Name: Enzo Daren B. Padual	**Date Performed: 08/24/2023**
Course/Section: CPE232	**Date Submitted: 08/28/2023**
Instructor: Engr. Roman Richard	**Semester and SY: 1st Sem – 3rd Year**

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.

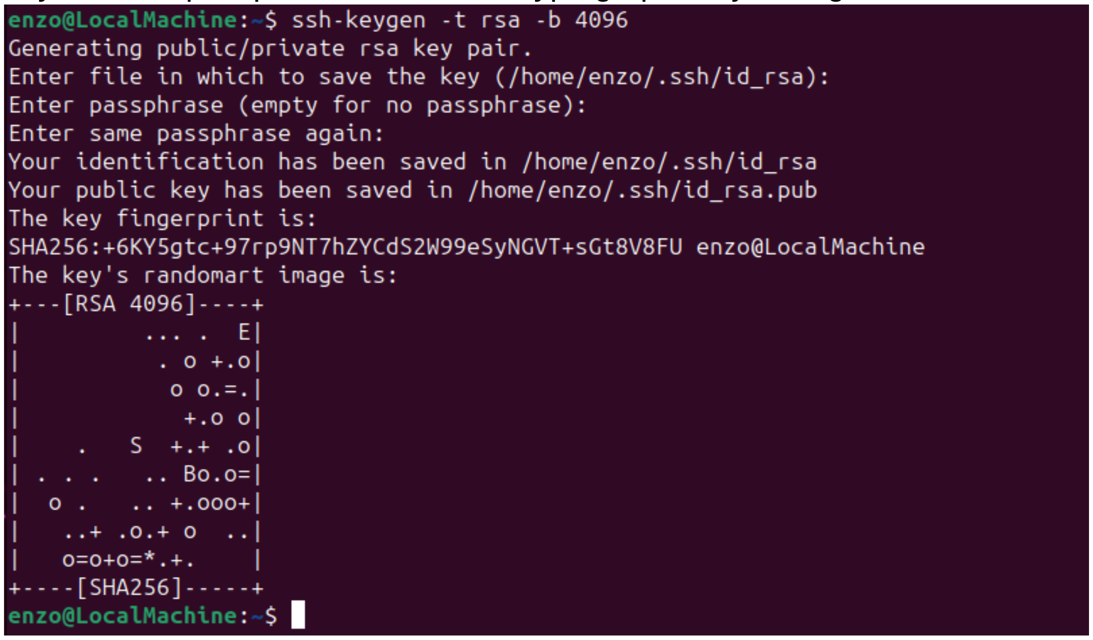
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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

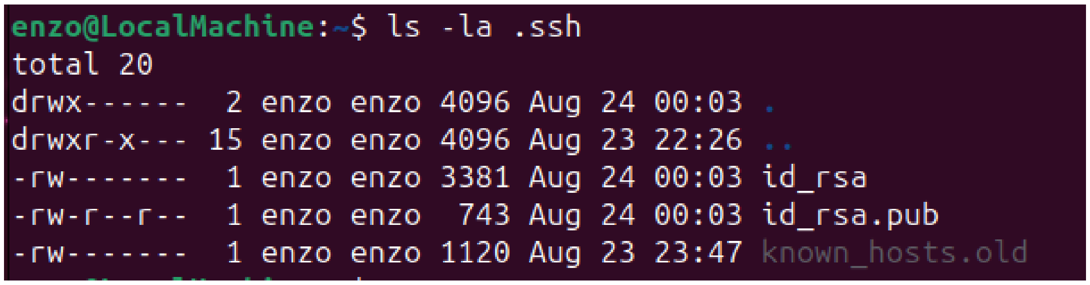
users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the default RSA algorithm. It could also be, for example, *id_dsa* or *id_ecdsa*.

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.



```
enzo@LocalMachine:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/enzo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/enzo/.ssh/id_rsa
Your public key has been saved in /home/enzo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:+6KY5gtc+97rp9NT7hZYCdS2W99eSyNGVT+sGt8V8FU enzo@LocalMachine
The key's randomart image is:
+---[RSA 4096]----+
|         ... .  E|
|          . o +.o|
|           o o.=.|
|            +.o o|
|    .   S  +.+ .o|
| . . .   .. Bo.o=|
|  o .   .. +.ooo+|
|   ..+ .o.+ o  ..|
|   o=o+o=*.+.    |
+----[SHA256]-----+
enzo@LocalMachine:~$
```

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.

