**Git and GitHub Test Questions**

**1. What is Git and why is it used?**

**Git** is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is used for tracking changes in source code during software development, enabling multiple developers to work together. Git allows users to keep a history of changes, revert to previous versions, and collaborate with others efficiently.

**2. Explain the difference between Git pull and Git fetch.**

**Git pull** is a command that updates the local repository with changes from the remote repository and merges them into the current branch. It is essentially a combination of git fetch followed by git merge.

**Git fetch** downloads changes from the remote repository into local branches but does not merge them. It allows users to see the changes in the remote repository without altering their current working directory.

**3. How do you revert a commit in Git?**

To revert a commit in Git, you can use the git revert command followed by the commit hash. This command creates a new commit that undoes the changes introduced by the specified commit.

git revert <commit\_hash>

Alternatively, if you want to undo the last commit, you can use:

git revert HEAD

**4. Describe the Git staging area.**

The **Git staging area** is an intermediate area where Git holds changes that have been added using the git add command before they are committed. It allows users to prepare and review changes before committing them to the project history.

**5. What is a merge conflict, and how can it be resolved?**

A **merge conflict** occurs when Git is unable to automatically resolve differences between two commits. This typically happens when changes are made to the same line of code in different branches.

To resolve a merge conflict:

1. Identify the files with conflicts.
2. Open the files and look for conflict markers (<<<<<<, ======, >>>>>>).
3. Edit the file to resolve the conflict manually.
4. Mark the conflict as resolved with git add <filename>.
5. Commit the changes using git commit.

**6. How does Git branching contribute to collaboration?**

**Git branching** allows multiple developers to work on different features, bug fixes, or experiments simultaneously without interfering with the main codebase. Each branch can be developed, tested, and merged independently, facilitating parallel development and reducing integration issues.

**7. What is the purpose of Git rebase?**

**Git rebase** is a command that allows you to integrate changes from one branch into another by moving or combining a sequence of commits to a new base commit. It is used to create a linear project history, making it easier to understand and maintain. Rebase can be used to keep a feature branch up to date with the main branch by replaying commits from the feature branch on top of the main branch.

**8. Explain the difference between Git clone and Git fork.**

**Git clone** creates a local copy of a remote repository, allowing you to work on it locally. Changes made in the local repository can be pushed back to the remote repository if you have the necessary permissions.

**Git fork** creates a personal copy of someone else's repository on your GitHub account. You can make changes to your forked repository without affecting the original repository. Changes can be submitted to the original repository through pull requests.

**9. How do you delete a branch in Git?**

To delete a branch in Git, you can use the following command:

* Locally: git branch -d <branch\_name>
* Remotely: git push origin --delete <branch\_name>

**10. What is a Git hook, and how can it be used?**

**Git hooks** are scripts that run automatically on specific Git events, such as committing changes or merging branches. They can be used to enforce coding standards, run tests, or automate tasks. Git hooks are stored in the .git/hooks directory of a repository.