Name: Bernice M. Peña	**Date Performed:** 08/24/2023
Course/Section: Managing Enterprise Severs/CPE31S5	**Date Submitted:** 08/26/2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st semester, SY 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.

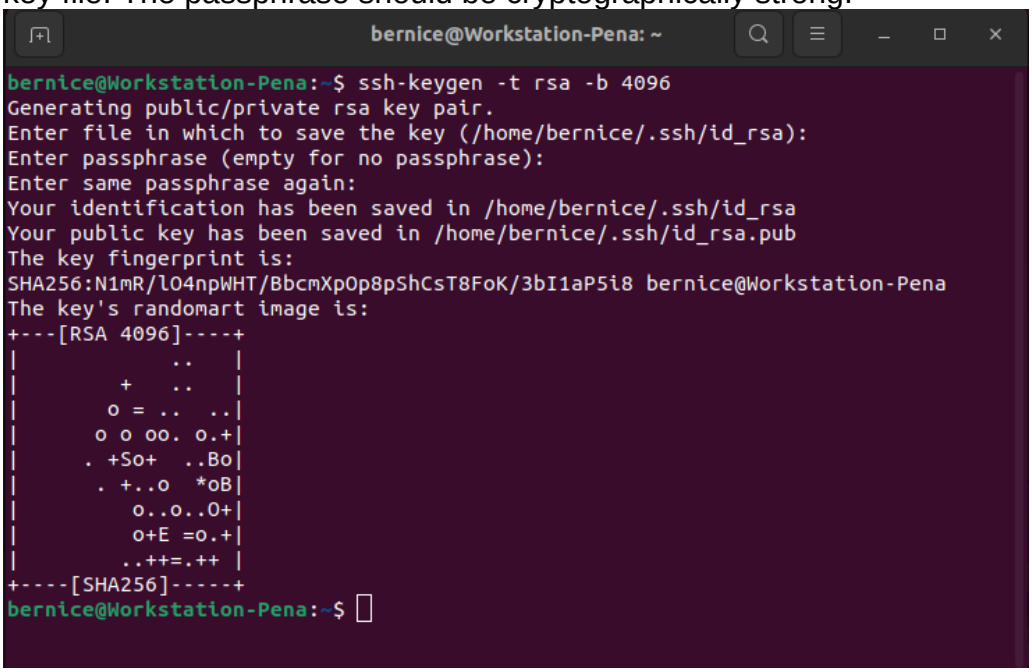
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

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*.

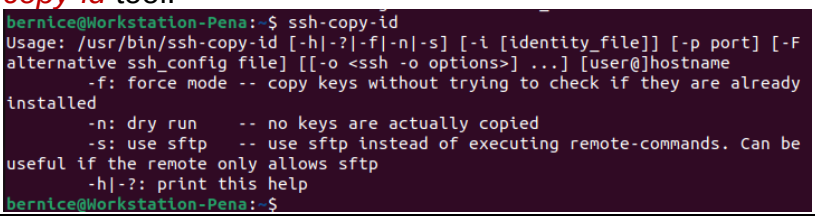
2. Issue the command *ssh-keygen -t rsa -b 4096.* The algorithm is selected using the -t option and key size using the -b option.
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.



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.

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.



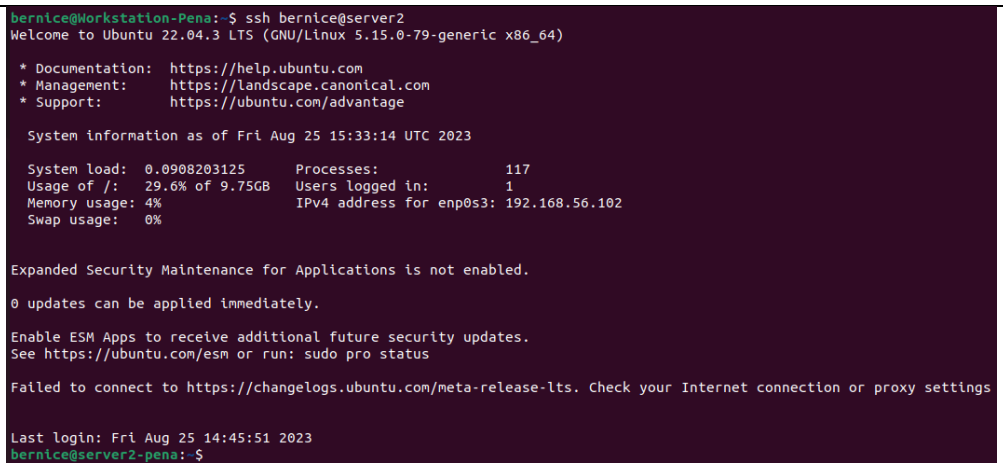
2. Issue the command similar to this: *ssh-copy-id -i ~/.ssh/id_rsa user@host*
3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.

```
bernice@Workstation-Pena:~$ ssh-copy-id -i ~/.ssh/id_rsa bernice@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/bernice/.ss
h/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
bernice@192.168.56.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'bernice@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.

bernice@Workstation-Pena:~$
```

```
bernice@Workstation-Pena:~$ ssh-copy-id -i ~/.ssh/id_rsa bernice@192.168.56.102
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/bernice/.ss
h/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
bernice@192.168.56.102's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'bernice@192.168.56.102'"
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?