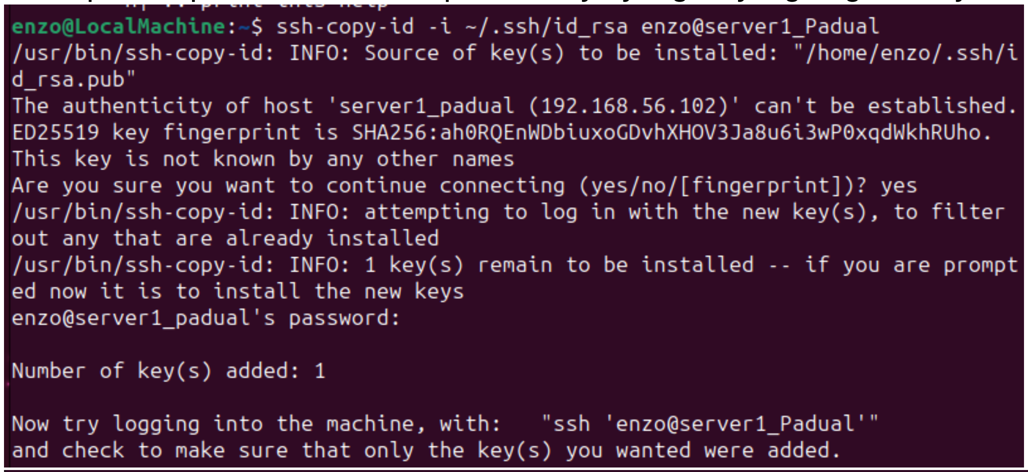


```
enzo@LocalMachine:~$ ls -la .ssh
total 20
drwx------  2 enzo enzo 4096 Aug 24 00:03 .
drwxr-x--- 15 enzo enzo 4096 Aug 23 22:26 ..
-rw-------  1 enzo enzo 3381 Aug 24 00:03 id_rsa
-rw-r--r--  1 enzo enzo  743 Aug 24 00:03 id_rsa.pub
-rw-------  1 enzo enzo 1120 Aug 23 23:47 known_hosts.old
```

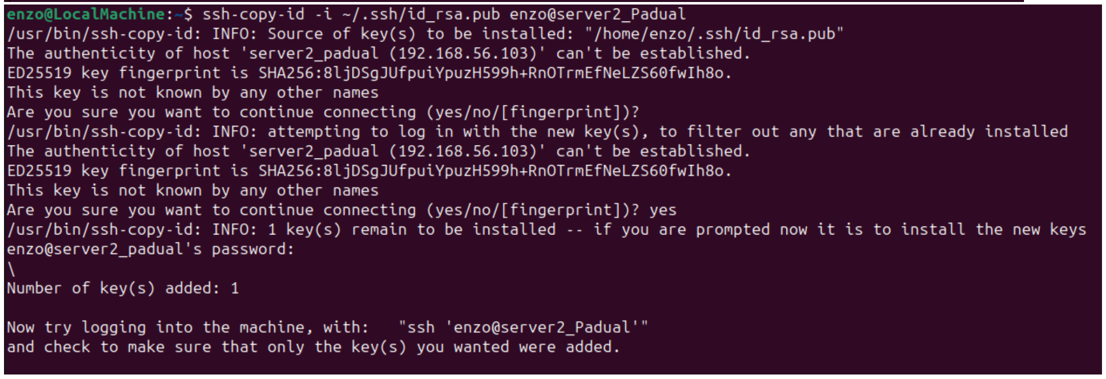
Task 2: Copying the Public Key to the remote servers

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 ~/.ssh/id_rsa user@host*

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.

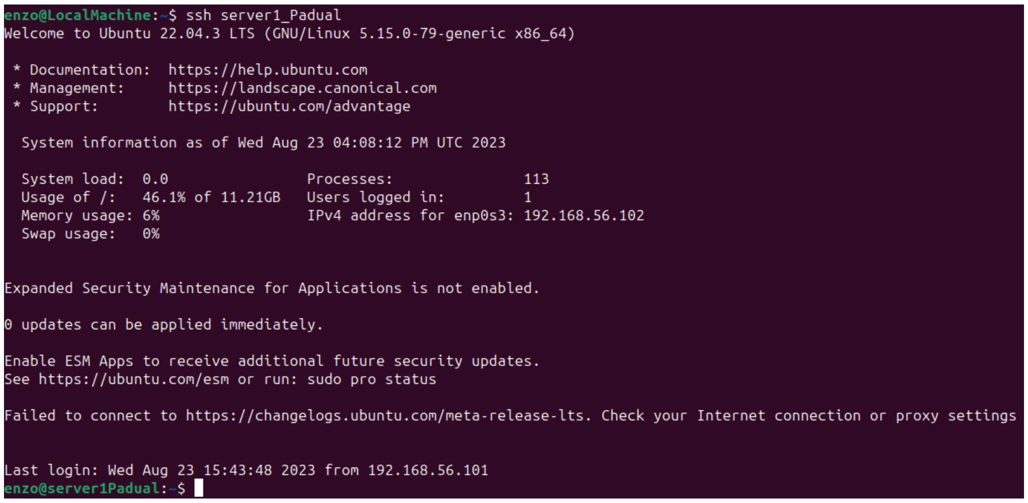


```
enzo@LocalMachine:~$ ssh-copy-id -i ~/.ssh/id_rsa enzo@server1_Padual
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/enzo/.ssh/i
d_rsa.pub"
The authenticity of host 'server1_padual (192.168.56.102)' can't be established.
ED25519 key fingerprint is SHA256:ah0RQEnWDbiuxoGDvhXHOV3Ja8u6i3wP0xqdWkhRUho.
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 prompt
ed now it is to install the new keys
enzo@server1_padual's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'enzo@server1_Padual'"
and check to make sure that only the key(s) you wanted were added.
```



```
enzo@LocalMachine:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub enzo@server2_Padual
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/enzo/.ssh/id_rsa.pub"
The authenticity of host 'server2_padual (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:8ljDSgJUfpuiYpuzH599h+RnOTrmEfNeLZS60fwIh8o.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
The authenticity of host 'server2_padual (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:8ljDSgJUfpuiYpuzH599h+RnOTrmEfNeLZS60fwIh8o.
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: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
enzo@server2_padual's password:
\
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'enzo@server2_Padual'"
and check to make sure that only the key(s) you wanted were added.
```

