Name: Mark Andrei Ponayo	**Date Performed:** August 28, 2023
Course/Section: BSCPE31S5	**Date Submitted:**
Instructor: Engr. Roman Richard	**Semester and SY:** 3rd Year 1st Sem

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

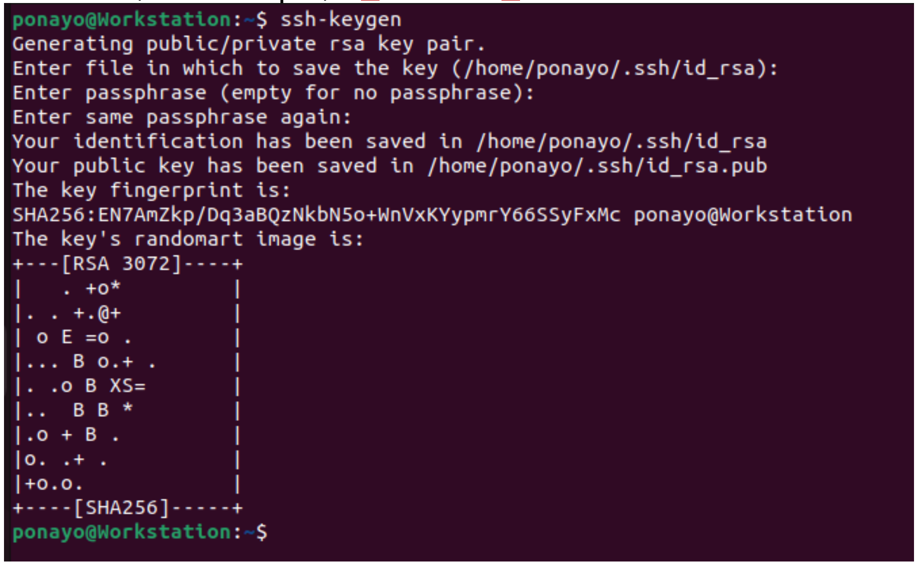
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

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.

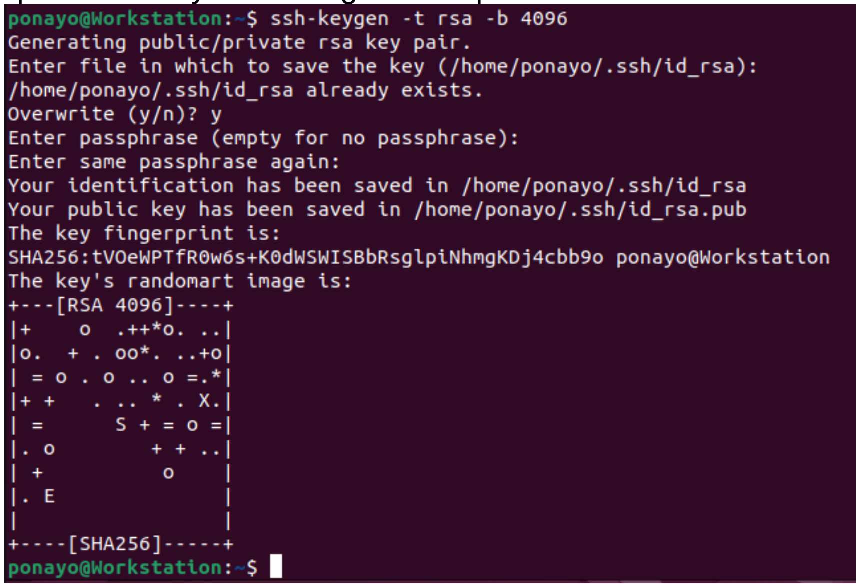
Task 1: Create an SSH Key Pair for User Authentication

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*.



```
ponayo@Workstation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ponayo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ponayo/.ssh/id_rsa
Your public key has been saved in /home/ponayo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:EN7AmZkp/Dq3aBQzNkbN5o+WnVxKYypmrY66SSyFxMc ponayo@Workstation
The key's randomart image is:
+---[RSA 3072]----+
|   . +o*         |
|. . +.@+         |
| o E =o .        |
|... B o.+ .      |
|. .o B XS=       |
|..  B B *        |
|.o + B .         |
|o. .+ .          |
|+o.o.            |
+----[SHA256]-----+
ponayo@Workstation:~$
```

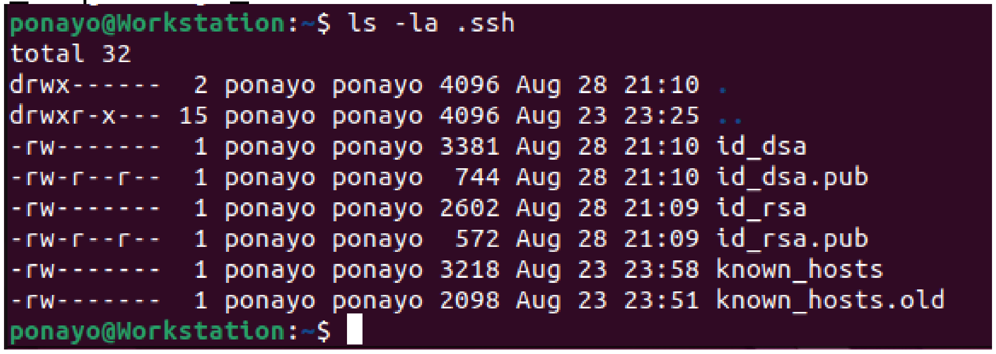
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.



```
ponayo@Workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ponayo/.ssh/id_rsa):
/home/ponayo/.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 /home/ponayo/.ssh/id_rsa
Your public key has been saved in /home/ponayo/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tVOeWPTfR0w6s+K0dWSWISBbRsglpiNhmgKDj4cbb9o ponayo@Workstation
The key's randomart image is:
+---[RSA 4096]----+
|+    o  .++*o. ..|
|o.  + . oo*. ..+o|
| = o . o .. o =.*|
|+ +   . .. * . X.|
| =      S + = o =|
|. o        + + ..|
| +          o    |
|. E              |
|                 |
+----[SHA256]-----+
ponayo@Workstation:~$
```

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.

