Lab Assignment 13, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Version Control System, Git, and GitHub

This is a bonus lab assignment where you can get 70% of the grade by creating an account on GitHub.com, creating a project on Eclipse and submitting the link to your remote repository. Rest 30% you can get by following the step-by-step process described in this document. I intentionally kept it simple so that you can get done with it fast and focus on the project due this Friday, May 1. Remember that, there will not be any extension to the project 4.

Pre-Lab Lecture Video on Git

You must watch this pre-lab lecture video on git before you start working on this assignment. This video lecture shows the activity or step-by-step process of this lab assignment.

https://miamioh.instructure.com/courses/117194/discussion_topics/880330

Introduction:

When you are working on a term paper and make a single tiny change it rarely wrecks the paper. But software developers and programmers have a very different experience: A single small change or stray character can turn a working system into a big pile of garbage. As a result, most successful developers and designers become fastidious and paranoid about saving older versions of their work … every time they make even a small piece of progress, they save a copy before making the next change.

Version Control System:

The purpose of a **version control system** or (aka **source control system** or **revision control system**) is to automate the process, to make the saving and organizing of small increments of progress easy. These systems often allow you to **revert** to previous versions, or to compare your current version with a previous one in a way that makes the changes easy to see. In systems where multiple people are working together the systems provide **attribution** so that we know who made each change, and why. What is version control? Watch: https://git-scm.com/video/what-is-version-control

Today we are going to learn the basics of a **version control system** called **git**. git was created by **Linus Torvalds**, who is best known for creating **Linux** operating system.

Step by Step worksheet

Install git

1. Make sure that Git is installed on the machine you are working at. You can download and install Git the software for Windows, Mac OS and Linux machines from here: https://git-scm.com/downloads

   Read this link install github [it has three tabs for mac, linux and windows] or watch the following videos to see how to install for different operating systems:

   - Windows 10: https://www.youtube.com/watch?v=nbFwejIsHlY
   - Mac OS: https://www.youtube.com/watch?v=sJ4zr0a4GAs&

Create A GitHub Account

2. We are going to host a remote repository from GitHub website. What is GitHub?
3. If you don't have a GitHub account yet, go to the Git Hub website and create an account for yourself --> https://github.com/ with your Miami Email address. If you already have an account, then you don't need to create one. Log in to your account on https://github.com.

   Miami University CSE department has its own version control system, GitLab Enterprise Edition, available at https://gitlab.csi.miamioh.edu. You can use the ***LDAP*** tab to provide your Miami username and password to sign-in. Today we are going to use github.com which is publicly available for everyone.

Create A Remote Repository on GitHub

4. Click on the button that says "**New Repository**" in the top right corner (the icon that says + with a drop-down arrow). This repository is going to be publicly visible to everyone.
5. Give it the name "***MyFirstRepo***" and make it Public. Do **NOT** check the box that says "Initialize this repository with a README"
6. Click on the **Create repository** Button

Create A Project on Eclipse

7. The next screen is VERY important as it gives you some of the commands you will need AND it tells you where you repo is located THUS LEAVE THIS WEBPAGE OPEN WHILE YOU DO THE NEXT FEW STEPS in Eclipse.
8. Go into Eclipse and create a new Java Project called "GitTesting", create a new class in it called HelloWorld. Put the following code in the HelloWorld Class:

```
public class HelloWorld {
         public static void main(String[] args) {
                 System.out.println("Hello World!");
     }
}
```

9. Run the code and make sure it works.
10. While in Eclipse, right click on the Project and select Properties. Figure out the location of the project (the directory with the drive that everything is stored in)

Open and Use Git Application or Terminal

11. Based on the operating system installed on your computer, do one of the following:
    - WINDOWS USERS: Go to the Start Menu and open **Git Bash** --> All Programs>Git>Git Bash. At this point you are opening a little window that accepts Unix type commands on a Window machine.
    - MAC/OSX USERS: Open the terminal application. You can look for it on the spotlight search.

