Name: Issa Victoria Peñas	**Date Performed:** 08/23/2022
Course/Section: CPE31S2	**Date Submitted:** 08/23/2022
Instructor: Dr. Jonathan Taylar	**Semester and SY:** 1st Semester (SY: 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 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*.

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.
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.

```
penas@penas-workstation-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/penas/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/penas/.ssh/id_rsa
Your public key has been saved in /home/penas/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:baZYiCO7QmHohLsbtTjKImwi5jLDvxmNEHSbW06uOLA penas@penas-workstation-Virtu
alBox
The key's randomart image is:
+---[RSA 4096]----+
| . .             |
|. . o            |
|o. o o           |
|o+. *. . .       |
|=o+.oo. S +      |
|o*o++. o +       |
|E+o+ .. .        |
|&B..o            |
|%Bo+.            |
+----[SHA256]-----+
penas@penas-workstation-VirtualBox:~$
```

Figure 1.1. Entering the given command

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.

```
penas@penas-workstation-VirtualBox:~$ ls -la .ssh
total 24
drwx------  2 penas penas 4096 Aug 23 11:29 .
drwxr-x--- 15 penas penas 4096 Aug 23 11:22 ..
-rw-------  1 penas penas 3401 Aug 23 11:29 id_rsa
-rw-r--r--  1 penas penas  760 Aug 23 11:29 id_rsa.pub
-rw-------  1 penas penas  978 Aug 23 11:22 known_hosts
-rw-r--r--  1 penas penas  142 Aug 23 11:22 known_hosts.old
penas@penas-workstation-VirtualBox:~$
```

Figure 1.2. Shows the Detailed List of the .ssh folder

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

```
penas@penas-workstation-VirtualBox:~$ ssh-copy-id penas@Server_1
/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 prompt
ed now it is to install the new keys
penas@server_1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'penas@Server_1'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 2.1. Copying the public key of the Server 1

```
penas@penas-workstation-VirtualBox:~$ ssh-copy-id penas@Server_2
/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 prompt
ed now it is to install the new keys
penas@server_2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'penas@Server_2'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 2.2. Copying the public key of the Server 2

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.
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?

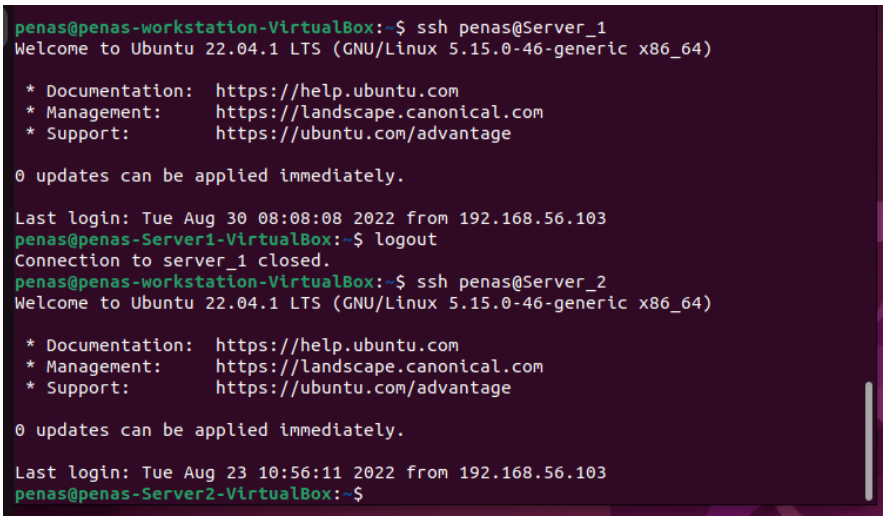


Figure 2.3. Logging In Between Servers 1 and 2 without requesting any passwords

- The Local Machine gave the admin privileges to access both Servers 1 and 2 without requesting any form of security such as User Passwords.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
    - Think of it as a duplicated keys on a hotel in which the hotel manager or staffs can access multiple rooms anytime they wanted to as long they had the keys, since they are given certain privileges which has a similar relationship where one admin or multiple admins

can access servers using their respective passwords depending on the topology of their networks. But in this case where the student copied each other public keys for efficiency on accessing a server to another server without requiring any passwords but its connectivity is more secure as it was connected in a SSH key.

