Name: Ejercito, Marlon Jason A.	**Date Performed: 08/22/2023**
Course/Section: CPE31S4	**Date Submitted: 08/23/2023**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st Semester, 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

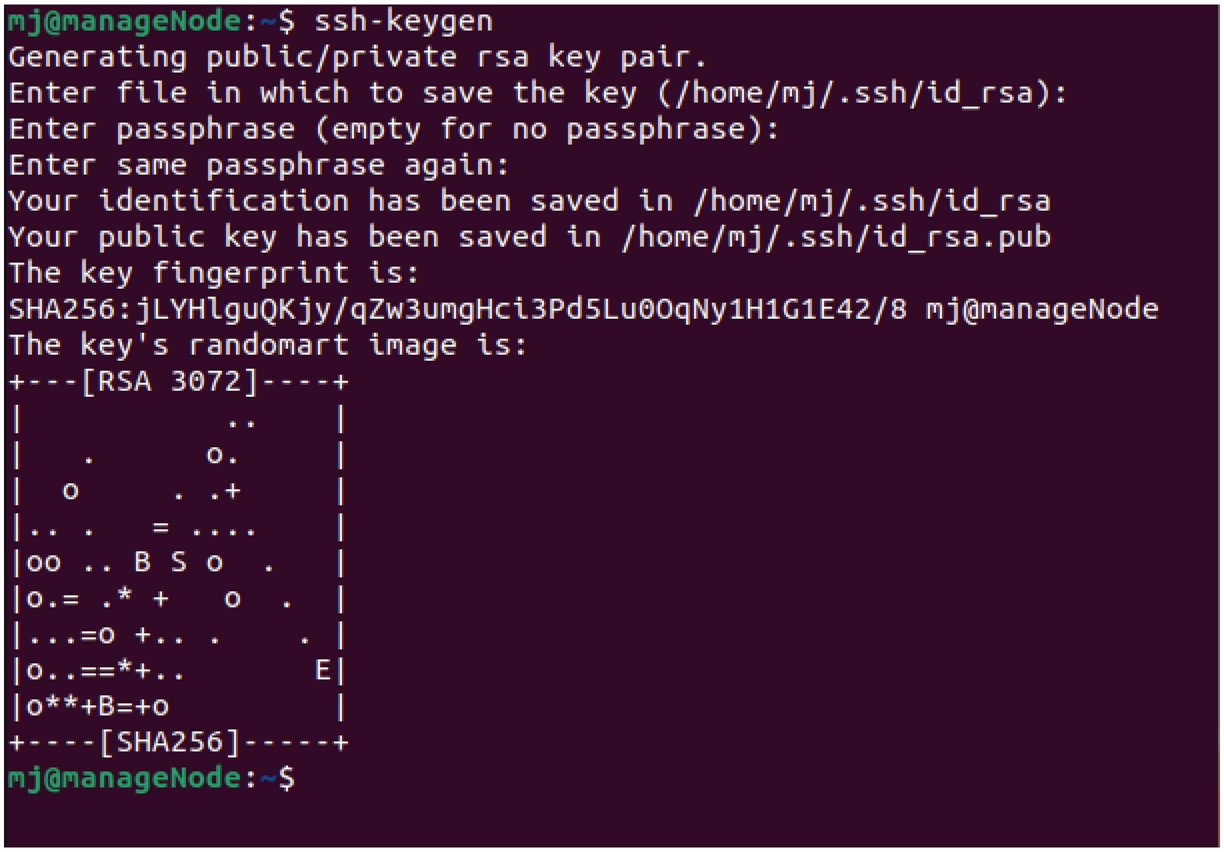
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

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.

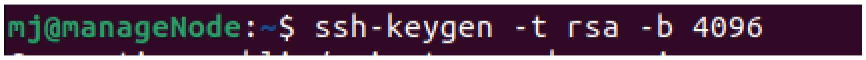
Task 1: Create an SSH Key Pair for User Authentication

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*.



