Name: Aquino, John Kennedy A.	**Date Performed:** August 30, 2022
Course/Section: CPE31S22	**Date Submitted:** August 30, 2022
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem, 2022-2023

Activity 2: SSH Key-Based Authentication and Setting up Git

1. **Objectives:**

1.1 Configure remote and local machine to connect via SSH using a KEY instead of using a password
1.2 Create a public key and private key
1.3 Verify connectivity
1.4 Setup Git Repository using local and remote repositories
1.5 Configure and Run ad hoc commands from local machine to remote servers

Part 1: Discussion

It is assumed that you are already done with the last Activity (**Activity 1: Configure Network using Virtual Machines).** *Provide screenshots for each task*.

It is also assumed that you have VMs running that you can SSH but it requires a password. Our goal is to remotely login through SSH using a key without using a password. In this activity, we create a public and a private key. The private key resides in the local machine while the public key will be pushed to remote machines. Thus, instead of using a password, the local machine can connect automatically using SSH through an authorized key.

What Is ssh-keygen?

Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

SSH Keys and Public Key Authentication

The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Task 1: Create an SSH Key Pair for User Authentication

1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.

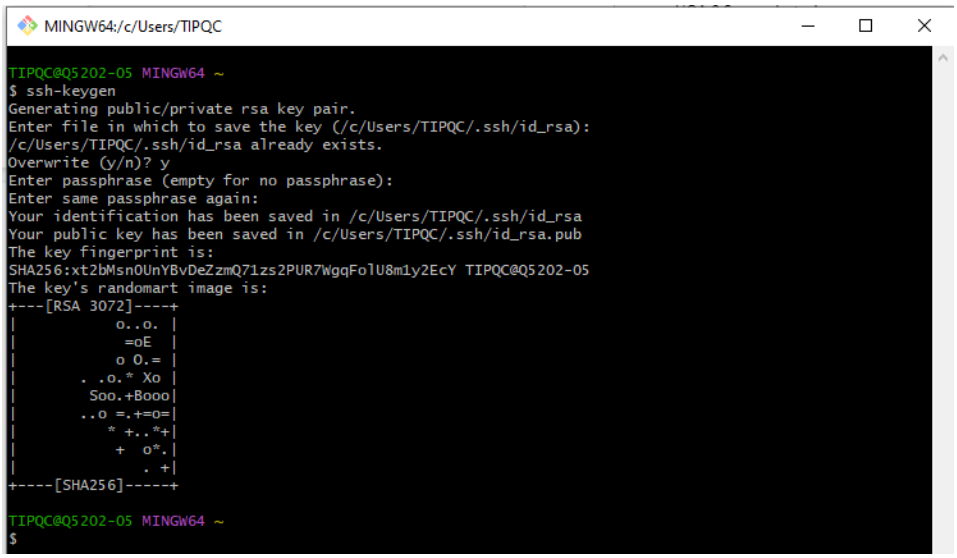


Figure 1: Running ssh-keygen

2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.

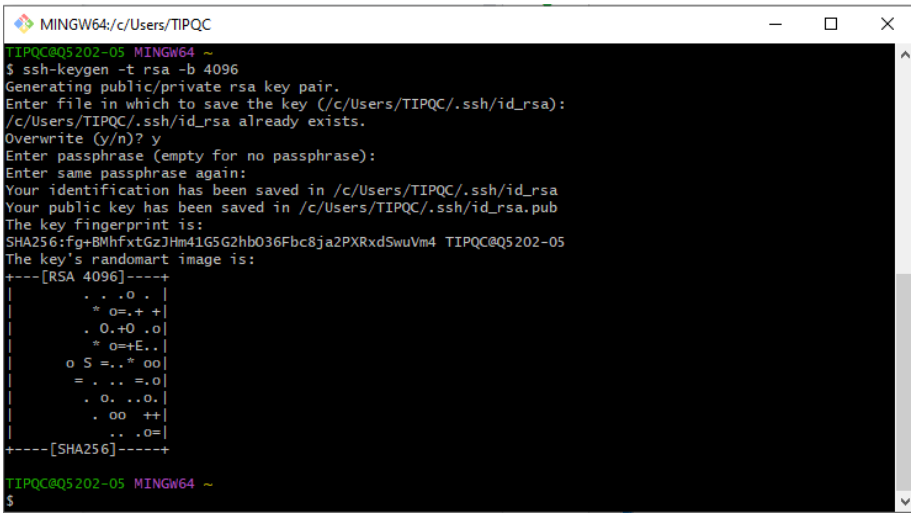


Figure 2: ssh-keygen -t rsa -b 4096 command issued

3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.