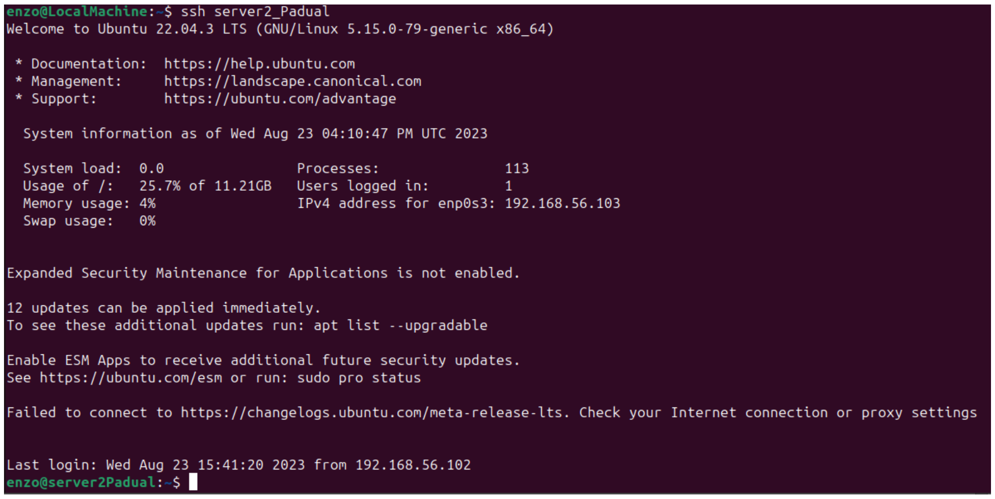
4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?



```
enzo@LocalMachine:~$ ssh server1_Padual
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

  System information as of Wed Aug 23 04:08:12 PM UTC 2023

  System load:  0.0               Processes:               113
  Usage of /:   46.1% of 11.21GB  Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.102
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Aug 23 15:43:48 2023 from 192.168.56.101
enzo@server1Padual:~$
```

It did not ask for a password because with the SSH key pair, it authenticates the Local Machine to the server 1 and 2 by sending an SSH key to both, but for that to be authenticated it needs password of server 1

