Name: John Erwin S. Ballesteros	**Date Performed:** 9/9/2024
Course/Section: CpE 31S2	**Date Submitted:** 9/10/2024
Instructor: Mr. Robin Valenzuela	**Semester and SY:** 1st Sem - 24'-25'

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).**

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.

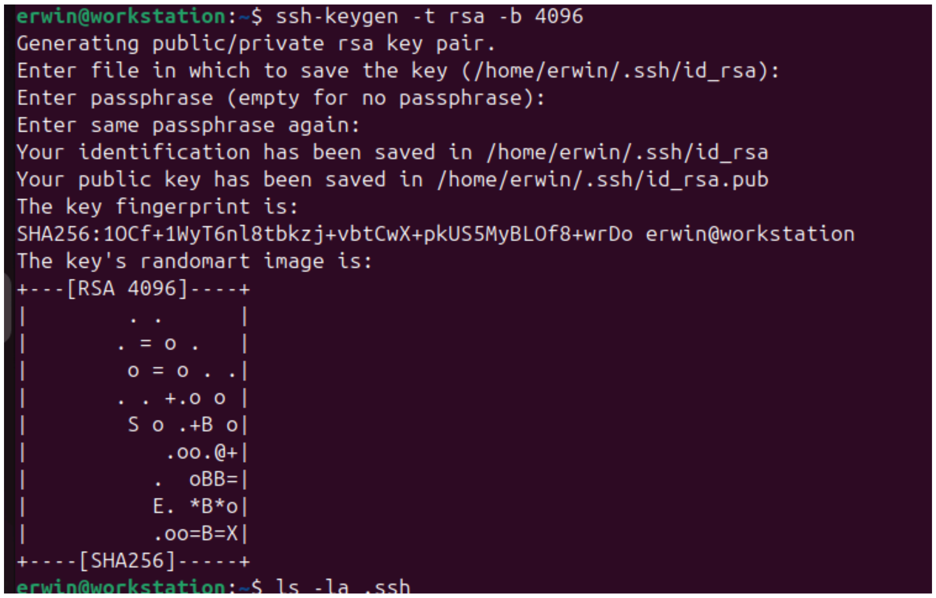
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Task 1: Create an SSH Key Pair for User Authentication

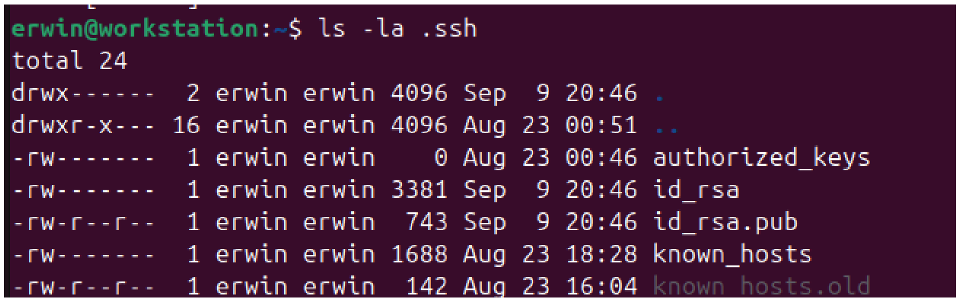
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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.
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 -LA and id_rsa.



```
erwin@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/erwin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/erwin/.ssh/id_rsa
Your public key has been saved in /home/erwin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1OCf+1WyT6nl8tbkzj+vbtCwX+pkUS5MyBLOf8+wrDo erwin@workstation
The key's randomart image is:
+---[RSA 4096]----+
|        . .      |
|       . = o .   |
|        o = o . .|
|       . . +.o o |
|        S o .+B o|
|           .oo.@+|
|            . oBB=|
|          E. *B*o|
|          .oo=B=X|
+----[SHA256]-----+
erwin@workstation:~$ ls -la .ssh
```