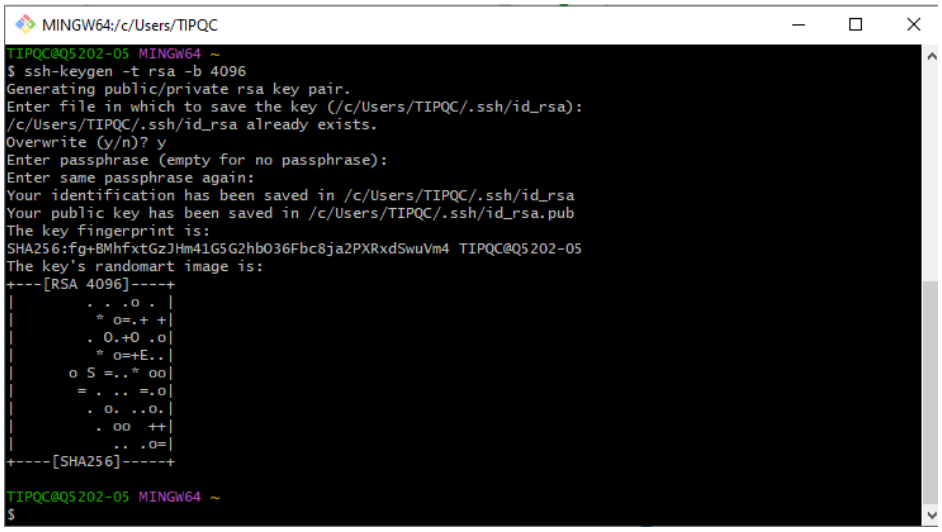


Figure 3:Passphrase for key encryption

4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.

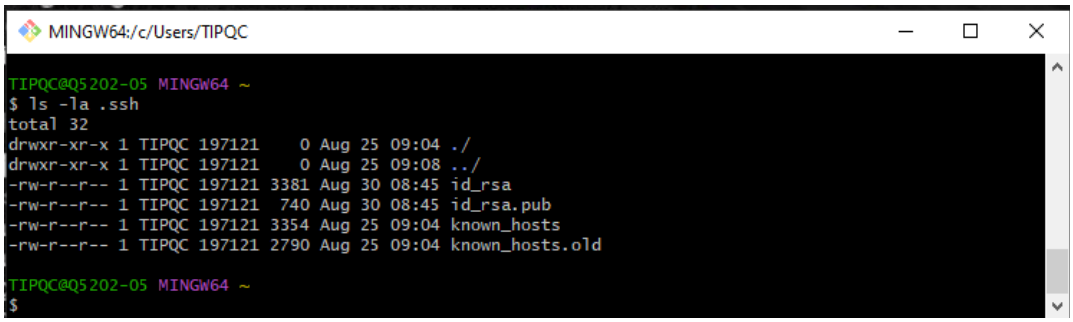


Figure 4: Verification of the key

Task 2: Copying the Public Key to the remote servers

1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

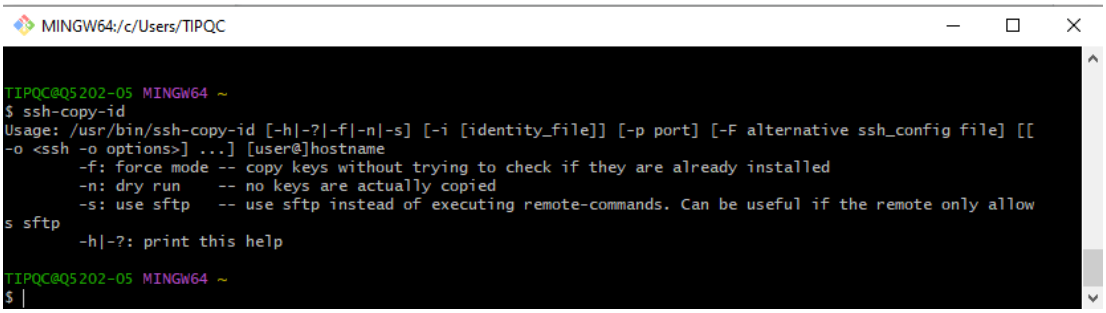


Figure 5: ssh-copy-id options

2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*

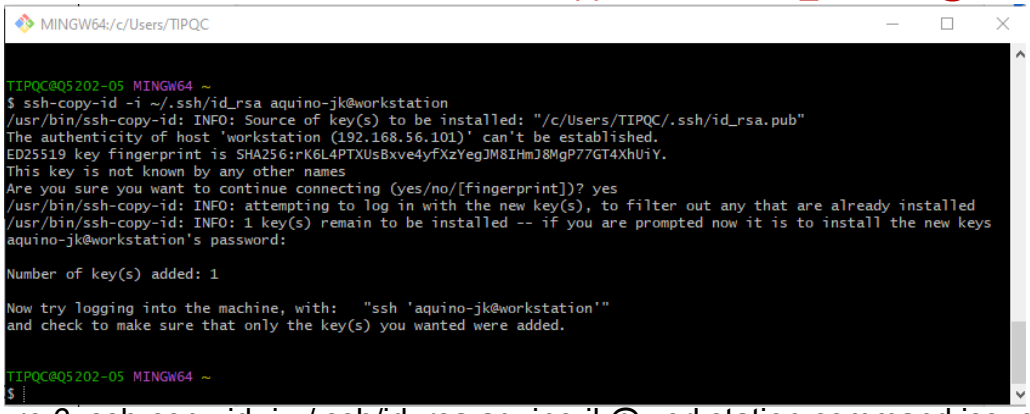


Figure 6: ssh-copy-id -i ~/.ssh/id_rsa aquino-jk@workstation command issued

3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.

