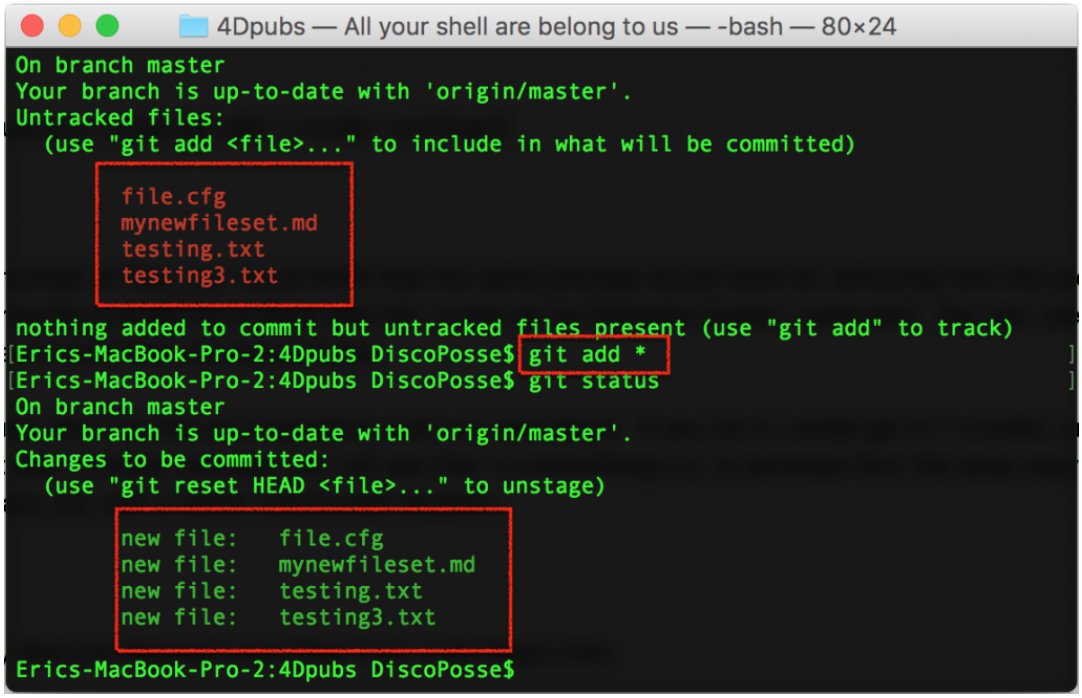
Git Remove Multiple Deleted Files

When working in the Git version control system, you may find yourself doing some handling of large numbers of files in a single commit. The commit part is the easy part. Adding files is very simple by using the `git add *` command which adds all of the new files that appear as new since the most recent commit.

Running a `git status` shows a few files to be added. We add them all using a `git add *` command, and see that the files are added and ready for a commit:



When you remove a large number of files, you would think that the same process would work for removing from the previous state of the repository. Removing a single file is done with the `git rm filename` command. You can use wildcards, but that's going to do a lot more than you would hope.

WARNING: Seriously, don't try this on a repository that you care about. If you run a `git rm *` just like you did with the `git add *` process, you will see that it could be nothing is removed from the local copy of your repo. In worst situations, you may also find that a lot is removed. A new commit will leave you with a rather unfortunate situation.

How to Safely Remove Deleted Local Files From a Git Repo

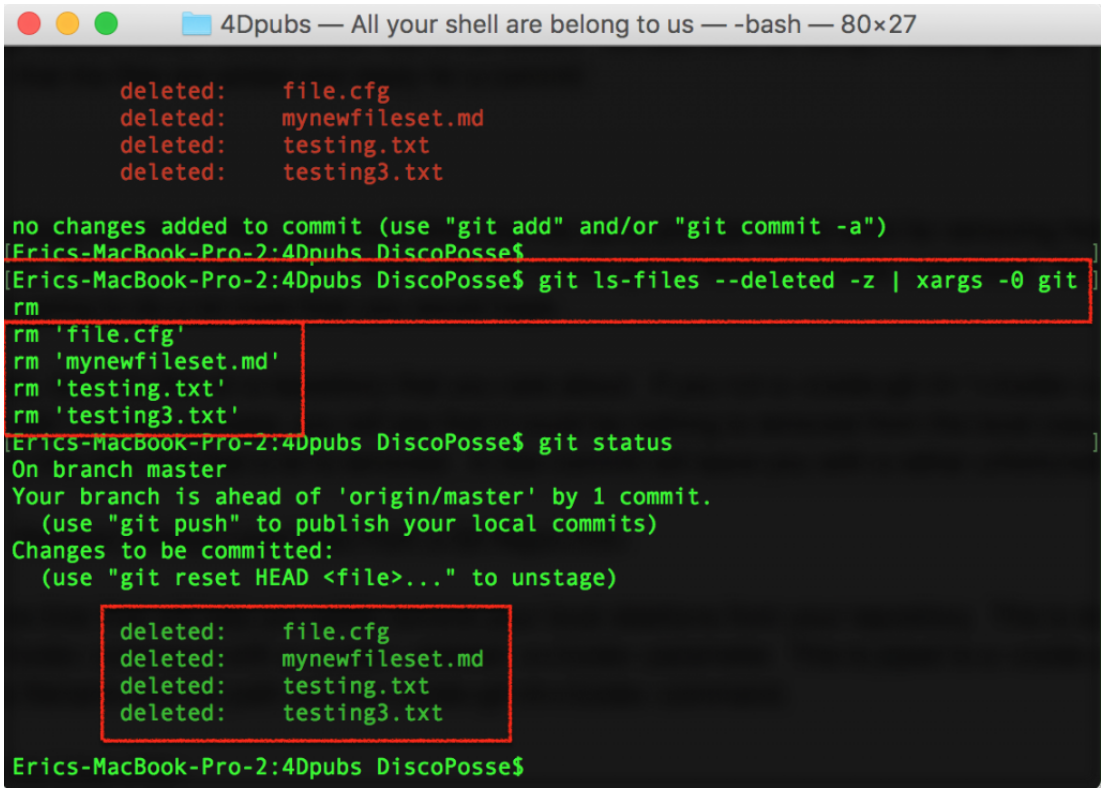
There is a simple one-liner that will help you safely remove your local deletions from your repository. This is done by using the `git ls-files` command with a `--deleted -z` parameter. This is piped to a `git rm` command using the filename and full path into the `git rm` command.

The Magical One-Liner

This is the full one-liner:

```
git ls-files --deleted -z | xargs -0 git rm
```

This is the result:



Using that command is much safer. This lets you remove all of the files marked as deleted to ensure your next commit is cleaned of your deleted files and nothing that you unexpectedly removed by a slip of a wildcard statement.