**How to delete a Git branch**

Question and answers: <http://stackoverflow.com/questions/2003505/delete-a-git-branch-both-locally-and-remotely?rq=1>

**The Short Answers**

If you want more detailed explanations of the following commands, then see the long answers in the next section.

Deleting a remote branch:

git push origin --delete <branch> # Git version 1.7.0 or newer

git push origin :<branch> # Git versions older than 1.7.0

Deleting a local branch:

git branch --delete <branch>

git branch -d <branch> # Shorter version

git branch -D <branch> # Force delete un-merged branches

Deleting a local remote-tracking branch:

git branch --delete --remotes <remote>/<branch>

git branch -dr <remote>/<branch> # Shorter

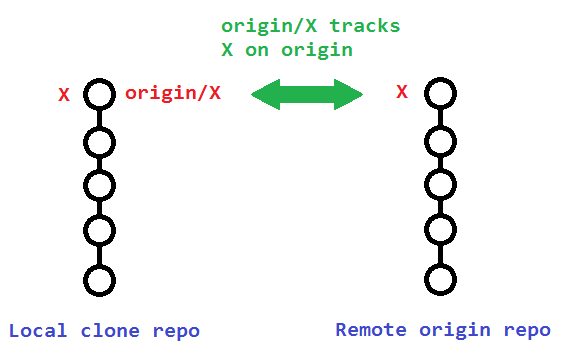
git fetch <remote> --prune # Delete multiple obsolete tracking branches

git fetch <remote> -p # Shorter

**The Long Answer: there are 3 different branches to delete!**

When you're dealing with deleting branches both locally and remotely, keep in mind that **there are 3 different branches involved**:

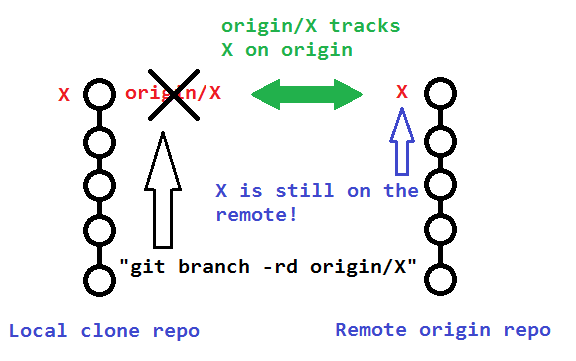
1. The local branch X.
2. The remote origin branch X.
3. The local remote-tracking branch origin/X that tracks the remote branch X.



The original poster used

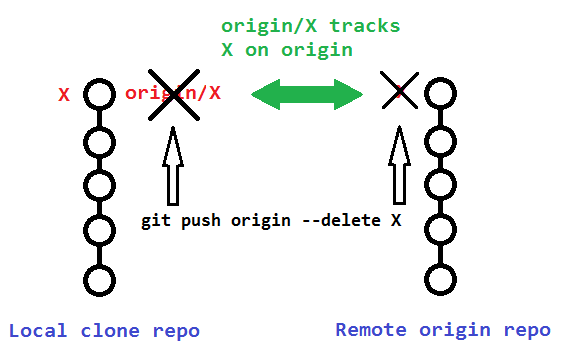
git branch -rd origin/bugfix

which only deleted his **local remote-tracking branch** origin/bugfix, and not the actual remote branch bugfix on origin.



**To delete that actual remote branch**, you need

git push origin --delete bugfix



**Additional Details**

The following sections describe additional details to consider when deleting your remote and remote-tracking branches.

**Pushing to delete remote branches also deletes remote-tracking branches**

Note that deleting the remote branch X from the command line using a git push **will also delete the local remote-tracking branch** origin/X, so it is not necessary to prune the obsolete remote-tracking branch with git fetch --prune or git fetch -p, though it wouldn't hurt if you did it anyway.

You can verify that the remote-tracking branch origin/X was also deleted by running the following:

# View just remote-tracking branches

git branch --remotes

git branch -r

# View both strictly local as well as remote-tracking branches

git branch --all

git branch -a

**Pruning the obsolete local remote-tracking branch origin/X**

If you didn't delete your remote branch X from the command line (like above), then your local repo will still contain (a now obsolete) remote-tracking branch origin/X. This can happen if you deleted a remote branch directly through GitHub's web interface, for example.

A typical way to remove these obsolete remote-tracking branches (since Git version 1.6.6) is to simply run git fetch with the --prune or shorter -p.

* **Note that this removes all obsolete local remote-tracking branches for any remote branches that no longer exist on the remote**:

git fetch origin --prune

git fetch origin -p # Shorter

Here is the relevant quote from the [1.6.6 release notes](https://github.com/git/git/blob/v2.0.0/Documentation/RelNotes/1.6.6.txt#L162-L166) (emphasis mine):

**"git fetch" learned** --all and --multipleoptions, to run fetch from many repositories, and **--prune option to remove remote tracking branches that went stale.** These make "git remote update" and "git remote prune" less necessary (there is no plan to remove "remote update" nor "remote prune", though).

**Alternative to above automatic pruning for obsolete remote-tracking branches**

Alternatively, instead of pruning your obsolete local remote-tracking branches through git fetch -p,*you can avoid making the extra network operation* by just manually removing the branch(es) with the --remote or -r flags:

git branch --delete --remotes origin/X

git branch -dr origin/X # Shorter

**See Also**

* [git-branch(1) Manual Page](http://jk.gs/git-branch.html).
* [git-fetch(1) Manual Page](http://jk.gs/git-fetch.html).
* [Pro Git § 3.5 Git Branching - Remote Branches](http://git-scm.com/book/en/Git-Branching-Remote-Branches).