Name: Karlo Santos	**Date Performed:08/27/23**
Course/Section: CPE31S5	**Date Submitted:08/28/23**
Instructor: Engr. Roman Richard	**Semester and SY: 1st sem, 2023-2024**

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.

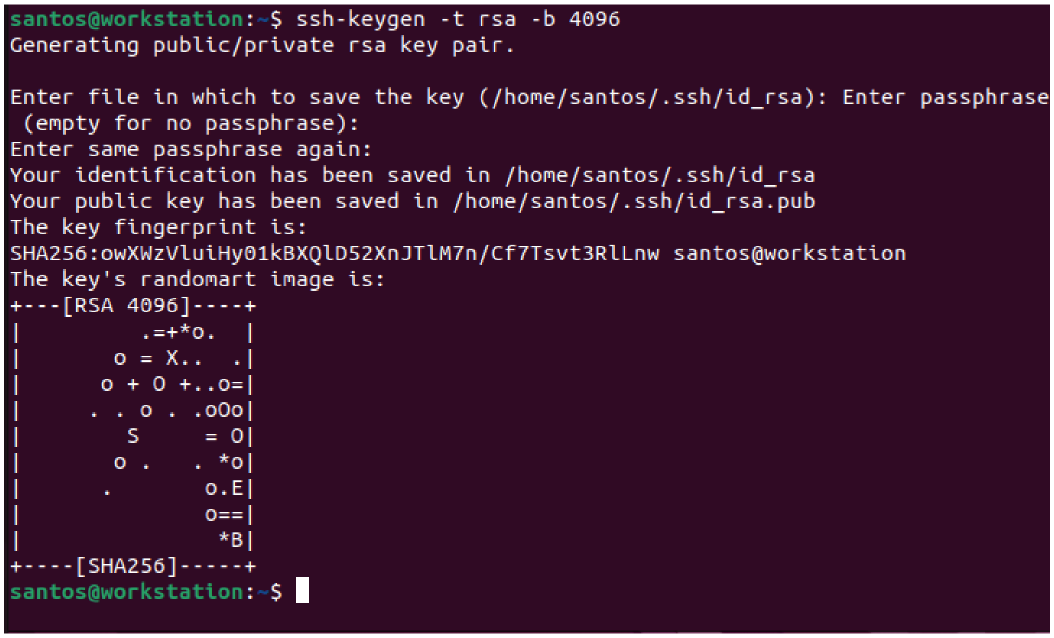
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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,

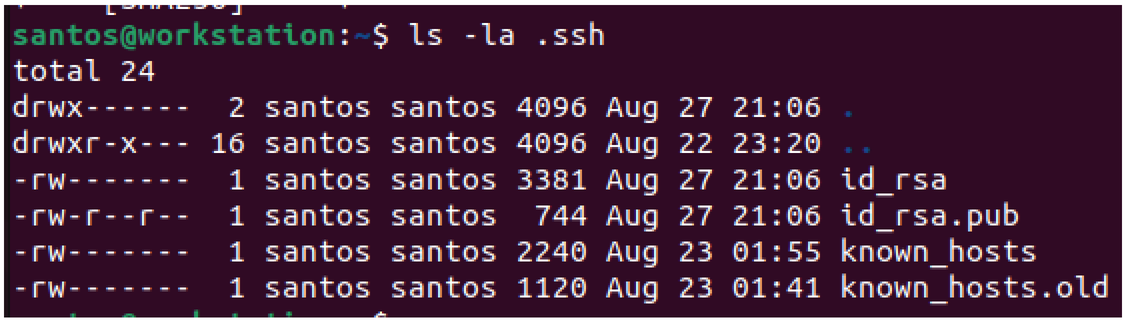
the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. However, in enterprise environments, the location is often different. The default key file name depends on the algorithm, in this case *id_rsa* when using the 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.



```
santos@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.

Enter file in which to save the key (/home/santos/.ssh/id_rsa): Enter passphrase
 (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/santos/.ssh/id_rsa
Your public key has been saved in /home/santos/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:owXWzVluiHy01kBXQlD52XnJTlM7n/Cf7Tsvt3RlLnw santos@workstation
The key's randomart image is:
+---[RSA 4096]----+
|         .=+*o.  |
|       o = X..   .|
|      o + O +..o=|
|     . . o . .oOo|
|        S     = O|
|       o .   . *o|
|      .       o.E|
|              o==|
|               *B|
+----[SHA256]-----+
santos@workstation:~$
```

