Git Quick Start Guide

A simple cheat sheet for Fedora developers new to Git.

You can run `git status` at any time to get a snapshot of your current state. You can also examine your differences with the master branch on github at any time by executing these two commands:

```
git fetch origin master
git diff origin master
```

Creating and working with a published branch

1. Get the repository:

   ```
   git clone git@github.com:fcrepo/fcrepo.git
   cd fcrepo
   ```

   or

   ```
   git clone http://github.com/fcrepo/fcrepo.git
   cd fcrepo
   ```

2. Create the branch where you'll do your work:

   ```
   git branch fcrepo-780
   git checkout fcrepo-780
   ```

   The `checkout` command makes whatever branch you specify the local active branch. Make your changes, test...

3. Add your edited/new files, then commit your branch:

   ```
   git add myfile.java
   git commit myfile.java
   ```

4. If you're working on a branch for some time, you may want to update your branch with the latest changes to master.

5. Push the branch back up to github

   ```
   git push origin fcrepo-780
   ```

6. Check out master (formerly known as 'trunk'):

   ```
   git checkout master
   ```

   Make sure it's current.

7. Merge the branch into master (formerly known as 'trunk')

   ```
   git merge fcrepo-780
   ```

   Resolve any conflicts, and test again.

8. Update master on github:

   ```
   git push origin master
   ```

9. Once you've received word that the build has completed correctly, delete the branch

```
git push origin :fcrepo-780
```

Creating and working with an unpublished branch (with *git rebase*)

1. Get the repository:

   ```
   git clone git@github.com:fcrepo/fcrepo.git
   cd fcrepo
   ```

2. Create the branch where you'll do your work:

   ```
   git branch fcrepo-780
   git checkout fcrepo-780
   ```

   The `checkout` command makes whatever branch you specify the local active branch. Make your changes, test...

3. Add your edited/new files, then commit your branch:

   ```
   git add myfile.java
   git commit myfile.java
   ```

4. Graft your changes (and commit histories) onto the end of master:

   ```
   git rebase master
   ```

   You may need resolve conflicts, then re-add and re-commit the merged files. If this is the case, you can pick up where you left off with the command `git rebase --continue`.
   The command `git rebase -i master` allows you to interactively edit, suppress, combine the commits in your branch, to eliminate non-useful or trivial commit messages in the final result.

5. Switch to the master branch, update it to the latest version:

   ```
   git checkout master
   git pull
   ```

6. Merge in the changes from your unpublished, rebased branch:

   ```
   git merge fcrepo-780
   ```

   Merges are automatically committed locally.

7. Update master on github:

   ```
   git push origin master
   ```

Examining and merging in upstream changes

If any time has passed since you began working on your local branch, make sure that you also merge any upstream changes to master into your local copy before pushing your changes back up:

```
git fetch origin master
```

Examine the changes:

```
git diff origin master
```

then:

```
git merge origin/master
```

If there were no conflicts, the merge will be automatically committed. If there are conflicts you will need to resolve them and then commit.
(The command `git pull` is a shortcut for doing a combined fetch-and-merge, but read this on why that's a bad idea.)