- LINUX (or Ubuntu) USERS: Open the terminal application.
- ALL USERS: NOTE: if at some point, you get stuck in the Git bash window and it keeps beeping, read this article: http://stackoverflow.com/questions/3923596/git-windows-command-prompt-gets-stuck-during-git-commands-with-end (It tells you to push Q to get out of it)

12. For the first command, type ***cd drivename***: and hit enter to move to the drive that contains you eclipse workspace. For example, cd c:

13. Then type ***cd foldername*** to move to the location that the project is stored in. For example, ***cd eclipseworkspace/GitTesting*** (Note: the slash will be the opposite direction here, use this forward slash / to separate folders)

14. Type ***ls -l*** to see the list of files and directories in your project folder.
    Notice your src and bin directories

15. Type ***git status***
    Notice that git tells us that “Not a git repository”, that is, we haven't set up the repository yet. git status is a very useful command that will tell us what git thinks is happening. Use the "***git status***" command often to check that current state of the repository.

16. Now type ***git status -h***
    For any command where you need more info, just type git command -h (e.g. git commit -h)

Set git and Create A Local Repository on Your Computer

17. We need to do a little setup before we create any local repositories so that our name and email will be associated with the commits (A commit is when you change the code and send it to the repository).

    - type ***which git***
      this will tell you where your system thinks git is installed
    - type **git config -l**
      this will tell you the default values. We want to set the user.name and the user.email
    - type **git config --global user.name "FirstName LastName"**
      type your first and last names
    - type **git config --global user.email your@email.address**
      type your email address
    - type **git config -l**
      to make sure it all worked!
18. Let’s create a **LOCAL** repository on our machine to store this project. (We will create a repo in the same location that contains the project, it will NOT move it up to GitHub yet).
    Type **git init**
    git init is the command to initialize a new repository.

19. Again type **git status**
    Notice that there is now a repo and it is telling you that it can see "untracked" files, in other

words, there are files (and folders) that you haven't put in your repo yet, let's now add the files and folders:

20. Now type **git add .**
    This command will "stage" all the files (that means make them ready to go into the repository). If you just want to get some files ready, for example HelloWorld.java, then do
    git add src/HelloWorld.java

21. Now type **git status**
    Notice that the files are now ready to be committed to the local repo

22. We probably don't want to keep track of binary files (such as bin/HelloWorld.class), so let's set up our repository so that it will not keep track of those files. In the same directory as your GitTesting project, create (using new→File in Eclipse or using Notepad) a file called .gitignore (IT MUST BE CALLED .gitignore and in the same directory where you created your local repository). Put the following into .gitignore:

    ```
    # Ignore my bin folder
    bin
    ```

    and save the file.
    - Now type **git add .** and the type **git status**
    - Notice that /bin is still ready to be committed. To fix this you need to remove it just once as follow, type: **git rm --cache -r bin**
    - Now type **git status**
    - Any files or folder that you want to ignore, just add them to the bottom of the file called **.gitignore** (NOTE: this is also a good place to tell it to ignore your .pub key you created (if that key is in the same area as the files going into your repository) so that just in case you have a public repository, others can't see your public key and use it to mess around with your repository).

Commit the Changes You Made

23. Now type **git commit -m "My first commit in Git"**
    We are now adding the files to the local repo with a comment. We should add a comment for every commit to tell our group members what we were doing.

24. Now type **git status**
    Notice that everything in the repo. The repo now has a branch called master. Think of master (or a branch) as one version of your repository, someone else might "Branch off" from the master code and make another version and then you and this other person will "merge" your versions together. REMEMBER the local repository is initially called master

Push Committed Changes to The Remote Repository on GitHub

25. Now we need to tell our local repo which repo on GitHub it will be associated with, i.e. we need to make a link with GitHub between our remote repo and our local repo. This is when we need to go back to our webpage that we left open that contains the name of repo. Right under where it says Quick setup you will see a box next to the buttons HTTP and SSH, make sure the SSH

button is clicked, you should see something in the box like:
git@github.com:username/MyFirstRepo.git,

copy the contents of the box. We are going to give our remote repo the name of "origin" on our local machine (this is just some GitHub conventions but you could call the remote repo anything you wanted on your local machine) and then connect origin to our GitHub repository. Go back to your Git Bash window on windows and type the command:

git remote add origin git@github.com:username/MyFirstRepo.git

26. Now type **git status**
Notice that nothing has really happened. All we have done in the previous step is set up the link between our local repo and the remote repo. But we still need to "PUSH" our code up to the remote repo. The command: **git remote** will show you the name of your remote in case you forget!
27. Now type **git push -u origin master**
This is trying to put your local repository (the master) onto the remote one (that you called origin) Notice that you are getting a Permissions error. This is because GitHub is not sure if you are who you say you are. You could just keep signing in and out of GitHub with a password, but this would get to be a pain, so instead you can generate a key pair. This is a set of two keys that "fit" together or go together. Every time you try to push stuff up to GitHub, GitHub will check if the key on GitHub FITS with the key on your local machine. If they match, you will be allowed to push and pull from the remote repository. First, we need to generate two keys.