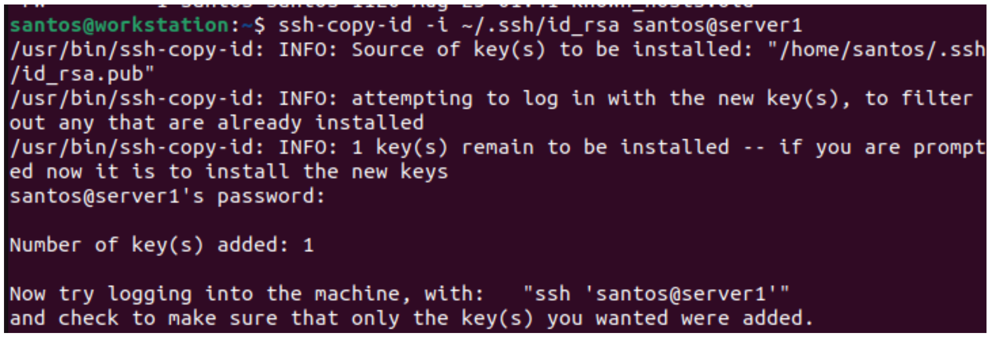
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



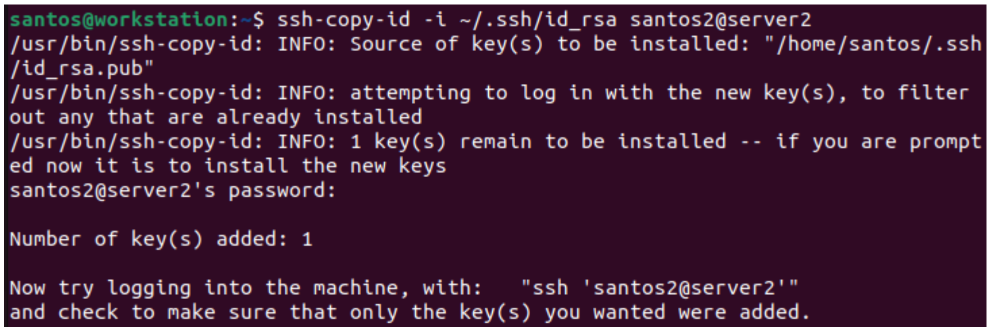
```
santos@workstation:~$ ls -la .ssh
total 24
drwx------  2 santos santos 4096 Aug 27 21:06 .
drwxr-x--- 16 santos santos 4096 Aug 22 23:20 ..
-rw-------  1 santos santos 3381 Aug 27 21:06 id_rsa
-rw-r--r--  1 santos santos  744 Aug 27 21:06 id_rsa.pub
-rw-------  1 santos santos 2240 Aug 23 01:55 known_hosts
-rw-------  1 santos santos 1120 Aug 23 01:41 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.

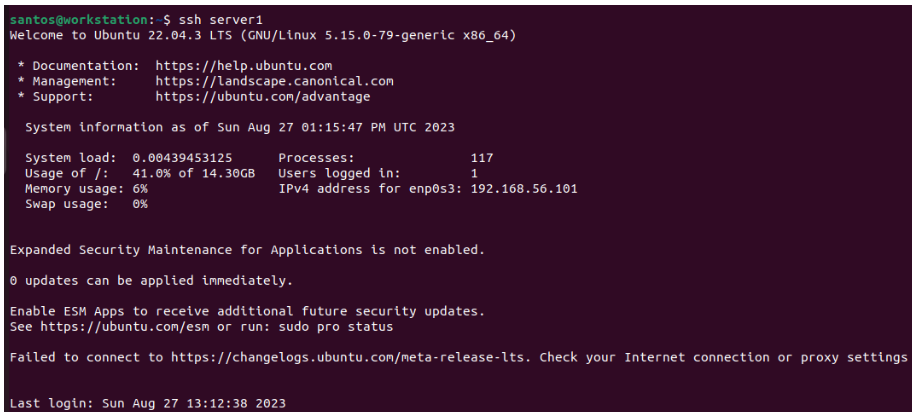


```
santos@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa santos@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/santos/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: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

Number of key(s) added: 1

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

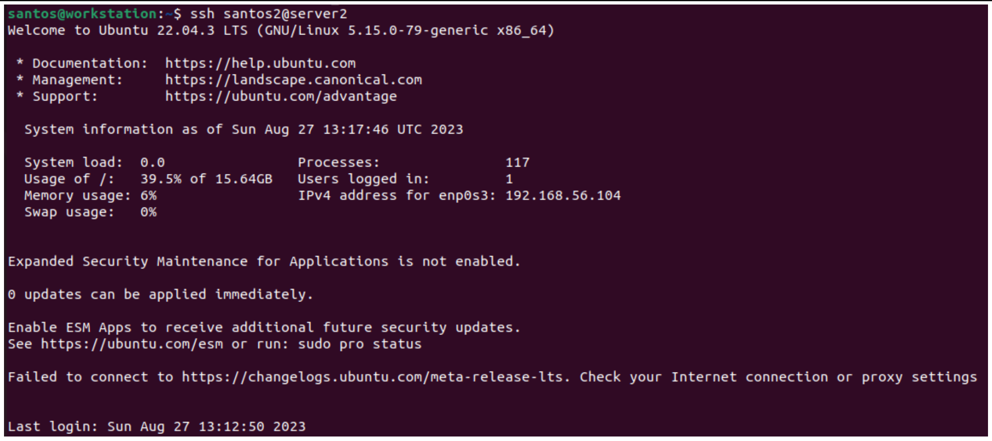


```
santos@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa santos2@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/santos/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
santos2@server2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'santos2@server2'"
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?