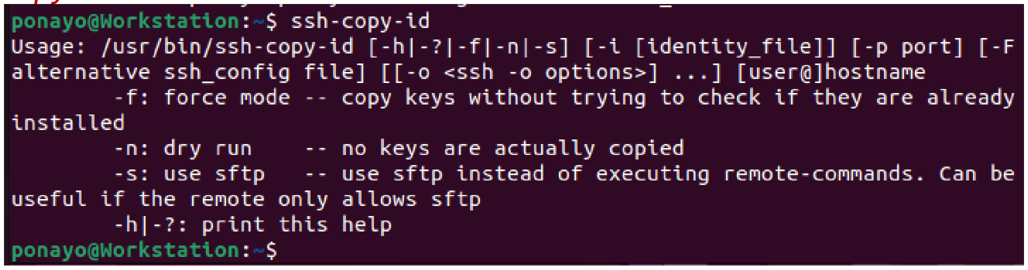
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
ponayo@Workstation:~$ ls -la .ssh
total 32
drwx------  2 ponayo ponayo 4096 Aug 28 21:10 .
drwxr-x--- 15 ponayo ponayo 4096 Aug 23 23:25 ..
-rw-------  1 ponayo ponayo 3381 Aug 28 21:10 id_dsa
-rw-r--r--  1 ponayo ponayo  744 Aug 28 21:10 id_dsa.pub
-rw-------  1 ponayo ponayo 2602 Aug 28 21:09 id_rsa
-rw-r--r--  1 ponayo ponayo  572 Aug 28 21:09 id_rsa.pub
-rw-------  1 ponayo ponayo 3218 Aug 23 23:58 known_hosts
-rw-------  1 ponayo ponayo 2098 Aug 23 23:51 known_hosts.old
ponayo@Workstation:~$
```

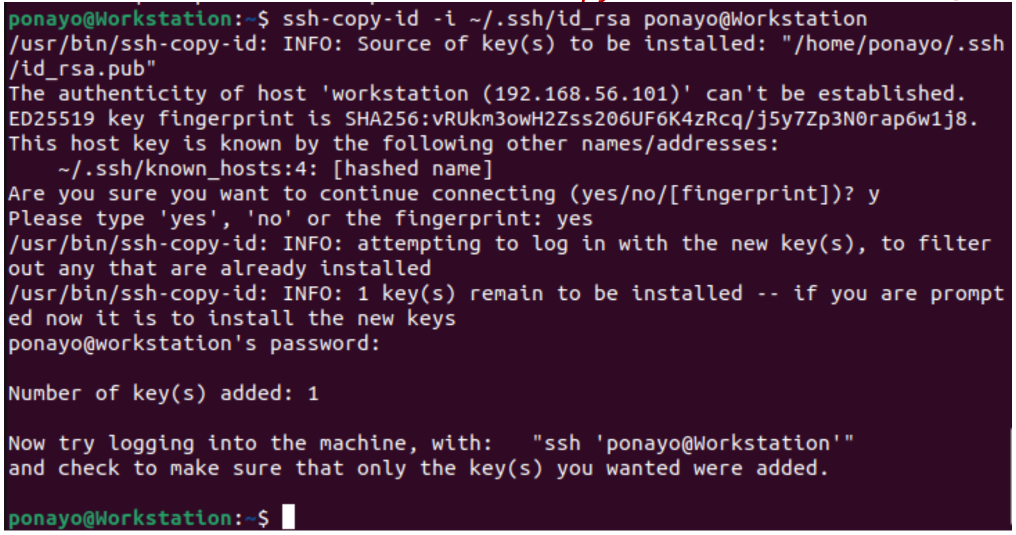
Task 2: Copying the Public Key to the remote servers

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.



```
ponayo@Workstation:~$ 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 allows sftp
        -h|-?: print this help
ponayo@Workstation:~$
```

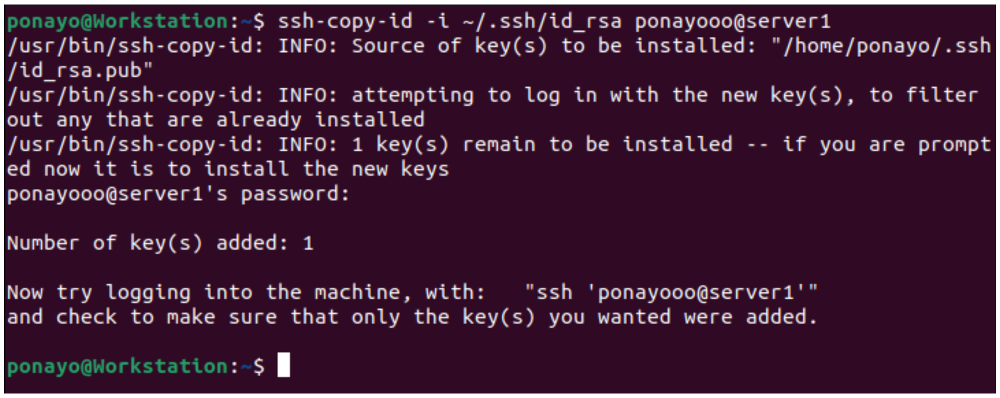
2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*