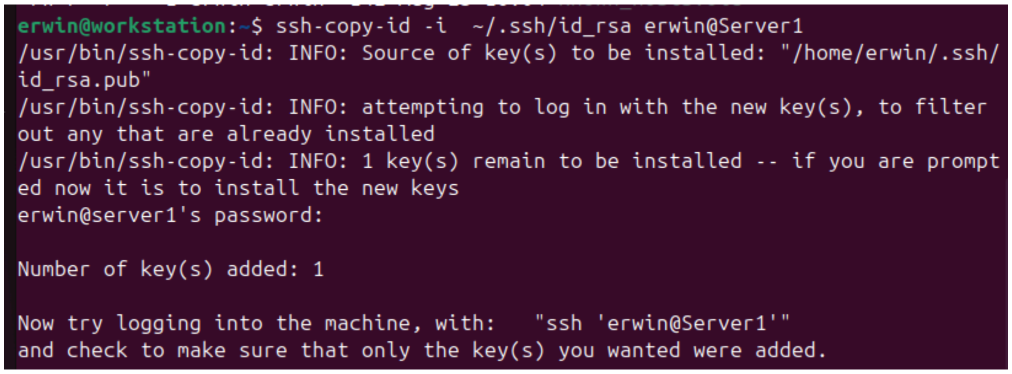
```
erwin@workstation:~$ ls -la .ssh
total 24
drwx------  2 erwin erwin 4096 Sep  9 20:46 .
drwxr-x--- 16 erwin erwin 4096 Aug 23 00:51 ..
-rw-------  1 erwin erwin    0 Aug 23 00:46 authorized_keys
-rw-------  1 erwin erwin 3381 Sep  9 20:46 id_rsa
-rw-r--r--  1 erwin erwin  743 Sep  9 20:46 id_rsa.pub
-rw-------  1 erwin erwin 1688 Aug 23 18:28 known_hosts
-rw-r--r--  1 erwin erwin  142 Aug 23 16:04 known_hosts.old
```

Figure 1 & 2 Creating SSH Keygen

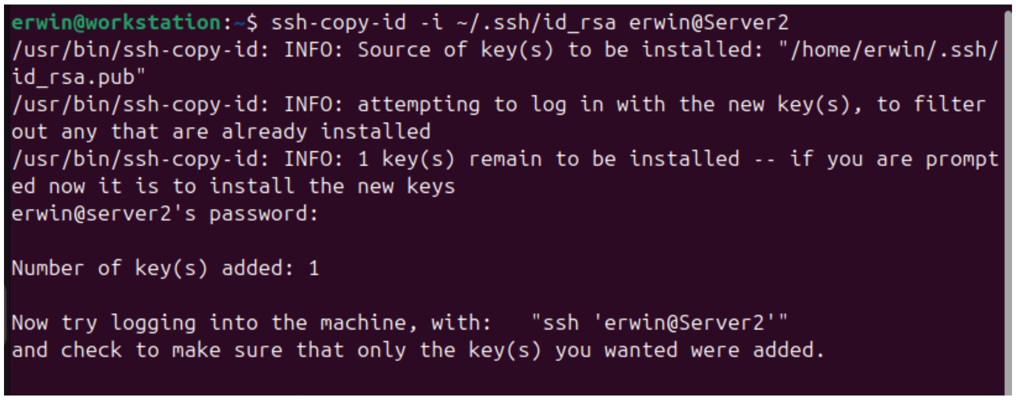
Task 2: Copying the Public Key to the remote servers

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.
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?

   When performing the common 'ssh user@host' the CLI automatically connects it and does not ask for password. it did not ask for password because the copied ssh key to the remote server is checking the connection between the user and remote server.

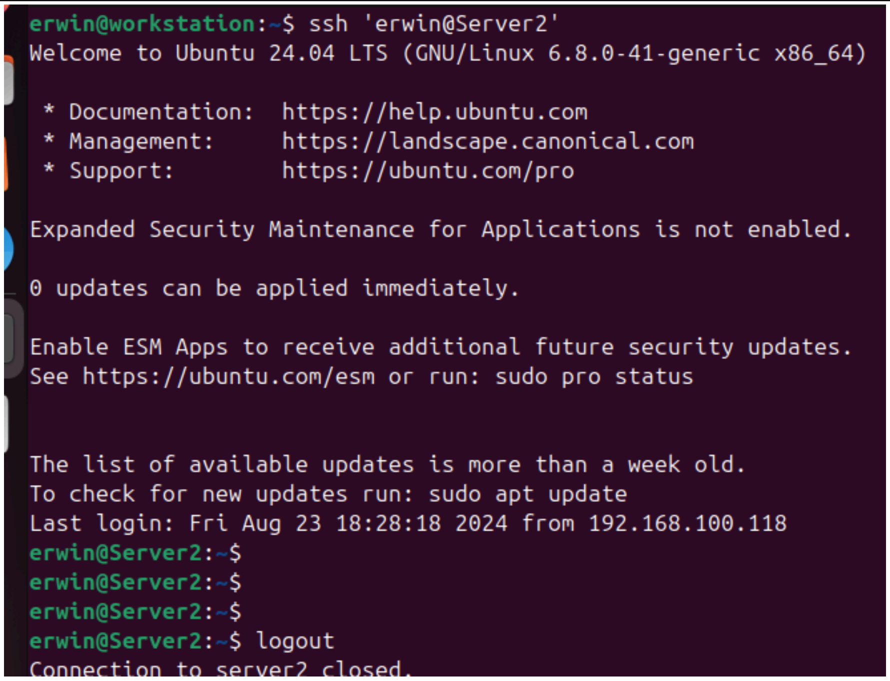


```
erwin@workstation:~$ ssh-copy-id -i  ~/.ssh/id_rsa erwin@Server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/erwin/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
erwin@server1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'erwin@Server1'"
and check to make sure that only the key(s) you wanted were added.
```