```
santos@workstation:~$ ssh server1
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 Sun Aug 27 01:15:47 PM UTC 2023

  System load:  0.00439453125      Processes:               117
  Usage of /:   41.0% of 14.30GB   Users logged in:         1
  Memory usage: 6%                 IPv4 address for enp0s3: 192.168.56.101
  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: Sun Aug 27 13:12:38 2023
```



```
santos@workstation:~$ ssh santos2@server2
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 Sun Aug 27 13:17:46 UTC 2023

  System load:  0.0               Processes:               117
  Usage of /:   39.5% of 15.64GB   Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 192.168.56.104
  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: Sun Aug 27 13:12:50 2023
```

After I verify the SSH access to server 1 and server 2 using the local machine. It shows that it didn't ask for a password to authenticate the cp+onnections. With the help of the SSH key, it will serve as a password to the servers. This will results for password to be not required while connection to the other servers using SSH access.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   - **SSH or Secure Shell is a network protocol that helps the user and also the system administrators to create secured connections in connecting to computers in not secured network. Secure shell can also be used in making secured tunnels that can be used for other protocols. The SSH applies encryption that will help to make sure that other people cannot see the information between the connection of two devices. This protocol does various things to ensure security, like key-based authentication and strong password authentication.**
2. How do you know that you already installed the public key to the remote servers?
   - **Connecting to the remote server via SSH is a way to know if you already installed the public key. Since, if you are able to connect and have access without a password that means the public key is already installed. This happened because the server already accepts the unique key that will help to access it without a password. This is a secure way to access and also a way to know if you already installed a public key in a remote server.**

Part 2: Discussion

Provide screenshots for each task.

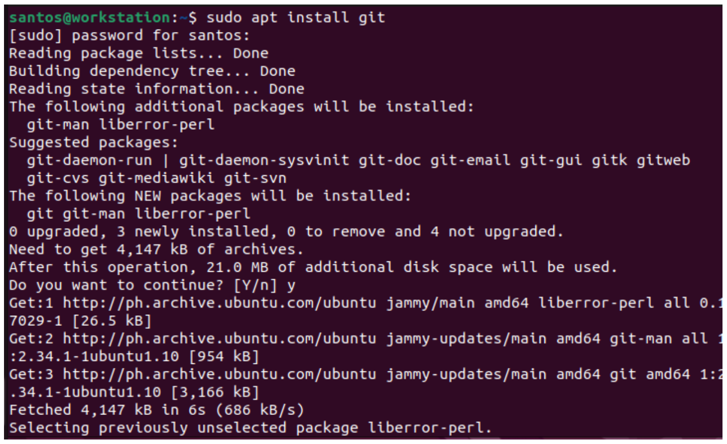
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

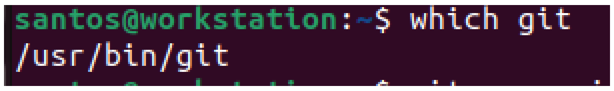
Task 3: Set up the Git Repository

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*