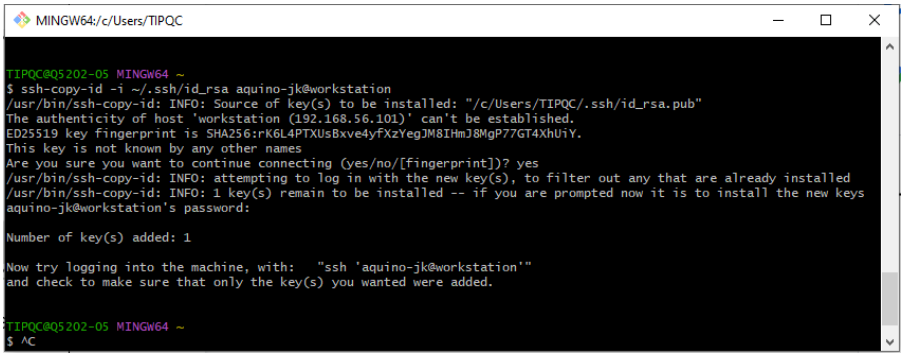


Figure 7: Login Process

4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

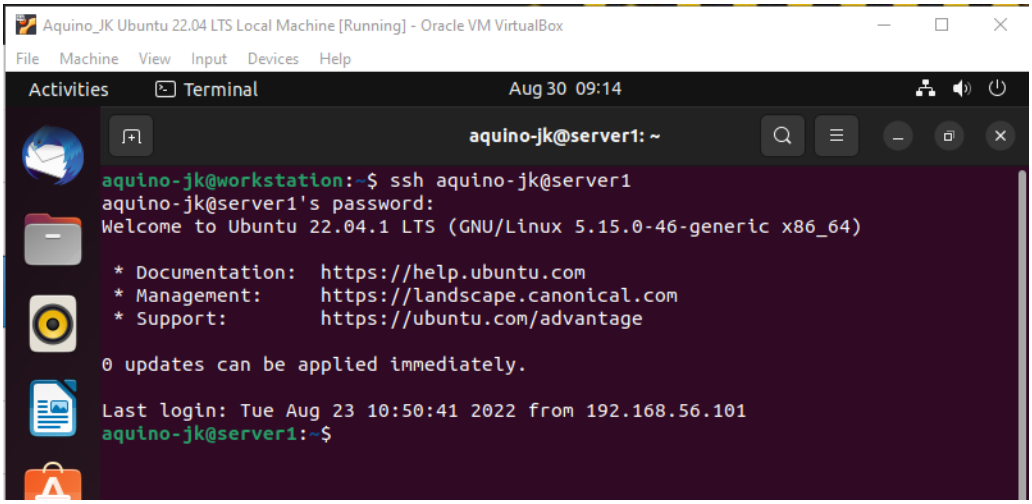


Figure 8: Verification of SSH connection with server 1

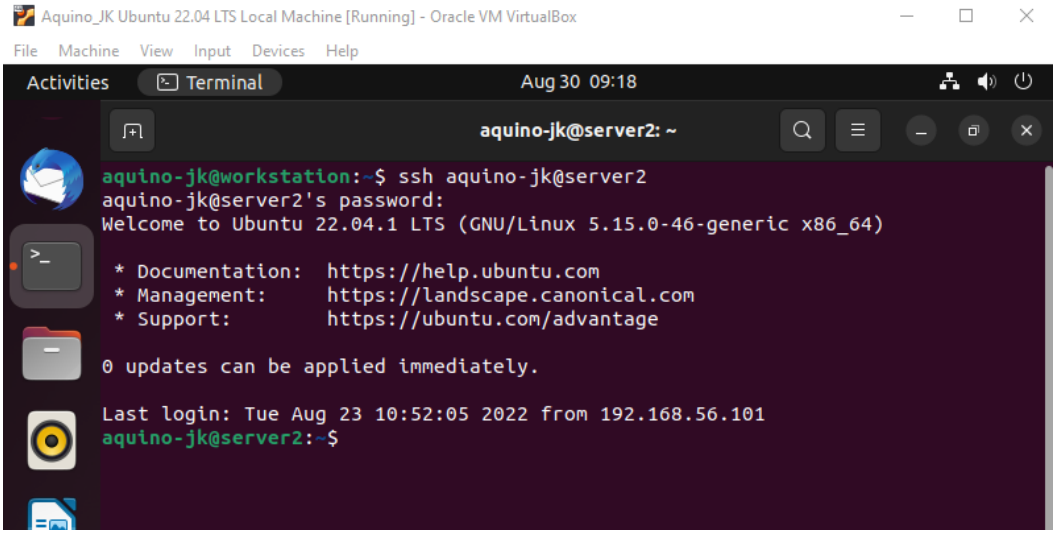


Figure 9: Verification of SSH connection with server 2

I have noticed that after verifying the connection for servers 1 and 2, we can see that the connection is verified since both servers can be connected with the workstation and we can see the login history. Yes, the connection asked for a password.

Reflections:
Answer the following:

1. How will you describe the ssh-program? What does it do?
   - Secure Shell or also known as the SSH Program is a protocol which is designed as a secure alternative for the remote shell protocols that are unsecured. It does provide greater security by means of authentication instead of just using a password.