and 2 first, to have a seamless login to the servers. It can also ask for password, but it is not called a password but a passphrase, it can be activated and register a new pass key when using SSH to connect remotely through servers.



```
enzo@LocalMachine:~$ ssh server2_Padual
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-79-generic x86_64)

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

  System information as of Wed Aug 23 04:10:47 PM UTC 2023

  System load:  0.0               Processes:               113
  Usage of /:   25.7% of 11.21GB   Users logged in:         1
  Memory usage: 4%                IPv4 address for enp0s3: 192.168.56.103
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Aug 23 15:41:20 2023 from 192.168.56.102
enzo@server2Padual:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   SSH (Secure Shell) is a cryptographic network protocol that ensures secure communication over insecure networks. It enables users to remotely access systems, transfer files, and execute commands on remote servers while maintaining data confidentiality and integrity. SSH employs strong encryption algorithms to protect transmitted information, thwarting eavesdropping and unauthorized access. The protocol offers multiple authentication methods, including key-based authentication, enhancing security beyond traditional password-based methods. SSH also facilitates port forwarding, allowing secure routing of network traffic through encrypted connections. Tools like scp and sftp enable safe file transfer between local and remote systems. Remote command execution via SSH grants users the ability to administer servers as if physically present, enhancing administrative efficiency. Furthermore, SSH tunneling enables the establishment of secure connections for various network services, making it a fundamental tool for secure remote management and data transfer.
2. How do you know that you already installed the public key to the remote servers?

- **Verifying the installation of your public key on remote servers involves multiple steps:**
- **Locate the public key file, often named id_rsa.pub or id_dsa.pub, situated in the .ssh directory within your home folder.**

- **Ensure proper authentication by entering the correct SSH username for your account on the remote server you intend to SSH into.**
- **For the directory check, employ the cd command to move to the .ssh directory on the remote server. If needed, you can create this directory.**
- **File Inspection: Within this directory, list the contents using ls to confirm the presence of the authorized_keys file.**
- **Key Comparison: Open the authorized_keys file with a text editor such as nano or cat, and confirm that its content matches the public key from your local id_rsa.pub (or equivalent) file.**
- **Key Entry Format: Each public key entry in the authorized_keys file generally starts with "ssh-rsa" or "ssh-dss," followed by the key content and an optional comment.**
- **Multiple Keys: If multiple public keys are used, expect to see multiple entries in the authorized_keys file, each corresponding to different key pairs.**
- **Permissions Check: Ensure that the authorized_keys file has the appropriate permissions (usually 600) and is owned by the user account you're connecting with.**
- **Functional Test: Test the setup by attempting SSH login from your local machine to the remote server again; a successful login without password prompts confirms the public key's proper installation.**
- **Repetition: Repeat the process for each remote server where you've intended to install the public key, ensuring consistent verification across all servers.**

Part 2: Discussion

Provide screenshots for each task.

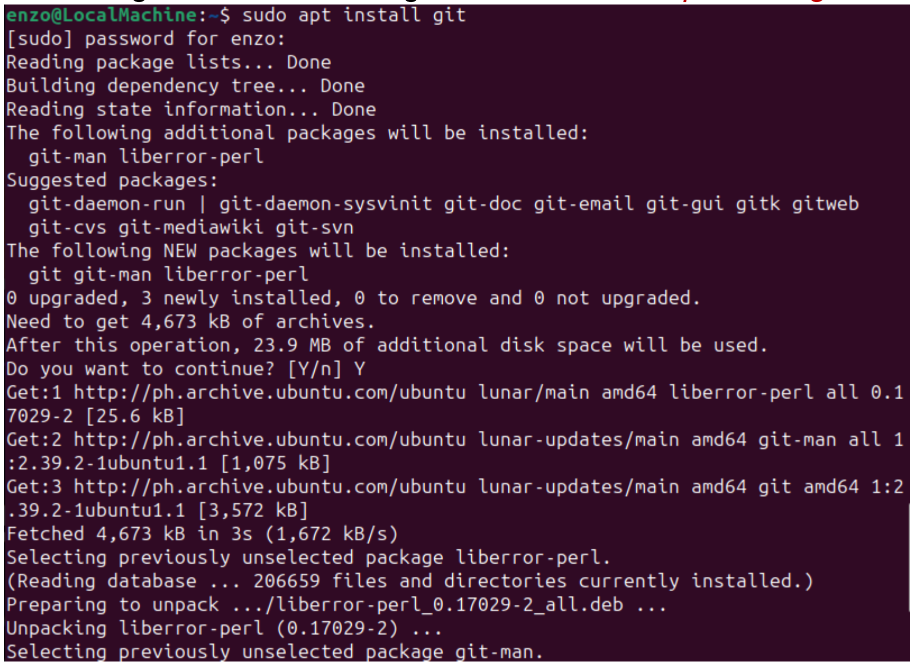
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

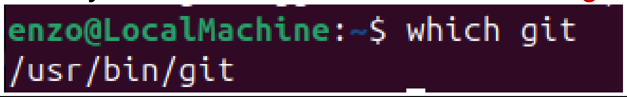
Task 3: Set up the Git Repository

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*



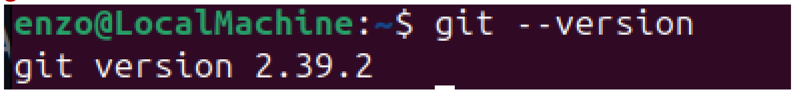
```
enzo@LocalMachine:~$ sudo apt install git
[sudo] password for enzo:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit 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 0 not upgraded.
Need to get 4,673 kB of archives.
After this operation, 23.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ph.archive.ubuntu.com/ubuntu lunar/main amd64 liberror-perl all 0.1
7029-2 [25.6 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu lunar-updates/main amd64 git-man all 1
:2.39.2-1ubuntu1.1 [1,075 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu lunar-updates/main amd64 git amd64 1:2
.39.2-1ubuntu1.1 [3,572 kB]
Fetched 4,673 kB in 3s (1,672 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 206659 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-2_all.deb ...
Unpacking liberror-perl (0.17029-2) ...
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.*



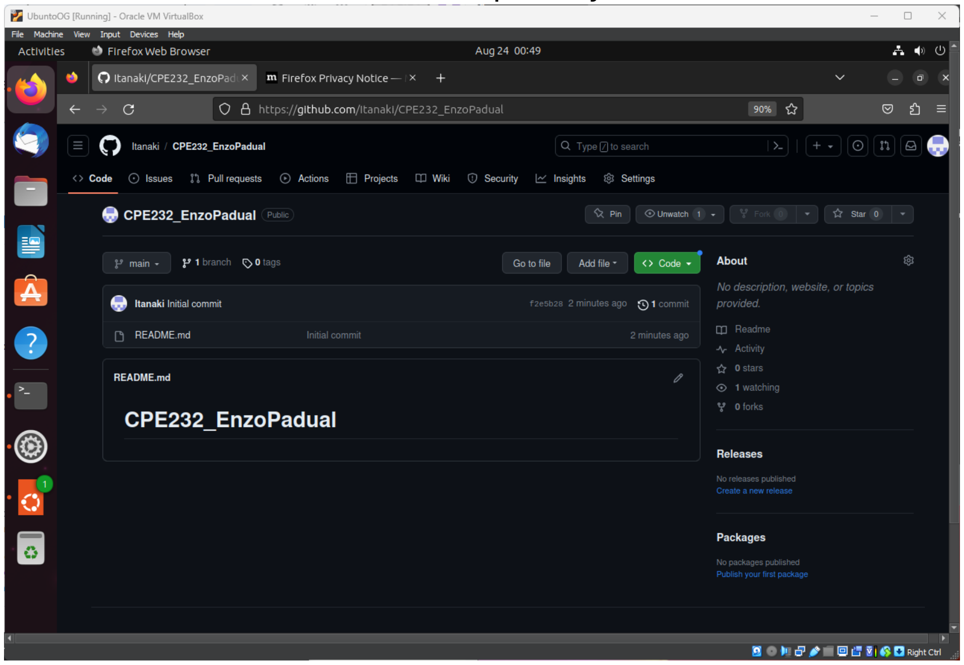
```
enzo@LocalMachine:~$ which git
/usr/bin/git
```

