How To Combine Multiple Commits

To combine multiple commits into a single commit, you can use **interactive rebase**. Here's how to do it step-by-step:

Steps to Combine Commits:

1. **Start an Interactive Rebase:** Open your terminal and type the following command to start an interactive rebase. The number `N` in the command represents how many commits you want to combine. For example, if you want to combine the last 3 commits, use `HEAD~3`.

   ```
   git rebase -i HEAD~3
   ```

   This will open an editor showing a list of the last 3 commits in your branch.

2. **Choose Commits to Combine:** In the editor, you will see something like this:

   ```
   pick <commit-hash> Commit message 1
   pick <commit-hash> Commit message 2
   pick <commit-hash> Commit message 3
   ```

   The `pick` command means that Git will include these commits in the rebase.

   To combine the commits, you need to **change `pick` to `squash` (or `s`)** for the second and third commits (or whichever ones you want to combine into the first one). Here's how it should look:

   ```
   pick <commit-hash> Commit message 1
   squash <commit-hash> Commit message 2
   squash <commit-hash> Commit message 3
   ```

   This tells Git to squash the second and third commits into the first one.

3. **Save and Close the Editor:** After making the changes, save and close the editor. Git will now combine the commits.
4. **Edit the Commit Message:** Git will open another editor where you can **edit the commit message** for the new combined commit. You can choose to keep all the commit messages, edit them, or write a new one.

   The editor will display something like this:

   ```
   # This is a combination of 3 commits.
   # The first commit's message is:
   Commit message 1

   # The 2nd commit's message is:
   Commit message 2

   # The 3rd commit's message is:
   ```

```
Commit message 3
```

You can edit the message as you prefer. After editing, save and close the editor.

5. **Complete the Rebase:** Git will now combine the commits and apply the changes. If there are no conflicts, the rebase will finish successfully.
6. **Force Push (if the branch is pushed to a remote):** If the branch was already pushed to a remote (like GitHub), you'll need to **force push** the changes because you've rewritten history:

```
git push --force
```

Important Notes:

- Be cautious when force-pushing, especially if others are working on the same branch, as it rewrites history.
- If there are conflicts during the rebase process, Git will prompt you to resolve them before you can continue the rebase.
- Squashing commits is a good way to clean up your commit history before merging into a main branch.