```
ponayo@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ponayo@Workstation
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ponayo/.ssh
/id_rsa.pub"
The authenticity of host 'workstation (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:vRUkm3owH2Zss206UF6K4zRcq/j5y7Zp3N0rap6w1j8.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the 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
ponayo@workstation's password:

Number of key(s) added: 1

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

ponayo@Workstation:~$
```

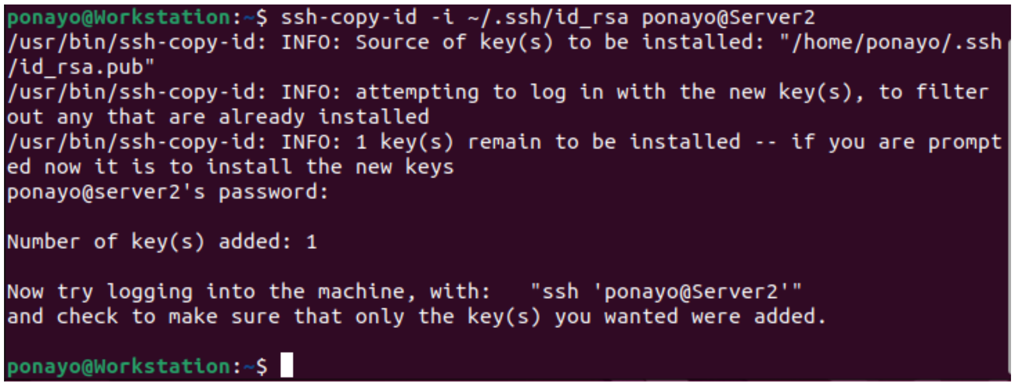
3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.

Live Server1:



```
ponayo@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ponayooo@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ponayo/.ssh
/id_rsa.pub"
/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
ponayooo@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ponayooo@server1'"
and check to make sure that only the key(s) you wanted were added.

ponayo@Workstation:~$ 
```

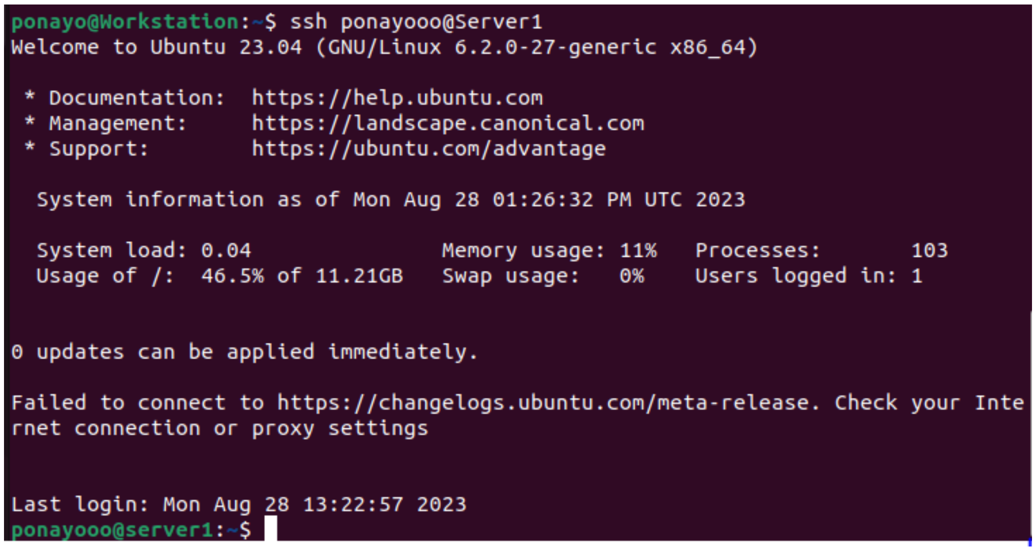
Live Server2



```
ponayo@Workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa ponayo@Server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ponayo/.ssh
/id_rsa.pub"
/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
ponayo@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ponayo@Server2'"
and check to make sure that only the key(s) you wanted were added.

