git rebase -i HEAD~<number_of_commits>.

15. Explain the difference between git fetch and git pull.

git fetch retrieves changes from a remote repository but does not merge them into the current branch, while git pull fetches changes and merges them into the current branch.

16. What is a detached HEAD in Git?

A detached HEAD state occurs when you check out a commit directly instead of a branch. Any commits made in this state will not be attached to any branch.

17. How do you rename a Git branch?

You can rename a Git branch using the command

$ git branch -m <old_branch_name> <new_branch_name>.

18. What is Git rebase and when would you use it?

Git rebase is a command used to reapply commits on top of another base tip. It is often used to maintain a linear history and incorporate changes from one branch into another.

19. How do you delete a Git branch?

You can delete a Git branch using the command $ git branch -d <branch_name>.

20. Explain what Git bisect is used for.

Git bisect is a command used for binary search through the commit history to find the commit that introduced a bug.

21. What is Git cherry-pick and when would you use it?

Git cherry-pick is a command used to apply a specific commit from one branch to another. It is often used to pick individual commits for merging into another branch.

22. What is Git submodule?

Git submodule is a way to include one Git repository as a subdirectory of another Git repository.

23. Explain what Git rebase --onto is used for.

Git rebase --onto is used to reapply commits from one branch onto another branch starting from a specific commit.

24. How do you view the commit history in Git?

You can view the commit history in Git using the command git log.

25. Explain the difference between git reset and git revert.

git reset is used to move the HEAD and current branch pointer to a specific commit, potentially discarding changes, while git revert creates a new commit that undoes the changes made by a previous commit.

26. What is Git stash and how do you use it?

Git stash is a command used to temporarily store changes that are not ready to be committed. You can use git stash save to stash changes and git stash apply to reapply them later.

27. Explain the difference between Git and SVN.

Git is a distributed version control system, while SVN (Subversion) is a centralized version control system. Git allows for more flexible branching and merging, while SVN requires a central server for collaboration.

28. What is the purpose of the .gitignore file?

The .gitignore file is used to specify intentionally untracked files that Git should ignore. This can include files generated by the build process, temporary files, and configuration files.

29. How do you undo the last commit in Git?

You can undo the last commit in Git using the command git reset HEAD^.

30. Explain what Git reflog is used for.

Git reflog is a command used to record the history of Git commands executed in the repository. It can be used to recover lost commits or undo changes.

31. What is Git blame and how do you use it?

Git blame is a command used to show the revision and author of each line in a file. You can use git blame <file> to display line-by-line changes along with the commit and author information.

32. How do you set up SSH keys for Git authentication?

You can set up SSH keys for Git authentication by generating a new SSH key using the ssh-keygen command and adding the public key to your Git hosting service, such as GitHub or GitLab.

33. What is Git cherry-pick and when would you use it?

Git cherry-pick is a command used to apply a specific commit from one branch to another. It is often used to pick individual commits for merging into another branch.

34. What is a Git tag and how do you create one?

A Git tag is a way to mark a specific commit as being important or significant. You can create a Git tag using the command $ git tag <tag_name>.

35. What is Git rebase interactive and how do you use it?

Git rebase interactive is a command used to interactively reapply commits on top of another base tip. You can use it to squash, edit, or reorder commits before applying them.

36. What is the difference between Git rebase and Git merge?

Git rebase moves the entire branch to a new base commit and re-applies commits on top of it, resulting in a linear history. Git merge combines changes from different branches, creating a merge commit and preserving the branch history.

37. Explain what Git hooks are and give examples of how they can be used.

Git hooks are scripts that run automatically before or after certain Git events, such as committing or merging. Examples include pre-commit hooks to run code linting or formatting checks, and post-commit hooks to send notifications or trigger automated builds.

38. How do you create and apply a Git patch?

You can create a Git patch using the git format-patch command and apply it using the git apply or git am command.

39. What is the purpose of Git bisect and how do you use it?

Git bisect is a command used to find the commit that introduced a bug by performing a binary search through the commit history. You can use git bisect start, git bisect bad, and git bisect good to perform the binary search and locate the faulty commit.

40. How do you set up a Git repository?

You can set up a Git repository using the git init command to initialize an empty repository, or git clone to clone an existing repository from a remote URL.

41. What is Git submodules and how do you use them?

Git submodules are repositories nested inside a parent repository. They allow you to include external repositories as dependencies in your project. You can add a submodule using the git submodule add command and update submodules using git submodule update.

42. How do you revert a commit in Git?

You can revert a commit in Git using the $ git revert <commit_id> command. This creates a new commit that undoes the changes introduced by the specified commit.

43. What is Git squash and how do you use it?

Git squash is a technique used to combine multiple commits into a single commit. You can use interactive rebasing with git rebase -i to squash commits together before merging them into the main branch.

44. How do you rename a Git remote?

You can rename a Git remote using the git remote rename command followed by the old and new remote names.

45. What is Git reflog and how do you use it?

Git reflog is a log of all Git operations performed in the repository, including commits, merges, and resets. You can use it to recover lost commits or undo changes by referencing the commit IDs in the log.

46. How do you set up Git to use an external diff tool?

You can set up Git to use an external diff tool by configuring the diff.tool and difftool.<tool>.path settings in your Git configuration file.

47. What is the purpose of Git sparse checkout?

Git sparse checkout is a feature that allows you to check out only specific files or directories from a repository, rather than the entire repository. It can be useful for reducing the size of your working directory or focusing on specific parts of the project.

48. What is Git rerere and how do you use it?

Git rerere (reuse recorded resolution) is a feature that allows Git to remember how merge conflicts were resolved and automatically apply the same resolution to future conflicts. You can enable rerere using the $ git config --global rerere.enabled true command.

49. How do you set up Git to use a proxy?

You can configure Git to use a proxy server for network operations by setting the http.proxy and https.proxy settings in your Git configuration file.

50. What is Git LFS and how do you use it?

Git LFS (Large File Storage) is an extension for Git that allows you to store large files outside of the Git repository and replace them with pointers. You can use it to manage large binary files more efficiently and reduce the size of your Git repository.