```
bernice@Workstation-Pena:~$ ssh bernice@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 Fri Aug 25 03:31:16 PM UTC 2023

  System load:  0.2275390625      Processes:               114
  Usage of /:   29.7% of 9.75GB   Users logged in:         1
  Memory usage: 3%                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: Fri Aug 25 15:30:15 2023 from 192.168.56.103
bernice@server1-pena:~$
```



When I tested SSH access to Server 1 and Server 2 on my local machine, I noticed that the connections did not ask for a password. This is because I had set up SSH key-based authentication correctly, with SSH keys, my private key is used to prove my identity to the servers, so there's no need for password entry during the connection process.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   SSH is powerful used to create secure connections between your computer and remote servers or devices. It achieves this security by encoding the information that's sent back and forth, making sure it can't be intercepted and read by unauthorized parties. SSH also uses strong methods to confirm your identity when connecting to another system, adding an extra layer of safety. It's incredibly useful for tasks like sending files securely, managing distant computers, and keeping your online interactions private. In short, SSH is a crucial tool that enhances the security of your remote connections and is used for a wide range of purposes, from managing systems to exchanging sensitive data.

2. How do you know that you already installed the public key to the remote servers?
   It can be determined whether your public key is installed on remote servers by attempting to connect via SSH without a password. If the connection is successful and you gain access without being asked for a password, it's a clear sign that the public key is properly installed on the remote server, this means the server recognizes the unique key and grants I secure access, which is much safer and more convenient than traditional password-based authentication.

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*

```
bernice@Workstation-Pena:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man
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
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 4,120 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Ign:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Fetched 4,120 kB in 36s (115 kB/s)
Selecting previously unselected package git-man.
(Reading database ... 164972 files and directories currently installed.)
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.10_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.10) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.10_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.10) ...
Setting up git-man (1:2.34.1-1ubuntu1.10) ...
Setting up git (1:2.34.1-1ubuntu1.10) ...
Processing triggers for man-db (2.10.2-1) ...
bernice@Workstation-Pena:~$
```

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

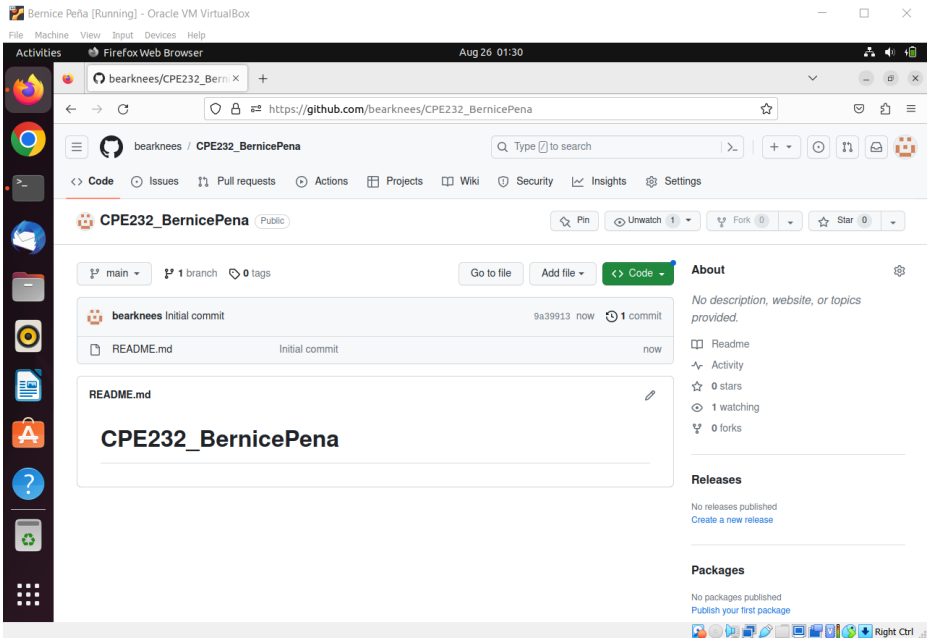
```
bernice@Workstation-Pena:~$ which git
/usr/bin/git
bernice@Workstation-Pena:~$
```

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