2. How do you know that you already installed the public key to the remote servers?
   - In order to verify if the public SSH key was installed to the system, the programmer must change its directory to *"~/.ssh"* in order to access the folder named 'authorized keys' where all registered public keys were stored, in order to do that enter the code *'ll'* which lists all the data of the *"~/.ssh"* then *"cat authorized keys"* to view the key itself.

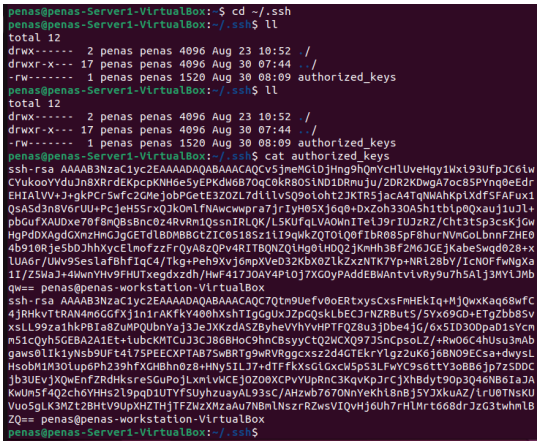


Figure 3.1. Shows how was the 'authorized keys' were being shown and to verify if the public key was already installed

Part 2: Discussion

