DevOps SHACK

DevOps Shack

Git Interview Questions

Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps

1. What is Git?

- **Answer:** Git is a distributed version control system used to track changes in files and coordinate work on those files among multiple people.

2. What is the difference between Git and SVN?

- **Answer:** Git is a distributed version control system, while SVN (Subversion) is a centralized version control system. In Git, every user has a local copy of the repository, allowing them to work offline and commit changes locally.

3. What is a repository in Git?

- **Answer:** A repository in Git is a data structure that stores metadata and objects representing a project. It contains all versions of the project's files and directories.

4. How do you create a new Git repository?

- **Answer:** You can create a new Git repository using the command `git init` followed by the project directory.

5. What is a commit in Git?

- **Answer:** A commit is a snapshot of changes made to a repository at a specific point in time. It includes a unique identifier, a commit message, and a pointer to the parent commit(s).

6. How do you commit changes in Git?

- **Answer:** To commit changes in Git, you use the command `git commit -m "commit message"`. This will record the changes in the local repository.

7. What is the purpose of the staging area (index) in Git?

- **Answer:** The staging area is where you prepare and review changes before committing them to the repository. It allows you to selectively include or exclude specific changes from a commit.

8. How do you add files to the staging area in Git?

- **Answer:** You can add files to the staging area using the command `git add filename` or `git add .` to add all modified files.

9. What is a branch in Git?

- **Answer:** A branch in Git is a separate line of development that allows you to work on features or bug fixes without affecting the main codebase.

10. How do you create a new branch in Git? - **Answer:** You can create a new branch using the command `git checkout -b branch_name`.
11. How do you switch between branches in Git? - **Answer:** You can switch between branches using the command `git checkout branch_name`.
12. What is a merge in Git? - **Answer:** A merge in Git combines the changes from one branch into another branch. It is used to integrate feature branches back into the main branch.

13. How do you merge branches in Git? - **Answer:** To merge branches in Git, you first switch to the target branch (`git checkout target_branch`) and then use the command `git merge source_branch`.
14. What is a conflict in Git? - **Answer:** A conflict in Git occurs when two branches have made changes to the same part of a file and Git is unable to automatically resolve the differences.

15. How do you resolve a merge conflict in Git? - **Answer:** To resolve a merge conflict, you need to manually edit the conflicting file(s), choose which changes to keep, and then commit the changes.

16. What is a remote repository in Git? - **Answer:** A remote repository in Git is a version of a project hosted on a server that allows collaboration between multiple contributors.

17. How do you clone a Git repository? - **Answer:** You can clone a Git repository using the command `git clone repository_url`.
18. What is the difference between `git pull` and `git fetch`? - **Answer:** `git pull` fetches and merges changes from a remote repository to the current branch,

while `git fetch` only downloads changes from the remote repository without merging them.
19. What is a tag in Git? - **Answer:** A tag in Git is a reference that points to a specific commit. It is used to mark specific points in history, such as release versions.

20. How do you create a tag in Git? - **Answer:** You can create a tag in Git using the command `git tag tag_name`.
21. How do you push a tag to a remote repository? - **Answer:** You can push a tag to a remote repository using the command `git push origin tag_name`.
22. What is a Git hook? - **Answer:** A Git hook is a script that Git executes before or after certain events, such as committing changes or merging branches.

23. What is Git bisect? - **Answer:** Git bisect is a command used for binary search of bugs. It helps you find the commit that introduced a bug.

24. What is Git rebase? - **Answer:** Git rebase is a command used to integrate changes from one branch into another by moving or combining commits.

25. What is the difference between a merge and a rebase in Git? - **Answer:** Merging creates a new commit that combines the changes from two branches, while rebasing moves or combines a sequence of commits to a new base commit.

26. How do you undo the last Git commit? - **Answer:** You can undo the last Git commit using the command `git reset HEAD~1`.
27. What is Git cherry-pick? - **Answer:** Git cherry-pick is a command used to apply a specific commit from one branch to another.

28. What is Git stash? - **Answer:** Git stash is a command used to save changes that are not ready to be committed, allowing you to switch branches without losing your work.

29. How do you list all the branches in a Git repository? - **Answer:** You can list all branches in a Git repository using the command `git branch`.
30. What is the purpose of the `.gitignore` file? - **Answer:** The `.gitignore` file is used to specify which files and directories Git should ignore, meaning they will not be tracked or committed to the repository.
31. How do you remove a file from Git without deleting it? - **Answer:** You can remove a file from Git without deleting it using the command `git rm --cached filename`.
32. How do you rename a file in Git? - **Answer:** You can rename a file in Git using the command `git mv old_filename new_filename`.
33. What is a Git submodule? - **Answer:** A Git submodule is a separate Git repository embedded within another Git repository. It allows you to include other projects as dependencies.