ponayo@Workstation:~$ 
```

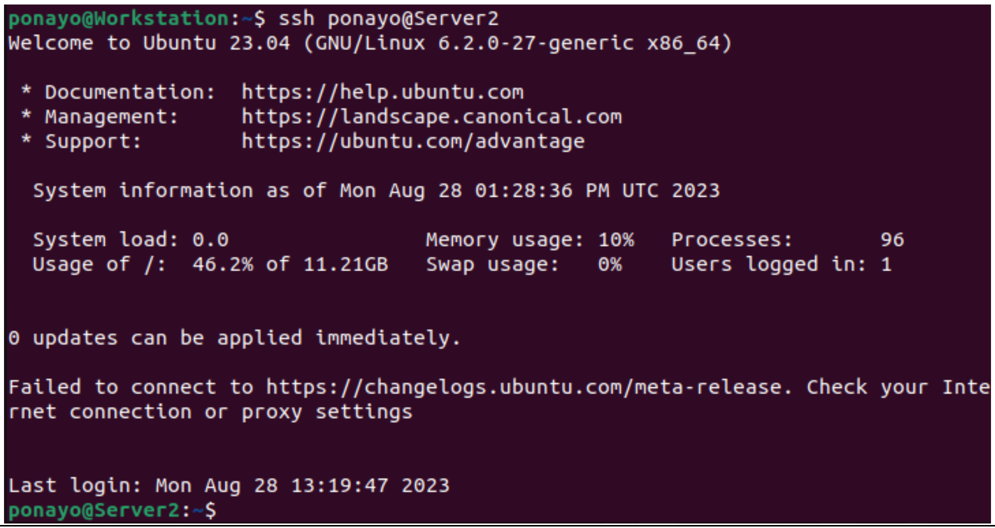
4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

Live Server1



```
ponayo@Workstation:~$ ssh ponayooo@Server1
Welcome to Ubuntu 23.04 (GNU/Linux 6.2.0-27-generic x86_64)

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

  System information as of Mon Aug 28 01:26:32 PM UTC 2023

  System load: 0.04              Memory usage: 11%   Processes:       103
  Usage of /:  46.5% of 11.21GB  Swap usage:   0%    Users logged in: 1


0 updates can be applied immediately.

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


Last login: Mon Aug 28 13:22:57 2023
ponayooo@server1:~$ 
```

Live Server2



```
ponayo@Workstation:~$ ssh ponayo@Server2
Welcome to Ubuntu 23.04 (GNU/Linux 6.2.0-27-generic x86_64)

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

  System information as of Mon Aug 28 01:28:36 PM UTC 2023

  System load: 0.0              Memory usage: 10%   Processes:       96
  Usage of /:  46.2% of 11.21GB  Swap usage:   0%    Users logged in: 1


0 updates can be applied immediately.

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


Last login: Mon Aug 28 13:19:47 2023
ponayo@Server2:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   - The ssh program does is it securs and interactive file transfers sessions. It can also automated and secures file transers.
2. How do you know that you already installed the public key to the remote servers?
   - If you type the command “ssh-copy-id -I /.ssh/id_rsa user@host” it will show the log detail installation of the public key.

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**).

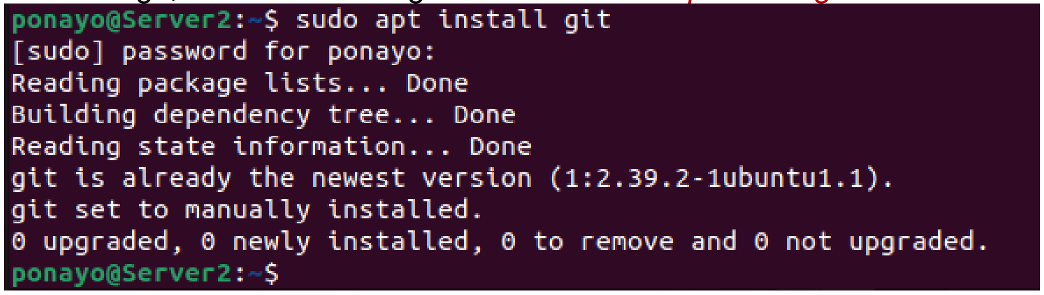
Set up Git

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

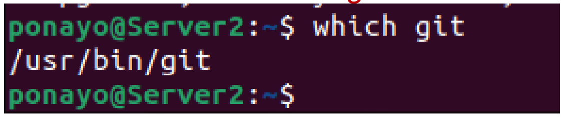
Task 3: Set up the Git Repository

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*



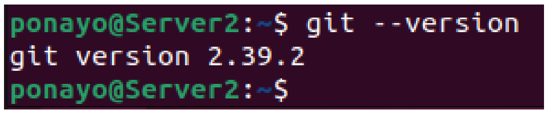
```
ponayo@Server2:~$ sudo apt install git
[sudo] password for ponayo:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.39.2-1ubuntu1.1).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ponayo@Server2:~$
```

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



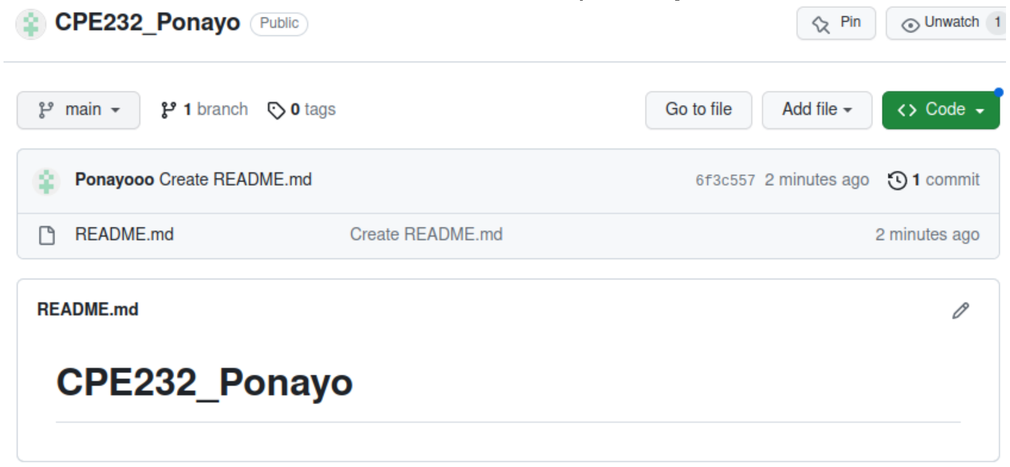
```
ponayo@Server2:~$ which git
/usr/bin/git
ponayo@Server2:~$
```

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



```
ponayo@Server2:~$ git --version
git version 2.39.2
ponayo@Server2:~$
```

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.