```
bernice@Workstation-Pena:~$ git --version
git version 2.34.1
bernice@Workstation-Pena:~$
```

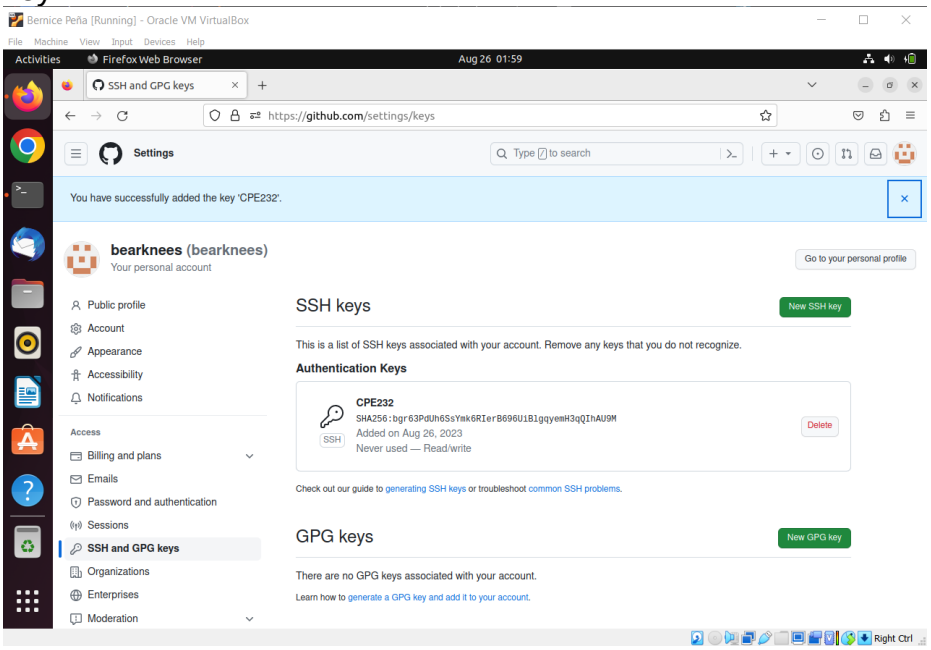
4. Using the browser in the local machine, go to www.github.com.
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.

a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.

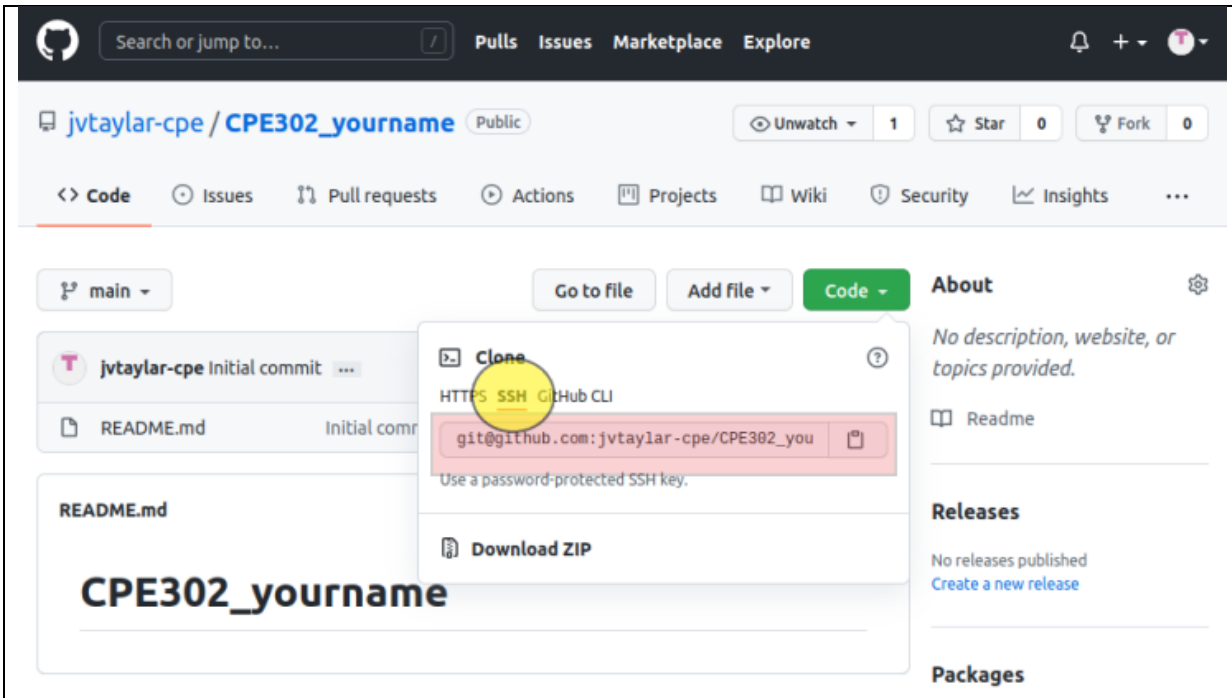


b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

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.



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.