34. How do you update a Git submodule? - **Answer:** You can update a Git submodule using the command `git submodule update --remote`.
35. What is Git flow? - **Answer:** Git flow is a branching model that defines a consistent way to organize branches and merges in a Git repository.

36. What is GitLab/GitHub Actions? - **Answer:** GitLab/GitHub Actions are Continuous Integration/Continuous Deployment (CI/CD) services provided by GitLab/GitHub for

automating tasks like testing and deploying code.

37. How do you revert a commit in Git? - **Answer:** You can revert a commit in Git using the command `git revert commit_sha`.
38. What is the difference between `git rebase` and `git merge`? - **Answer:** `git rebase` integrates changes from one branch onto another by moving or combining commits, resulting in a linear history. `git merge` creates a new commit that combines changes from two branches, resulting in a branch-like history.
39. How do you squash multiple commits into one? - **Answer:** You can squash multiple commits into one using the interactive rebase command: `git rebase -i HEAD~n`, where `n` is the number of commits you want to squash.
40. How do you amend the last commit message in Git? - **Answer:** You can amend the last commit message using the command `git commit --amend`.
41. What is Git LFS (Large File Storage)? - **Answer:** Git LFS is an extension to Git that allows the versioning of large files by storing them on a remote server, while keeping lightweight references in the repository.

42. How do you configure Git user information? - **Answer:** You can configure Git user information using the commands: `git config --global user.name "Your Name" git config --global user.email "your.email@example.com"`
43. How do you show the commit history in Git? - **Answer:** You can show the commit history using the command `git log`.
44. What is Git Cherry-Pick and when would you use it? - **Answer:** Git Cherry-Pick is a command used to apply a specific commit from one branch to another. It is useful when you want to apply a specific fix or feature to a different branch without merging the entire branch.

45. How do you create and apply patches in Git? - **Answer:** To create a patch, use the command `git format-patch -1 <commit>`. To apply a patch, use the command `git am <patch>`.
46. How do you view the changes between two Git commits? - **Answer:** You can view the changes between two Git commits using the command `git diff commit1 commit2`.
47. What is Git reflog? - **Answer:** Git reflog is a log of all reference updates in a repository. It allows you to recover lost commits or branches.

48. What is Git blame? - **Answer:** Git blame is a command that shows information about who last modified each line of a file, and in which commit.

49. How do you delete a branch in Git? - **Answer:** You can delete a branch in Git using the command `git branch -d branch_name`.
50. How do you force push in Git? - **Answer:** You can force push in Git using the command `git push -f origin branch_name`.
Remember, these are just interview questions, and in a real-world scenario, understanding the concepts and being able to apply them is crucial.

50 Advanced Git Questions with answers

50 advanced Git interview questions with detailed answers:

1. **What is Git and how does it differ from other version control systems?**
   - **Answer:** Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Unlike centralized version control systems, Git allows every developer to have a full copy of the repository, enabling them to work offline.
2. **Explain the difference between Git and GitHub.**
   - **Answer:** Git is a version control system, while GitHub is a web-based platform for hosting and collaborating on Git repositories. GitHub provides additional features such as pull requests, issue tracking, and collaborative tools.
3. **What is a Git repository?**
   - **Answer:** A Git repository is a collection of files and their history, stored in a `.git` directory at the root of the project. It contains the complete history of the project, along with metadata and configuration settings.
4. **What is a commit in Git?**
   - **Answer:** A commit is a snapshot of changes made to the repository at a specific point in time. It includes a unique identifier (SHA-1 hash), author information, timestamp, and a reference to the previous commit.
5. **Explain the Git staging area.**
   - **Answer:** The staging area (or index) is a crucial part of Git that acts as a middle-ground between the working directory and the repository. Changes are first staged using `git add` before being committed to the repository.
6. **What is a branch in Git?**

- **Answer:** A branch in Git is a lightweight movable pointer to a commit. It allows developers to work on separate features or fixes without affecting the main codebase. Branches can be easily created, merged, or deleted.