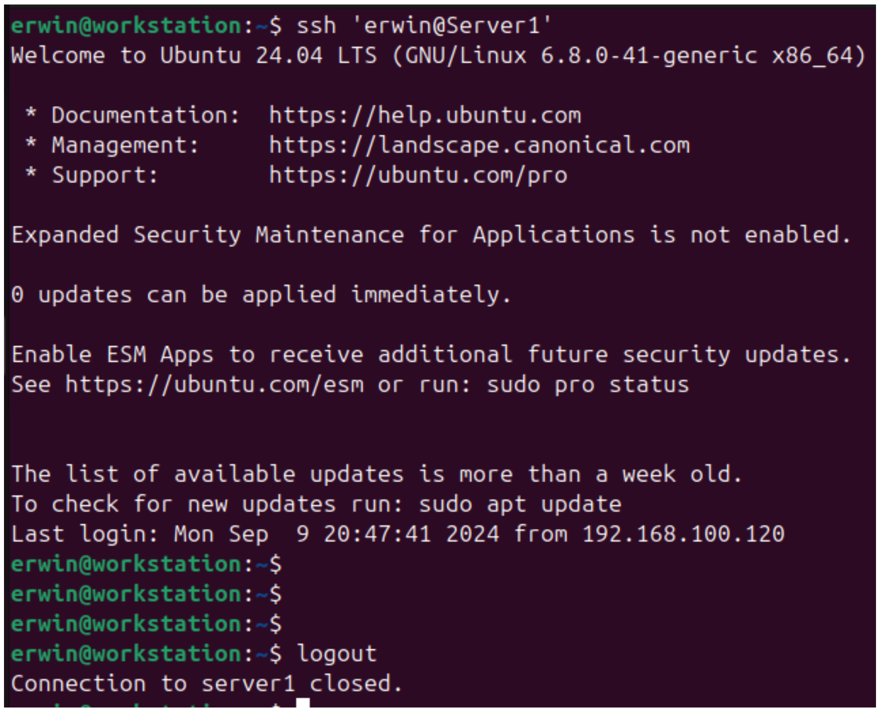


```
erwin@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa erwin@Server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/erwin/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
erwin@server2's password:

Number of key(s) added: 1

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

FIgure 1 & 2 Copying the SSH keygen for Server 1 and 2



```
erwin@workstation:~$ ssh 'erwin@Server2'
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-41-generic x86_64)

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

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Fri Aug 23 18:28:18 2024 from 192.168.100.118
erwin@Server2:~$
erwin@Server2:~$
erwin@Server2:~$
erwin@Server2:~$ logout
Connection to server2 closed.
```



```
erwin@workstation:~$ ssh 'erwin@Server1'
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-41-generic x86_64)

