**Lab 5 (plus HW)**

Oakland University / CSI3660

Fall 2018

**Due November 12th @ 11:55pm**

This lab is intended to provide you experience with:

1. Installing and using Git

**LAB PROCEDURE**

One important aspect of any project is **revision control**.

1. **Install and configure Git on your server.**

This one is fairly easy.

**$ sudo yum install git-all**

This will install git on your server. Fairly straightforward process. Now, let’s customize our install.

The command to do this is **git config.** This command allows you to modify various aspects of git.

First, set your global git user. I will use CIT348 as the username, but you can really use any user. You can use your own username, since right now you’ll be the only one accessing git on your system:

**$ git config --global user.name "CSI3660"**

**$ git config --global user.email CSI3660@csi3660.com**

This should set up our global user, where global impacts all users of your system.

Next, set your default editor. This will open up the text editor of your choice when you edit files or commit changes without a message. This will be either nano, vim, or emacs. I will show an example for vim, but you would replace vim with your editor of choice.

**$ git config --global core.editor vim**

Now, check your settings with the following command to make sure that they are valid. **Copy this output or take a screenshot for your lab report**.

**$ git config --list**

This will show the configured settings.

1. **Create a Git repository to manage your course project**

Creating a git repository is fairly easy. Make a new folder in your home directory called CSI3660-Lab5, and enter it.

Create an empty git repository:

**$ git init**

This will create a .git folder. **For your lab report, show me that this folder was created (hint: this folder is hidden by default). Don’t show me the git init command, show me that it exists on your filesystem.**

Now move some files in there for tracking. On your VMs you \*should\* have an index.php from the last lab. Copy it over. (Note, if you didn't give it a .php extension then it might be .html instead).

**$ cp /var/www/html/index.php .**

Then add it to version tracking.

**$ git add \***

Let’s check the status of our repository to see what’s going on. **Copy/paste this output for your report**:

**$ git status**

We then need to “commit” this change to git, otherwise it will hang in limbo. For this, we should also add a good description of what this commit means for future employees who check out this version.

**$ git commit -m "Initial commit for CSI3660"**

You’ve just committed your first revision to the git repository. Congrats…you can now always go back to this current state. **Run git status again, and for your report, tell me what the difference is from the last time you ran it.**

Let’s add a new file now to show the difference between tracked and untracked files. Create a docs folder inside the CSI3660-Lab folder, and inside that folder should be a file named TODO

So your directory should look like:

CSI3660

docs

TODO

In this file, write a few items that need to be done in order for your term project to be successful. Think of this file as a checklist of your items that need to be done. This file might be useful in distributing the workload if you’re on a team.

**Run git status again.**  **For your report, show me this output and describe what the difference between a tracked and untracked file is**.

Now, let’s add that file. Do it as you did before, but now just for a single file:

**$ git add docs/TODO**

**$ git commit -m "Added the TODO file"**

Lastly, let’s see how we can move and rename files. Git does not explicitly track any metadata changes to files (renaming, moving, etc.). If you want to move or rename a file and have Git track that change, then you need to use the git mv command.

First, try just renaming the file:

**$ mv docs/TODO docs/TODO.txt**

**$ git status**

**Copy and paste the output from that command. What does git think happened here?**

**Now, move it back:**

**$ mv docs/TODO.txt docs/TODO**

**$ git status**

**Copy and paste this output. What happened?**

Finally, let’s try renaming a file in the git fashion. But, let’s make sure that all of our files are named in a similar fashion

**$ git mv docs/TODO docs/Project\_TODO**

**$ git status**

**Copy and paste the output. What has happened here that is different from before?**

Now, commit the change using the appropriate command. By now, you should remember how to commit in git. Commit the change with an appropriate status message.

Now, let’s check the commit history. **Copy and paste this output into your report**:

**$ git log**

1. **Work with git from another directory**

Let’s now see what cloning a git repository does for us. Navigate to your home directory and clone the git repository there. This will pull down a working copy of that project so you can work with it in a different directory:

**$ cd /tmp**

**$ git clone /home/<your username>/CSI3660-Lab5**

You now have a clone of your project in your local directory. Navigate into the newly created CSI3660-Lab5 in your home directory and **show the log**:

**$ git log**

You now have a working repository in your home directory. If you make any changes and commit, those changes will be reflected within /home/<your username/CSI3660-Lab5.

1. **Term Project Checkpoint**

It is now time to make some forward progress on your term project, if you have not already. Based on your project overview you described on your team website, do the following.

1. Select which team member will host the official website, and which team member will host the service. Remember that the website should be on your Google Cloud microinstance, and your service can be hosted wherever you prefer (on the school VM might be easiest).

**For this, provide the name of the team member and the IP address of their respective VM for that role.**

1. Based on where you currently are with your project, perform one task related to moving it forward. If you have not started yet, try to install a basic requirement (e.g., download and try the Minecraft server, install the TF2 service, etc.). Look at Homework Question 6 for more details.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**HOMEWORK**

For this assignment, you’ll need to do some Googling. For these questions, please include your references that you used to answer the questions.

1. **What exactly is Git and why is it important?**
2. **What is the purpose of *tagging*? Look up the command for creating a tag. Add a tag for your current commit on your virtual machine, naming the tag v0.1 (remember that you need to be in /home/<username/CSI3660-Lab5 or in a cloned repository). Show me that the tag was successfully created (using the command for *listing* tags, not *creating* tags).**
3. **How does Git differ from more standard types of source control (e.g., Subversion)?**
4. **What is the purpose of branching and merging?**
5. **Why is the addition of a *staging area* helpful in Git?**
6. **As promised, you will now kickstart your course project if you haven’t started it yet. For this question, you need to demonstrate that you have at least part of your project working on your server. For example, if you were setting up an ownCloud server, you could show me that you installed ownCloud on the system and take a screenshot of it minimally working. Or, if you were running a Minecraft server you could at least show me that you compiled the server and can execute it from the command line.**
7. **In a short paragraph, discuss what the next step of your project is. Basically, Question (6) made sure that your service is at least started. What is the next step to success?**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**LAB REPORT REQUIREMENTS**

Copy and paste this document into a new document, but remove all text except everything in **red and bold.** For each of those points, put the output that was requested, and discuss any points that were asked about. For instance, if I ask you to ‘show me that a file exists’ then you need to use the appropriate **ls** command to do so.

Make sure you answer the homework questions as well. You can submit it all in one document.