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

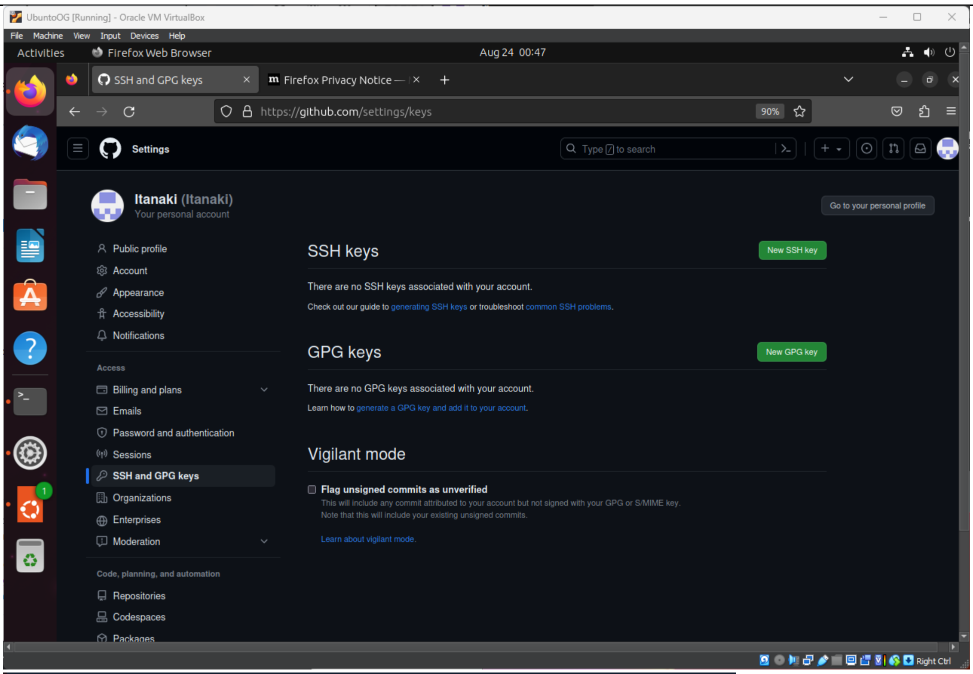


```
enzo@LocalMachine:~$ git --version
git version 2.39.2
```

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



   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.



Itanaki (Itanaki)
Your personal account

SSH keys

There are no SSH keys associated with your account.

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

GPG keys

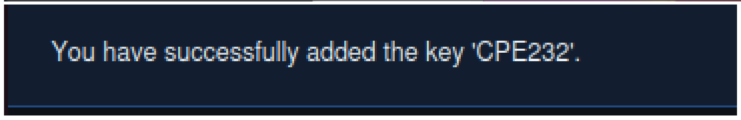
There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account.

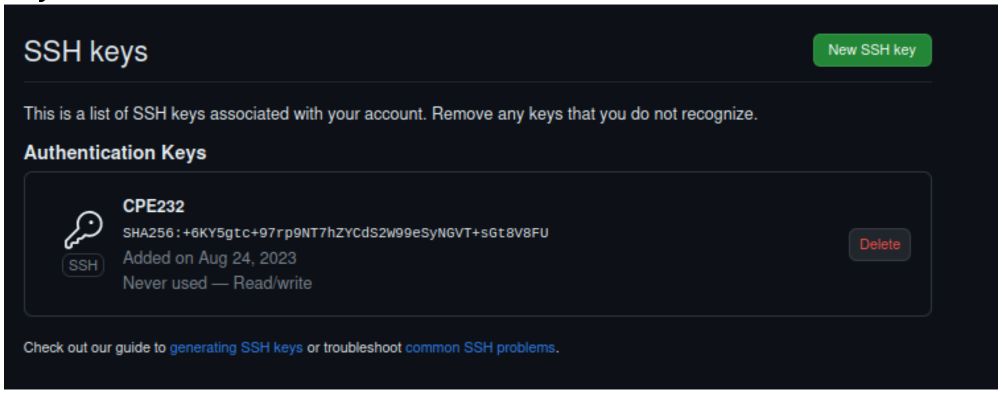
Vigilant mode

☐ Flag unsigned commits as unverified
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.

Learn about vigilant mode.


You have successfully added the key 'CPE232'.

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.



