Amend a Git Commit
Amend a Git Commit
Philip Wilkinson
Software Engineer, Amazon

Changing the last commit with --amend

You may want to amend a commit because you forgot to stage a file, mistakes were made in the original commit, or because some small additional changes should be bundled in the last commit rather than a new one. The --amend flag allows you to make changes to the previous commit.

For this, you need to stage the files that you missed or want to modify using git add <file_name> and then use git commit --amend. A typical workflow for this may take the form:

```
$ git add <new_file>
$ git commit -m “Create new component
# edit the file to remove unnecessary code
$ git add <existing_file>
$ git commit --amend
```

Staging the <existing_file> with the second git add allows the git commit --amend functionality to bundle the second set of changes in with those in the last commit.

This allows you to change anything in the previous commit, whether that is removing lines from a file, removing a file, or adding a new file.

When using the --amend flag, Git will open your default text editor to allow you to change the commit message. If you only want to amend files and not the commit message, you can use the --no-edit flag.

Changing the commit message

One way to use the --amend flag is to change the message of the last commit. This can

be beneficial when a commit was made too early, a temporary commit message was used, or team conventions weren't followed in the original message. Changing the commit message can ensure that it accurately describes the previous commit so that others know exactly what the change contains.

To amend the message when no changes are current staged you can run:

```
$ git commit --amend
```

This will then open your chosen text editor to allow you to edit the previous message.

Alternatively, if you don't want to open the editor, you can simply add the -m flag which will allow you to specify the new message in the same command.

```
$ git commit --amend -m "New commit message"
```

Be careful when amending public commits

git commit --amend works by removing the previous commit and creating a new one. This means that it changes the history of the repository and can make things difficult and messy if the repository has been shared publicly, such as on GitHub, and other developers have built on the previous commit. Therefore it is not recommended to use `amend` for commits that have already been pushed to a shared repository as it can cause conflicts with other developers' work.

Amending specific commits

git commit --amend is typically used to change the last commit only. Since this command removes the specified commit and creates a new one this affects the commit history and is very likely to cause merge conflicts. If you do want to change specific commits you should instead use:

git reset: Which can be used to move the current tip of a branch to a previous commit and thus reset the branch to a previous state
git revert: This can be used to "undo" a specific commit by creating a new commit overwriting the old one.
git rebase: This can be used to navigate back to a specific commit in your history. You can then used git commit -amend to edit the changes to that specific commit

WRITTEN BY
Philip Wilkinson
Philip is a data visualisation engineer who also likes to write about data science, maching learning, and algorithms.
FILED UNDER
GIT
Related articles

Undo a Git Pull
This post will guide you through reverting a git pull and getting you back to your previous commits and history. Also learn how to undo GitHub pull requests.
GIT
Undo a Git Merge
This post will show you how to undo a merge operation using git revert and git reset.
GIT
Prompt Show Git Branch In Prompt
Enhance your terminal with a custom Git prompt. Learn different ways to integrate this contextual info, from custom shell functions to Warp context chips and toolkits like Starship and P10K.
GIT
How To Create a Git Repository
Creating repos in various scenarios with git
GIT
Git Clone, Push, And Pull Over SSH
use an SSH key for Git repos
GIT
Change Git Origin Remote URL
change git repo URL
GIT
Delete Local Git Branch
variety of situations to delete a local git branch
GIT
Git Commit History
Navigate the commit history of a repo or branch
GIT
Undoing Git Commits
Overview of the simple way to undo a commit using git reset
GIT
Git Push Tags
This post will show you how to push a single tag, multiple tags, all tags, and tags with commits.
GIT
Create Folder In GitHub Repository
Learn how to create and push one or more empty directories in a Git repository using `.placeholder` and `README.md` files using both the CLI and the GitHub interface.
GIT
Git Push Origin
A breakdown of git push origin
GIT

TERMINUS BY WARP
CONTENTS

Privacy
FAQs
Terms