Pull vs Fetch

Git fetch is used to retrieve commits from a remote repository's branch but it *does not* automatically merge the local branch with the remote tracking branch after those commits have been received.

The above paragraph is a little dense so why don't you reread it one more time.

You provide the exact same information to git fetch as you do for git pull. So you provide the shortname of the remote repository you want to fetch from and then the branch you want to fetch:

```
$ git fetch origin master
```

When git fetch is run, the following things happen:

- the commit(s) on the remote branch are copied to the local repository
- the local tracking branch (e.g. origin/master) is moved to point to the most recent commit

The important thing to note is that the local branch does not change at all.

You can think of git fetch as half of a git pull. The other half of git pull is the merging aspect.

One main point when you want to use git fetch rather than git pull is if your remote branch and your local branch both have changes that neither of the other ones has. In this case, you want to fetch the remote changes to get them in your local branch and then perform a merge manually. Then you can push that new merge commit back to the remote.

Let's take a look at that.

Recap

You can think of the git pull command as doing two things:

1. fetching remote changes (which adds the commits to the local repository and moves the tracking branch to point to them)
2. merging the local branch with the tracking branch

The git fetch command is just the first step. It just retrieves the commits and moves the tracking branch. It *does not* merge the local branch with the tracking branch. The same information provided to git pull is passed to git fetch:

- the shortname of the remote repository
- the branch with commits to retrieve

```
$ git fetch origin master
```