SSH keys

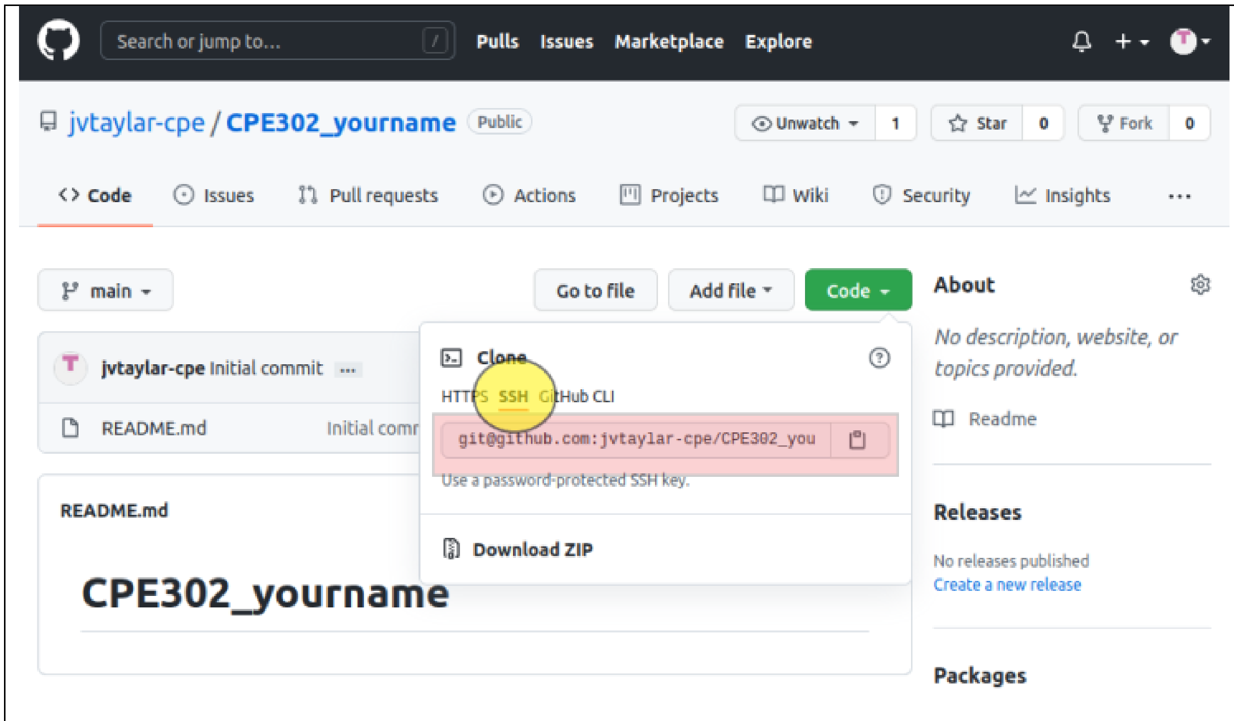
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

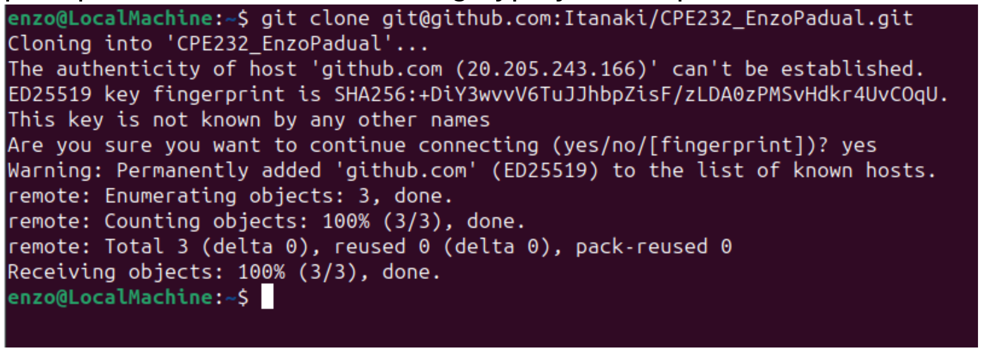
CPE232
SHA256:+6KY5gtc+97rp9NT7hZYCdS2W99eSyNGVT+sGt8V8FU
Added on Aug 24, 2023
Never used — Read/write

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

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.

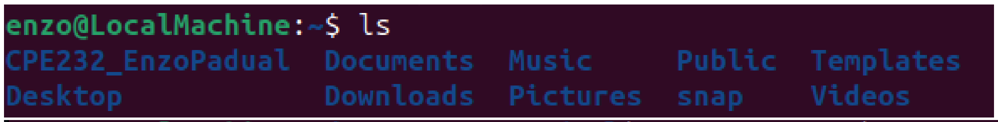


e. Issue the command git clone followed by the copied link. For example, *git clone* *git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.

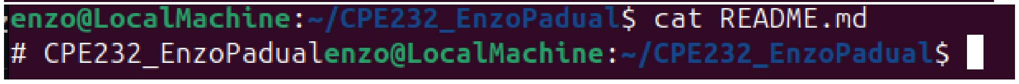


```
enzo@LocalMachine:~$ git clone git@github.com:Itanaki/CPE232_EnzoPadual.git
Cloning into 'CPE232_EnzoPadual'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any 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.
enzo@LocalMachine:~$
```

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.



```
enzo@LocalMachine:~$ ls
CPE232_EnzoPadual  Documents  Music     Public  Templates
Desktop            Downloads  Pictures  snap    Videos
```

