**Git Interview Questions and Answers Part 1**

**1. What is Git, and how does it differ from other version control systems?**

**Answer:**  
Git is a distributed version control system (VCS) that allows multiple developers to work on the same project without overwriting each other’s work. Unlike centralized version control systems, Git enables each developer to have a local copy of the repository, including the full history of commits. This distributed nature allows developers to work offline, commit changes, and later synchronize with the remote repository.

* **Git**: Distributed; each user has a full copy of the repository.
* **SVN/CVS**: Centralized; changes are stored in a central repository.

**2. What is the difference between a fork and a clone in Git?**

**Answer:**

* **Fork**: A fork is a copy of a repository on your own Git hosting service account (e.g., GitHub). It allows you to make changes without affecting the original project, typically used in open-source projects to propose changes via pull requests.
* **Clone**: A clone is a local copy of a remote repository that you create using the git clone command. You work on the clone directly, and changes can be pushed to the original repository if you have the necessary permissions.

**3. What is the purpose of the .gitignore file?**

**Answer:**  
The .gitignore file tells Git which files and directories to ignore when committing changes. It helps avoid committing unnecessary files like build artifacts, temporary files, or sensitive data, thus keeping the repository clean and focused on source code.

* Example:
* \*.log
* node\_modules/
* .env

**4. What is a Git commit, and how is it different from a Git push?**

**Answer:**

* **Git Commit**: A commit is a snapshot of your changes to the repository at a particular point in time. It includes metadata (author, timestamp) and the diff of changes.
* **Git Push**: Pushing sends your local commits to a remote repository. A commit is local until pushed, after which it becomes available to other collaborators.

**5. What are the different Git branch types?**

**Answer:**  
Git supports several types of branches:

* **Local Branch**: A branch stored on your local machine.
* **Remote Tracking Branch**: A reference to a branch in a remote repository (e.g., origin/main).
* **Remote Branch**: A branch that exists on a remote repository (e.g., origin/feature-branch).
* **Head**: A symbolic reference to the currently checked-out branch.

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

**Answer:**  
To revert a commit, you can use the git revert command, which creates a new commit that undoes the changes of the specified commit.

git revert <commit-hash>

* This is useful for safely undoing changes, especially in shared repositories.

**7. What is the difference between git merge and git rebase?**

**Answer:**

* **Git Merge**: Combines two branches into one. It creates a merge commit to bring together the histories of both branches.
* **Git Rebase**: Moves or "replays" your commits on top of another base commit. It rewrites history and results in a cleaner, linear commit history.
* **Use case**: git merge is good for preserving the full history, while git rebase is used to maintain a cleaner, linear history.

**8. What is a merge conflict, and how do you resolve it?**

**Answer:**  
A merge conflict occurs when Git is unable to automatically merge changes from two branches because both branches have made changes to the same part of a file.

To resolve a conflict:

1. Git marks the conflict in the file with <<<<<<<, =======, and >>>>>>>.
2. Edit the file to choose the correct changes.
3. Mark the conflict as resolved by running git add <file-name>.
4. Complete the merge with git commit.

**9. What is the purpose of git stash?**

**Answer:**  
git stash allows you to temporarily save changes that are not yet ready to be committed. This is useful when you need to switch branches but don’t want to commit the changes yet.

* To stash changes:
* git stash
* To apply the changes:
* git stash pop

**10. What is a fast-forward merge in Git?**

**Answer:**  
A fast-forward merge occurs when the current branch’s HEAD can be moved forward to the tip of the branch being merged because no changes exist in the current branch. This means no merge commit is created, and Git simply advances the branch pointer.

* **Fast-forward**: No merge commit, just moving the pointer.
* **Non-fast-forward**: Requires a merge commit.

**11. What is git rebase -i?**

**Answer:**  
git rebase -i (interactive rebase) allows you to rewrite commits by squashing, editing, or reordering them. It’s useful for cleaning up commit history before pushing changes.

Example:

git rebase -i HEAD~3

This opens an editor where you can modify the last three commits.