CPE232_Ponayo Public

Pin | Unwatch 1

main | 1 branch | 0 tags | Go to file | Add file | Code

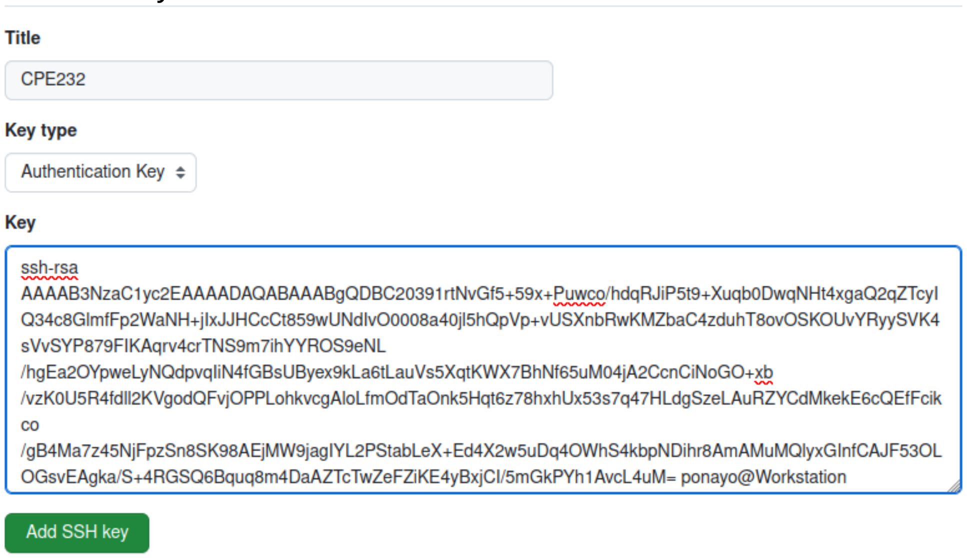
Ponayooo Create README.md 6f3c557 2 minutes ago 1 commit

README.md Create README.md 2 minutes ago

README.md

CPE232_Ponayo

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.