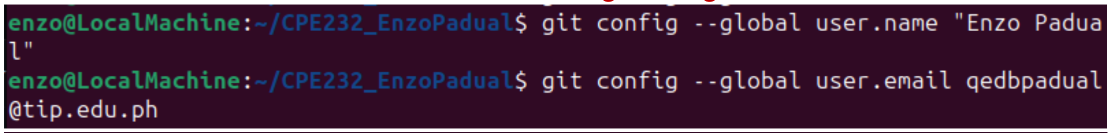


```
enzo@LocalMachine:~/CPE232_EnzoPadual$ cat README.md
# CPE232_EnzoPadualenzo@LocalMachine:~/CPE232_EnzoPadual$
```

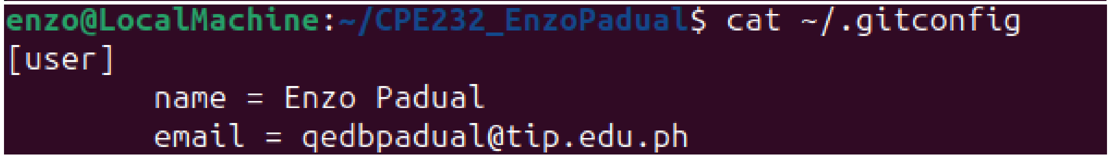
g. Use the following commands to personalize your git.

- *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*

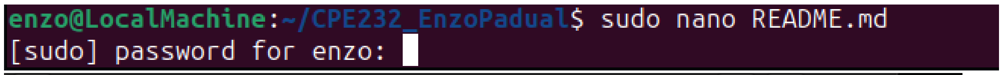


```
enzo@LocalMachine:~/CPE232_EnzoPadual$ git config --global user.name "Enzo Padua
l"
enzo@LocalMachine:~/CPE232_EnzoPadual$ git config --global user.email qedbpadual
@tip.edu.ph
```

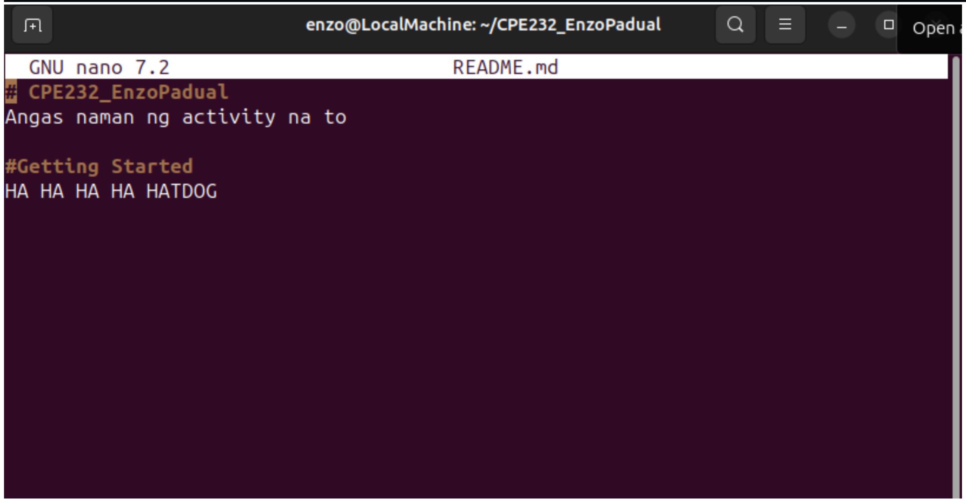


```
enzo@LocalMachine:~/CPE232_EnzoPadual$ cat ~/.gitconfig
[user]
        name = Enzo Padual
        email = qedbpadual@tip.edu.ph
```

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.

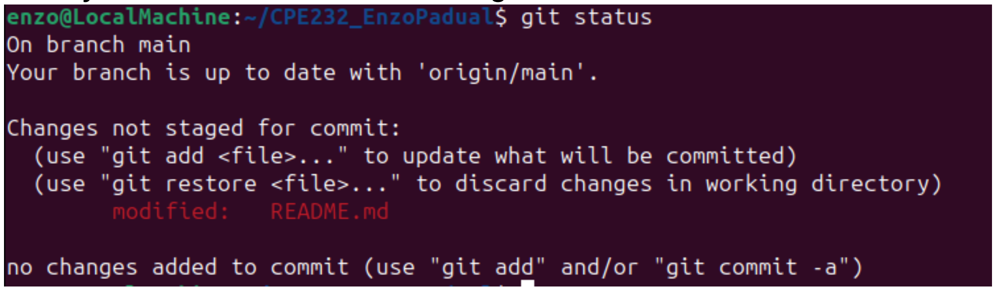


```
enzo@LocalMachine:~/CPE232_EnzoPadual$ sudo nano README.md
[sudo] password for enzo:
```



```
  GNU nano 7.2                          README.md
# CPE232_EnzoPadual
Angas naman ng activity na to

#Getting Started
HA HA HA HA HATDOG
```

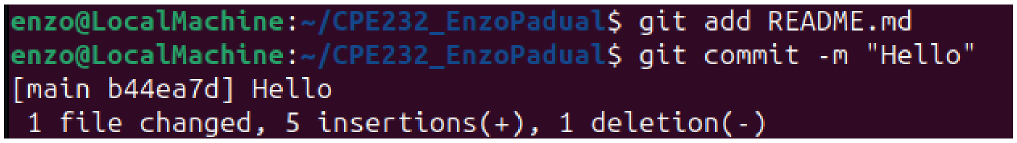
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What is the result of issuing this command?



```
enzo@LocalMachine:~/CPE232_EnzoPadual$ 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")
```

When you employ the git status command, it's akin to encapsulating the prevailing condition of your repository. It unravels the files made for committing, those that have undergone changes but are yet unsaved, and the novel files escaping tracking. Empowered with this data, you can exercise discretion regarding your commit selections and architect your subsequent maneuvers, all the while maintaining cognizance of any transformations materializing within your project.