2. How do you know that you already installed the public key to the remote servers?
   - In order for us to know or verify that we have installed the public key to the remote servers, we have to issue the command ssh username@machine.

Part 2: Discussion

Provide screenshots for each task.

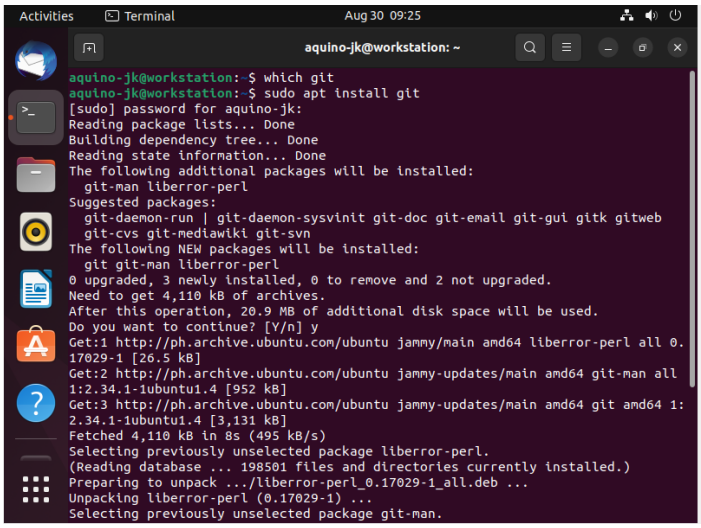
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

Set up Git
At the heart of GitHub is an open-source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer. To use Git on the command line, you'll need to download, install, and configure Git on your computer. You can also install GitHub CLI to use GitHub from the command line. If you don't need to work with files locally, GitHub lets you complete many Git-related actions directly in the browser, including:

- Creating a repository
- Forking a repository
- Managing files
- Being social

Task 3: Set up the Git Repository

1. On the local machine, verify the version of your git using the command *which git*. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*



2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

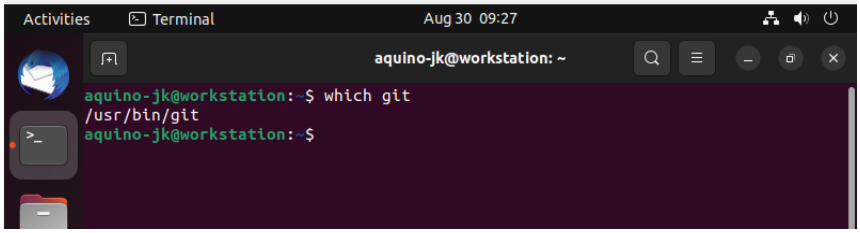


Figure 11: command which git issue

3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

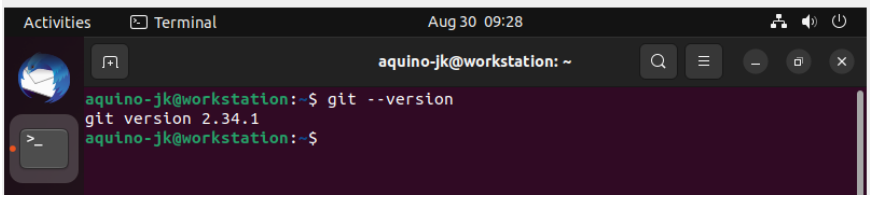


Figure 12: Version of git installed

4. Using the browser in the local machine, go to www.github.com.

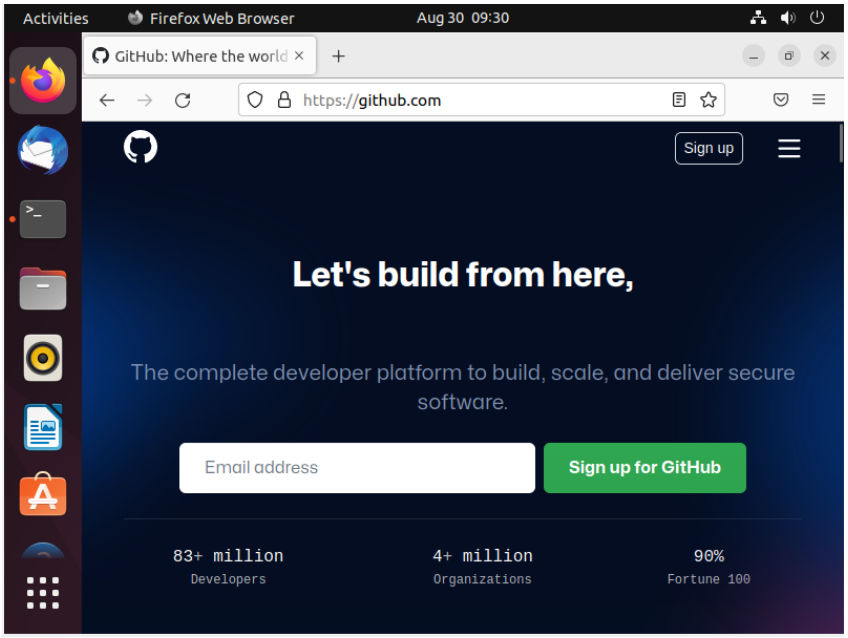


Figure 13: www.github.com website

5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
    a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