```
bernice@Workstation-Pena:~/.ssh$ git clone git@github.com:bearknees/CPE232_Berni
cePena.git
Cloning into 'CPE232_BernicePena'...
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.
```

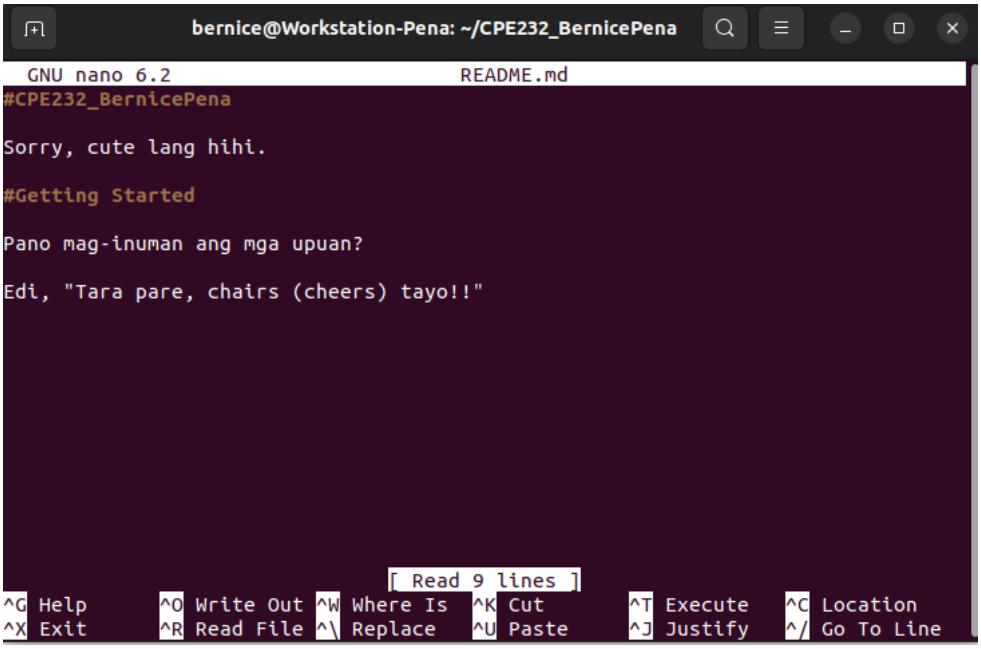
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

```
bernice@Workstation-Pena:~$ ls
CPE232_BernicePena  google-chrome-stable_current_amd64.deb  Pictures   Videos
Desktop             home                                    Public
Documents           home.pub                                snap
Downloads           Music                                   Templates
bernice@Workstation-Pena:~$ cd CPE232_BernicePena
bernice@Workstation-Pena:~/CPE232_BernicePena$ ls
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*

```
bernice@Workstation-Pena:~$ git config --global user.name "Bernice Pena"
bernice@Workstation-Pena:~$ git config --global user.email bernquinn2001@gmail.c
om
bernice@Workstation-Pena:~$ cat ~/.gitconfig
[user]
        name = Bernice Pena
        email = bernquinn2001@gmail.com
bernice@Workstation-Pena:~$
```

