Git: Merge vs Rebase, Hands-on, and How to Undo Rebase

# 1. Merge vs Rebase

Merge and Rebase are both Git commands used to integrate changes from one branch into another. However, they do this in fundamentally different ways, leading to different implications for the project history.

## Merge

Merge combines the histories of two branches into a single branch, typically creating a new 'merge commit'. This preserves the context of the branch development, making it clear that a merge has occurred. This method is non-destructive as it does not change existing history.

## Rebase

Rebase rewrites the commit history by placing the commits from one branch onto the base of another, creating a linear history. This can simplify project history but requires careful management as it changes commit hashes and can complicate shared branch histories if not used properly.

# 2. Hands-on Merge and Rebase

The following sections demonstrate how to perform both merge and rebase operations in Git, offering practical examples.

## Performing a Merge

To merge changes from one branch into another, ensure you're on the branch that you want to merge into and then use the 'git merge' command with the name of the other branch. This will create a merge commit if there are divergent histories.

Example:

git checkout master  
git merge feature

## Performing a Rebase

To rebase a branch onto another, check out the branch you want to rebase and use 'git rebase' with the target branch name. This will start reapplying commits from the current branch onto the target branch.

Example:

git checkout feature  
git rebase master

# 3. How to Undo Rebase

If a rebase does not go as planned, it can be undone using the 'git reflog' to find the state before the rebase started, and then performing a hard reset to that state.

Using git reflog and reset:

git reflog  
git reset --hard HEAD@{N}

Here, 'N' is the number from the reflog entry before the rebase. It’s essential to carefully select this to avoid losing work.

After resetting locally, you would need to force push to rewrite the history on the remote repository:

git push origin +branch-name

The plus sign + before the branch name indicates a forced push, which is necessary to overwrite the remote history. Be very careful with this, as it can disrupt other collaborators.