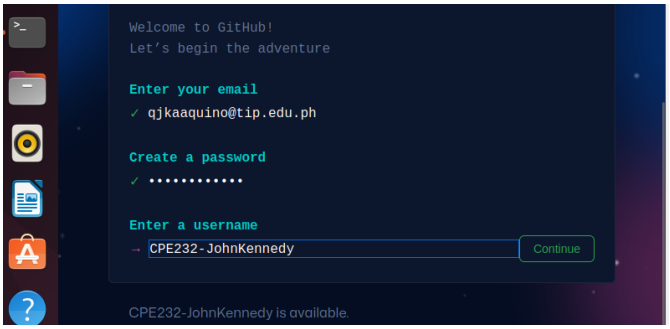
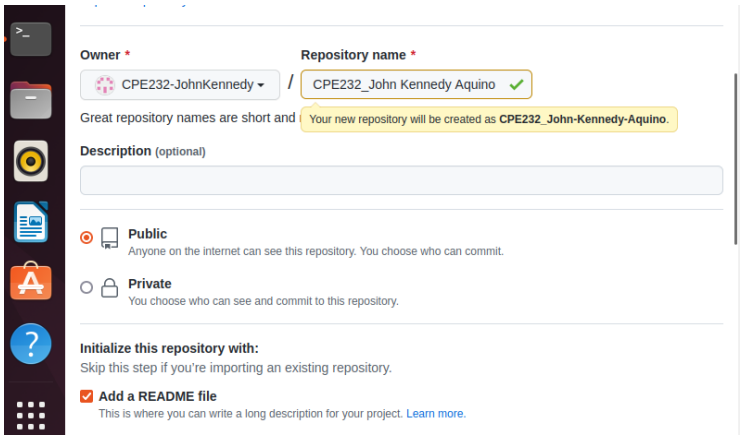


Figure 14: Creating an Account



Figue 15: Repository name and checked add README file

b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

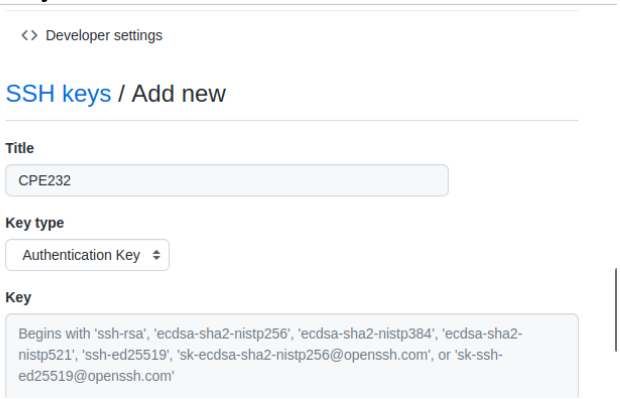


Figure 16: New SSH keys

c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.

