Updating Forked Git Repository from Upstream Source (aka Rebase all the things!)

As you can imagine, the world of IaC (Infrastructure-as-Code) means that we are going to have to dabble a lot more in the world of code repositories. Git is the most common tool I have found in use for code version control, and along with it, Github.com is the most common place that people (including myself https://github.com/DiscoPosse) store their project code.

All Forked Up

When the first work happens with using Github is that you may find yourself forking a repository to create a point-in-time snapshot under your own repositories. This is done for a variety of reasons like contributing upstream code, and keeping a "safe" stable release of a particular codebase that could change and affect other work you are doing with it.

As you can see in the image above, I have a forked copy of the Lattice framework from Cloud Foundry. Nice and easy to do, but as you look at the bottom of the image, you will also see that I've been falling behind on updates.

So, how does someone fix this situation? Let's assume that we are testing locally and find a bug, but then realize that the upstream repository has already fixed the bug. Rather than wiping out the repository altogether and re-cloning, let's fix it in place!

Updating a Forked Repository from Upstream Source

Let's jump in the command line, and see what's happening. In my lattice folder, I will do a `git status` to see the current state of things:

We can see that we show as up to date, but I know that I am 649 commits behind the upstream source. Time to get ourselves up to date for real.

The way we do this is by syncing up our repositories here locally, and then pushing the changes back up to our forked repository. First, let's check our remote source by typing `git remote` to see what's happening locally:

We have one source called `origin` which is our forked repository. We are going to add one more source called `upstream` to point to the original repo using the command `git remote add upstream https://github.com/cloudfoundry-incubator/lattice.git` in my case and then run our `git remote` again to confirm the change:

Now we can see both of our sources. We are assuming that you are using the master branch of your

repo, but just in case, we can also do a `git checkout master` first for safety. As you can see in my case, it will complain that I am already on 'master' and nothing will happen:

Now let's do the next steps which is to fetch the upstream and rebase our local repo. Yes, these are funny sounding terms to some, but you will get used to them. This is done by using the `git fetch upstream` command followed by the `git rebase upstream/master` to sync them up:

Lots of updates came down, and you can see that our rebase has done all the changes locally and if we had any updates, they would be left in place with the underlying repo updates done at the same time.

We need to check our status first using the `git status` and as you can see here, it will show the 649 commits ahead of origin/master which is my forked repo on Github:

Now it's time to push all the updates! This will commit the changes to the Github forked repo for you and then we are up to date with the upstream source. We will use the `git push origin master` which pushes the local changes to the master branch of our origin source (in our case discoposse/lattice) and then we can confirm the changes are committed with a `git status` afterwards:

There you go! Now your forked repository is up to date with the upstream source and you can get back to the coding!

If you check your Gitub page also, you will see the change there:

Hopefully this is helpful, because I have been asked a few times recently about dealing with this issue as people get started with Git. I hope to bring some more quick Git tips as I hear questions come from the community, so feel free to drop me a comment with any question you have!