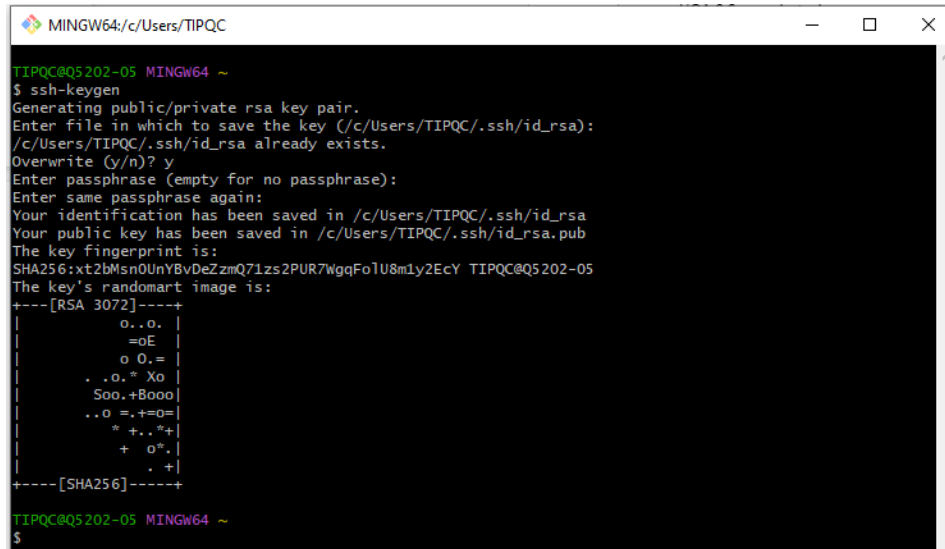


Name: Aquino, John Kennedy A.	Date Performed: August 30, 2022
Course/Section: CPE31S22	Date Submitted: August 30, 2022
Instructor: Dr. Jonathan V. Taylor	Semester and SY: 1st Sem, 2022-2023
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: <ul style="list-style-type: none"> 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 <p>It is assumed that you are already done with the last Activity (Activity 1: Configure Network using Virtual Machines). <i>Provide screenshots for each task.</i></p> <p>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.</p> <p>What Is ssh-keygen?</p> <p>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.</p> <p>SSH Keys and Public Key Authentication</p> <p>The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.</p> <p>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.</p> <p>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.</p>	

Task 1: Create an SSH Key Pair for User Authentication

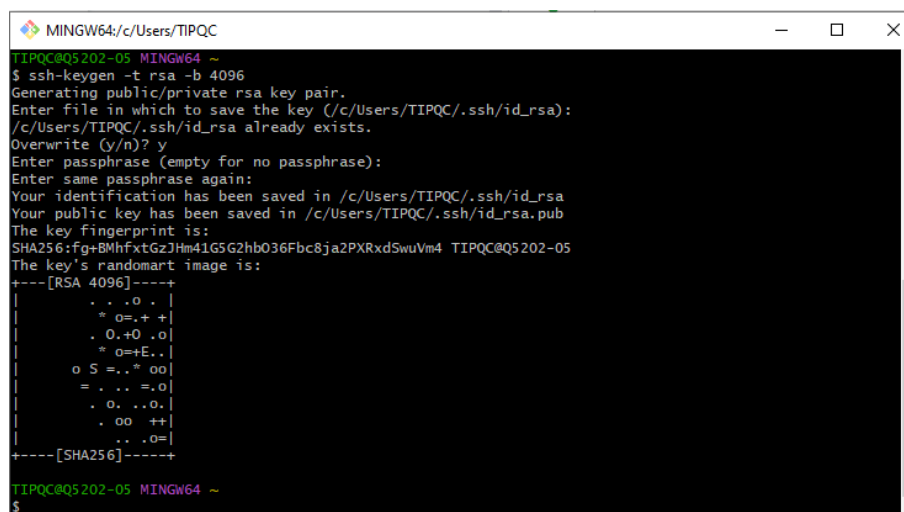
1. The simplest way to generate a key pair is to run **ssh-keygen** without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. 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**.



```
MINGW64:/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:xt2bMsn0UnYBvDeZzmQ71zs2PUR7WgqFo1U8m1y2EcY TIPQC@Q5202-05
The key's randomart image is:
+---[RSA 3072]-----+
|
| o..o. |
| =oE |
| o O.= |
| ..o.* Xo |
| Soo.+Booo|
| ..o =.+o=|
| * +. .*+|
| + o*.|
| . +|
+---[SHA256]-----+
TIPQC@Q5202-05 MINGW64 ~
$
```