Title

CPE232

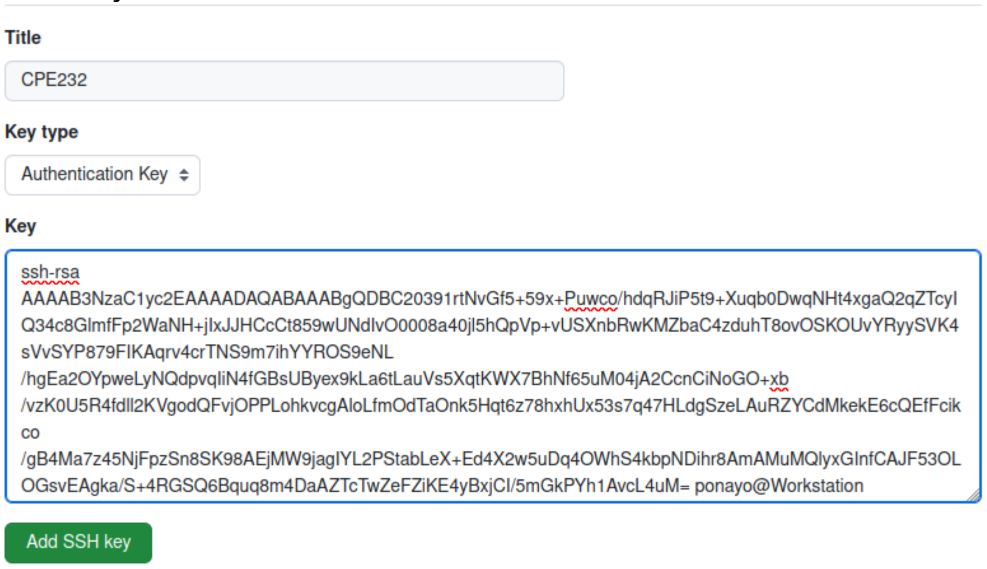
Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDBC20391rtNvGf5+59x+Puwco/hdqRJiP5t9+Xuqb0DwqNHt4xgaQ2qZTcyl
Q34c8GlmfFp2WaNH+jlxJJHCcCt859wUNdlvO0008a40jl5hQpVp+vUSXnbRwKMZbaC4zduhT8ovOSKOUvYRyySVK4
sVvSYP879FIKAqrv4crTNS9m7ihYYROS9eNL
/hgEa2OYpweLyNQdpvqliN4fGBsUByex9kLa6tLauVs5XqtKWX7BhNf65uM04jA2CcnCiNoGO+xb
/vzK0U5R4fdll2KVgodQFvjOPPLohkvcgAloLfmOdTaOnk5Hqt6z78hxhUx53s7q47HLdgSzeLAuRZYCdMkekE6cQEfFcik
co
/gB4Ma7z45NjFpzSn8SK98AEjMW9jaglYL2PStabLeX+Ed4X2w5uDq4OWhS4kbpNDihr8AmAMuMQlyxGlnfCAJF53OL
OGsvEAgka/S+4RGSQ6Bquq8m4DaAZTcTwZeFZiKE4yBxjCl/5mGkPYh1AvcL4uM= ponayo@Workstation
```

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.



Title

CPE232

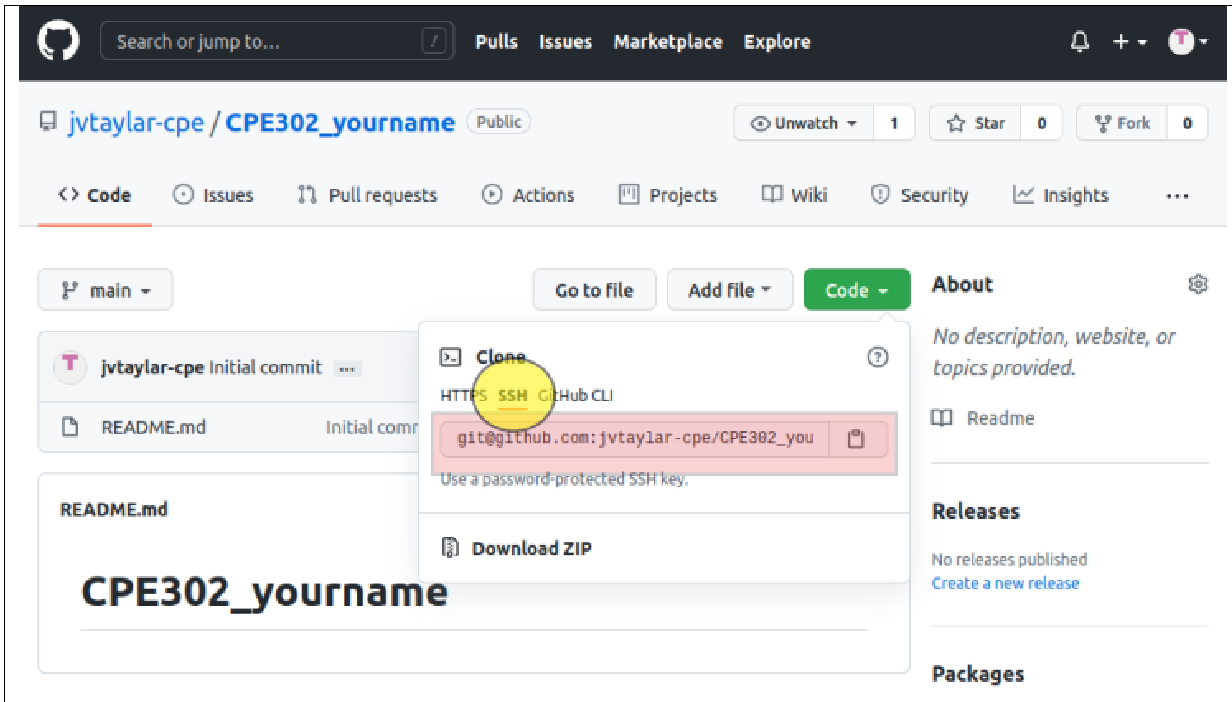
Key type

Authentication Key

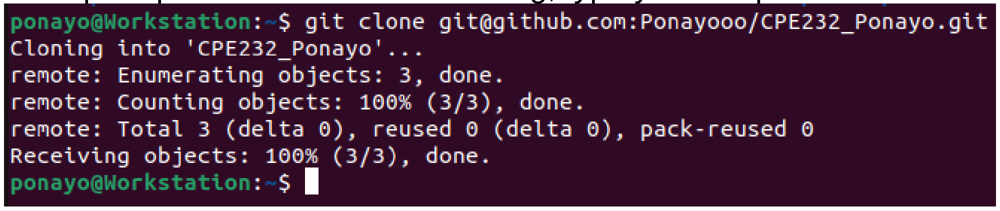
Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDBC20391rtNvGf5+59x+Puwco/hdqRJiP5t9+Xuqb0DwqNHt4xgaQ2qZTcyl
Q34c8GlmfFp2WaNH+jlxJJHCcCt859wUNdlvO0008a40jl5hQpVp+vUSXnbRwKMZbaC4zduhT8ovOSKOUvYRyySVK4
sVvSYP879FIKAqrv4crTNS9m7ihYYROS9eNL
/hgEa2OYpweLyNQdpvqliN4fGBsUByex9kLa6tLauVs5XqtKWX7BhNf65uM04jA2CcnCiNoGO+xb
/vzK0U5R4fdll2KVgodQFvjOPPLohkvcgAloLfmOdTaOnk5Hqt6z78hxhUx53s7q47HLdgSzeLAuRZYCdMkekE6cQEfFcik
co
/gB4Ma7z45NjFpzSn8SK98AEjMW9jaglYL2PStabLeX+Ed4X2w5uDq4OWhS4kbpNDihr8AmAMuMQlyxGlnfCAJF53OL
OGsvEAgka/S+4RGSQ6Bquq8m4DaAZTcTwZeFZiKE4yBxjCl/5mGkPYh1AvcL4uM= ponayo@Workstation
```

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.