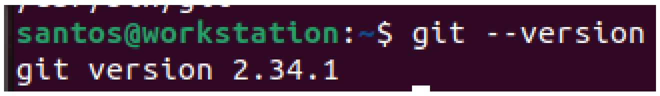
```
santos@workstation:~$ sudo apt install git
[sudo] password for santos:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit 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 4 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.1
7029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2
.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 6s (686 kB/s)
Selecting previously unselected package liberror-perl.
```

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



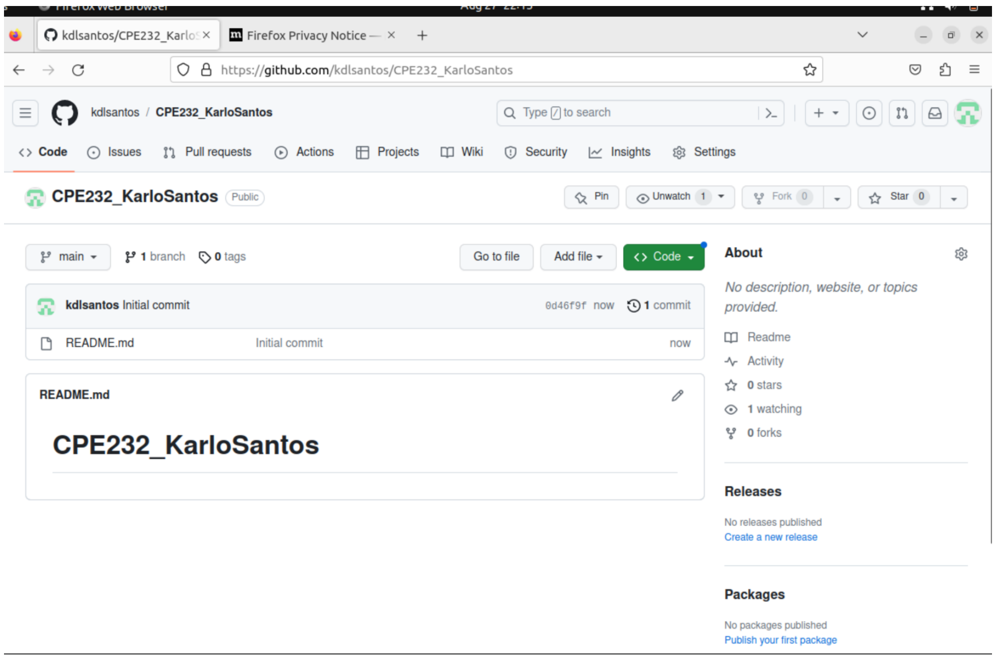
```
santos@workstation:~$ which git
/usr/bin/git
```

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

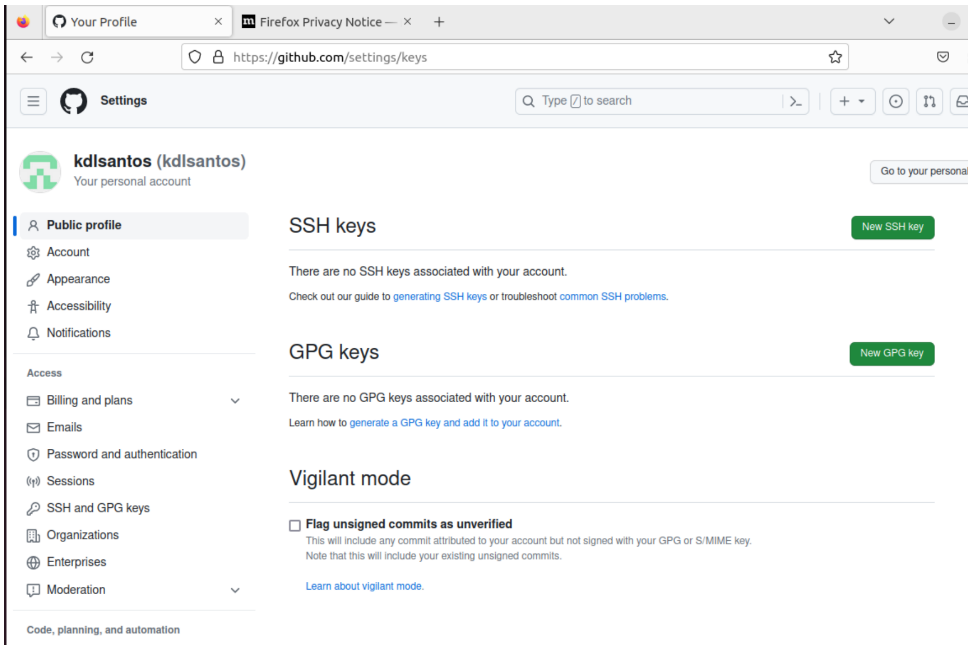


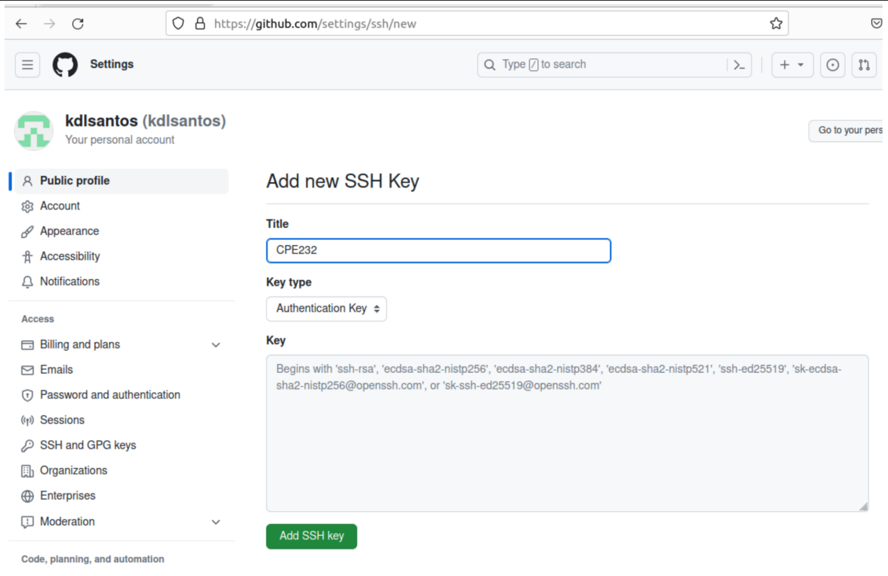
```
santos@workstation:~$ git --version
git version 2.34.1
```

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.


Your Profile
Firefox Privacy Notice
https://github.com/settings/keys
Settings
kdlsantos (kdlsantos)
Your personal account
Go to your personal
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
SSH keys
New SSH key
There are no SSH keys associated with your account.
Check out our guide to generating SSH keys or troubleshoot common SSH problems.
GPG keys
New GPG key
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.


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

kdlsantos (kdlsantos)
Your personal account

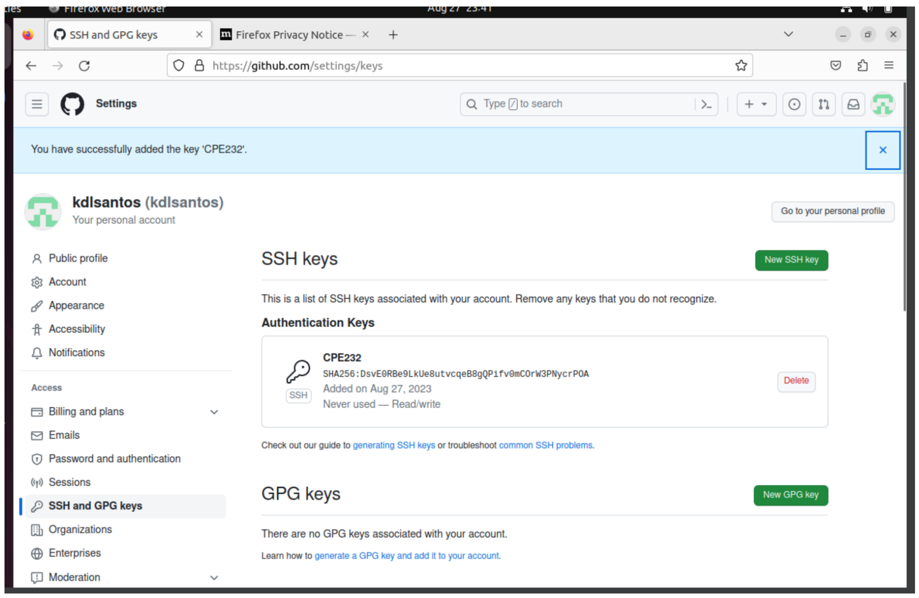
Add new SSH Key

Title
CPE232

Key type
Authentication Key

Key
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

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.


https://github.com/settings/keys

You have successfully added the key 'CPE232'.

kdlsantos (kdlsantos)
Your personal account

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:DsvE0RBe9LkUe8utvcqeB8gQPifv0mCOrW3PNycrPOA
Added on Aug 27, 2023
Never used — Read/write

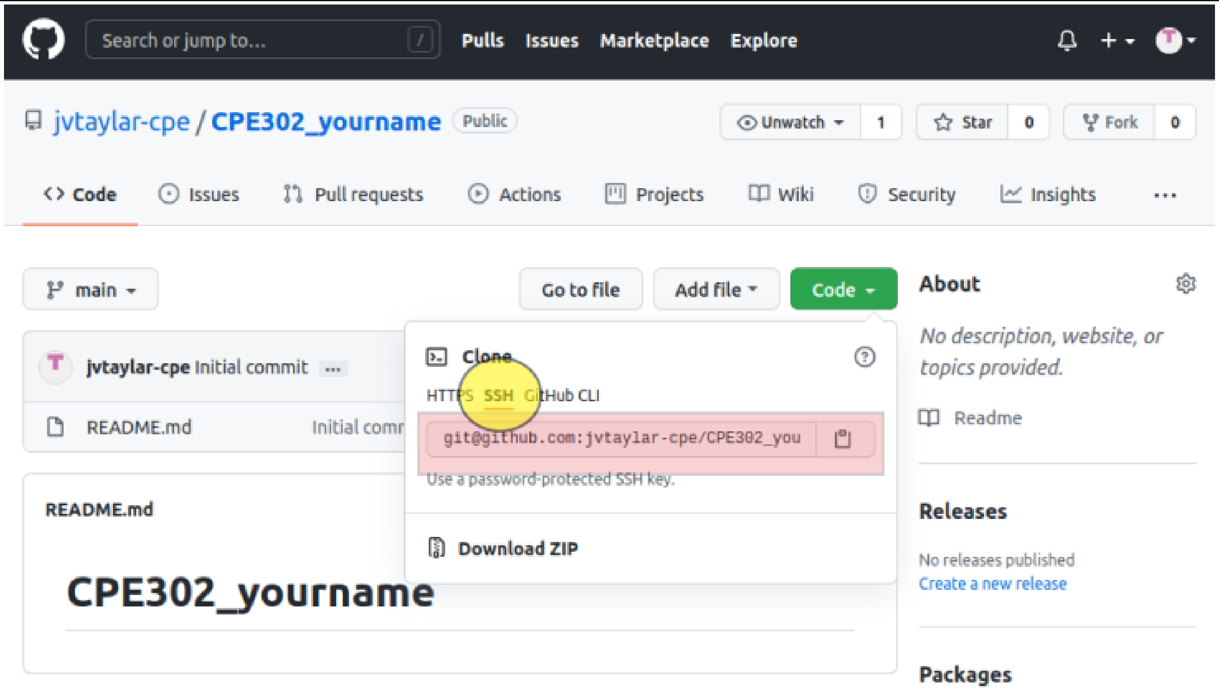
Check out our guide to generating SSH keys or troubleshoot common SSH problems.

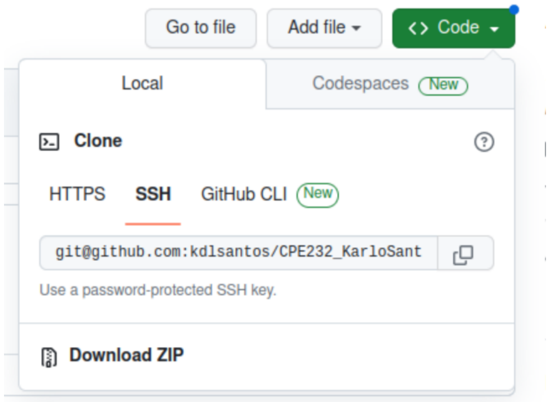
GPG keys

There are no GPG keys associated with your account.

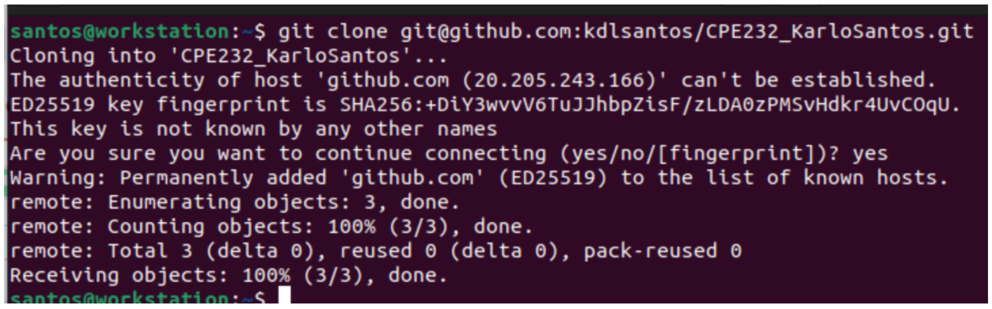
Learn how to generate a GPG key and add it to your account.

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.



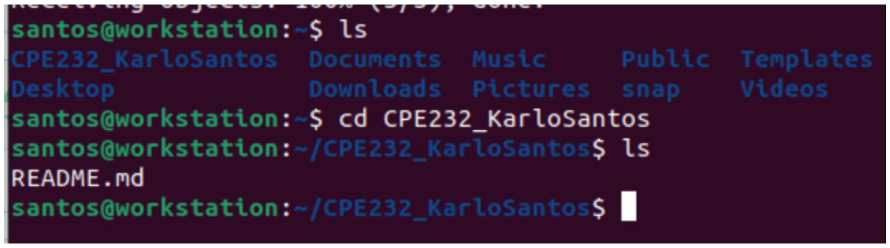


e. Issue the command git clone followed by the copied link. For example, *git clone git@github.com:jvtaylar-cpe/CPE232_yourname.git*. When prompted to continue connecting, type yes and press enter.