Figure 1: Running ssh-keygen

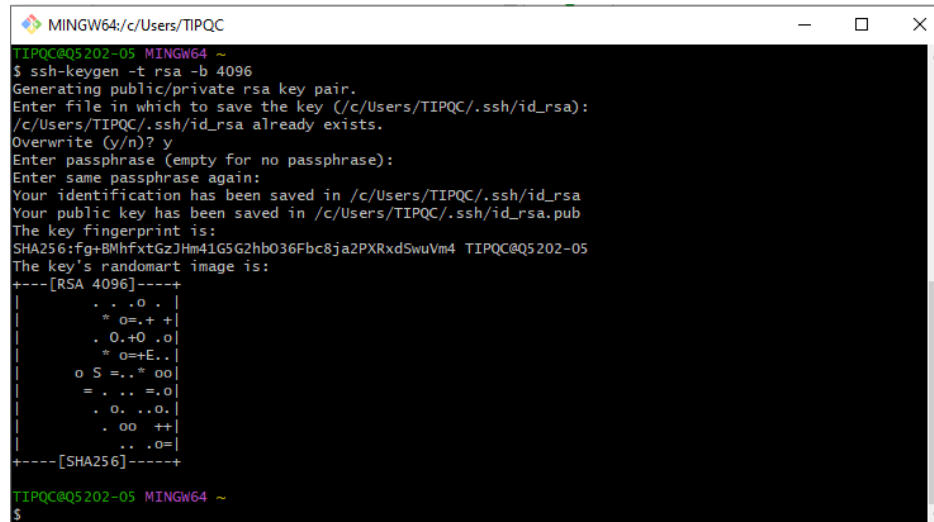
2. Issue the command **ssh-keygen -t rsa -b 4096**. The algorithm is selected using the -t option and key size using the -b option.



```
MINGW64:/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:fg+BMhftGzJHm41G5G2hb036Fbc8ja2PXRxdSwwVm4 TIPQC@Q5202-05
The key's randomart image is:
+---[RSA 4096]-----+
|
| ..o. |
| * o=+ +|
| . O.+O .o|
| * o+E..|
| o S =.* oo|
| = . . =.o|
| . O. .o.|
| . oo ++|
| ..o=|
+---[SHA256]-----+
TIPQC@Q5202-05 MINGW64 ~
$
```

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

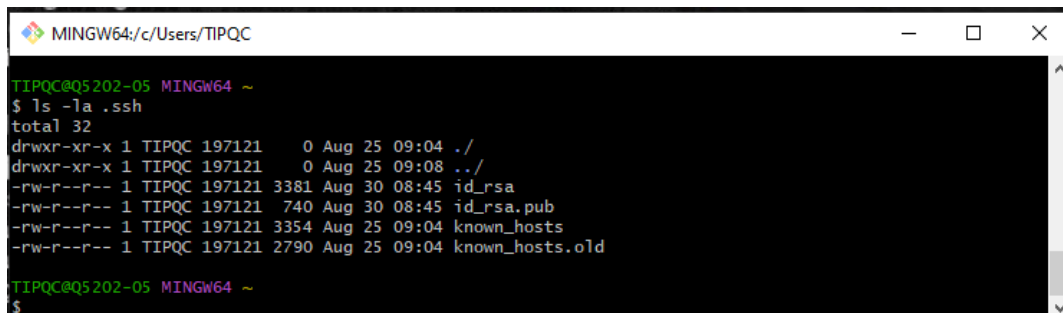
- When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.



```
MINGW64:/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/TIPQC/.ssh/id_rsa):
/c/Users/TIPQC/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/TIPQC/.ssh/id_rsa
Your public key has been saved in /c/Users/TIPQC/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:fg+BMhfxtGzJHm41G5G2hb036Fbc8ja2PXRxdSvuVm4 TIPQC@Q5202-05
The key's randomart image is:
+---[RSA 4096]-----+
|      . . . O . . |
|      * o = + + |
|      . O + O . o |
|      * o = + E . |
|      o S = . . oo |
|      = . . . = . o |
|      . O . . . O . |
|      . oo ++ |
|      . . . O = |
+---[SHA256]-----+
TIPQC@Q5202-05 MINGW64 ~
$
```

Figure 3:Passphrase for key encryption

- 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.

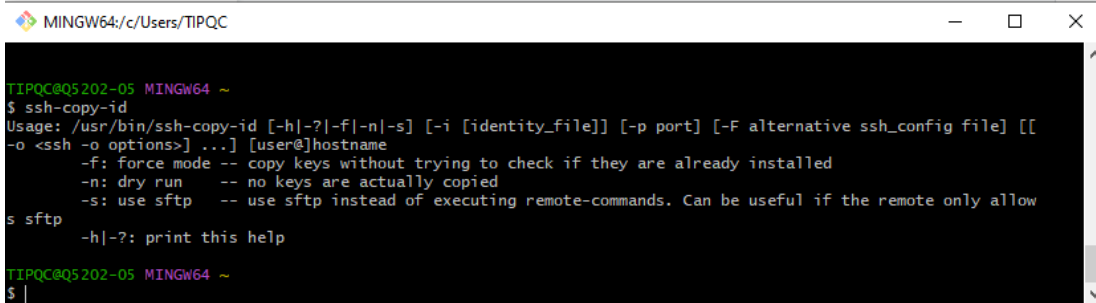


```
MINGW64:/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ls -la .ssh
total 32
drwxr-xr-x 1 TIPQC 197121  0 Aug 25 09:04 ./
drwxr-xr-x 1 TIPQC 197121  0 Aug 25 09:08 ../
-rw-r--r-- 1 TIPQC 197121 3381 Aug 30 08:45 id_rsa
-rw-r--r-- 1 TIPQC 197121  740 Aug 30 08:45 id_rsa.pub
-rw-r--r-- 1 TIPQC 197121 3354 Aug 25 09:04 known_hosts
-rw-r--r-- 1 TIPQC 197121 2790 Aug 25 09:04 known_hosts.old
TIPQC@Q5202-05 MINGW64 ~
$
```

