

# Unmodifying a Modified File

What if you realize that you don't want to keep your changes to the `CONTRIBUTING.md` file? How can you easily unmodify it — revert it back to what it looked like when you last committed (or initially cloned, or however you got it into your working directory)? Luckily, `git status` tells you how to do that, too. In the last example output, the unstaged area looks like this:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   CONTRIBUTING.md
```

It tells you pretty explicitly how to discard the changes you've made. Let's do what it says:

```
git checkout -- CONTRIBUTING.md
git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

       renamed:    README.md -> README
```

You can see that the changes have been reverted.

It's important to understand that `git checkout -- <file>` is a dangerous command. Any local changes you made to that file are gone — Git just replaced that file with the most recently-committed version. Don't ever use this command unless you absolutely know that you don't want those unsaved local changes.

If you would like to keep the changes you've made to that file but still need to get it out of the way for now, we'll go over stashing and branching in [Git Branching](#); these are generally better ways to go.

Remember, anything that is *committed* in Git can almost always be recovered. Even commits that were on branches that were deleted or commits that were overwritten with an `--amend` commit can be recovered (see [Data Recovery](#) for data recovery). However, anything you lose that was never committed is likely never to be seen again.