```
santos@workstation:~$ git clone git@github.com:kdlsantos/CPE232_KarloSantos.git
Cloning into 'CPE232_KarloSantos'...
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.
santos@workstation:~$
```

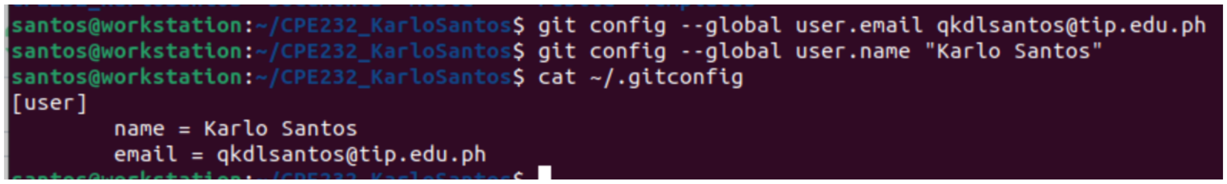
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
santos@workstation:~$ ls
CPE232_KarloSantos  Documents  Music     Public  Templates
Desktop             Downloads  Pictures  snap    Videos
santos@workstation:~$ cd CPE232_KarloSantos
santos@workstation:~/CPE232_KarloSantos$ ls
README.md
santos@workstation:~/CPE232_KarloSantos$
```