Figure 4: Verification of the key

Task 2: Copying the Public Key to the remote servers

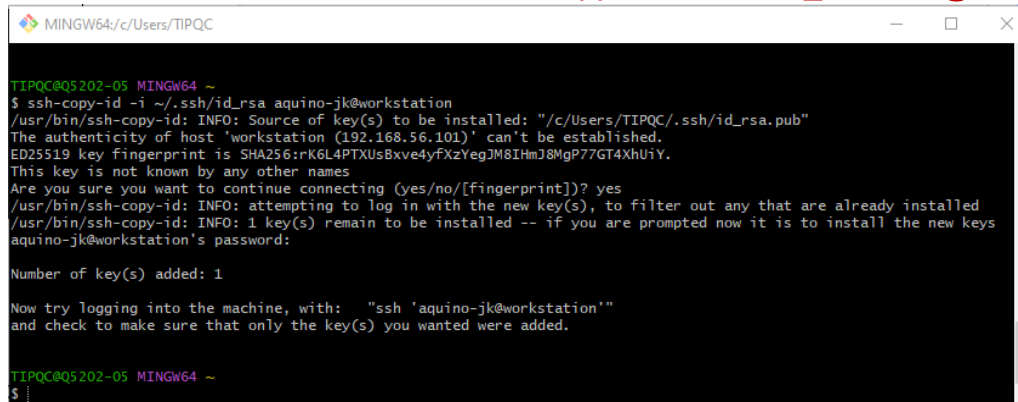
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.



```
MINGW64/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F alternative_ssh_config_file] [[
-o <ssh -o options>] ...] [user@]hostname
    -f: force mode -- copy keys without trying to check if they are already installed
    -n: dry run -- no keys are actually copied
    -s: use sftp -- use sftp instead of executing remote-commands. Can be useful if the remote only allow
s sftp
    -h|-?: print this help
TIPQC@Q5202-05 MINGW64 ~
$ |
```

Figure 5: ssh-copy-id options

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



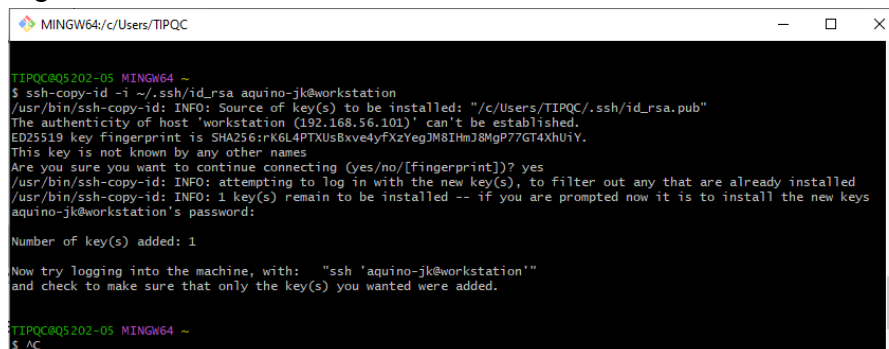
```
MINGW64/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa aquino-jk@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.ssh/id_rsa.pub"
The authenticity of host 'workstation (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:rK6L4PTXUs8xve4yFzYegJM8IHmJ8MgP77GT4XhUiY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
aquino-jk@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'aquino-jk@workstation'"
and check to make sure that only the key(s) you wanted were added.
TIPQC@Q5202-05 MINGW64 ~
$ |
```

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.



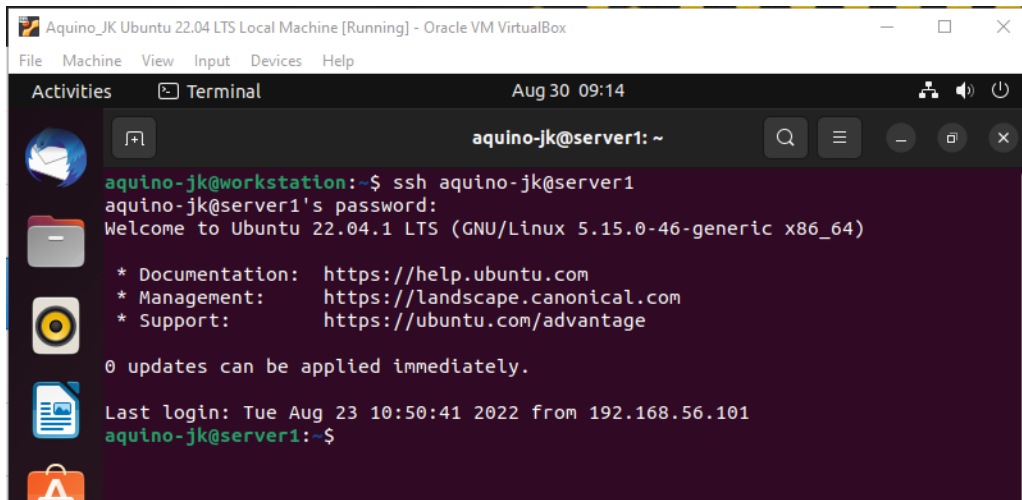
```
MINGW64/c/Users/TIPQC
TIPQC@Q5202-05 MINGW64 ~
$ ssh-copy-id -i ~/.ssh/id_rsa aquino-jk@workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/TIPQC/.ssh/id_rsa.pub"
The authenticity of host 'workstation (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:rK6L4PTXUs8xve4yFzYegJM8IHmJ8MgP77GT4XhUiY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
aquino-jk@workstation's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'aquino-jk@workstation'"
and check to make sure that only the key(s) you wanted were added.
TIPQC@Q5202-05 MINGW64 ~
$ ^C
```

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?



The screenshot shows a terminal window titled "Aquino_JK Ubuntu 22.04 LTS Local Machine [Running] - Oracle VM VirtualBox". The terminal is running a command to SSH into "server1". The output shows the password prompt, a welcome message for Ubuntu 22.04.1 LTS, system information, update status, and login history. The prompt changes from "aquino-jk@workstation" to "aquino-jk@server1".