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

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Mon Sep  9 20:47:41 2024 from 192.168.100.120
erwin@workstation:~$
erwin@workstation:~$
erwin@workstation:~$
erwin@workstation:~$ logout
Connection to server1 closed.
```

Figure 3 & 4 Verifying the Connection for Server 1 and 2

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   Secure shell (ssh) is a network protocol in Linux that can give us users a secure method to manage remote servers in an unsafe network. SSH can do this by the use of cryptography that includes password and public key.

2. How do you know that you already installed the public key to the remote servers?

   We can determine that we have installed the public key to the remove server by logging in it as usual. If it does not ask for the passkey of the machine, it means it has been successfully installed. We can also check it by looking inside the authorized keys and if our appropriate key is inside the directory.

Part 2: Discussion

Provide screenshots for each task. Screenshots are at the bottom.

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

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

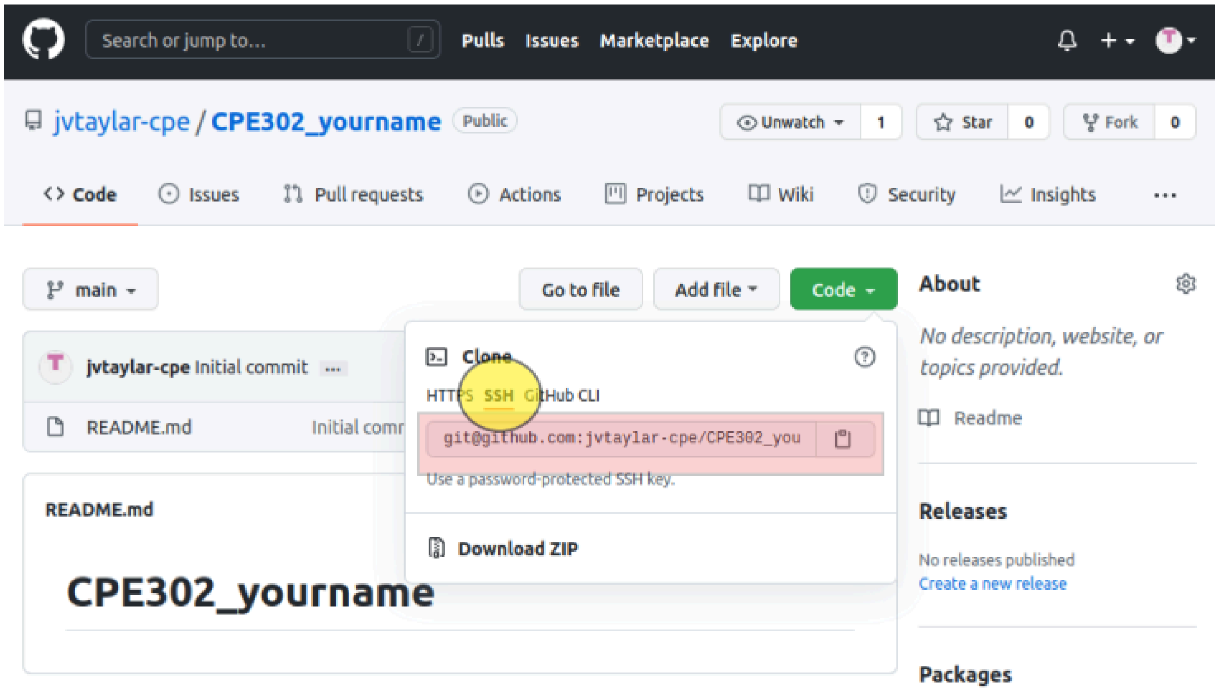
- Creating a repository
- Forking a repository
- Managing files
- Being social

Task 3: Set up the Git Repository

1. On the local machine, verify the version of your git using the command *which git.* If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git.*
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.
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.

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.

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.



git@github.com:jvtaylar-cpe/CPE302_you

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.

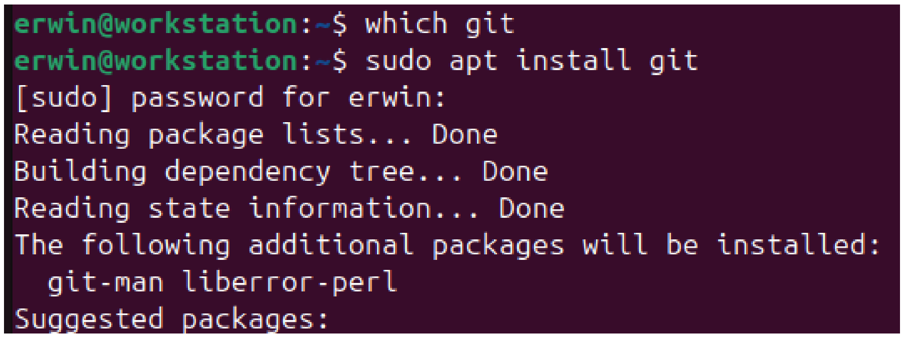
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

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*

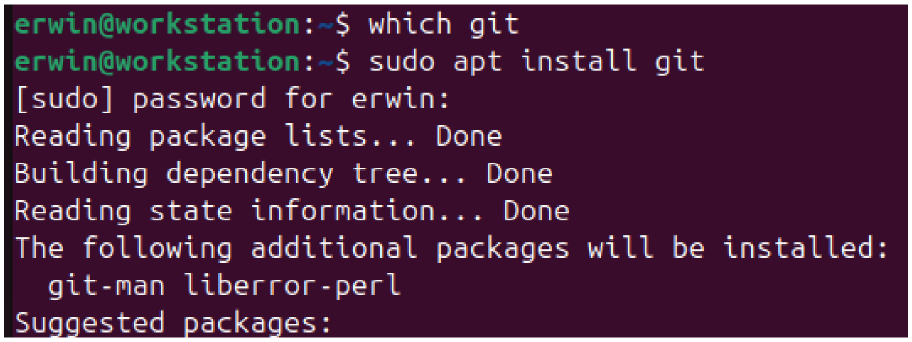
h. Edit the README.md file using nano command. Provide any information on the markdown file pertaining to the repository you created. Make sure to write out or save the file and exit.

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?
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.
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.*
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.

Screenshots

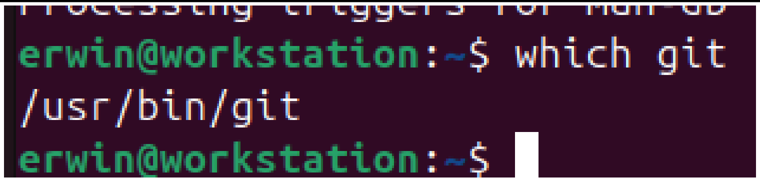