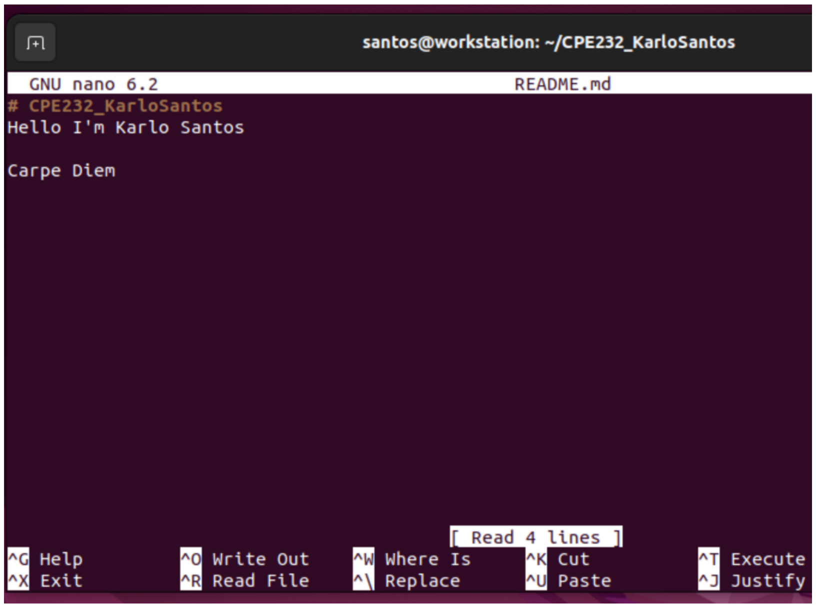
g. Use the following commands to personalize your git.

