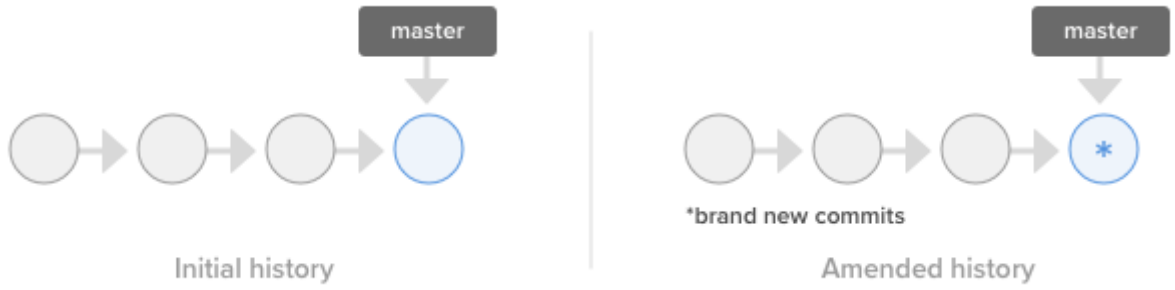
Rewriting history

- Git commmit --amend
- Git rebase
- Git cherry pick
- Git merge --squash

There are times when you need to revise your local commit history. This can include anything from changing your commit message to changing the order of your commits to squashing commits together. In this section, we'll discuss how to rewrite history before sharing your work with others.

Git commit --amend

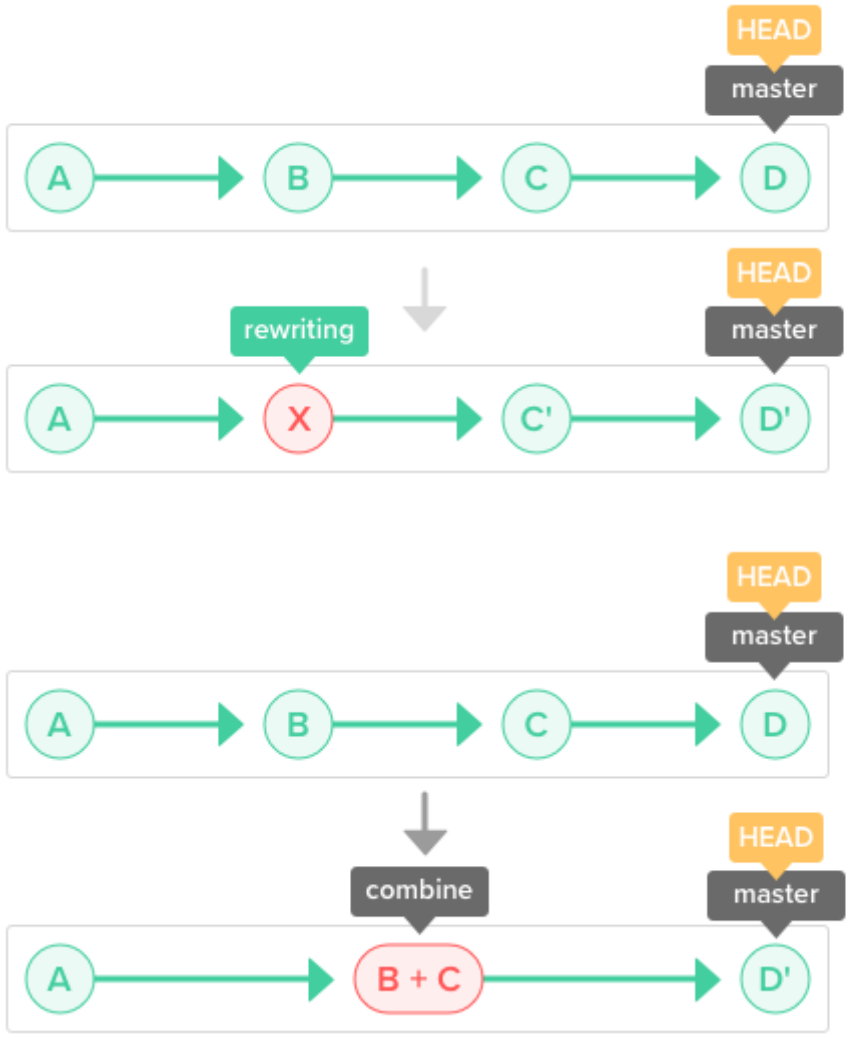
You can modify the most recent commit in the same branch by running **git commit --amend**. This command is convenient for adding new or updated files to the previous commit. It is also a simple way to edit or add comments to the previous commit.



Use git commit --amend to modify the most recent commit.

Run **git rebase** and add in the -i option to rewrite, replace, delete, and merge individual commits in the history. You can also:

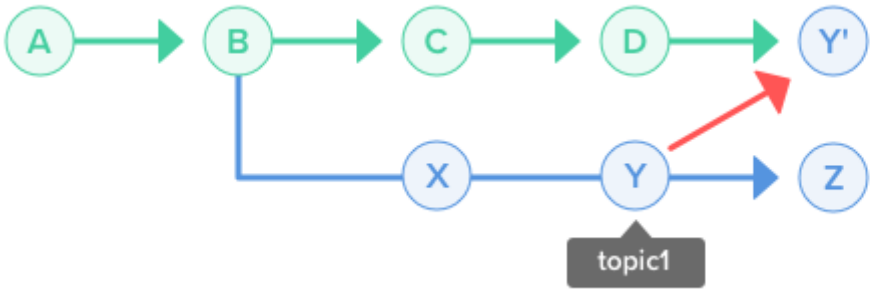
- Rewrite a past commit message
- Squash a group of commits together
- Add files that have not been committed



Identify the commit you want to rewrite and run the git rebase -i command.

Git cherry pick

You can apply an existing commit from another branch to the current branch within the repository by using the git cherry-pick command. Cherry-picking allows you to:

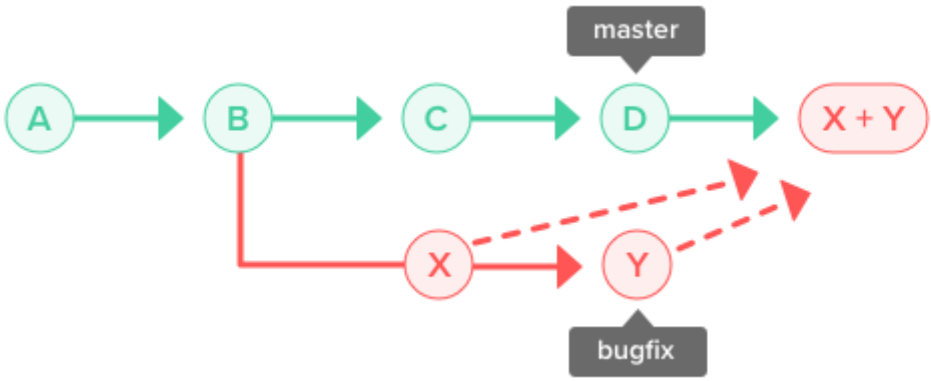


Use git cherry-pick to change the branch of a commit.

Git merge --squash

Squashing is the process of merging multiple commits into a single commit.

If you run **git merge** and the **--squash** option, a new commit will group all of the commits from that branch together. The commit will be used to merge into the current branch.



Use git merge --squash to unifying commits from a feature/topic branch into a single commit to be merged into your current branch.