```
erwin@workstation:~$ which git
erwin@workstation:~$ sudo apt install git
[sudo] password for erwin:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
```

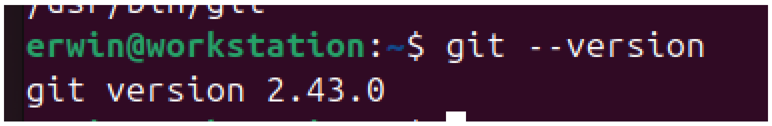


```
erwin@workstation:~$ which git
erwin@workstation:~$ sudo apt install git
[sudo] password for erwin:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
```

Figure 1 & 2: Installation and Checking of Git on CLI.

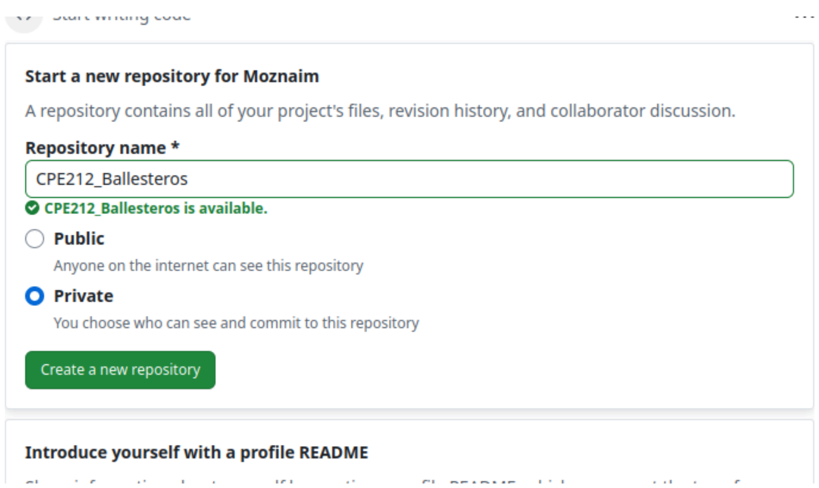


```
erwin@workstation:~$ which git
/usr/bin/git
erwin@workstation:~$
```



```
erwin@workstation:~$ git --version
git version 2.43.0
```

Figure 3 & 4: Checking the file location and version of git



Start a new repository for Moznaim

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

CPE212_Ballesteros

✓ CPE212_Ballesteros is available.

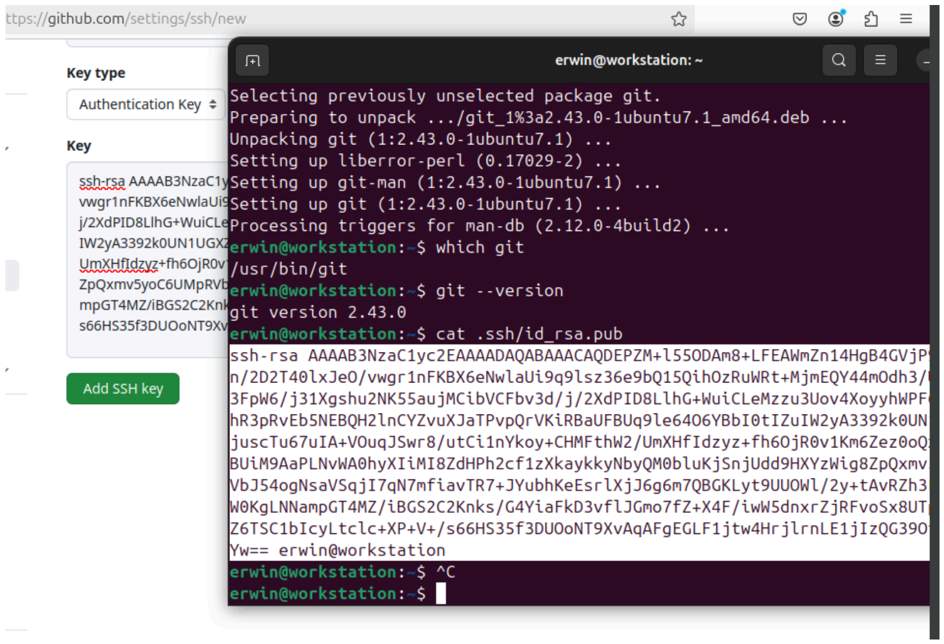
○ Public

Anyone on the internet can see this repository

● Private

You choose who can see and commit to this repository

Introduce yourself with a profile README



ttps://github.com/settings/ssh/new

Key type

Authentication Key

Key