- *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*



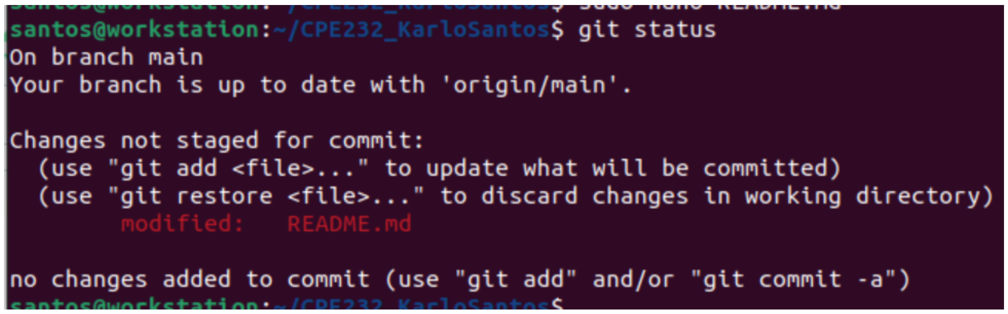
```
santos@workstation:~/CPE232_KarloSantos$ git config --global user.email qkdlsantos@tip.edu.ph
santos@workstation:~/CPE232_KarloSantos$ git config --global user.name "Karlo Santos"
santos@workstation:~/CPE232_KarloSantos$ cat ~/.gitconfig
[user]
        name = Karlo Santos
        email = qkdlsantos@tip.edu.ph
```

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.



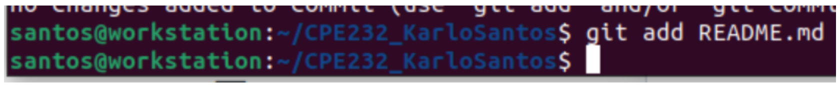
```
santos@workstation: ~/CPE232_KarloSantos
 GNU nano 6.2                          README.md
# CPE232_KarloSantos
Hello I'm Karlo Santos

Carpe Diem

                                  [ Read 4 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

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?



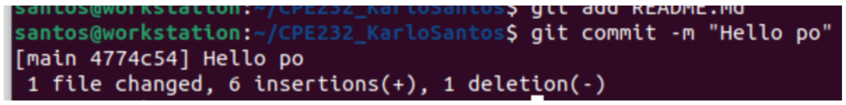
```
santos@workstation:~/CPE232_KarloSantos$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

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



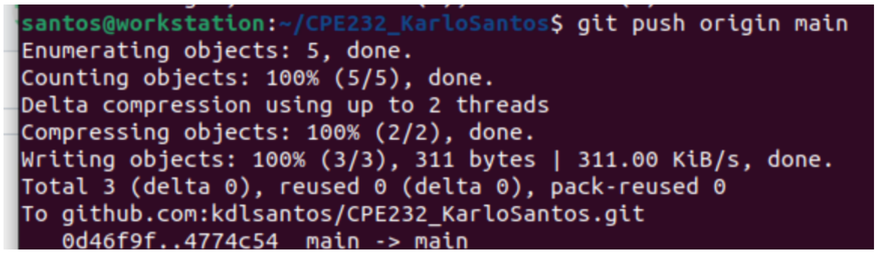
```
santos@workstation:~/CPE232_KarloSantos$ git add README.md
santos@workstation:~/CPE232_KarloSantos$
```

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.



