GIT Interview Questions

What is Git?
Answer: Git is a distributed version control system for tracking changes in source code during software development. It is designed to handle projects of all sizes with speed and efficiency.

What is the difference between Git and GitHub?
Answer: Git is a version control system that lets you manage and keep track of your source code history, whereas GitHub is a cloud-based hosting service that lets you manage Git repositories.

Can you explain what a commit is in Git?
Answer: A commit in Git is like a snapshot of your entire repository at one point in time. It captures the current state of the project's source code.

What is a branch in Git? And can you explain what 'master' branch is?
Answer: A branch in Git allows you to diverge from the main line of development and continue to work independently without affecting the main line. The 'master' branch is the default development branch.

How does Git handle conflicts?
Answer: Git handles conflicts by marking the files that have conflicting changes. It's then up to the developer to manually resolve the conflicts.

What is a merge in Git?
Answer: A merge in Git is an operation that takes the changes from one branch (source) and integrates them into another (target).

Describe the difference between a fast-forward merge and a three-way merge.
Answer: A fast-forward merge happens when the target branch's head is a direct ancestor of the source branch's head. A three-way merge is needed when the branches have diverged, creating a new commit that ties together the histories of both branches.

What is a rebase in Git?

Answer: Rebase is a way to move the base of a branch to a different commit, integrating changes from one branch onto another by replaying commits.

What are the common use-cases for using 'git stash'?
Answer: `git stash` is useful when you want to quickly switch context and work on something else without committing work-in-progress to a branch.

Explain the difference between 'git merge' and 'git rebase'.
Answer: `git merge` integrates the histories of the two branches together, whereas `git rebase` rewrites the commit history to apply changes from one branch onto another in a linear sequence.

What is a 'remote' in Git?
Answer: A 'remote' in Git is a common repository that all team members use to exchange their changes.

How can you undo a commit in Git?
Answer: You can undo a commit using commands like `git revert` to create a new commit that undoes the changes or `git reset` to move the branch pointer to a previous commit.

Explain 'git fetch' vs 'git pull'.
Answer: `git fetch` downloads all the changes in the remote repository that are not present in your current branch without integrating them. `git pull` fetches and immediately attempts to merge these changes into the current branch.

In Git, how do you create a new repository?
Answer: You create a new repository by running `git init`. This initializes a new Git repository in your current directory.

What is 'git clone', and how is it different from 'git init'?
Answer: `git clone` is used to create a copy of an existing repository into a new directory. `git init` simply initializes a new Git repository in an empty or existing directory without copying any existing data.

How can you make existing Git branches track a remote branch?

Answer: You can use `git branch --set-upstream-to` to set a local branch to track a remote branch.

What is a 'pull request'?
Answer: A pull request is a way to propose changes to a repository on a hosting service such as GitHub. It is a request to another developer to pull a branch from your repository into their repository.

How do you create a tag in Git?
Answer: You can create a tag in Git using the command `git tag`. Tags are typically used to mark release points or other important points in a repository's history.

What command would you use to review the commit history?
Answer: `git log` is used to review and read the chronological commit history of the current branch.

How can you revert a Git repository to a previous commit?
Answer: To revert to a previous commit, check it out with `git checkout <commit_id>`. If you want to reset the current branch to that commit, use `git reset --hard <commit_id>` (destructive, as it discards all changes). For a safe option that preserves changes, use `git revert`.