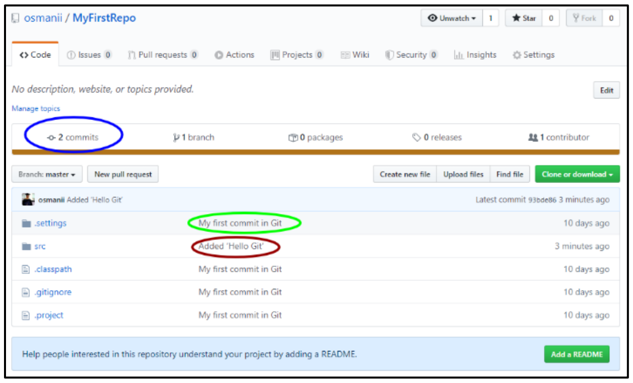
Generate and Add Public and Private SSH Keys

28. In Git Bash type **ssh-keygen -t rsa**
- Use the default file folder to save the key in (just hit **enter**)
If you get a warning that you are going to overwrite a file, select n and go to the next step.
- Hit **enter** for the passphrase (NO PASSPHRASE) (you don't really need to set one and it is just more passwords to remember, so don't bother setting one)
- Hit **enter** again
What you just did was generate two files: one called id_rsa (the private file that says on your machine) and one file called id_rsa.pub (the public file that you copy the contents of it up to GitHub)
29. **cd** to the drive where the keygen was placed and then **cd** to the directory where the above command stored your keys. (probably ~/.ssh). For example the location on a Windows machine is: c:/Users/YourUsername/.ssh/
30. type **ls**
31. The file called **id_rsa** is the private key (the local key) and the one called **id_rsa.pub** is the public key (the remote key)
32. Using Notepad or some editor, open the file called **id_rsa.pub** and copy the contents of it into your clipboard.
- You can use command like "**start id_rsa.pub**" or "**vim id_rsa.pub**" or "**vi id_rsa.pub**" to open the file in edit mode. Select and copy the content of the file. If the file has been opened in the GitBash or terminal then write "**:q**" to exit the editor. You can also open it using the notepad or other text editors using the command "**notepad id_rsa.pub**".

33. We now have to move the contents of the clipboard onto GitHub. Go back to the webpage for GitHub. Click on the icon in the top right with the profile picture and it will show a list of options once your click. Now select the option "**Settings**" and then "**SSH and GPG keys**". It should take you to the following url: https://github.com/settings/keys
34. Click on the green **New SSH Keys** on the top right
35. Give the key the title "MyWorkspaceKey" and past the contents of the clipboard into the key box below. Now click on the green **Add SSH key**

Push Committed Changes to The Remote Repository on GitHub [After Adding SSH Key]

36. Go back to Git Bash on Windows and move back (cd) to the directory where your repo is. You can type "**cd /c/Users/YourUsername/eclipse-workspace/GitTesting/**" on windows machine.
37. Type **git push -u origin master**
38. NOTE : you might get an error here because of simple vs. matching options for the push command. If you do get that error, type this command: **git config --global push.default simple**. Then try it again if there was an error. You will see messages related to file being uploaded to the remote git repository.
39. Go back to the GitHub webpage , in the "**Setting**" page , click on "**Your Repositories**", (right under "**Your Profile**"), then click on your repo link and you should see all your files there now (except for bin which was ignored).
40. Now, go back to the main method in the HelloWorld.java on Eclipse. Add a second print statement after the first one: `System.out.println("Hello Git!");`
41. type **git status**
    Notice that there is now a modified file that has "Changes not staged for commit:". That is, there is a modified file that you haven't staged, let's now add the modified file:
42. Now type **git add src/HelloWorld.java**
    This command will "stage" the modified file (that means make them ready to go into the repository).
43. Now type **git status**
    Notice that the modified file is now ready to be committed to the local repo.
44. Now type **git commit -m "Added 'Hello Git'"**
    We are now adding the modified file to the local repo with a comment. We should add a comment for every commit to tell our group members what we were doing.
45. Now type **git status**
46. Now type **git push -u origin master**
    This is going to put your committed changes in the local repository (the master) onto the remote one (that you called origin).
47. Now go back to your repository on github.com. You will see something similar to the Figure 1 below. The blue circle shows that there are two commits in this repo. You can click and see the list of commits. Also, you can see, in green and red circles, the files/folders that were changed for different commits. If you select a commit, then it will show the changes made, addition and deletion, to associated with that commit, as presented in Figure 2.