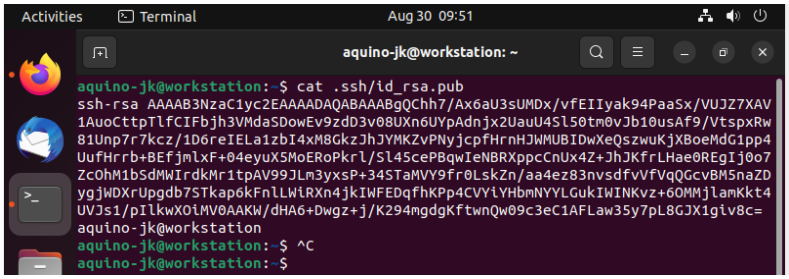


Figure 17: cat .ssh/id_rsapub command issued

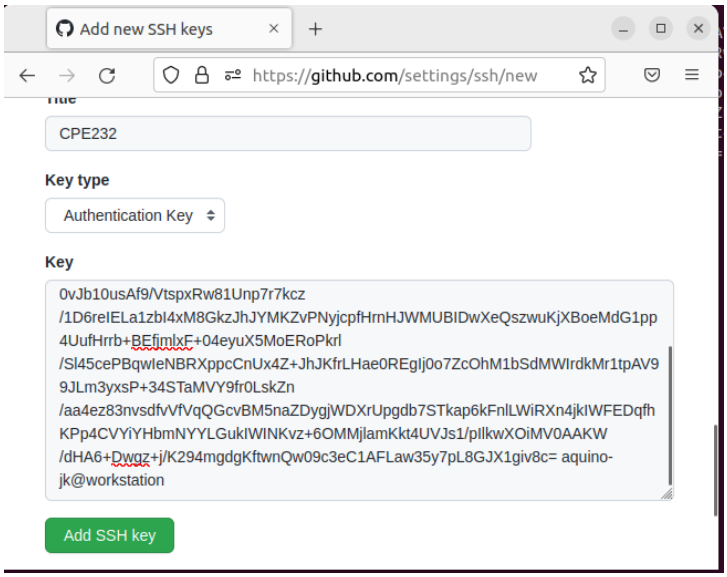
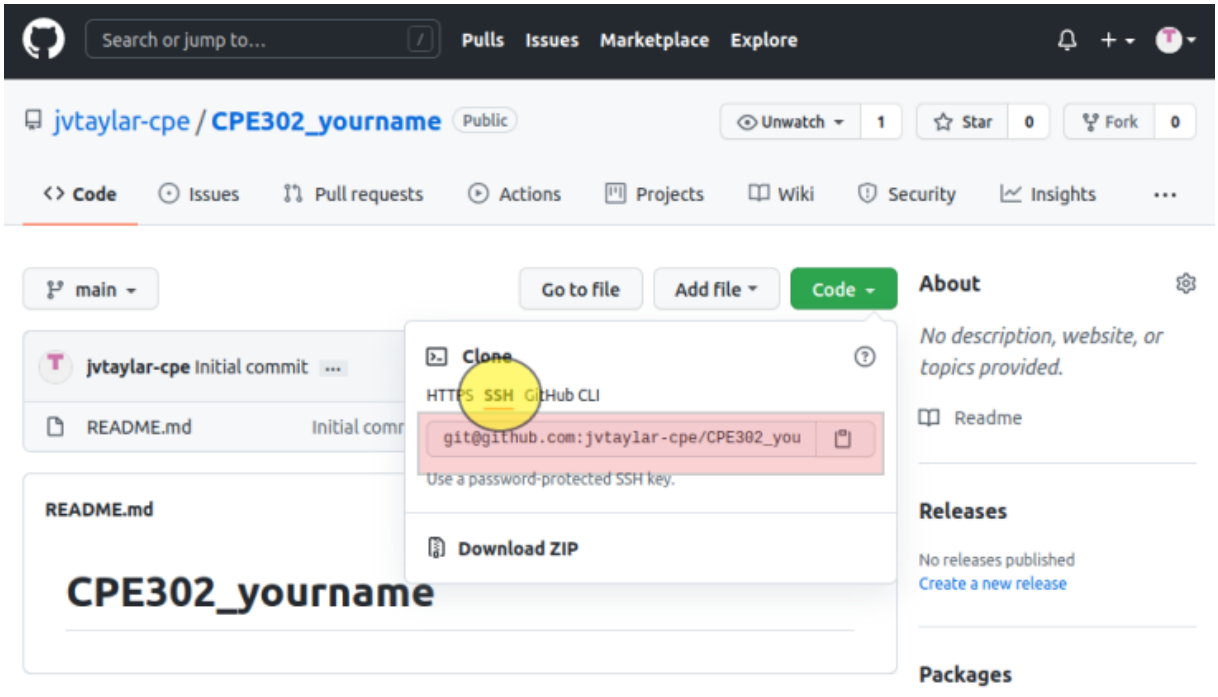


Figure 17: Adding SSh Key

d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



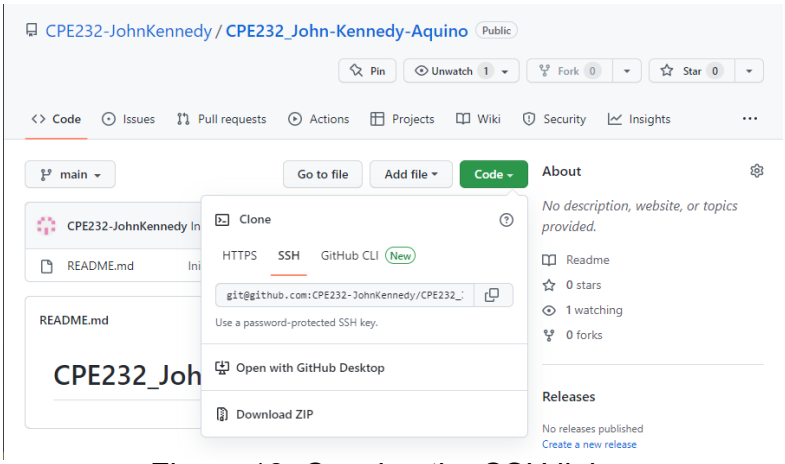


Figure 18: Copying the SSH link

e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.