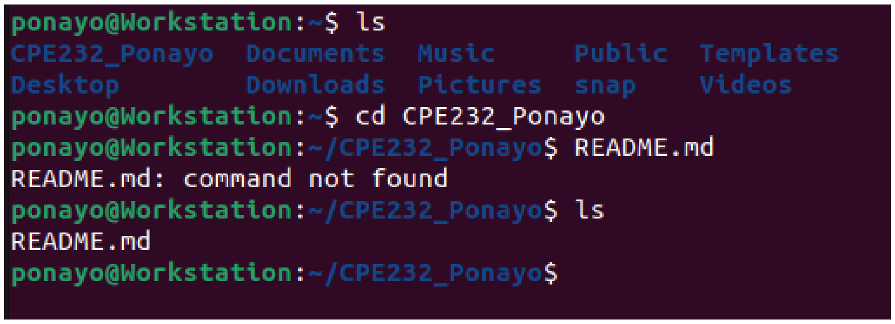


e. Issue the command git clone followed by the copied link. For example, *git clone* [*git@github.com:jvtaylar-cpe/CPE232_yourname.git*](git@github.com:jvtaylar-cpe/CPE232_yourname.git). When prompted to continue connecting, type yes and press enter.



```
ponayo@Workstation:~$ git clone git@github.com:Ponayooo/CPE232_Ponayo.git
Cloning into 'CPE232_Ponayo'...
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.
ponayo@Workstation:~$
```

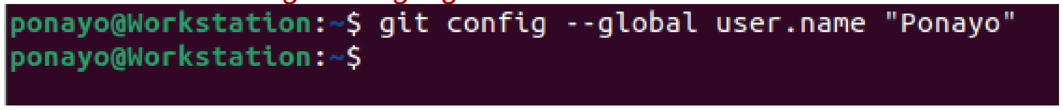
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
ponayo@Workstation:~$ ls
CPE232_Ponayo  Documents  Music     Public  Templates
Desktop        Downloads  Pictures  snap    Videos
ponayo@Workstation:~$ cd CPE232_Ponayo
ponayo@Workstation:~/CPE232_Ponayo$ README.md
README.md: command not found
ponayo@Workstation:~/CPE232_Ponayo$ ls
README.md
ponayo@Workstation:~/CPE232_Ponayo$
```

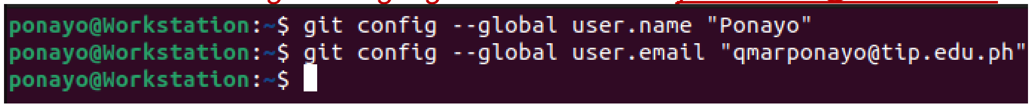
g. Use the following commands to personalize your git.

- *git config --global user.name "Your Name"*



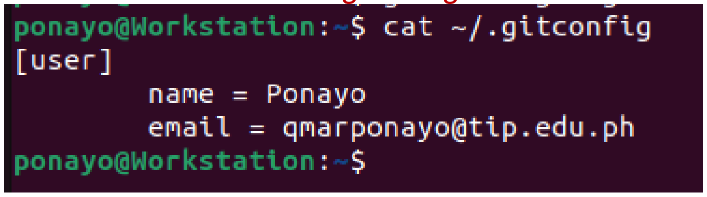
```
ponayo@Workstation:~$ git config --global user.name "Ponayo"
ponayo@Workstation:~$
```

- *git config --global user.email yourname@email.com*



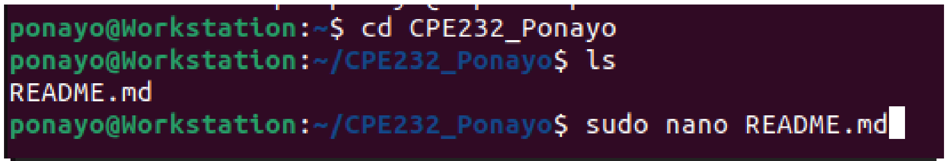
```
ponayo@Workstation:~$ git config --global user.name "Ponayo"
ponayo@Workstation:~$ git config --global user.email "qmarponayo@tip.edu.ph"
ponayo@Workstation:~$
```

- Verify that you have personalized the config file using the command *cat ~/.gitconfig*

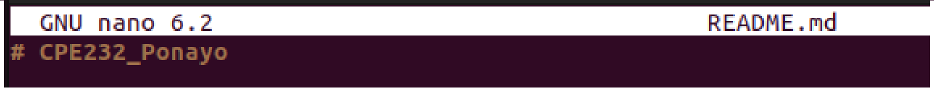


```
ponayo@Workstation:~$ cat ~/.gitconfig
[user]
        name = Ponayo
        email = qmarponayo@tip.edu.ph
ponayo@Workstation:~$
```

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.

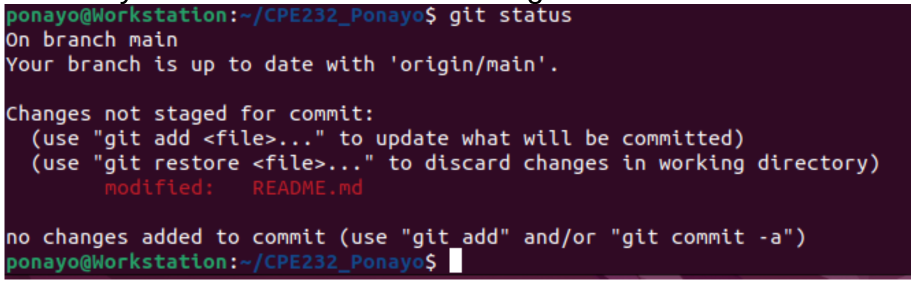


