DEVOPS INSIDERS

GIT INTERVIEW QUESTIONS

What is Git rebase, and when would you use it in a collaborative development environment?

Git rebase is a command used to reapply commits from one branch onto another, typically to integrate changes from one branch into another cleanly. It is commonly used to maintain a linear commit history, resolve merge conflicts, or incorporate changes from a feature branch into the main branch.

Describe the typical workflow for adding, committing, and pushing changes to a Git repository.

The workflow involves adding modified files to the staging area using git add, committing staged changes to the local repository using git commit, and pushing committed changes to a remote repository using git push, ensuring synchronization and collaboration in distributed development environments.

How do you differentiate between Azure Repos and GitHub ?

Azure Repos is a Git repository hosting service provided by Microsoft Azure, primarily integrated with Azure DevOps for seamless CI/CD pipelines and collaboration. On the other hand, GitHub is a widely-used Git repository hosting platform, known for its extensive community, open-source projects, and collaboration features beyond version control.

Can you describe the branching strategy used in our current project?

In our current project, we follow a Git branching strategy based on GitFlow, with main branches such as `master` and `develop`, feature branches for new developments, release branches for stable releases, and hotfix branches for addressing critical issues in production.

What is a pull request, and how does it facilitate collaboration in Git-based projects?

A pull request (PR) is a mechanism in Git for proposing changes and initiating code review before merging them into the main codebase. It allows team members to review, discuss, and provide feedback on proposed changes, ensuring code quality, consistency, and alignment with project goals.

Explain the concept of a branch in Git and its significance in version control.

In Git, a branch is a parallel line of development that diverges from the main codebase, allowing developers to work on features, fixes, or experiments independently without affecting the main codebase. Branches facilitate collaboration, experimentation, and isolation of changes before merging them back into the main branch

What is the relationship between `pull`, `fetch`, and `merge` commands in Git?

In Git, the `pull` command combines the actions of `fetch` and `merge`. It retrieves changes from the remote repository using `fetch` and then merges them into the current branch using `merge`, ensuring synchronization of local and remote repositories.

How does the `merge` operation work in Git, and when would you use it?

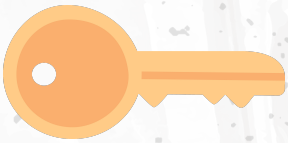
The `merge` operation in Git combines changes from different branches into the current branch, integrating new features, fixes, or updates. It is commonly used to incorporate changes from feature branches into the main branch or to resolve divergent changes between branches.

What is a merge conflict in Git, and how do you resolve it?

A merge conflict occurs when Git is unable to automatically merge changes from different branches due to conflicting modifications to the same file or lines of code. To resolve a merge conflict, developers need to manually reconcile the differences, choose the desired changes, and commit the resolution.

Why is the commit history important in Git, and how do you review it?

The commit history in Git provides a chronological record of changes made to the repository, including who made the changes, when they were made, and the nature of the changes. Reviewing the commit history helps track project progress, understand code evolution, and identify contributors and their contributions.



How do you switch to a different branch in Git using the `checkout` command?

To switch to a different branch in Git, you can use the `checkout` command followed by the name of the target branch. This command updates the working directory and the HEAD pointer to the specified branch, allowing you to continue work on that branch.

What is the process for creating a new branch in Git, and why would you create a branch?

To create a new branch in Git, you can use the `git checkout -b` command followed by the desired branch name. Branches are created to work on new features, bug fixes, or experiments without directly modifying the main codebase, providing isolation and versioning for changes.

How do you delete a branch in Git, and when would you delete a branch?

To delete a branch in Git, you can use the `git branch -d` command followed by the name of the branch to be deleted. Branches are typically deleted after their changes have been merged into the main codebase, or if they are no longer needed for ongoing development or maintenance.

What does the `git reset` command do, and how is it used in Git?

The `git reset` command is used to reset the current branch to a specific state, undoing changes or resetting the staging area. It is commonly used to undo local commits, unstage changes, or move the HEAD pointer to a different commit in the commit history.

What is the purpose of the `git diff` command in Git, and how is it used?

The `git diff` command compares changes between different versions of files in the repository, showing the differences line by line. It is used to review changes before committing, identify modifications between branches, or track file changes over time.

What is the `git stash` command used for in Git, and when would you use it?

The `git stash` command is used to temporarily store changes in the working directory without committing them, allowing you to switch branches or work on other tasks. It is useful when you need to switch contexts quickly or save work in progress temporarily.

What is a Git tag, and how is it used to mark significant points in a repository's history?

In Git, a tag is a reference pointing to a specific commit in the repository's history, marking it as a significant point such as a release or milestone. Tags are immutable and provide a stable reference for versioning, release management, and historical tracking of software releases.

How do you manage releases in Git, and what role do tags play in the release process?

Git releases are managed by creating tags that mark specific commits as release points. These tags serve as stable references for versioning and release management, enabling teams to track and distribute software releases efficiently.