Scenario: You have multiple commits on your local branch, but you only want to push some of them to the remote repository while keeping others local. What command would you use to push specific commits to the remote repository?Solution: To push specific commits to the remote repository, you would use the following Git command:

git push origin <commit-hash>:<branch-name>

Scenario: You've accidentally checked out to a different branch and want to switch back to the branch you were previously working on. What command would you use to switch back to the previous branch?Solution: To switch back to the previously checked-out branch, you would use the following Git command:

git checkout -

Scenario: You have multiple remote repositories configured for your Git project, and you want to push changes to a specific remote repository other than the default one. What command would you use to push to a non-default remote repository?Solution: To push changes to a specific remote repository other than the default one, you would use the following Git command:

git push <remote-name> <branch-name>

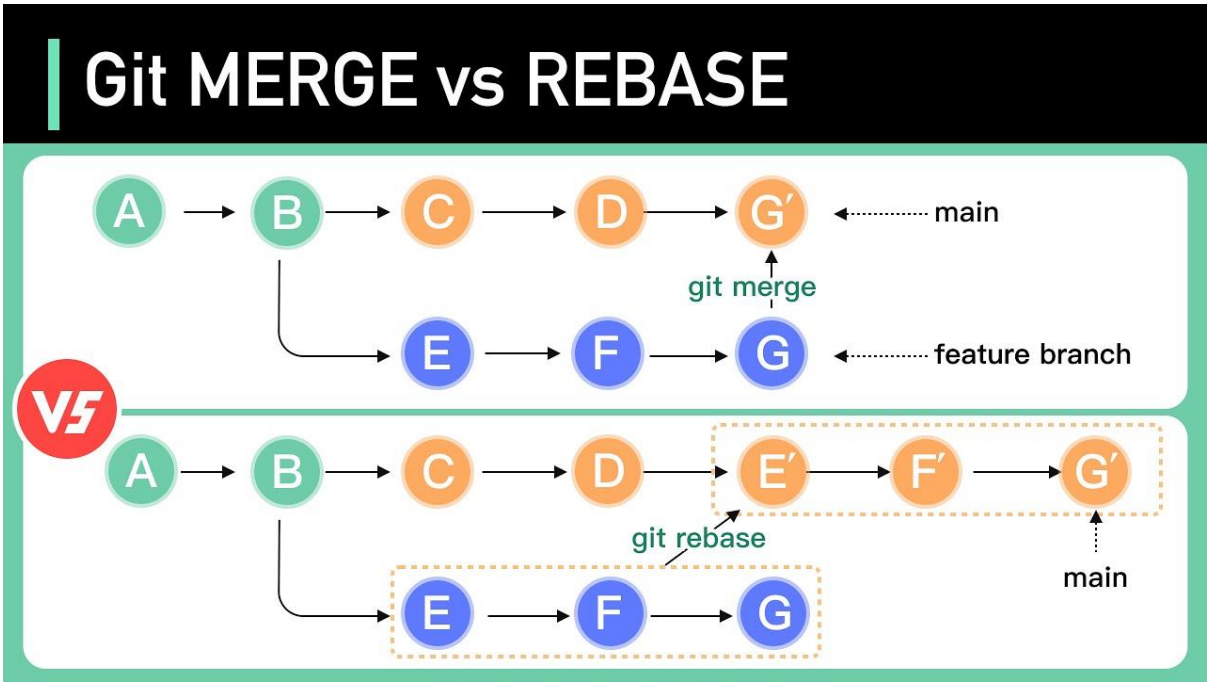
Scenario: You've made changes to a file and want to temporarily switch to another branch to work on a different task without committing the changes. What command would you use to stash the changes and switch branches?Solution: To temporarily stash changes and switch branches without committing them, you would use the following Git command:

git checkout <branch-name>

Scenario: You want to pull changes from the remote repository and automatically merge them into your local branch without requiring manual intervention. What command would you use to achieve this?Solution: To pull changes from the remote repository and automatically merge them into your local branch, you would use the following Git command:

git pull origin <branch-name>

Scenario: You're working on a feature branch called "feature-x" and have made several commits. Meanwhile, changes have been made to the main branch by other team members. You want to rebase your feature branch onto the main branch to incorporate these changes while preserving your commit history. However, you also want to squash your commits into a single commit before merging. How would you accomplish this?



git checkout feature-x

git fetch origin main

git rebase main

git reset --soft HEAD~<number of commits>

git commit -m "Squashed commits"

git checkout main

git merge feature-x

Scenario: You've been working on a project with a colleague, and you both have made changes to the same file, resulting in merge conflicts. You want to resolve these conflicts by keeping some changes from your colleague and some of your own changes. How would you resolve the merge conflict?

git pull origin main

[Resolve merge conflicts in the conflicted file manually]

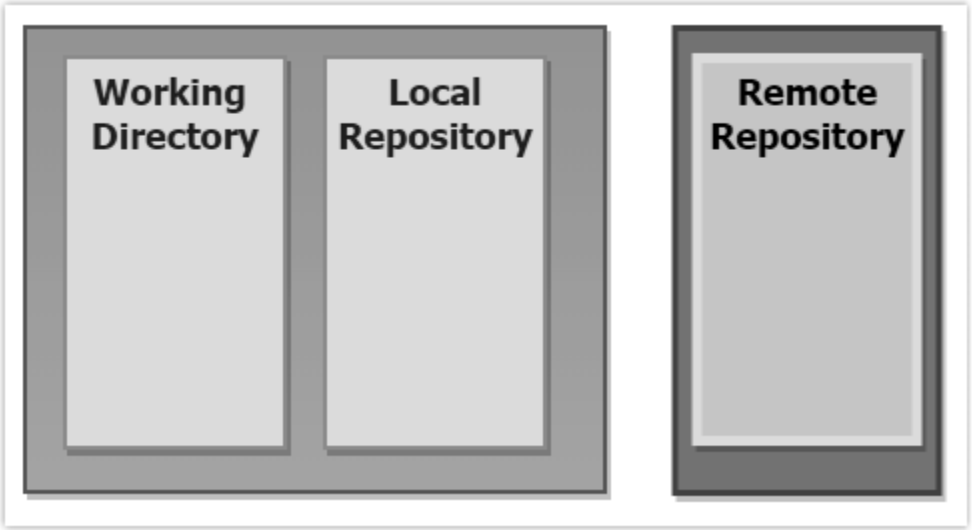
git add <conflicted-file>

git commit -m "Resolved merge conflict"

On a user's workstation, a Git installation includes the following items:

- The local Git repository where the history of all commits across all branches are maintained.

- A working directory where a developer actively edits and updates files that Git tracks.



The key difference between git fetch and pull is that git pull copies changes from a remote repository directly into your working directory, while git fetch does not. The git fetch command only copies changes into your local Git repo. The git pull command does both.

You're collaborating on a project with a remote team, and you've been asked to review changes made by another team member in their branch named "feature-y". However, you want to review the changes locally before merging them into your own feature branch named "feature-x". How would you fetch and review the changes from their branch without merging them into your branch?

git fetch origin feature-y

git checkout feature-y

[Review changes locally]

git checkout feature-x