Unstaging a Staged File

The next two sections demonstrate how to work with your staging area and working directory changes. The nice part is that the command you use to determine the state of those two areas also reminds you how to undo changes to them. For example, let's say you've changed two files and want to commit them as two separate changes, but you accidentally type `git add *` and stage them both. How can you unstage one of the two? The `git status` command reminds you:

```
git add *
git status
 branch master
hanges to be committed:
 (use "git reset HEAD <file>..." to unstage)

    renamed:     README.md -> README
    modified:    CONTRIBUTING.md
```

Right below the "Changes to be committed" text, it says use `git reset HEAD <file>...` to unstage. So, let's use that advice to unstage the `CONTRIBUTING.md` file:

```
git reset HEAD CONTRIBUTING.md
nstaged changes after reset:
CONTRIBUTING.md
git status
 branch master
hanges to be committed:
 (use "git reset HEAD <file>..." to unstage)

    renamed:     README.md -> README


hanges 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
```

The command is a bit strange, but it works. The `CONTRIBUTING.md` file is modified but once again unstaged.

It's true that `git reset` can be a dangerous command, especially if you provide the `--hard` flag. However, in the scenario described above, the file in your working directory is not touched, so it's relatively safe.

For now this magic invocation is all you need to know about the `git reset` command. We'll go into much more detail about what `reset` does and how to master it to do really interesting things in Reset Demystified.