How to Use "prune" in Git to Clean Up Remote Branches

git-tower.com/learn/git/faq/cleanup-remote-branches-with-git-prune

How to Use prune to Clean Up Remote Branches in Git

One of the great things about Git is that it's *very careful about deleting data*. This makes it pretty hard to lose commits or other valuable data in Git!

A small downside of this is that you might sometimes see stale data that you actually don't need anymore. One of the best examples of this are **references to remote branches that have already been deleted**: let's say that one of your teammates deletes a branch on your shared remote repository; the branch will still be displayed for you unless you explicitly instruct Git to clean up.

In this short article, we'll look at how to do this using the `prune` option in Git.

Using "prune" on a Remote Repository

"prune" is available as an option for the `git fetch` and `git remote` commands. (Don't confuse this with the stand-alone `git prune` command - this is used during garbage collection and is not what we're talking about here.)

The easiest way to use prune is to provide it as an option when fetching:

```
$ git fetch --prune origin
```

In cases where you'd like to *only* perform a prune and *not* fetch remote data, you can use it with the `git remote` command:

```
$ git remote prune origin
```

The result is the same in both cases: stale references to remote branches that don't exist anymore on the specified remote repository will be deleted. By the way: you never have to worry about your local branches, since prune will never affect those.

If you want to have prune executed with every fetch operation, you can configure Git accordingly:

```
$ git config --global fetch.prune true
```

In case you are using the Tower Git client, you don't have to worry about "prune" at all: it is activated by default (and can be switched off simply in the "Fetch" dialog options):