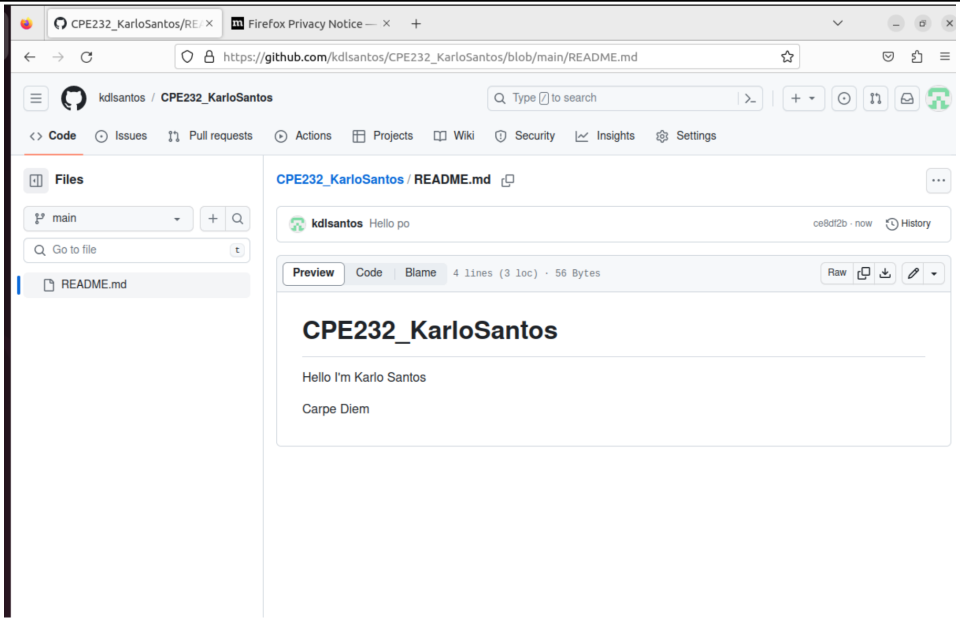
```
santos@workstation:~/CPE232_KarloSantos$ git commit -m "Hello po"
[main 4774c54] Hello po
 1 file changed, 6 insertions(+), 1 deletion(-)
```

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.*



```
santos@workstation:~/CPE232_KarloSantos$ 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), 311 bytes | 311.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kdlsantos/CPE232_KarloSantos.git
   0d46f9f..4774c54  main -> main
```

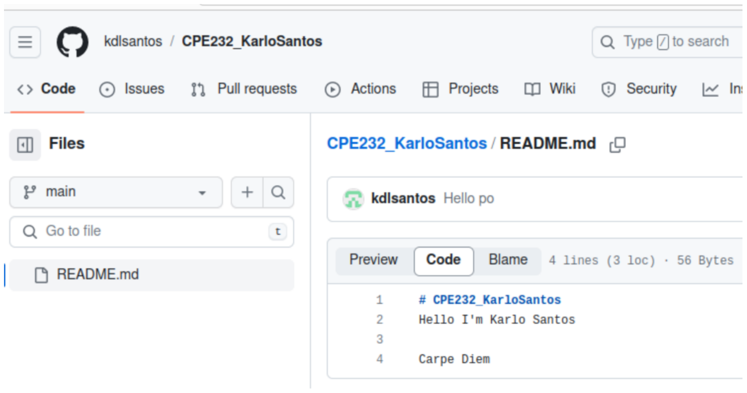
m. On the GitHub repository, verify that the changes have been made to README.md by refreshing the page. Describe the README.md file. You can notice the how long was the last commit. It should be some minutes ago and the message you typed on the git commit command should be there. Also, the README.md file should have been edited according to the text you wrote.



CPE232_KarloSantos

Hello I'm Karlo Santos

Carpe Diem



```
# CPE232_KarloSantos
Hello I'm Karlo Santos

Carpe Diem
```

Reflections:

Answer the following:

3. What sort of things have we so far done to the remote servers using ansible commands?
   - **We are able to do various things in managing remote servers. One of these is being able to connect to the github repository using the terminal in Ubuntu. Another one is being able to access the remote servers by using SSH key, this will result in not needing a password every time you need to access one, it also adds additional security by using a SSH key.**

4. How important is the inventory file?
   - **The inventory file is an important part in using Ansible and also for remote servers. Since, it will help and tell the Ansible different information like the location of the servers, what the purpose of it and other information of it that includes IP address. In short, the inventory file**

plays an important role for using Ansible commands, and being able to do tasks more efficiently and be organized in managing different servers.

Conclusions/Learnings:

In this laboratory activity I am able to learn and explore different things. First, I am able to use ssh in configuring remote and connecting to the local machine. I use a key as an alternative instead of using a password. Another one is I am able to use and explore the git website, wherein I am able to create and familiarize with the git repository. Also, in doing this successfully run some commands that help to verify the connectivity. The creation of the public and private key is also one thing that I learned and understand. Lasly, I learned how to configure and run the hoc command into the server from the local machine. Its good that I am able to gain knowledge and enhance my skill with the help of this activity. I'm looking forward in using this in the future path as a computer engineering student.