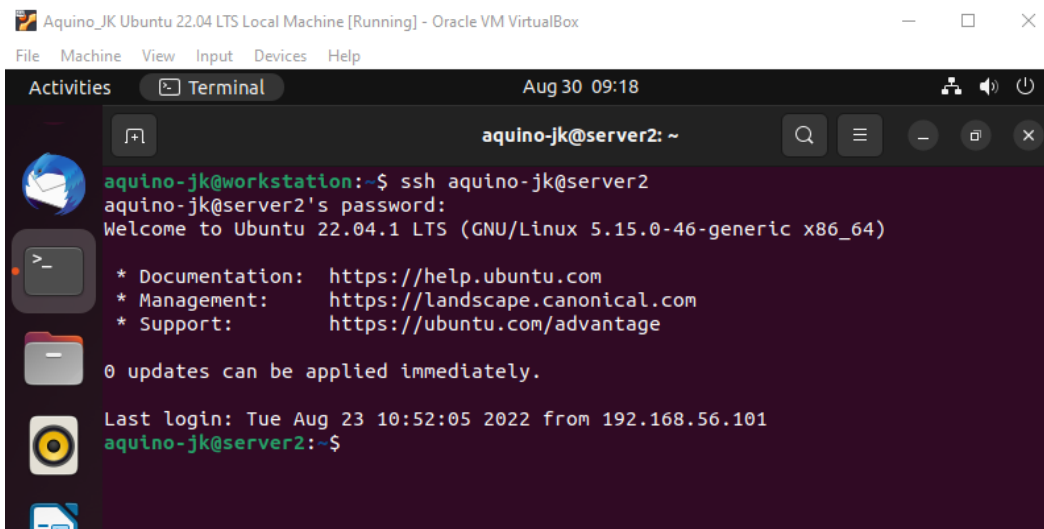
```
aquino-jk@workstation:~$ ssh aquino-jk@server1
aquino-jk@server1's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

Last login: Tue Aug 23 10:50:41 2022 from 192.168.56.101
aquino-jk@server1:~$
```

Figure 8: Verification of SSH connection with server 1



The screenshot shows a terminal window titled "Aquino_JK Ubuntu 22.04 LTS Local Machine [Running] - Oracle VM VirtualBox". The terminal is running a command to SSH into "server2". The output shows the password prompt, a welcome message for Ubuntu 22.04.1 LTS, system information, update status, and login history. The prompt changes from "aquino-jk@workstation" to "aquino-jk@server2".

```
aquino-jk@workstation:~$ ssh aquino-jk@server2
aquino-jk@server2's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

Last login: Tue Aug 23 10:52:05 2022 from 192.168.56.101
aquino-jk@server2:~$
```

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**).

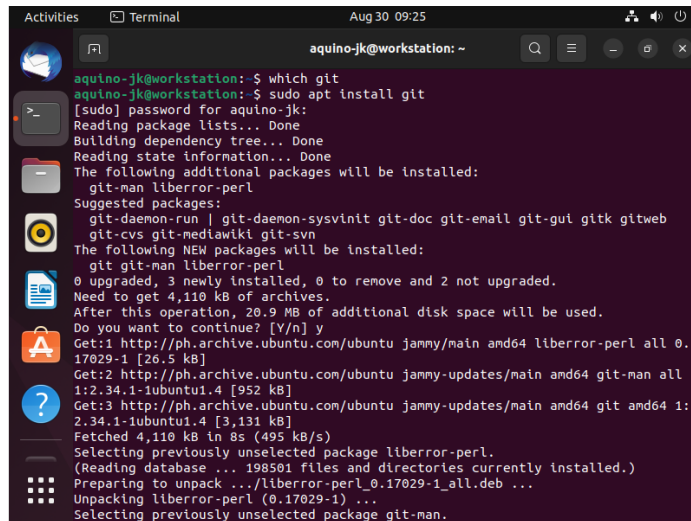
Set up Git

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

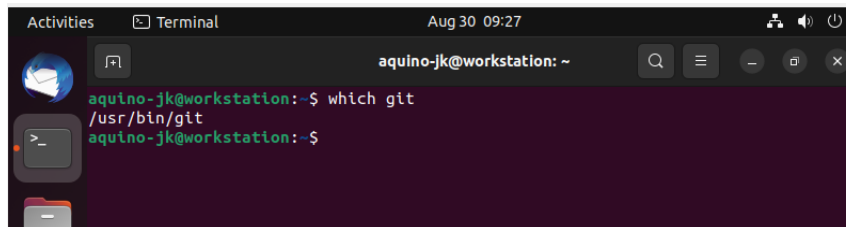
Task 3: Set up the Git Repository

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*



```
Activities Terminal Aug 30 09:25
aquino-jk@workstation: ~
aquino-jk@workstation:~$ which git
aquino-jk@workstation:~$ sudo apt install git
[sudo] password for aquino-jk:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
git-man liberror-perl
Suggested packages:
git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 2 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.4 [952 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.4 [3,131 kB]
Fetched 4,110 kB in 8s (495 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 198501 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
```

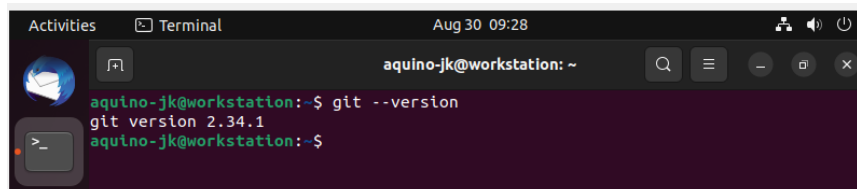
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.



```
Activities Terminal Aug 30 09:27
aquino-jk@workstation: ~
aquino-jk@workstation:~$ which git
/usr/bin/git
aquino-jk@workstation:~$
```

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.



```
Activities Terminal Aug 30 09:28
aquino-jk@workstation: ~
aquino-jk@workstation:~$ git --version
git version 2.34.1
aquino-jk@workstation:~$
```