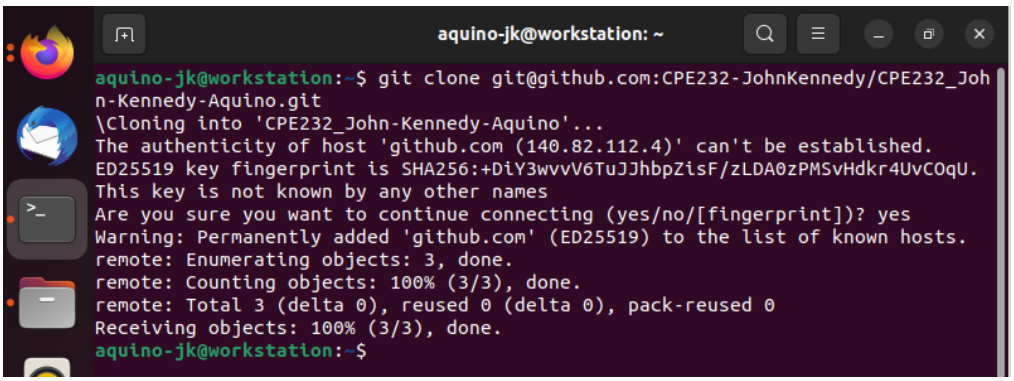


Figure 19: Cloning

g. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

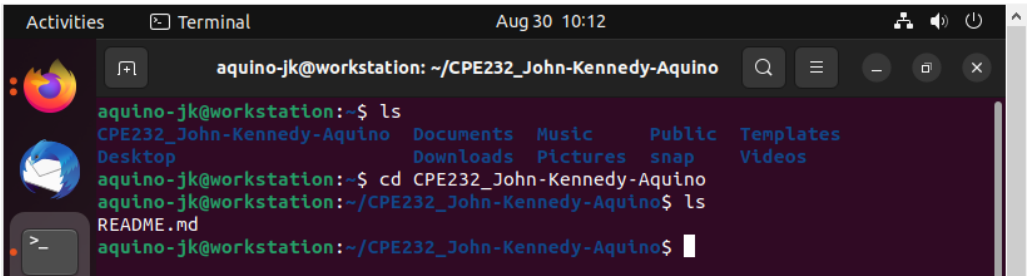


Figure 20: Verification of cloned GitHub repository

h. Use the following commands to personalize your git.
   - *git config --global user.name "Your Name"*
   - *git config --global user.email yourname@email.com*
   - Verify that you have personalized the config file using the command *cat ~/.gitconfig*

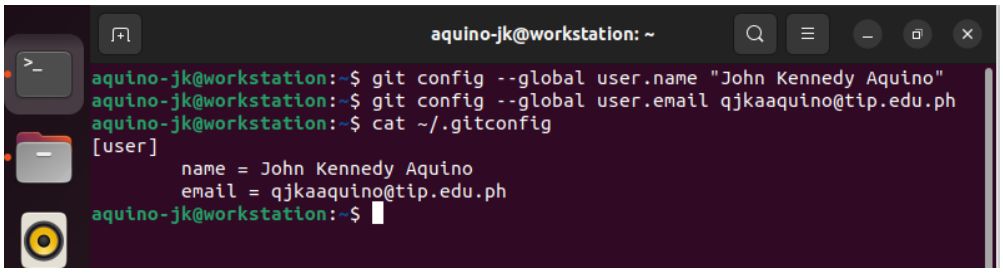


Figure 21: Verification of personalized configuration file

i. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.

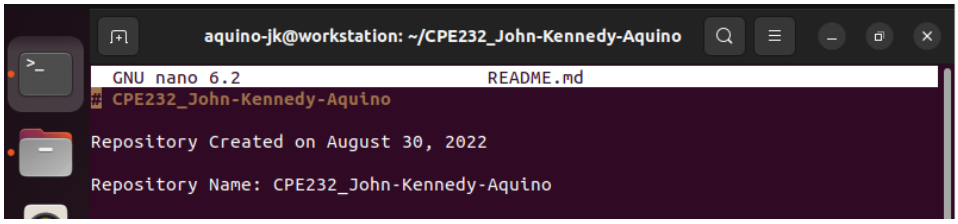


Figure 22: Editing the README.md file using nano command

j. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command? **By issuing this command, the result shows that changes are not staged for commit, and we can see that the README.md file is modified. It is also shown that there are no changes added to commit.**

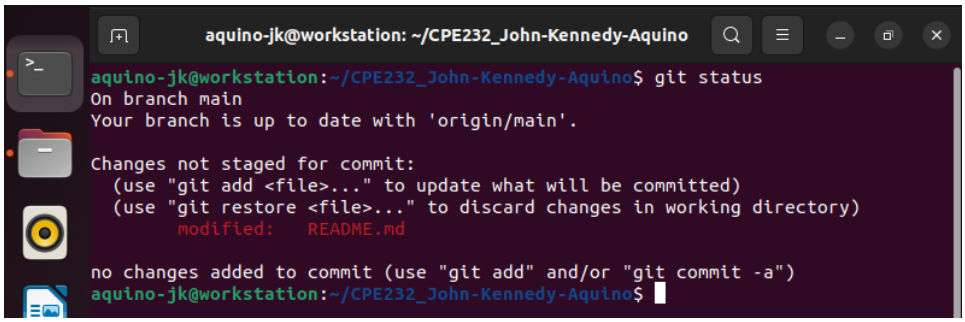


Figure 23: using the command git status

k. Use the command *git add README.md* to add the file into the staging area.