```
ssh-rsa AAAAB3NzaC1y
vwgr1nFKBX6eNwlaUi9
j/2XdPID8LlhG+WuiCLe
IW2yA3392k0UN1UGXZ
UmXHfIdzyz+fh6OjR0v
ZpQxmv5yoC6UMpRVb
mpGT4MZ/iBGS2C2Knk
s66HS35f3DUOoNT9Xv
```

erwin@workstation: ~

```
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.43.0-1ubuntu7.1_amd64.deb ...
Unpacking git (1:2.43.0-1ubuntu7.1) ...
Setting up liberror-perl (0.17029-2) ...
Setting up git-man (1:2.43.0-1ubuntu7.1) ...
Setting up git (1:2.43.0-1ubuntu7.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
erwin@workstation:~$ which git
/usr/bin/git
erwin@workstation:~$ git --version
git version 2.43.0
erwin@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDEPZM+l55ODAm8+LFEAWmZn14HgB4GVjP
n/2D2T40lxJeO/vwgr1nFKBX6eNwlaUi9q9lsz36e9bQ15QihOzRuWRt+MjmEQY44mOdh3/
3FpW6/j31Xgshu2NK55aujMCibVCFbv3d/j/2XdPID8LlhG+WuiCLeMzzu3Uov4XoyyhWPF
hR3pRvEb5NEBQH2lnCYZvuXJaTPvpQrVKiRBaUFBUq9le64O6YBbI0tIZuIW2yA3392k0UN
juscTu67uIA+VOuqJSwr8/utCi1nYkoy+CHMFthW2/UmXHfIdzyz+fh6OjR0v1Km6Zez0oQ
