Thomas Gibbons

October 7, 2016

ECE 3220

LAB 3

Objective

The objective of this lab was to learn about version control. We got familiar with version control using Git. We learned the basics of Git and how to push our repository online.

Discussion

We utilized many Git commands throughout the lab including “git init” where we started a repository within a directory. From there we used “git status” to check on all files modified or not added to the repository yet. This is an example (example 1-git status) from the next lab.

```
C:\Users\Thomas\Desktop\C code\LAB4>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   lab4.c

no changes added to commit (use "git add" and/or "git commit -a")
```

When we saw “.c” files not added we added all of them since there was nothing to exclude. We then could use “git commit” with “-m (message)” to create first version. After committing we can use the “git log” to check on different times you committed the work (Example 2- git log).

```
commit 0566aa3555b02d178bc49f15f8f330cc38a2dd36
Author: Thomas Gibbons <gibbons.thomas95@gmail.com>
Date:   Tue Sep 20 18:27:53 2016 -0500

    v1.3

commit b991a3c570bb6fcf7023d3cfaf15f13964145104
Author: Thomas Gibbons <gibbons.thomas95@gmail.com>
Date:   Tue Sep 20 17:18:19 2016 -0500

    v1.2

commit 2f0de8cad7645486413224cccb61b7cda3214089
Author: Thomas Gibbons <gibbons.thomas95@gmail.com>
Date:   Tue Sep 20 17:14:24 2016 -0500

    v1.1

commit ae49e02d68f86b18e9a01e47a0bee0f772205838
Author: Thomas Gibbons <gibbons.thomas95@gmail.com>
Date:   Tue Sep 20 16:44:55 2016 -0500

    v1.0
```

We could modify the code and check “git status” and will result in example 1 again. We used “git diff” to show us what line was modified in the code. We then can add and commit again with a new message for example “v1.1”. We can also undo changes. After we made changes to our code, we can check “git status” and it shows the changes. Instead of committing, we can undo the changes by using the “git checkout -- <file changed>” command. This undoes what was changed and does not commit it. When we check the “git log”, we see there are no new changes. We explored the “git show” command and it shows more than just “git diff” does as it shows all various types of objects.

We also used GitHub to push our repositories online for storing versions online and others to see our code. The link to my GitHub account is http://github.com/tgibbons95.

After creating an online account, we created a new directory. We used the “git clone” command and copied the directory from GitHub to our machine in the new directory. We then added a new file, added it and committed it. And updated it online by pushing to GitHub.

We then went back to the original directory on our computer and added a new file. We added and committed it as well. Next, we used the “git pull” command and then “git push”. What this did is pulled the updated directory from online with the file from the other local directory before we pushed our update within the current directory. You pull before you push because had we not pulled we would have overwritten the directory and lost changes made in the second directory. When we check the log though we see that there was a merge that happened. This happened because we were working from two directories so when we pulled it merged online with the local directory.

We then learned how to use the “git tag” command which shows you all the tags currently on the branch. We can also use it to add a tag for example by putting “git tag v1.0”. This adds the v1.0 tag and can be viewed by using “git tag” again. We can then make changes, add, and commit. Then we can add a new tag for example v1.1. We learned to use the checkout command again with tags to go back to points we tagged. This could be useful if v1.1 was not working at all and you wanted to go back to v1.0. After using “git checkout v1.0” it brings all the files to where they were when you made the tag.