**12. What is the git reflog command used for?**

**Answer:**  
git reflog is a tool to view the history of HEAD and branch references. It is useful for recovering lost commits or branches, as it tracks changes to HEAD that may not be part of the current branch history.

* Example:
* git reflog

**13. What does git pull --rebase do?**

**Answer:**  
git pull --rebase fetches changes from the remote repository and rebases your local commits on top of them instead of creating a merge commit. This creates a cleaner, linear history.

**14. What is the .gitmodules file?**

**Answer:**  
The .gitmodules file is used to configure Git submodules. Submodules are repositories embedded within a Git repository, allowing you to manage external repositories as part of your project.

**15. How can you delete a branch in Git?**

**Answer:**  
To delete a branch in Git:

* **Delete local branch**:
* git branch -d <branch-name>
* **Delete remote branch**:
* git push origin --delete <branch-name>

**16. What is the difference between git reset and git checkout?**

**Answer:**

* **git reset**: Moves the current branch’s HEAD to a specified commit, modifying the staging area and optionally the working directory.
* **git checkout**: Switches branches or restores a file to its state in a specific commit.

**17. What is git bisect, and how does it work?**

**Answer:**  
git bisect helps you find which commit introduced a bug by performing a binary search through your commit history. You mark a known "good" commit and a "bad" commit, and Git will help you narrow down the problematic commit.

**18. What is the use of git diff?**

**Answer:**  
git diff shows the changes between the working directory and the staging area, or between commits.

* To see unstaged changes:
* git diff
* To see staged changes:
* git diff --staged

**19. How do you rename a Git branch?**

**Answer:**  
To rename the current branch:

git branch -m <new-branch-name>

To rename a branch that is not currently checked out:

git branch -m <old-branch-name> <new-branch-name>

**20. What is git tag, and how is it used?**

**Answer:**  
git tag is used to mark specific commits as important, typically for releases or versions. There are two types of tags:

* **Lightweight tags**: Pointers to a commit.
* **Annotated tags**: Tags with metadata such as the tagger's name, email, and date.
* Example:
* git tag -a v1.0 -m "Version 1.0 release"

**21. What is a Git submodule?**

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

* **Add a submodule**:
* git submodule add <repository-url> <path>

**22. How do you configure user details in Git?**

**Answer:**  
You can set your Git configuration globally or per repository using git config.

* Global configuration:
* git config --global user.name "Your Name"
* git config --global user.email "youremail@example.com"

**23. What is git merge --no-ff?**

**Answer:**  
git merge --no-ff forces a merge commit even if a fast-forward merge is possible. This preserves the history of the feature branch and creates a clear boundary in the commit history.

**24. What is the git gc command used for?**

**Answer:**  
git gc (garbage collection) optimizes the repository by removing unnecessary files and packing objects to improve performance.

* Example:
* git gc

**25. How do you show commit history in a simplified manner?**

**Answer:**  
To view commit history in a concise form:

git log --oneline

This will display each commit as a one-line summary.

**26. What is the purpose of git log --graph?**

**Answer:**  
git log --graph

shows the commit history in a tree-like graph format, making it easier to visualize branching and merging history.

**27. How do you create an alias for a Git command?**

**Answer:**  
You can create a Git alias for frequently used commands to speed up your workflow.

* Example:
* git config --global alias.st status

This lets you use git st instead of git status.

**28. What is the git ls-files command?**

**Answer:**  
git ls-files shows all the files that are currently being tracked by Git in the repository. It lists files that are staged for commit or are part of the project history.

**29. What is git cherry-pick?**

**Answer:**  
git cherry-pick allows you to apply the changes from a specific commit onto your current branch, without merging the entire branch.

* Example:
* git cherry-pick <commit-hash>

**30. What is git push --force?**

**Answer:**  
git push --force forces Git to push changes to a remote repository, even if it would overwrite changes. It is often used when rewriting history, such as after a rebase.