BUiM9AaPLNvWA0hyXIiMI8ZdHPh2cf1zXkaykkyNbyQM0bluKjSnjUdd9HXYzWig8ZpQxmv
VbJ54ogNsaVSqjI7qN7mfiavTR7+JYubhKeEsrlXjJ6g6m7QBGKLyt9UUOWl/2y+tAvRZh3
W0KgLNNampGT4MZ/iBGS2C2Knks/G4YiaFkD3vflJGmo7fZ+X4F/iwW5dnxrZjRFvoSx8UT
Z6TSC1bIcyLtclc+XP+V+/s66HS35f3DUOoNT9XvAqAFgEGLF1jtw4HrjlrnLE1jIzQG39O
Yw== erwin@workstation
erwin@workstation:~$ ^C
erwin@workstation:~$
```

Figure 5 & 6: Creation of Repository and Adding SSH key

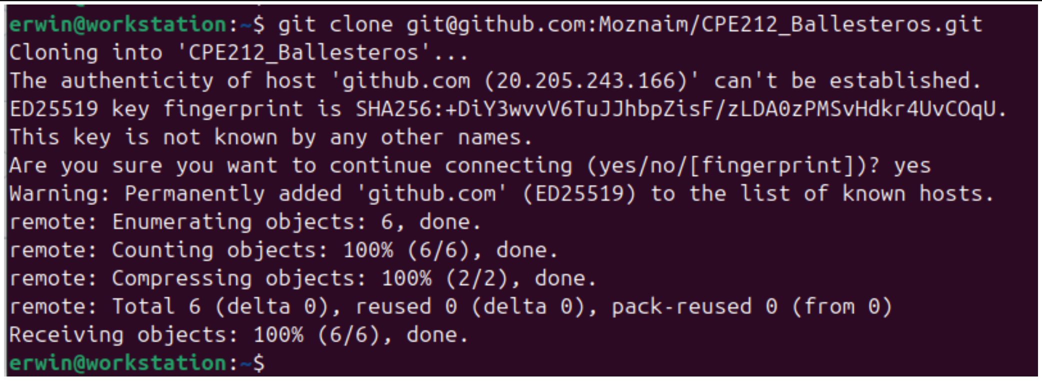
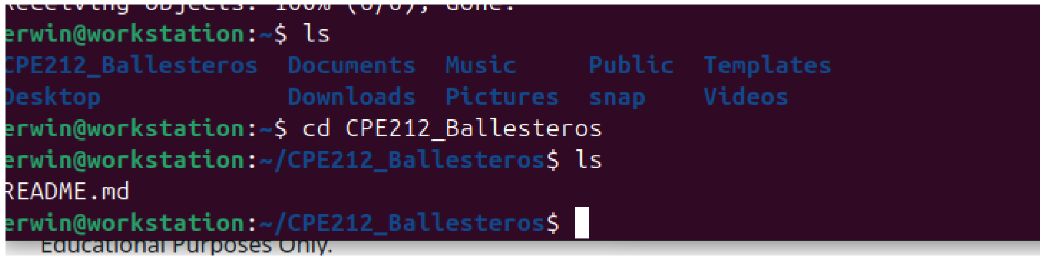
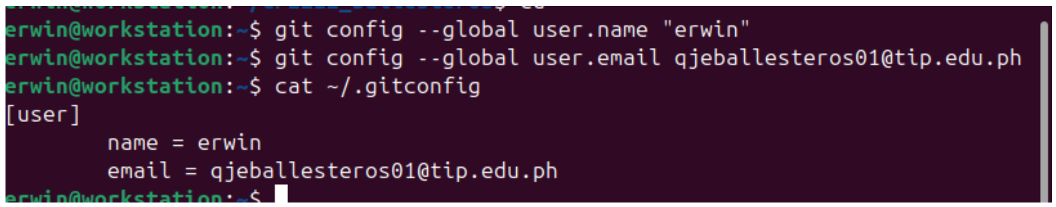
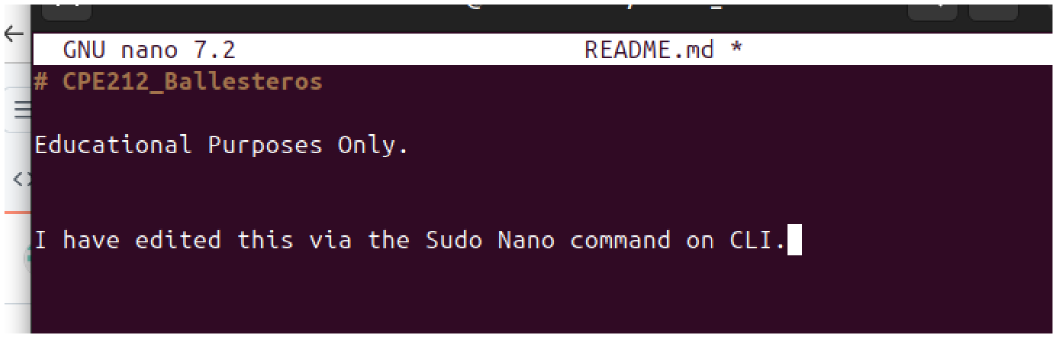
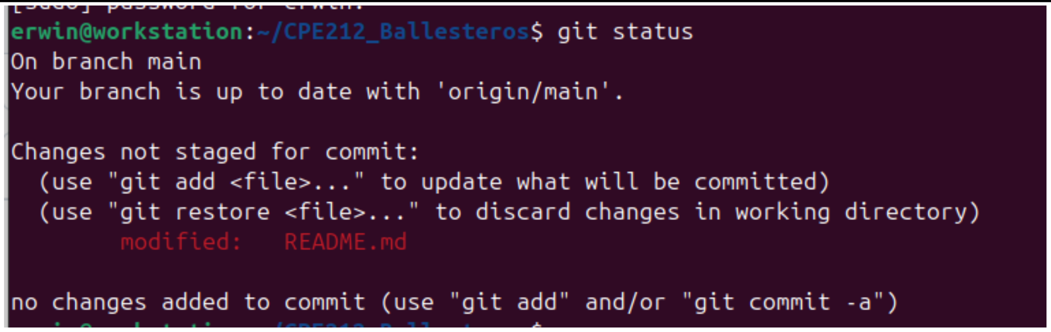

```
erwin@workstation:~$ git clone git@github.com:Moznaim/CPE212_Ballesteros.git
Cloning into 'CPE212_Ballesteros'...
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any 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: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
erwin@workstation:~$
```