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*

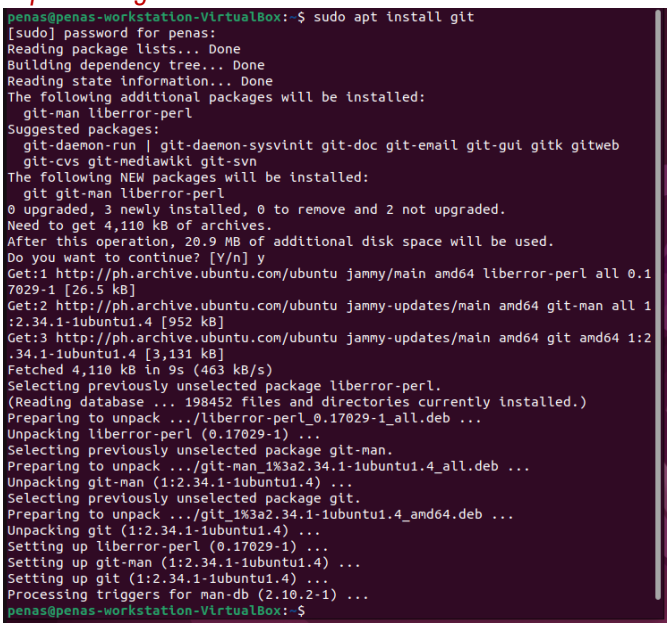


Figure 4.1. Downloading git library in Linux

2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

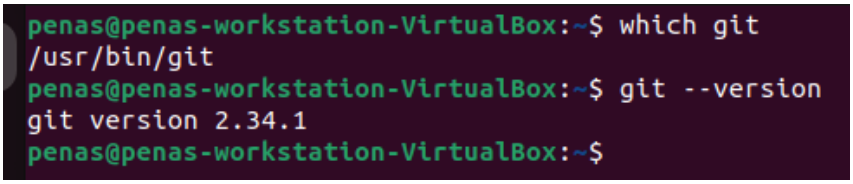


Figure 4.2. Showing where was the git library location and its version

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

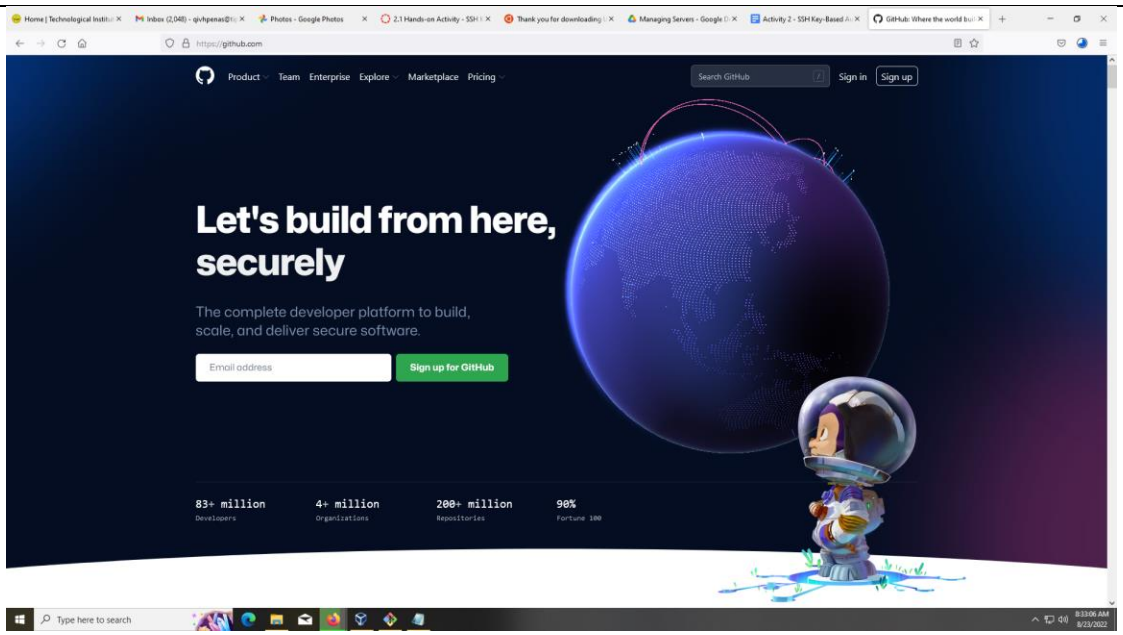


Figure 5.1. Shows the Front Page of the website GitHub

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.

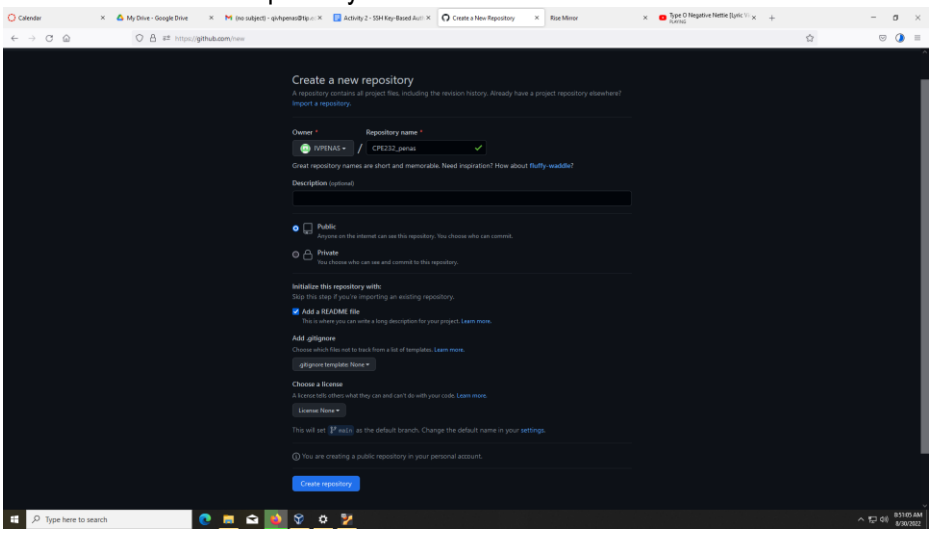


Figure 5.2. Creating new repository in GitHub

    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.