Figure 12: Version of git installed

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

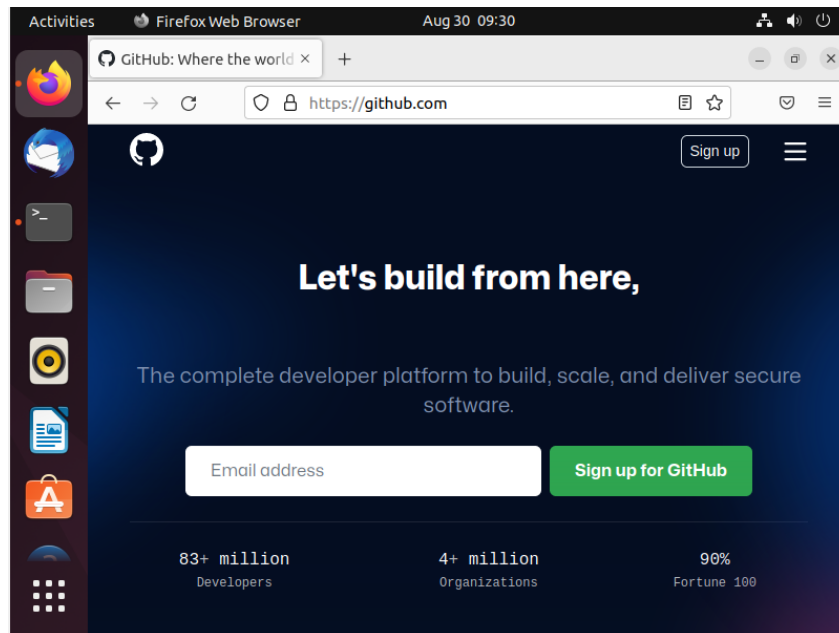


Figure 13: www.github.com website

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

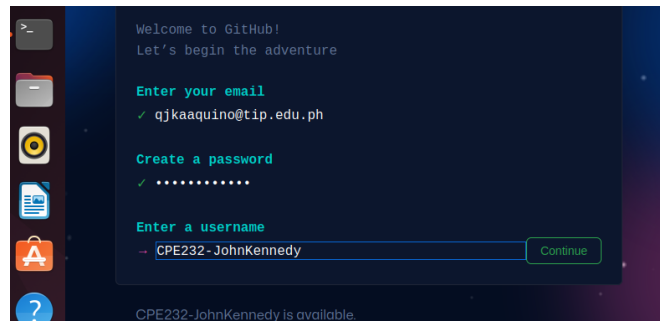


Figure 14: Creating an Account

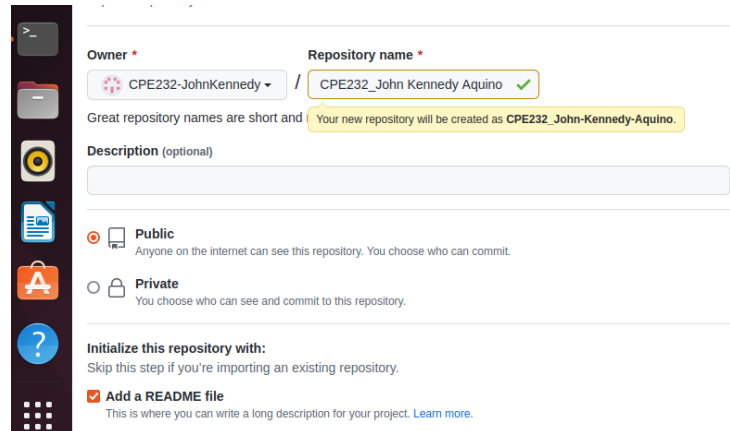


Figure 15: Repository name and checked add README file

- 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.

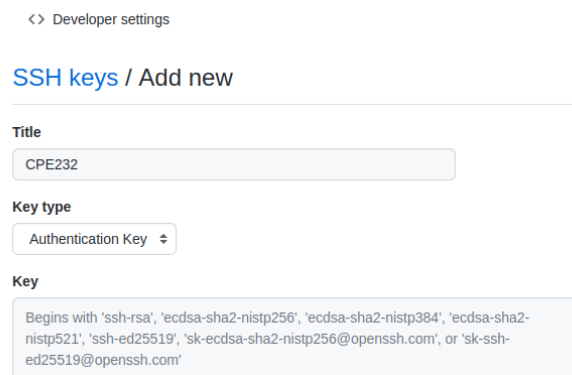


Figure 16: New SSH keys

- 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.

