

## Study guide: Git Revert

When writing and committing code, making mistakes is a common occurrence. Thankfully, there are multiple ways for you to revert or undo your mistakes. Take a look at the helpful commands below.

`git checkout` [↗](#) is used to switch branches. For example, you might want to pull from your main branch. In this case, you would use the command `git checkout main`. This will switch to your main branch, allowing you to pull. Then you could switch to another branch by using the command `git checkout <branch>`.

`git reset` [↗](#) can be somewhat difficult to understand. Say you have just used the command `git add` to stage all of your changes, but then you decide that you are not ready to stage those files. You could use the command `git reset` to undo the staging of your files.

There are some other useful articles online, which discuss more aggressive approaches to [resetting the repo](#) [↗](#) (Git repository). As discussed in this article, doing a hard reset can be extremely dangerous. With a hard reset, you run the risk of losing your local changes. There are safer ways to achieve the same effect. For example, you could run `git stash`, which will temporarily shelve or stash your current changes. This way, your current changes are kept safe, and you can come back to them if needed.

`git commit --amend` [↗](#) is used to make changes to your most recent commit after-the-fact, which can be useful for making notes about or adding files to your most recent commit. Be aware that this `git --amend` command rewrites and replaces your previous commit, so it is best not to use this command on a published commit.

`git revert` [↗](#) makes a new commit which effectively rolls back a previous commit. Unlike the `git reset` command which rewrites your commit history, the `git revert` command creates a new commit which undoes the changes in a specific commit. Therefore, a `revert` command is generally safer than a `reset` command.

For more information on these and other methods to undo something in Git, checkout this [Git Basics - Undoing Things](#) [↗](#) article.

Additionally, there are some interesting considerations about how git object data is stored, such as the usage of SHA-1.

SHA-1 is what's known as a *hash function*, a cryptographic function that generates a digital fingerprint of a file. Theoretically, it's impossible for two different files to have the same SHA-1 hash, which means that SHA-1 can be used for two things:

- Confirming that the contents of a file have not changed (digital signature).
- Serving as an identifier for the file itself (a token or fingerprint).




Git calculates a hash for every commit. Those hashes are displayed by commands like git log or in various pages on Github. For commands like git revert, you can then use the hash to refer to a specific commit.

Feel free to read more here:

- [SHA-1 collision detection on GitHub.com](#) [↗](#)

Even the most accomplished developers make mistakes in Git. It happens to everyone, so don't stress about it. You have these and other methods to help you revert or undo your mistakes.

Mark as completed

 Like  Dislike  Report an issue