7. **How do you create a new branch in Git?**
   - **Answer:** Use the command `git branch <branch_name>` to create a new branch. To switch to the new branch, use `git checkout <branch_name>` or `git switch <branch_name>`.
8. **Explain the concept of Git merging.**
   - **Answer:** Merging in Git combines changes from different branches. It can be done using `git merge` to integrate changes from a source branch into a target branch.
9. **What is a Git merge conflict, and how can it be resolved?**
   - **Answer:** A merge conflict occurs when Git cannot automatically reconcile changes between branches. To resolve conflicts, manually edit the conflicting files, mark them as resolved using `git add`, and complete the merge with `git merge --continue`.
10. **Describe the rebase operation in Git.**
    - **Answer:** Git rebase is the process of moving or combining a sequence of commits to a new base commit. It can be used to maintain a cleaner and linear project history.
11. **What is a detached HEAD state in Git?**
    - **Answer:** A detached HEAD state occurs when the HEAD points directly to a specific commit instead of a branch. It is a common state during operations like checking out a specific commit.
12. **Explain the difference between git pull and git fetch.**
    - **Answer:** `git pull` fetches changes from a remote repository and automatically merges them into the current branch. `git fetch` only retrieves changes but does not automatically merge them, allowing the user to review and decide how to integrate the changes.
13. **How do you revert a commit in Git?**
    - **Answer:** To revert a commit, use `git revert <commit_hash>` to create a new commit that undoes the changes introduced by the specified commit.
14. **What is a Git submodule?**

- **Answer:** A Git submodule is a repository embedded within another repository. It allows you to include external repositories as a subdirectory of your project.

15. **Explain the purpose of Git hooks.**
    - **Answer:** Git hooks are scripts that can be triggered at specific points in the Git workflow. They enable custom actions such as pre-commit validation or post-receive deployment.
16. **How do you squash commits in Git?**
    - **Answer:** Use `git rebase -i HEAD~n` (replace n with the number of commits) and mark commits as "squash" or "s" to combine them into a single commit.
17. **Describe the .gitignore file.**
    - **Answer:** The .gitignore file specifies files and directories that should be ignored by Git. It helps exclude unnecessary files (e.g., build artifacts, logs) from version control.
18. **What is Git bisect, and how is it used?**
    - **Answer:** Git bisect is a binary search tool for finding the commit that introduced a bug. It involves marking known good and bad commits, allowing Git to narrow down the faulty commit.
19. **Explain the purpose of Git cherry-pick.**
    - **Answer:** Git cherry-pick is used to apply a specific commit from one branch to another. It allows you to selectively choose and apply changes.
20. **What is Git reflog, and how is it useful?**
    - **Answer:** Git reflog is a log of reference updates in the repository. It helps recover lost commits or branches and provides a history of recent operations.
21. **How do you sign commits in Git?**
    - **Answer:** Sign commits using `git commit -S` to add a GPG signature. Configure Git with your GPG key, and use `git log --show-signature` to verify signatures.
22. **What is Git cherry and when would you use it?**
    - **Answer:** Git cherry is used to find commits that are in one branch but not in another. It is helpful when you want to identify changes that have not been merged between branches.

23. **Describe Git rebase --onto.**
    - **Answer:** `git rebase --onto` is used to reapply commits onto a new base. It allows you to move a branch or a series of commits from one branch to another.
24. **Explain the purpose of Git subcommands like git clean and git reset.**
    - **Answer:** `git clean` removes untracked files, and `git reset` allows you to reset the current branch to a specific commit or undo changes.
25. **What is the purpose of the git worktree command?**
    - **Answer:** The `git worktree` command allows you to maintain multiple working directories (worktrees) associated with a single Git repository. Each worktree can have its own branch and changes.
26. **Describe Git shallow cloning.**
    - **Answer:** Shallow cloning involves cloning a Git repository with a limited history. It can be done using `--depth` option with `git clone` to specify the number of commits to retrieve.
27. **How do you configure Git to use an HTTP proxy?**
    - **Answer:** Set the HTTP proxy for Git using the `http.proxy` configuration. For example, use `git config --global http.proxy http://proxy.example.com:8080`.
28. **Explain the purpose of Git LFS (Large File Storage).**
    - **Answer:** Git LFS is an extension to Git that allows for the storage of large files outside the Git repository, reducing the repository size. It is useful for managing binary

files and large assets.

29. **What is the Git "refspec"?**
    - **Answer:** A refspec in Git is a string that defines the mapping between remote and local branches during fetch or push operations. It specifies the source and destination references.
30. **Describe Git worktrees and their advantages.**
    - **Answer:** Git worktrees allow you to work on multiple branches simultaneously without switching back and forth. They provide a clean and isolated environment for each branch.