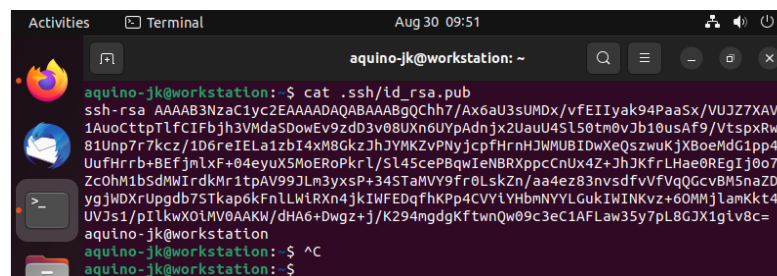


Figure 17: `cat .ssh/id_rsapub` command issued

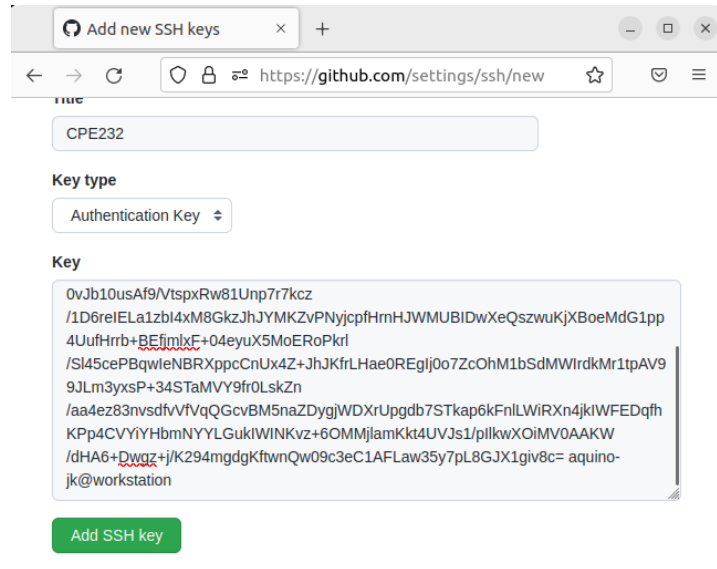
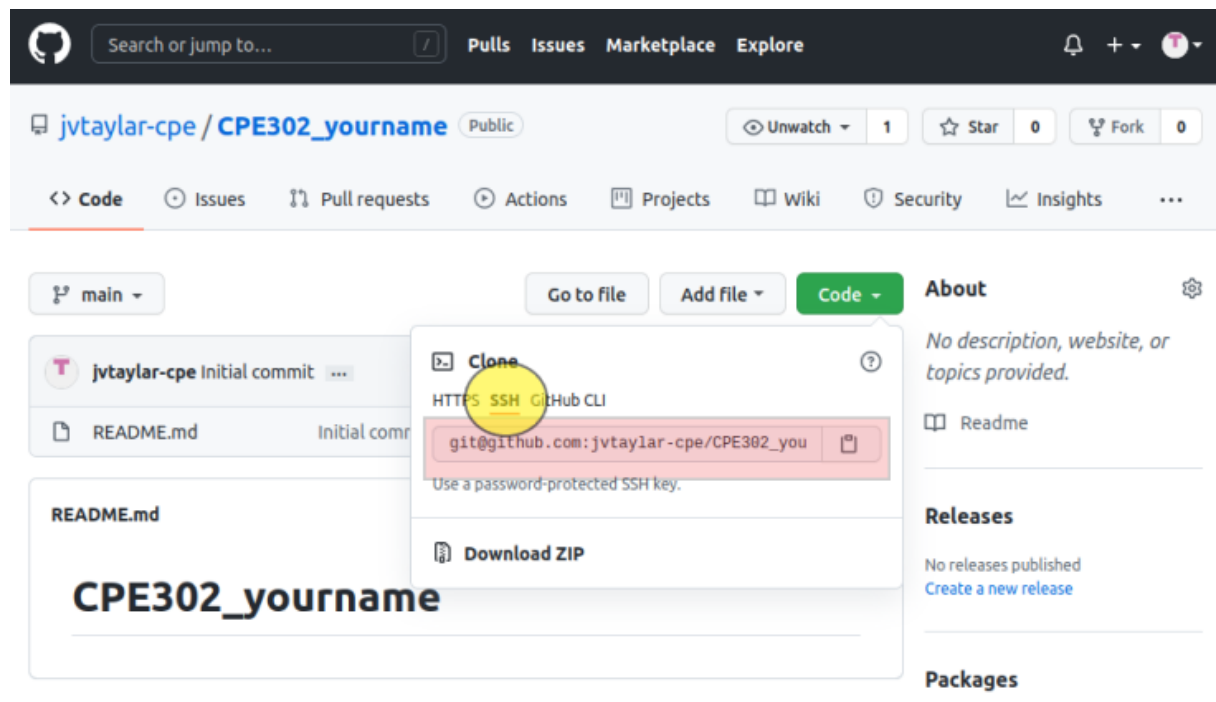


Figure 17: Adding SSh Key

- 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.



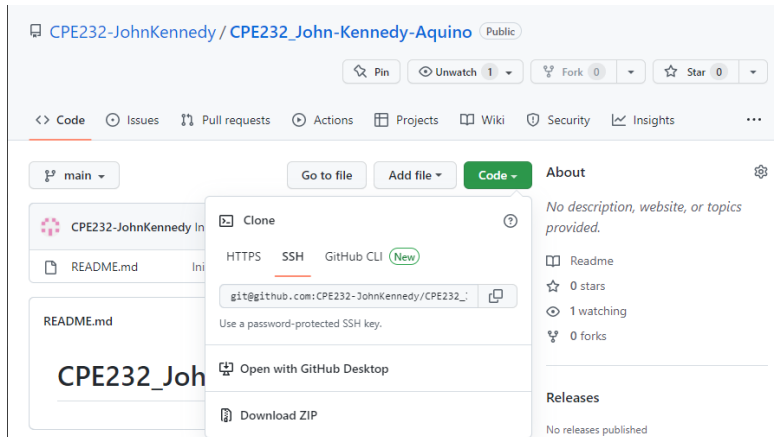


Figure 18: Copying the SSH link

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

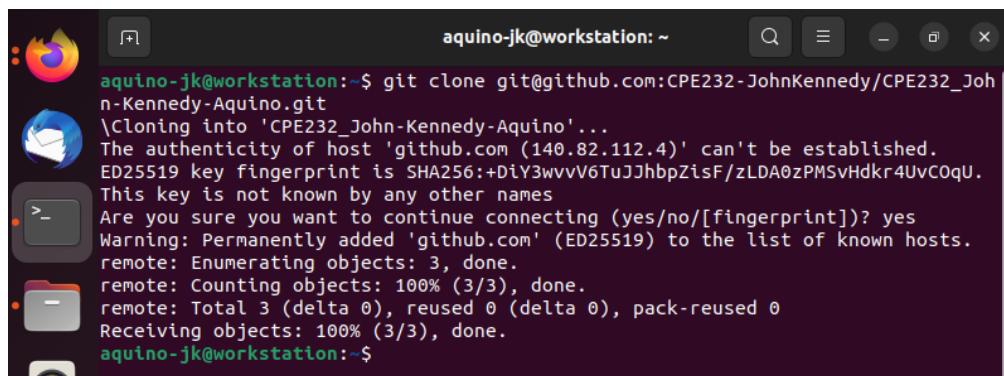


Figure 19: Cloning

- 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`.