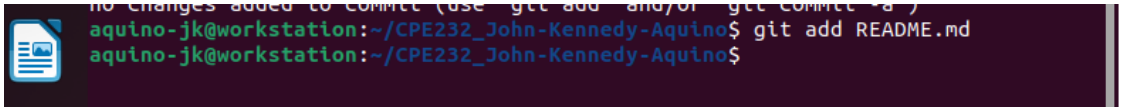


Figure 24: Adding README.md into the staging area

l. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

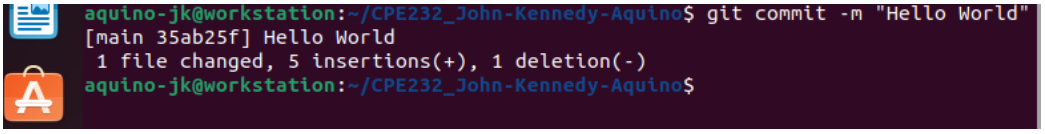


Figure 25: creating a snapshot of the staged changes

m. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main.*

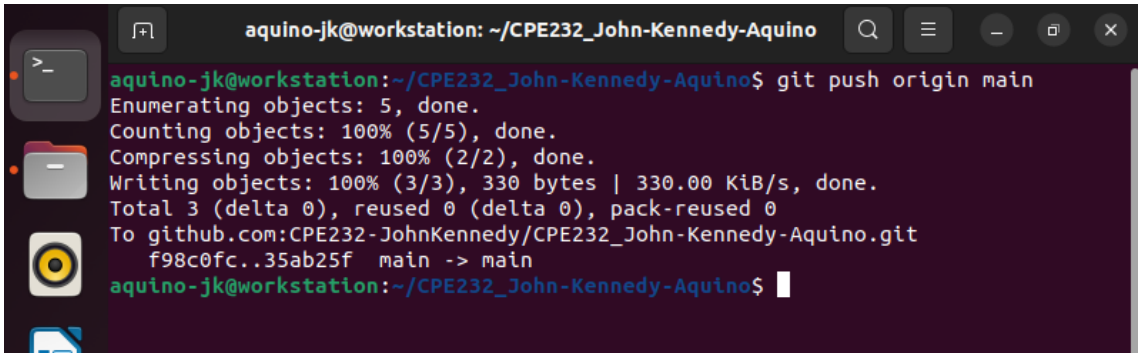
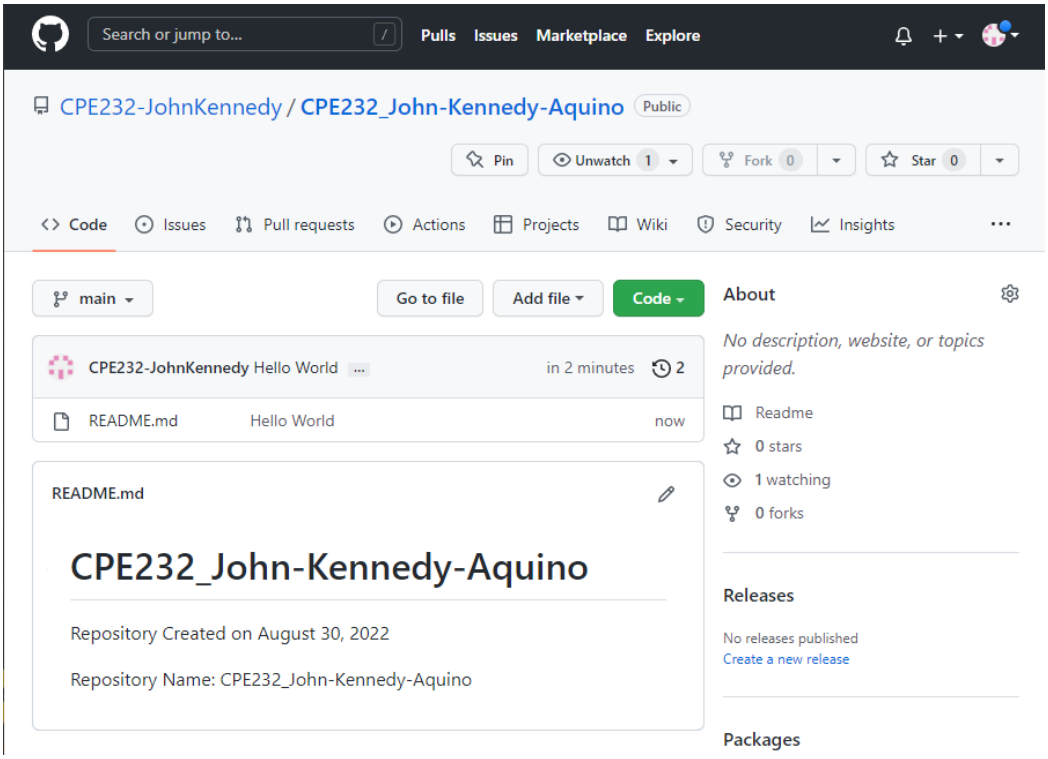


Figure 26: Uploaded local repository content to GitHub

n. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.



Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
    - In this activity we were able to connect and verify the SSH connectivity of the local machine, server1, and server2 and it was successful. After the creation of my GitHub activity we have inputted some commands in order to upload the modified README.md file to GitHub, after that, we have verified the changes we have done.

4. How important is the inventory file?
    - Inventory files are very essential in order for us to manage the files and monitor the changes that we have done in the files. It is also used to store the codes, documentations, and the files needed for the project.

Conclusions/Learnings:

- In conclusion, this activity helped to understand the configuration of the remote and local machine connection via SSH using a key instead of using a password, and I also learned how to create a public key and a private key, and verify the connectivity in the machines. Lastly, I was to create my GitHub account using remote and local repositories.