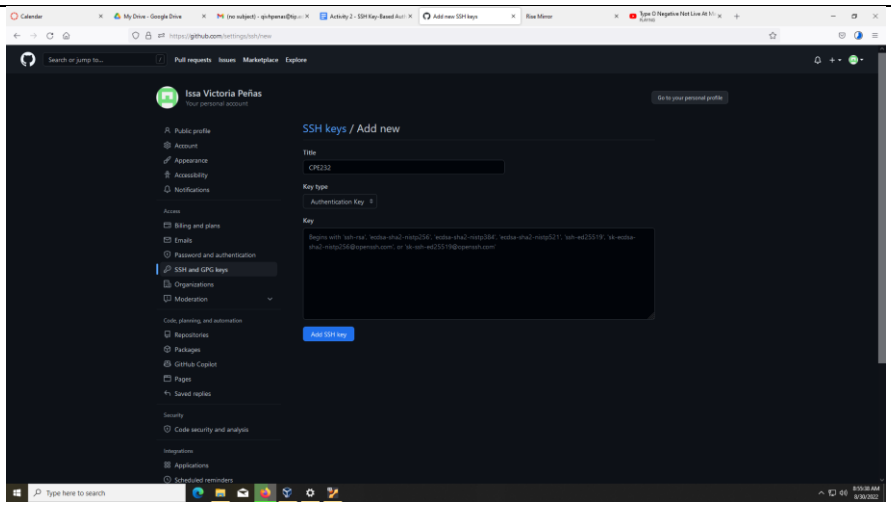


Figure 5.3. Adding new SSH Keys

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.

```
penas@penas-workstation-VirtualBox:~$ which git
/usr/bin/git
penas@penas-workstation-VirtualBox:~$ git --version
git version 2.34.1
penas@penas-workstation-VirtualBox:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC7Qtm9Uefv0oERtxysCxsFmHEkIq+MjQwxKaq68wfC
4jRHkvTtRAN4m6GGfXj1n1rAKfkY400hXshTIgGgUxJZpGQskLbECJrNZRButS/5Yx69GD+ETgZbb8Sv
xsLL99za1hkPBIa8ZuMPQUbnYaj3JeJXKzdASZByheVYhYvHPTFQZ8u3jDbe4jG/6x5ID3ODpaD1sYcm
m51cQyh5GEBA2A1Et+iubcKMTCuJ3CJ86BHoC9hnCBsyyCtQ2WCXQ97JSnCpsoLZ/+RwO6C4hUsu3mAb
gaws0lIk1yNsb9UFt4i75PEECXPTAB7SwBRTg9wRVRggcxsz2d4GTEkrYlgz2uK6j6BNO9ECsa+dwysL
HsobM1M3Oiup6Ph239hfXGHBhn0z8+HNy5ILJ7+dTFfkXsGiGxcW5pS3LFwYC9s6ttY3oBB6jp7zSDDC
jb3UEvjXQwEnfZRdHksreSGuPojLxmivWCEjOZO0XCPvYUpRnC3KqvKpJrCjXhBdyt9Op3Q46NB6IaJA
KwUm5f4Q2ch6YHHs2l9pqD1UTYfSUyhzuayAL93sC/AHzwb767ONnYeKhi8nBj5YJXkuAZ/irU0TNsKU
Vuo5gLK3MZt2BHtV9UpXHZTHjTFZWzXMzaAu7NBmlNszrRZwsVIQvHj6Uh7rHlMrt668drJzG3twhmlB
ZQ== penas@penas-workstation-VirtualBox
penas@penas-workstation-VirtualBox:~$
```

Figure 5.4. Copying the Public Key from Linux

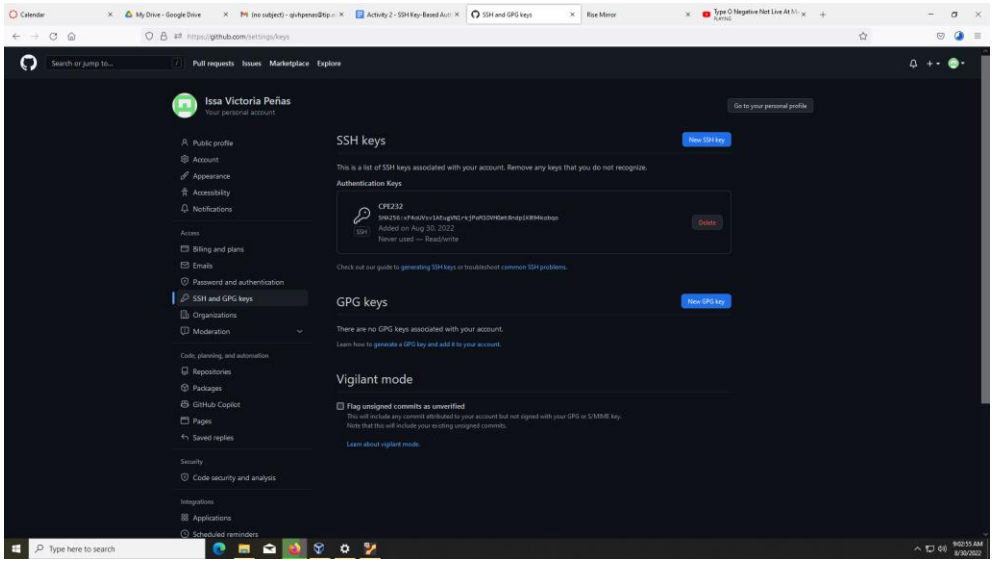


Figure 5.4. Added a new SSH Key from Local Host-Linux

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.

