**Git Interview Questions**

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

* Tests understanding of Git’s distributed nature vs. centralized systems.
* **Sample Answer**: Git is a distributed version control system that tracks changes in source code. Unlike SVN (centralized, single server), Git allows every developer to have a full repository copy, enabling offline work and faster operations.

**2. What is git cherry-pick, and when would you use it?**

* + **Sample Answer**: git cherry-pick <commit> applies a specific commit’s changes to the current branch, creating a new commit. It’s used to backport fixes or selectively apply changes without merging an entire branch. For example, picking a bug fix from a feature branch to main.
  + **Follow-Up**: Can you provide an example of cherry-picking a commit?
    - **Answer**: git checkout main; git cherry-pick abc123 applies the abc123 commit from another branch to main.

**3. How do you resolve merge conflicts in Git?**

* + **Sample Answer**: When a merge conflict occurs, Git pauses the merge. Open the conflicting files, resolve the marked conflicts manually, then run git add <file> and git commit to complete the merge. Alternatively, use git merge --abort to cancel.

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

* + **Sample Answer**: git merge combines branches, creating a merge commit, preserving history. git rebase rewrites history by moving the current branch’s commits onto another branch’s tip, creating a linear history. Merge is safer for shared branches; rebase is cleaner for local work.

**5. What is git fetch vs. git pull?**

* + **Sample Answer**: git fetch downloads remote repository data without merging, while git pull fetches and merges changes into the current branch. Use fetch to review changes before merging.

**6. How do you undo a commit that hasn’t been pushed yet?**

* + **Sample Answer**: Use git reset --soft HEAD^ to keep changes in the working directory or git reset --hard HEAD^ to discard them. Alternatively, git commit --amend modifies the last commit.

**7. What happens during a git cherry-pick if there’s a conflict? How do you resolve it?**

* **Sample Answer**: If git cherry-pick encounters a conflict (e.g., overlapping changes), Git pauses. Resolve conflicts in the affected files, then run git add <file> and git cherry-pick --continue. To abort, use git cherry-pick --abort. For example, picking a commit that modifies the same lines as the current branch may cause conflicts.

**8. Explain the Git rebase interactive mode (git rebase -i).**

* **Sample Answer**: git rebase -i <commit> allows editing, squashing, or reordering commits. For example, git rebase -i HEAD~3 opens an editor to modify the last three commits. Use it to clean up history before pushing.

**9. How would you recover a deleted branch in Git?**

* **Sample Answer**: Find the branch’s last commit hash using git reflog, then recreate the branch with git branch <branch-name> <commit-hash>. For example, git branch feature-x abc123.

**10. What are the risks of using git cherry-pick extensively?**

* **Sample Answer**: Overusing git cherry-pick can duplicate commits, complicating history and making it hard to track changes. It may also lead to conflicts if the same changes are later merged. Use it sparingly and prefer merging or rebasing for larger integrations.

**11. You accidentally committed sensitive data to a public repository. How do you remove it?**

* **Sample Answer**: Rewrite history with git filter-branch or git rebase -i to remove the commit, then force-push with git push --force. Use tools like BFG Repo-Cleaner for efficiency. Notify collaborators and rotate any exposed credentials.

**12. A teammate’s branch has a commit you need, but you don’t want to merge their entire branch. How do you proceed?**

* **Sample Answer**: Use git cherry-pick <commit-hash> to apply only that commit to your branch. For example, git cherry-pick abc123 copies the desired commit’s changes.

**13. You’ve pushed a commit to a shared branch but realize it’s incorrect. How do you fix it safely?**

* **Sample Answer**: Avoid rewriting shared history. Instead, create a new commit to revert the changes with git revert <commit-hash>, then push it. Communicate with the team to avoid conflicts.