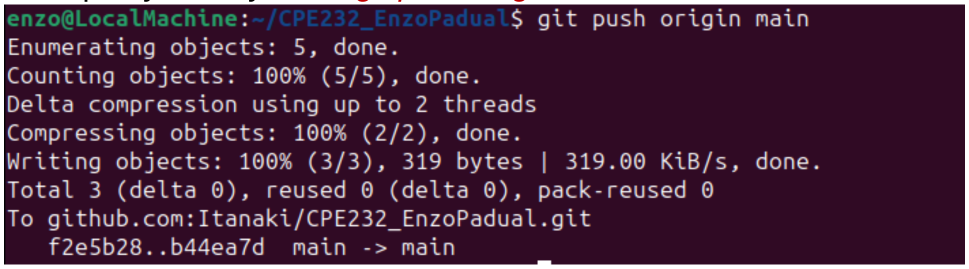
j. Use the command *git add README.md* to add the file into the staging area.
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.



```
enzo@LocalMachine:~/CPE232_EnzoPadual$ git add README.md
enzo@LocalMachine:~/CPE232_EnzoPadual$ git commit -m "Hello"
[main b44ea7d] Hello
 1 file changed, 5 insertions(+), 1 deletion(-)
```

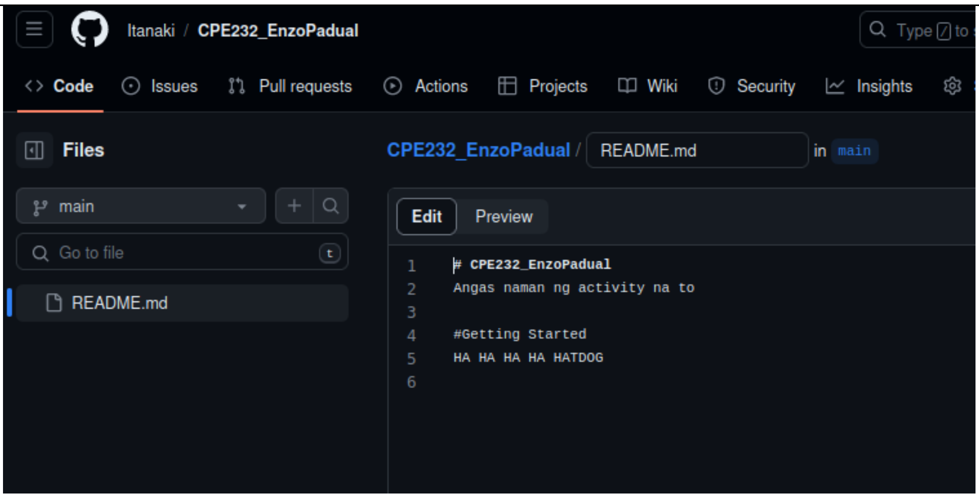
The -m is an option for users to provide short descriptions of the changes in the commit.

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.*



```
enzo@LocalMachine:~/CPE232_EnzoPadual$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Itanaki/CPE232_EnzoPadual.git
   f2e5b28..b44ea7d  main -> main
```

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.



```
# CPE232_EnzoPadual
Angas naman ng activity na to

#Getting Started
HA HA HA HA HATDOG

```

Reflections:

Answer the following:

1. What sort of things have we so far done to the remote servers using ansible commands?

   We were able to connect to github and access and changed the contents of the repository using the terminal of Ubuntu. I can remotely access different servers by knowing their IP addresses, and make an SSH key that will be a key for permanent authentication, so it does not need password every time you want to access the server remotely.
2. How important is the inventory file?

   The inventory file is like a core piece for remote servers and ansible commands. It's like the base for talking about target servers and what they're like – you know, names, IP stuff, and all that. This inventory part really matters 'cause it's what gives ansible the info it needs to do tasks better and smoother across all the servers. Makes managing and running things on the servers clear and easy, kinda automatic too.

Conclusions/Learnings:

During this laboratory endeavor, I have acquired the art of deploying SSH for the establishment of passphrases, discreetly enshrining them within files. This artifice empowers my computing machinery to securely safeguard these veritable keys of access for subsequent allusions. Furthermore, I've delved into the symbiotic junction of Git and Github, orchestrating an assembly of my endeavors within the confines of Linux Ubuntu on my desktop, ultimately ushering them onto the realm of my Github profile. This orchestration transcends mere confinement to my Oracle virtual desktop, furnishing me with a medium that traverses diverse platforms. This symphony of integration begets a facile retrospective, aiding in scrutinizing miscues and avenues for refinement within this professional trajectory.