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?

```
bernice@Workstation-Pena:~/CPE232_BernicePena$ 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")
bernice@Workstation-Pena:~/CPE232_BernicePena$
```

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

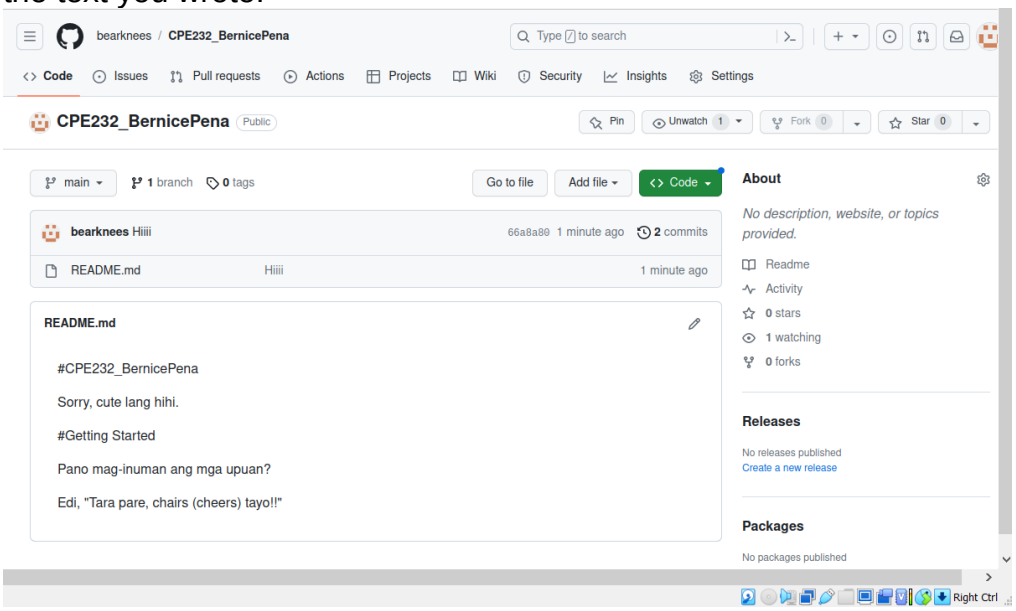
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
bernice@Workstation-Pena:~/CPE232_BernicePena$ git add README.md
bernice@Workstation-Pena:~/CPE232_BernicePena$ git commit -m "Hiiii"
[main 66a8a80] Hiiii
 1 file changed, 9 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.*

```
bernice@Workstation-Pena:~/CPE232_BernicePena$ 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), 364 bytes | 182.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:bearknees/CPE232_BernicePena.git
   9a39913..66a8a80  main -> main
bernice@Workstation-Pena:~/CPE232_BernicePena$
```

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.



Reflections:

Answer the following:

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

   With Ansible, we've accomplished several key tasks for managing remote servers effectively. First, we set up secure connections to these servers allowing me to work with GitHub repositories directly from the Ubuntu terminal. This means that we can easily retrieve, modify, and organize files in those repositories. To enhance both security and usability, we implemented

SSH key pairs. These keys act as digital authentication tokens, eliminating the need to enter passwords each time we access a remote server, this not only boosts security but also streamlines the remote server access process especially in scenarios involving multiple servers.

4. How important is the inventory file?
   The inventory file in Ansible is like a phonebook for all the servers you want to manage. It's really important because it helps Ansible know where these servers are and what they do. You can group servers together in this file, like putting all the web servers in one group and the database servers in another, plus, you can tell Ansible special things about each server, like how to connect to it securely. This file also works with ever-changing environments which is very handy. So, in a nutshell, the inventory file is like a control center that helps Ansible keep everything organized and running smoothly when managing lots of servers.

Conclusions/Learnings:

In this activity, I successfully accomplished a series of objectives related to secure remote server management and efficient code collaboration. I configured SSH key-based authentication, security and simplifying remote connections by eliminating the need for passwords. The creation of public and private key pairs further enhanced access control. I verified SSH connectivity to ensure reliable server access. I set up Git repositories both locally and remotely, gaining the ability to manage version control effectively and collaborate on coding projects in an organized manner. Aside from these, I learned to configure and execute ad hoc commands on remote servers, streamlining server maintenance. These achievements collectively equip me with essential skills for secure and efficient remote server management and server administration. This knowledge is highly valuable and applicable across various fields.