SE101—`git` Assignment

Rollen D'Souza, Derek Rayside

Fall 2017
Last Updated October 18, 2017
Due : October 26, 2017 @ 1200pm

Introduction

As part of your demo, we would like you to perform the tasks listed in this document. It is expected that Q1 is completed in order to correctly assess the demo. Q2 will not be directly assessed; we trust you will complete it for your own learning purposes.

Relevant Readings

- Git Book (2.3) : Viewing Commit History
- Git Book (2.6) : Tagging

Q1 : Tagging

Recall that `git` retains the history of changes in atomic units known as commits. If you perform `git log` in the local copy of your repository, you will be able to see all the commits made to your repository. You will notice that these commits are associated with a sequence of letters and numbers that appear random. This sequence is a unique identifier for each commit and is known as a commit hash.

For example, consider the commit f58cce03a729cf8451a6d7013178eeb86352a66a in the `se101-notes` repository. If you perform `git checkout f58cce03a729cf8451a6d7013178eeb86352a66a`, your local files match that of the particular commit. This is extremely useful if, for example, you wish to inspect some older code without having to browse the online user interface.

Of course, no one can remember a large number of important commit hashes. Tags allow us to give clean names to an important commit in the development history. In practice, these are often used to give names to specific changes that mark a particular complete version of software. This allows users to clone a repository and checkout a given version without taking any future changes made to the project. In the lab, we would like you to tag the version of your software that you wish to submit, prior to the deadline.

Lets say you committed a working prototype and you now wish to continue working on the project without ruining the prototype. You first should execute `git log` or view the commit history online to find out what the commit hash of the commit associated with your complete prototype. Then you will perform `git tag lab-prototype COMMIT_HASH` which will give the name `lab-prototype` to that commit. Finally you will push that tag using `git push origin lab-prototype`. We will ask you, in your demo, to checkout this particular version when demoing.

*WARNING: You **do not** have permissions to replace the tag on the server once you have pushed it. **Do not** push the tag until you are **certain** that that is the submission you wish to provide.*

Q2 : Merging

In order to ensure you are using `git` effectively, we would like you to perform the following tasks with your group members at the same time, in each others' presence, in the following order:

1. Every group member must create a file named mygroup.txt in *their copy* of the group repository with their username in it. You must do it on your own computers so that every person has a file that has different content, but the same file name.

2. Every group member creates a commit for that file.

3. One group member must push their commit (`git push origin master`)

4. Another group member must perform a pull (merging the others content) (`git pull origin master`), creating another commit, and push their changes.

5. Repeat (4) until the repository has a file `mygroup.txt` that has all members names.