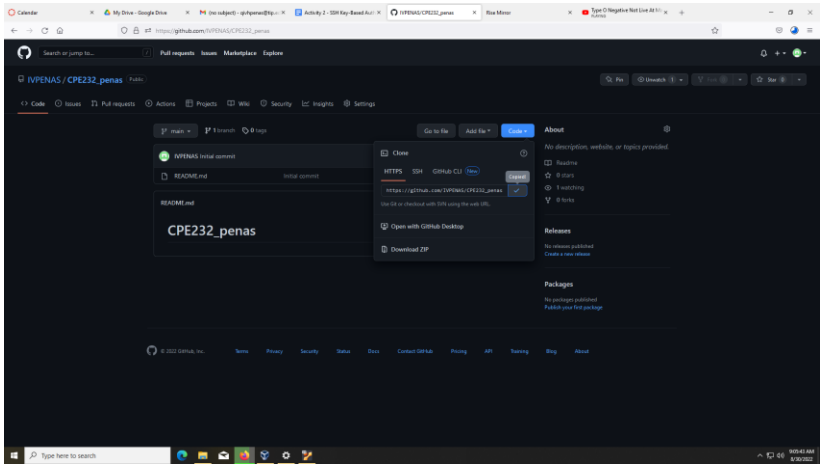
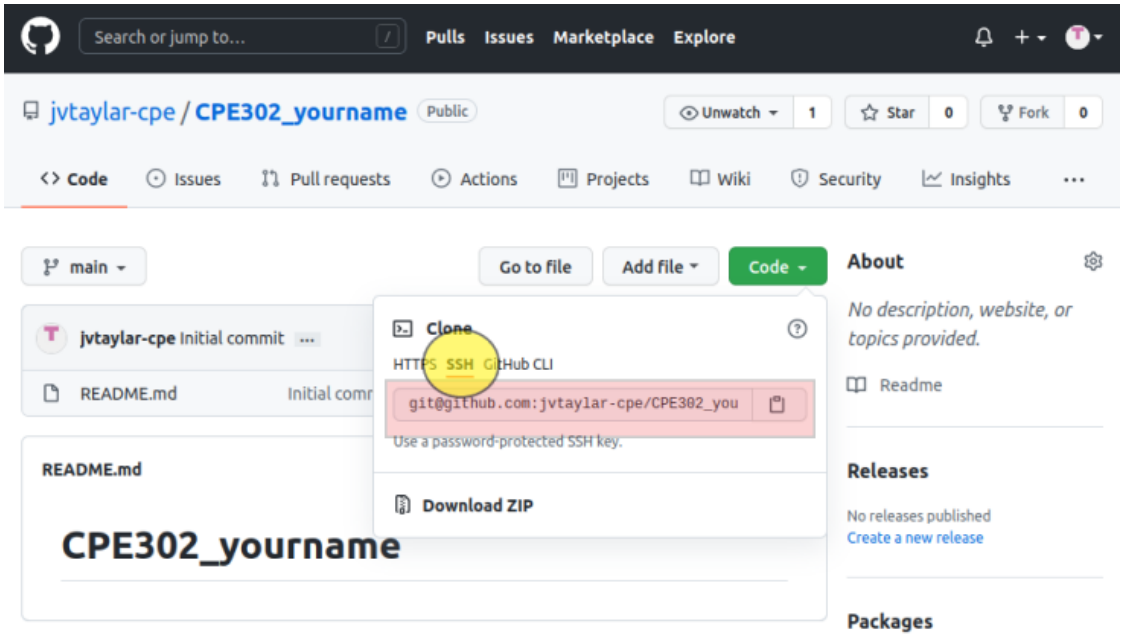


Figure 5.5. Copying Link of the Created Repository



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.

```
penas@penas-workstation-VirtualBox:~$ git clone git@github.com:IVPENAS/CPE232_pe
nas.git
Cloning into 'CPE232_penas'...
The authenticity of host 'github.com (140.82.113.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Figure 5.6. Cloning the git in Linux using the copied link from github

f. 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.

```
penas@penas-workstation-VirtualBox:~$ ls
CPE232_penas  Documents  Music     Public  Templates
Desktop       Downloads  Pictures  snap    Videos
penas@penas-workstation-VirtualBox:~$ cd CPE232_penas
penas@penas-workstation-VirtualBox:~/CPE232_penas$
```

Figure 5.7. Verifying the cloned git repository in Linux

g. 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*

```
penas@penas-workstation-VirtualBox:~/CPE232_penas$ git config --global user.name
 "issa_victoria_penas"
penas@penas-workstation-VirtualBox:~/CPE232_penas$ git config --global user.emai
l "qivhpenas@tip.edu.ph"
penas@penas-workstation-VirtualBox:~/CPE232_penas$ cat ~/.gitconfig
[user]
        name = issa_victoria_penas
        email = qivhpenas@tip.edu.ph
penas@penas-workstation-VirtualBox:~/CPE232_penas$
```

Figure 5.8. Inputting Private Information Details in Git Repository

h. 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.