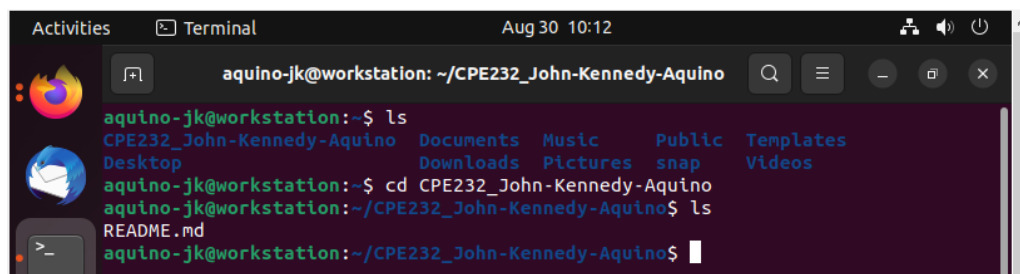
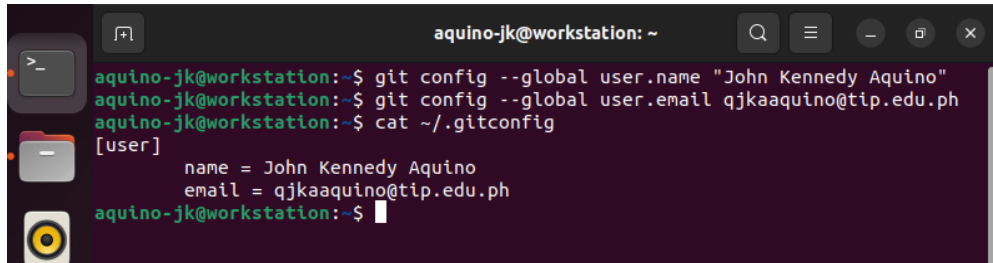


Figure 20: Verification of cloned GitHub repository

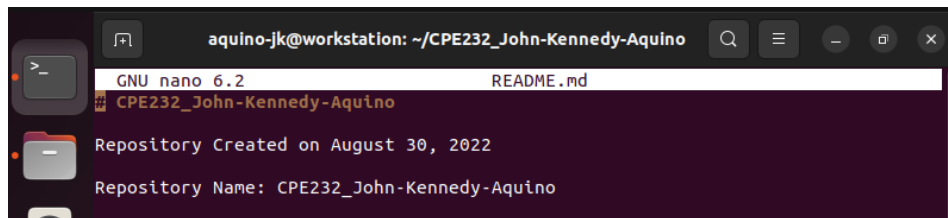
- h. Use the following commands to personalize your git.
- `git config --global user.name "Your Name"`
 - `git config --global user.email yourname@email.com`
 - Verify that you have personalized the config file using the command `cat ~/.gitconfig`



```
aquino-jk@workstation: ~  
aquino-jk@workstation:~$ git config --global user.name "John Kennedy Aquino"  
aquino-jk@workstation:~$ git config --global user.email qjkaaquino@tip.edu.ph  
aquino-jk@workstation:~$ cat ~/.gitconfig  
[user]  
    name = John Kennedy Aquino  
    email = qjkaaquino@tip.edu.ph  
aquino-jk@workstation:~$
```

Figure 21: Verification of personalized configuration file

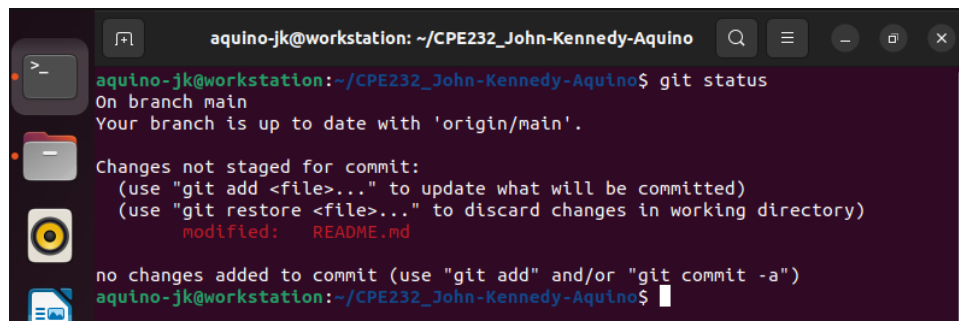
- i. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.



```
aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino  
GNU nano 6.2 README.md  
CPE232_John-Kennedy-Aquino  
Repository Created on August 30, 2022  
Repository Name: CPE232_John-Kennedy-Aquino
```

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.**

A terminal window titled 'aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino'. The user has entered 'git status'. The output shows the current branch is 'main' and it is up to date with 'origin/main'. It lists 'changes not staged for commit' for 'README.md' and provides instructions on how to stage or discard changes. It also states 'no changes added to commit' and provides instructions on how to commit.

