Git Workflow and Tips

The following is one Islandora developer's workflow for dealing with Islandora commits in git.

Workflow

You may wish to make sure your 6.x/7.x/master branch always tracks the main Islandora repository; they may be a few commits behind (someone committed code in the interim) but it's **never** different. If you never have commits in your 6.x/7.x/master branches that are not already present in the Islandora repository, you will avoid merge problems and keep from committing code that you don't want to commit.

When you want to create a new feature or fix a bug, create a topic branch like so:

First checkout your master branch.

```
git checkout 7.x
```

Then update it so you know you're running the latest code that is available in the public islandora/discoverygarden repository.

```
git update
```

The git update is a custom alias created by Nigel Banks. It basically grabs the latest for the current branch from the remote upstream then pushes those changes into the remote origin. In this way your upstream remote is always the public canonical Islandora repository, and your remote origin is your personal repository on Github.

It expands into this:

```
git pull upstream 7.x && git push origin 7.x
```

So now that your master branch has the latest code, you create your topic branch:

```
git checkout -b 7.x-fixes-some-bug
```

Now you do whatever work is required.

If some other issue pops up, you can create another topic branch to deal with it separately. First switch to your master branch:

```
git checkout 7.x

git update

git checkout -b 7.x-fixes-some-different-bug
```

You can do this with any number of topic branches, and send pull requests for each of them individually, or after working on several separate topic branches you may want to make them into a single branch. This can be done with rebases/merges:

```
git checkout 7.x-fixes-some-bug

git merge 7.x-fixes-some-different-bug or git rebase 7.x-fixes-some-different-bug
```

When you're ready with your changes you can push my topic branch to your personal remote, then issue a pull request.

```
git push -u origin 7.x-fixes-some-bug
```

Cherry-Picking Commits

The following example takes a code contribution from a member of the Islandora community and cherry-picks commits into a separate pull request:

```
git remote add rosiel git://github.com/rosiel/islandora_xml_forms.git

git fetch rosier
```

You need to find out the commit hashes, so you can look at the user's history:

```
git log rosiel/7.x
```

Which returns:

> *commit 120075920a631339719728dd793dfb686e795ca5*
>
> *Author: Rosemary Le Faive*
>
> *Date: Tue Apr 23 17:37:02 2013 -0400*
>
> *Insert missing ['#autocomplete_path'] element*
>
> *commit d6e0b46564f541e3bcea5ad44099d566fe9bfc50*
>
> *Author: Rosemary Le Faive*
>
> *Date: Tue Apr 23 17:27:15 2013 -0400*
>
> *Tag elements autocomplete.*
>
> …..

With this information you can then cherry-pick the two commits.

Before doing so, create a topic branch (So your local 7.x will still always track against your islandora 7.x)

```
git checkout -b 7.x-rosiel-tag-autocomplete
```

At this point your topic branch is the same as your 7.x branch which is the same as the islandora/7.x branch.

```
git cherry-pick d6e0b46564f541e3bcea5ad44099d566fe9bfc50

git cherry-pick 120075920a631339719728dd793dfb686e795ca5
```

Now that your topic branch has your two commits, you now push my whole topic branch to my personal Github repository.

You can use the custom alias "remote-branch" which is defined in Nigel's configuration gist below.

```
git remote-branch
```

Which is equivelent to:

```
git push -u origin 7.x-rosiel-tag-autocomplete
```

Then go onto Github and trigger issue the pull request.

Resources

Free Book on using Git:

A free Pro Git book can be found here: http://git-scm.com/book

Custom Git Aliases:

Also checkout some custom git aliases from Nigel Banks to save you some time: https://gist.github.com/nigelgbanks/5116805