```
  GNU nano 6.2
# CPE232_penas

# THIS IS A SAMPLE README FILE
### Name: Issa Victoria H. Penas
### Age: 20
### Gender: Female
```

Figure 5.9. Inputting Private Information Details in README.md

i. 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?

- It'll notify that the README.md was been modified

```
penas@penas-workstation-VirtualBox:~/CPE232_penas$ 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")
penas@penas-workstation-VirtualBox:~/CPE232_penas$
```

```
penas@penas-workstation-VirtualBox:~/CPE232_penas$ git add README.md
penas@penas-workstation-VirtualBox:~/CPE232_penas$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

Figure 5.11. Shows the status of the git repository

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

k. 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.

```
penas@penas-workstation-VirtualBox:~/CPE232_penas$ git commit -m "First Commit"
[main 254c241] First Commit
 1 file changed, 6 insertions(+), 1 deletion(-)
```

Figure 5.10. Commiting to the git repository

l. 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.*

```
penas@penas-workstation-VirtualBox:~/CPE232_penas$ 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), 345 bytes | 345.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:IVPENAS/CPE232_penas.git
   a2ffa2a..254c241  main -> main
```

Figure 5.12. The modification has been pushed (Saved) to the repository

m. 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.

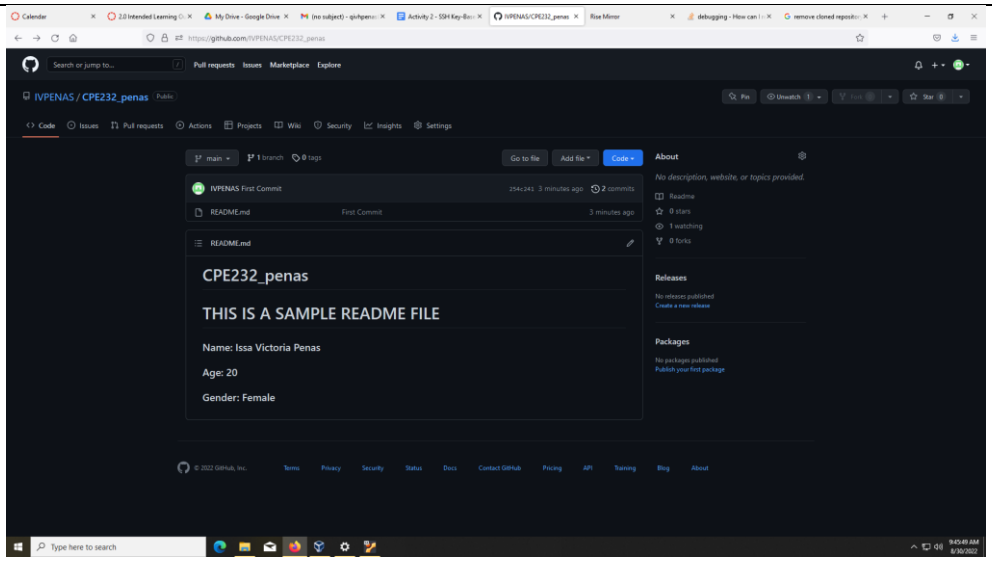


Figure 5.13. The modification has been pushed (Saved) in to the GitHub

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
    - Some of the ansible command was used in this activity which are the ssh-keygen" that allows the programmer to generate a key pair for the local machine and remote machines which is encrypted and "ssh-copy-id" which copies the public key of a server or local machine and pasting it to the remote machine inorder to have connectivity between the two networks.
4. How important is the inventory file?
    - Firstly, an Ansible Inventory Files is like a Log Book which can be in multiple format where it is mostly used on big companies handling multiple programmers as it was efficient enough to track the progress and fix bugs of a certain project.

Conclusions/Learnings:

- In this Hands-On Activity, the student was able to do each tasks where generating new keygen for the system using the command "ssh-keygen", to copy the encrypted Public Key of a certain Local Host to connect on another Network inorder to access it more efficiently and lastly connecting the Local machine online using the website Github where all modifications from Linux Terminal was appended Online by cloning its repository after intputing the Public Key of a network in order to make modifications.