```
erwin@workstation:~$ ls
CPE212_Ballesteros  Documents  Music     Public  Templates
Desktop             Downloads  Pictures  snap    Videos
erwin@workstation:~$ cd CPE212_Ballesteros
erwin@workstation:~/CPE212_Ballesteros$ ls
README.md
erwin@workstation:~/CPE212_Ballesteros$
```
Educational Purposes Only.

Figure 7 & 8: Creating and Checking for cloned git

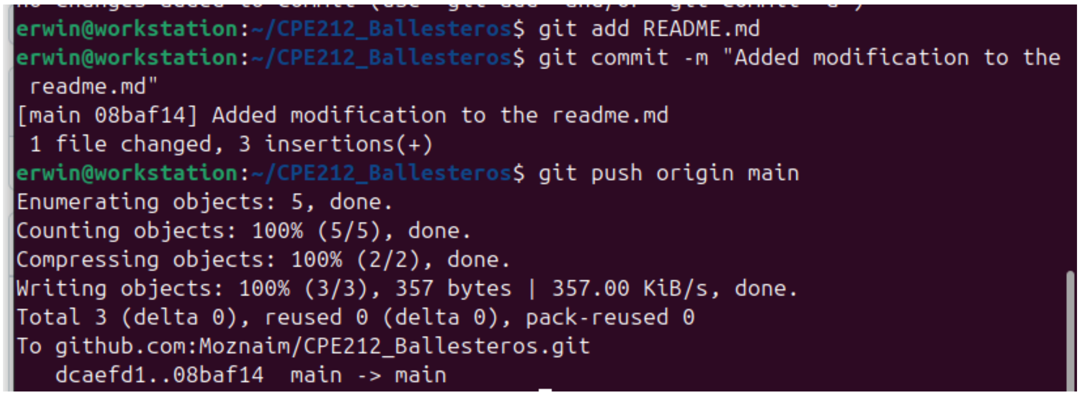

```
erwin@workstation:~$ git config --global user.name "erwin"
erwin@workstation:~$ git config --global user.email qjeballesteros01@tip.edu.ph
erwin@workstation:~$ cat ~/.gitconfig
[user]
        name = erwin
        email = qjeballesteros01@tip.edu.ph
erwin@workstation:~$
```
Figure 9: Configuring username and email


```
  GNU nano 7.2                        README.md *
# CPE212_Ballesteros

Educational Purposes Only.


I have edited this via the Sudo Nano command on CLI.
```
Figure 10: Editing README.md via sudo nano



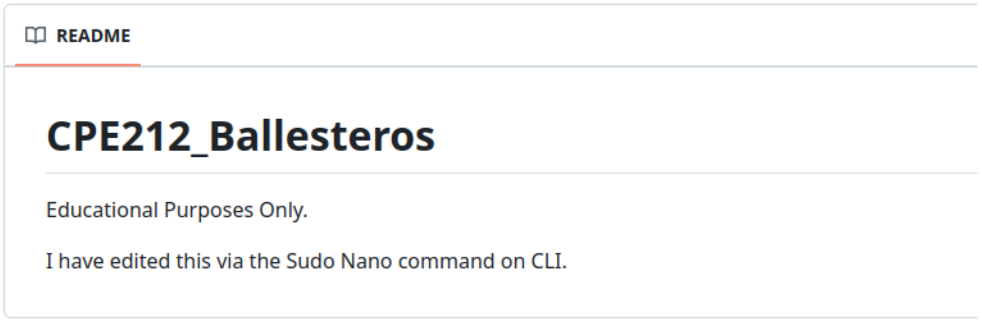
```
erwin@workstation:~/CPE212_Ballesteros$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

Figure 11: Checking status of the difference between the main



```
erwin@workstation:~/CPE212_Ballesteros$ git add README.md
erwin@workstation:~/CPE212_Ballesteros$ git commit -m "Added modification to the
 readme.md"
[main 08baf14] Added modification to the readme.md
 1 file changed, 3 insertions(+)
erwin@workstation:~/CPE212_Ballesteros$ 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), 357 bytes | 357.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Moznaim/CPE212_Ballesteros.git
   dcaefd1..08baf14  main -> main
```

Figure 12: Adding README.md to the final changes and committing it



README

CPE212_Ballesteros

Educational Purposes Only.

I have edited this via the Sudo Nano command on CLI.

Figure 13: Changes on the README.md

Reflections:

Answer the following:

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

   The different things we have utilized ansible are many but the most prominent of them all currently are updating and installing software packages. We have also utilized it in communicating to different remote server/s.

4. How important is the inventory file?

   The inventory file is very important as it is the one that is keeping track of the server/s that we have and how Ansible operates on it. This allows

ansible to correct machines for the different tasks they are given and can make configuration management easier for network administrators.

Conclusions/Learnings:

For this activity I have learned that when the proper connection are setup between the host and the different server, connecting them via the SSH key can be relatively easy as you only need to copy it. By performing the activity, I also discovered that setting up git and connecting it to Github by using SSH is also easy and not really intimidating. To conclude, this activity teaches us the use of SSH key and how we can utilize it by using github and streamlining the workflow of our codes.