```
ponayo@Workstation:~$ cd CPE232_Ponayo
ponayo@Workstation:~/CPE232_Ponayo$ ls
README.md
ponayo@Workstation:~/CPE232_Ponayo$ sudo nano README.md
```



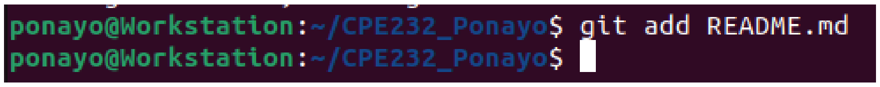
```
  GNU nano 6.2                                    README.md
# CPE232_Ponayo
```

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?



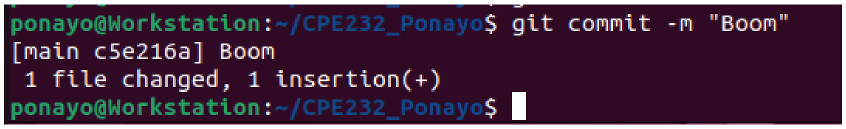
```
ponayo@Workstation:~/CPE232_Ponayo$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
ponayo@Workstation:~/CPE232_Ponayo$
```

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



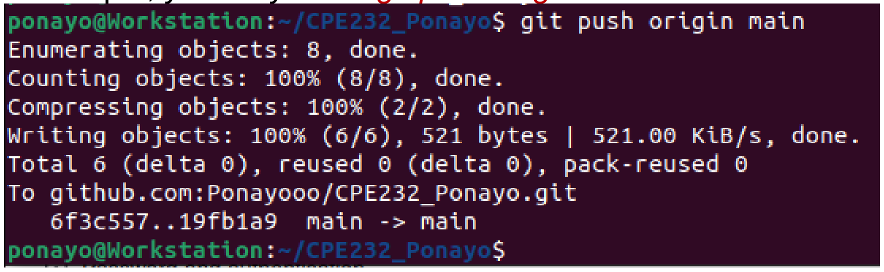
```
ponayo@Workstation:~/CPE232_Ponayo$ git add README.md
ponayo@Workstation:~/CPE232_Ponayo$
```

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.



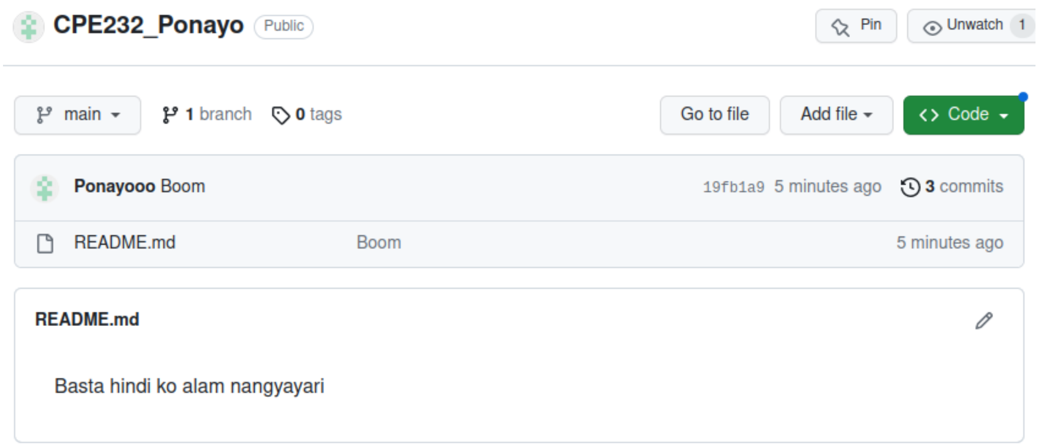
```
ponayo@Workstation:~/CPE232_Ponayo$ git commit -m "Boom"
[main c5e216a] Boom
 1 file changed, 1 insertion(+)
ponayo@Workstation:~/CPE232_Ponayo$
```

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.*



```
ponayo@Workstation:~/CPE232_Ponayo$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 521 bytes | 521.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ponayooo/CPE232_Ponayo.git
   6f3c557..19fb1a9  main -> main
ponayo@Workstation:~/CPE232_Ponayo$
```

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.



Ponayooo Boom 19fb1a9 5 minutes ago 3 commits

README.md Boom 5 minutes ago

README.md

Basta hindi ko alam nangyayari

Reflections:

Answer the following:

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

Using Ansible commands, we've accomplished several on remote servers. Firstly, we've configured server wide settings by changing system parameters and setups. Also, software packages and updates have been efficiently installed and managed across many servers, securing consistent Environment. Ansible enabled us to deploy application perfectly, Ensuring they run Optimally through configuration Adjustments. Security patches have been applied, helping server defenses. These actions have all together improved server performance, security, maintainability while Streamlining Administrative tasks.

4. How important is the inventory file?

The inventory file holds critical importance as it serves as a basic record of available valuable things, products, or resources within a business or system. A well-maintained inventory file improves enhances decision-making processes, minimizes stockouts, and optimizes useful resources.

Conclusions/Learnings:

On this activity, I learned on how to connect github into the linux operating system. Learning on how to connect the github is a huge benefit to me since it might help me in the near future.