Git Etiquette RFC

March 2018

1 Key Terms

1. origin - alias for the remote address
2. HEAD - alias for the most recent commit on your current branch
3. rebase - changing the commit history
4. squash - combine multiple commits into a single commit

2 Goals for Different Branches

2.1 origin/master

1. This should be the clean copy of our code. It should be the case that checking out to any commit in this branch will give us a copy of our code base that is free of bugs (not including changes in API, of course). This means that commits should be modular.
2. Commit messages should be descriptive. It should tell us what changes were implemented in each commit and should note any changes in the API.
3. If merging code from a feature branch causes code base to break, make sure to repair the code before pushing to origin, and noting down what changes you made in your commit message

2.2 origin/feature/server

1. This should be the working copy of our server/business logic code. Code written by people working on the server code should be merged and tested thoroughly on this branch before promoting to master.
2. Before transferring code from your user branch to this branch, it is a good idea to make sure your changes work in isolation. Ideally, you should only use this branch to resolve incompatibilities that arise among different people that work on the server
3. While this need not be a clean copy, please also keep commit messages as descriptive as possible. For example, if merging code causes problems, note them down and note down how they are solved in the commit messages

2.3 origin/feature/web-app

Analogous to origin/feature/server

2.4 origin/user/*

1. These are your personal user branches. Do whatever you want.
2. Test your changes on this branch before promoting code to feature branch.

3 How to Move Code between Branches

The following is a recommended set of procedures for moving code around. Note, if you are promoting code to a different branch, please resolve merge conflicts locally and squash into a single commit before pushing to origin.

1. Call “git rebase -i” and squash all the code you want to move around into a single commit
2. Call “git format-patch HEAD 1..HEAD” to get a patch file
3. Call “git checkout branch-name” to shift onto the branch you are transferring to
4. Call “git am name-of-patch” to apply the changes
5. If there are merge conflicts, resolve them and squash into a single commit before pushing to origin.