osmanii / MyFirstRepo

No description, website, or topics provided.

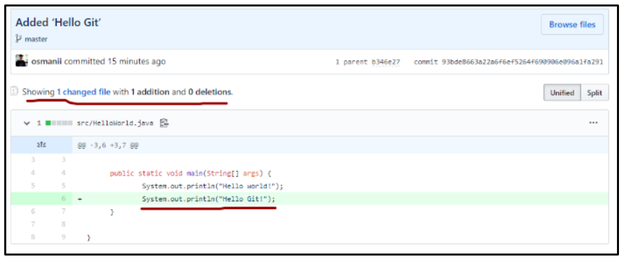
2 commits · 1 branch · 0 packages · 0 releases · 1 contributor

osmanii Added 'Hello Git' — Latest commit 93bde86 3 minutes ago

.settings	My first commit in Git	10 days ago
src	Added 'Hello Git'	3 minutes ago
.classpath	My first commit in Git	10 days ago
.gitignore	My first commit in Git	10 days ago
.project	My first commit in Git	10 days ago

Help people interested in this repository understand your project by adding a README.

Figure 1: Commits



Added 'Hello Git'
master

osmanii committed 15 minutes ago — 1 parent b346e27 commit 93bde8663a22a6f6ef5264f690906e096a1fa291

Showing 1 changed file with 1 addition and 0 deletions.

```
src/HelloWorld.java
@@ -3,6 +3,7 @@

        public static void main(String[] args) {
                System.out.println("Hello world!");
+               System.out.println("Hello Git!");
        }

 }
```

Figure 2: Changes associated with a commit

Submission:

Please submit the link (URL) of the remote repository you created on https://github.com. Also, please submit username associated with your github.com account. You can write this information on Canvas or write in a text file and submit the text file on Canvas. We will verify your work by visiting the link to your remote repository. If we don't see any committed changes to your repository then you will receive 70 out of 100 assuming you did install git, created a github.com account, an eclipse project and a local repository.

Grading Rubric:

Task	Points
Install git	20
Create GitHub Account	30
Create A Project on Eclipse	
Create A Remote Repository on GitHub	
Open and Use Git Application or Terminal	20
Set git and Create A Local Repository on Your Computer	
Commit the Changes You Made	30
Push Committed Changes to The Remote Repository on GitHub	
Generate and Add Public and Private SSH Keys	
Push Committed Changes to The Remote Repository on GitHub [After Adding SSH Key]	

Review for initial creation of Git Hub and Git repository:

You don't have a repository locally or on Git Hub:

1. Create a new repository on Git Hub, do NOT check the box that says "Initialize this repository with a README"
2. On the local machine, move to the area where the files/folders are that you want to put in the repository and do the following commands on the local machine:

```
git init
git add .
git commit -m "first commit"
git remote add origin git@github.com:yourgithubname/reponame.git
git push -u origin master
```

You have an existing EMPTY repository on Git Hub:

1. On the local machine, move to the area where the files/folders are that you want to put in the repository and do the following commands on the local machine:

```
git init
git add .
git commit -m "first commit"
git remote add origin git@github.com:yourgithubname/reponame.git
git push -u origin master
```

You have an existing repository on GitHub that contains files that you want to copy to your area [Something we did not discuss above but you might find it helpful later]:

1. On the local machine, move to the area where the files/folders are that you want to put in the repository and do the following commands on the local machine:

```
git clone git@github.com:yourgithubname/reponame.git
```

2. NOTE: if you want to clone into an existing empty directory (such as the src directory created by Eclipse for a Java project), on the local machine, move to that directory and type the following:

```
git clone git@github.com:yourgithubname/reponame.git .
```