**31. What is the purpose of git config --list?**

**Answer:**  
git config --list displays all the current configuration settings in Git, including global, system, and repository-specific configurations.

**32. How do you delete a Git repository?**

**Answer:**  
To delete a Git repository, you simply delete the .git directory. This removes all versioning information and history, effectively turning the project into a regular folder.

rm -rf .git

**33. What is git reset --hard?**

**Answer:**  
git reset --hard resets the repository’s state to the specified commit and discards all changes in the working directory and staging area.

* Example:
* git reset --hard <commit-hash>

**34. What is git pull --no-ff?**

**Answer:**  
git pull --no-ff performs a fetch followed by a merge, and ensures that a merge commit is created even if the merge could be fast-forwarded. This is useful for maintaining clear histories when merging feature branches.

**35. What are the benefits of using Git over other version control systems?**

**Answer:**

* **Distributed system**: Each user has a full history of commits.
* **Offline capability**: You can work without an internet connection.
* **Branching and merging**: Git’s branching model makes it easy to work on multiple features or fixes simultaneously.
* **Speed**: Git is fast, particularly for operations like committing and branching.

**36. How do you squash commits in Git?**

**Answer:**  
To squash commits (combine multiple commits into one), you can use an interactive rebase:

git rebase -i HEAD~3

Then change pick to squash for the commits you want to combine.

**37. What is git fetch?**

**Answer:**  
git fetch downloads objects and refs from a remote repository, without merging them into your current branch. It updates your local references, but leaves your working directory unchanged.

**38. What is the difference between git pull and git fetch?**

**Answer:**

* **git fetch**: Downloads changes from the remote but doesn’t apply them to your working directory.
* **git pull**: Fetches changes from the remote and automatically merges them into your current branch.

**39. What are Git hooks?**

**Answer:**  
Git hooks are scripts that run automatically at certain points in the Git workflow, such as before or after a commit, push, or merge. These can be used for tasks like running tests, linting, or enforcing policies.

**40. How do you track a remote branch in Git?**

**Answer:**  
You can track a remote branch by using:

git checkout --track origin/<branch-name>

This sets up a tracking branch that tracks the remote branch.

**41. What does the git commit --amend command do?**

**Answer:**  
git commit --amend modifies the most recent commit. It opens an editor to allow changes to the commit message, or it can be used to add changes to the last commit.

**42. What is git clean?**

**Answer:**  
git clean removes untracked files from the working directory, including files that are not part of the version control system. This helps clean up unnecessary files.

* Example:
* git clean -f

**43. What are Git tags used for?**

**Answer:**  
Git tags are used to mark specific points in the commit history, often used to mark releases or important milestones in a project’s lifecycle.

* Example:
* git tag v1.0

**44. What is git diff --staged?**

**Answer:**  
git diff --staged shows the differences between the staged files (those added to the index) and the last commit. It helps you see what changes will be committed.

**45. What is the git log --oneline command used for?**

**Answer:**  
git log --oneline displays the commit history in a simplified form, showing only the commit hash and the commit message on one line for each commit.

**46. What does git push --tags do?**

**Answer:**  
git push --tags pushes all local tags to the remote repository. This is necessary if you've created new tags locally and want to share them with others.

**47. How do you undo a commit in Git?**

**Answer:**  
To undo the last commit:

git reset --soft HEAD~1

This removes the commit but keeps your changes in the working directory.

**48. What is a git rebase conflict, and how do you resolve it?**

**Answer:**  
A rebase conflict occurs when Git encounters a conflict while attempting to reapply commits during a rebase operation. To resolve it:

1. Fix the conflict manually.
2. Stage the resolved files.
3. Continue the rebase:
4. git rebase --continue

**49. How do you view changes between commits in Git?**

**Answer:**  
To view changes between two commits, use:

git diff <commit-hash-1> <commit-hash-2>

**50. What is git submodule update?**

**Answer:**  
git submodule update initializes, fetches, and checks out the submodule’s latest commit. It ensures that the submodule is in sync with the repository.