31. **How do you handle confidential information, such as API keys, in a Git repository?**
    - **Answer:** Confidential information should be stored in environment variables or configuration files outside the repository. Use tools like `.gitignore` and `git-crypt` to avoid accidentally committing sensitive data.
32. **What is Git rebase -i used for, and how does it work?**
    - **Answer:** `git rebase -i` opens an interactive rebase session, allowing you to modify, reorder, or squash commits. It provides a text editor interface where you can choose actions for each commit.
33. **Explain the purpose of Git sparse-checkout.**
    - **Answer:** Git sparse-checkout is used to limit the working directory to specific directories or files, enabling users to check out only the necessary parts of a repository.
34. **How do you recover from a detached HEAD state in Git?**
    - **Answer:** To recover from a detached HEAD state, create a new branch using `git checkout -b new_branch_name` or switch to an existing branch using `git checkout branch_name`.
35. **What is the Git "rerere" (Reuse Recorded Resolution) feature?**
    - **Answer:** The `rerere` feature records previous conflict resolutions, allowing Git to automatically apply them to future conflicts. It stands for "Reuse Recorded Resolution."
36. **Explain Git's "worktree add" command.**
    - **Answer:** The `git worktree add` command creates a new linked working directory associated with the current repository. It enables you to work on multiple branches simultaneously.
37. **How do you amend the last commit message in Git?**
    - **Answer:** Use `git commit --amend` to modify the last commit message. This opens the default text editor for you to make changes.
38. **Describe the purpose of Git "revert" vs. "reset."**
    - **Answer:** `git revert` creates a new commit that undoes changes from a specific commit. `git reset` is used to reset the current branch to a specified commit, potentially discarding changes.

39. **What is the Git "worktree move" command?**
    - **Answer:** The `git worktree move` command allows you to move or rename an existing linked working directory created with `git worktree add`.
40. **Explain the concept of Git "subversion" (svn) integration.**
    - **Answer:** Git can integrate with Subversion repositories using `git svn` commands, providing a bridge between Git and Subversion version control systems.
41. **How do you use Git to show the changes introduced by a specific commit?**
    - **Answer:** Use `git show <commit_hash>` to display the changes introduced by a specific commit, including the commit message and diff.
42. **What is Git "cherry-pick range"?**
    - **Answer:** Git cherry-pick range involves picking a range of commits from one branch and applying them to another branch. It can be achieved using `git cherry-pick <start_commit>^..<end_commit>`.
43. **Explain the purpose of the "git bisect" command.**
    - **Answer:** The `git bisect` command helps in finding a specific commit that introduced a bug by using a binary search algorithm. It requires marking known good and bad commits to narrow down the problematic commit.
44. **How does Git handle line endings, and what are autocrlf and core.autocrlf?**
    - **Answer:** Git handles line endings based on platform conventions. The `core.autocrlf` setting controls automatic conversion between Unix and Windows line endings. `autocrlf` is used for automatic line ending conversion.
45. **What is the purpose of "git blame"?**
    - **Answer:** `git blame` shows the commit and author information for each line of a file, helping to identify who made changes and when.
46. **How can you rename a Git branch?**
    - **Answer:** To rename a Git branch, use `git branch -m <new_branch_name>` to rename the current branch, or `git branch -m <old_branch_name> <new_branch_name>` to rename a different branch.
47. **Explain the difference between "merge" and "rebase" workflows.**

- **Answer:** The merge workflow combines changes from different branches, creating a new merge commit. The rebase workflow moves or combines a sequence of commits onto a new base, creating a linear history.

48. **What is Git "filter-branch" used for?**
    - **Answer:** `git filter-branch` is used to rewrite Git repository history by applying filters. It can be used for tasks like renaming files, removing sensitive data, or restructuring the repository.

49. **Describe Git "force push" and its implications.**
    - **Answer:** A force push (`git push --force`) is used to overwrite remote branch history with local changes. It should be used with caution, as it can lead to loss of commits and disrupt collaboration.

50. **How do you squash multiple commits into a single commit during an interactive rebase?**
    - **Answer:** During an interactive rebase, mark commits as "squash" or "s" to combine them into a single commit. The rebase process will prompt you to edit the commit message before finalizing the changes.

These questions cover various advanced Git topics, and the answers provide detailed explanations. Make sure to understand these concepts thoroughly for a successful Git interview.