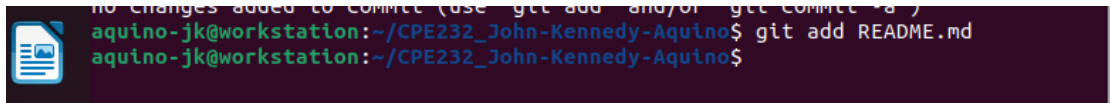
```
aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$
```

Figure 23: using the command git status

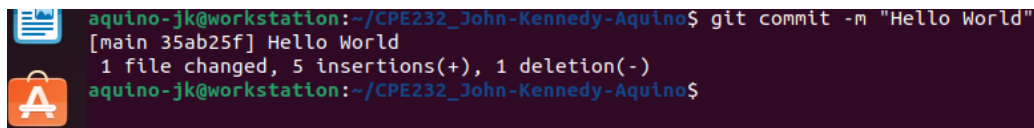
- k. Use the command *git add README.md* to add the file into the staging area.

A terminal window showing the execution of the 'git add README.md' command. The prompt is 'aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino\$'. The command is entered and executed, resulting in a new prompt.

```
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$ git add README.md
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$
```

Figure 24: Adding README.md into the staging area

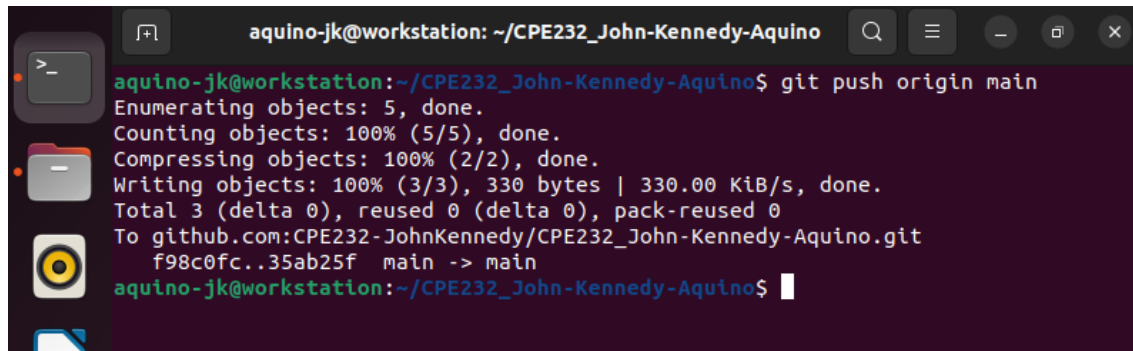
- l. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

A terminal window showing the execution of the 'git commit -m "Hello World"' command. The prompt is 'aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino\$'. The command is entered and executed, resulting in a new commit with hash '35ab25f' and the message 'Hello World'. It also shows '1 file changed, 5 insertions(+), 1 deletion(-)'.

```
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$ git commit -m "Hello World"
[main 35ab25f] Hello World
1 file changed, 5 insertions(+), 1 deletion(-)
aquino-jk@workstation:~/CPE232_John-Kennedy-Aquino$
```

Figure 25: creating a snapshot of the staged changes

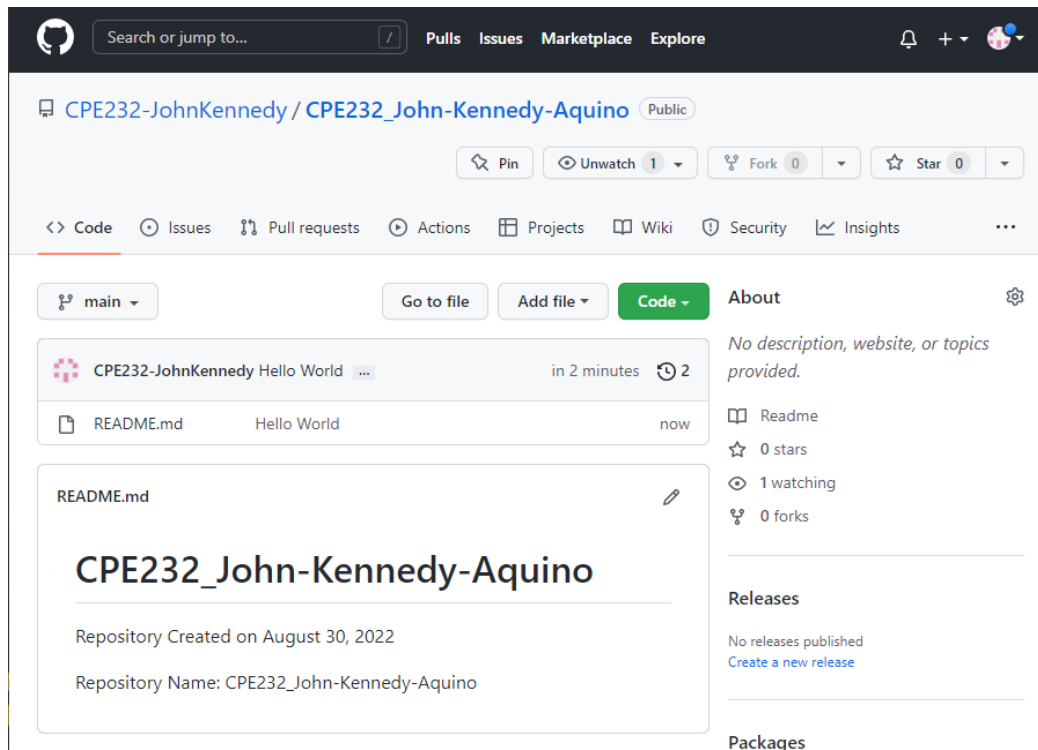
- m. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main*.

A terminal window with a dark background. The title bar shows 'aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino'. The command 'git push origin main' has been executed. The output shows the progress of pushing the commit to GitHub, including object enumeration, counting, and compression. The commit is successfully pushed to the 'main' branch.

```
aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 330 bytes | 330.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:CPE232-JohnKennedy/CPE232_John-Kennedy-Aquino.git
f98c0fc..35ab25f main -> main
aquino-jk@workstation: ~/CPE232_John-Kennedy-Aquino$
```

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.