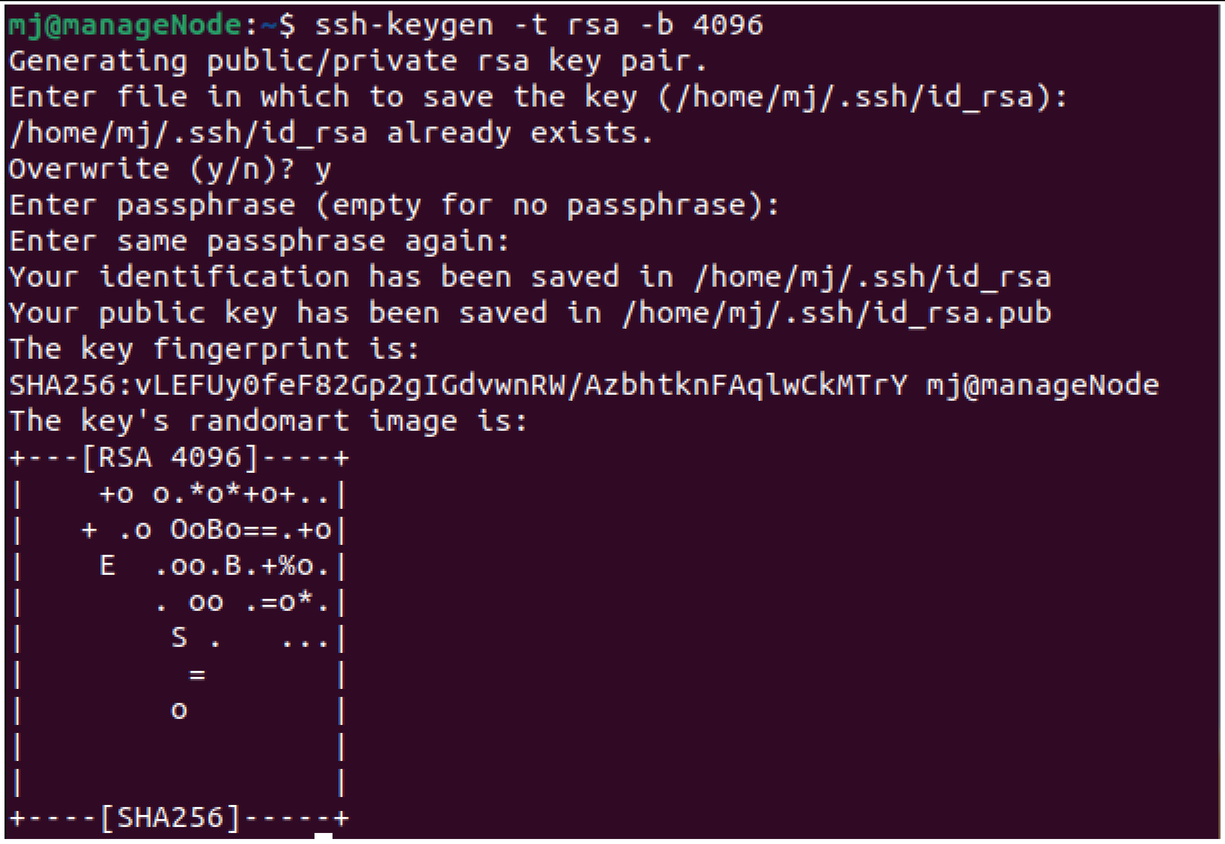
```
mj@manageNode:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mj/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mj/.ssh/id_rsa
Your public key has been saved in /home/mj/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jLYHlguQKjy/qZw3umgHci3Pd5Lu0OqNy1H1G1E42/8 mj@manageNode
The key's randomart image is:
+---[RSA 3072]----+
|           ..    |
|   .      o.     |
|  o     . .+     |
|.. .   = ....    |
|oo .. B S o  .   |
|o.= .* +   o  .  |
|...=o +.. .    . |
|o..==*+..       E|
|o**+B=+o         |
+----[SHA256]-----+
mj@manageNode:~$
```

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.



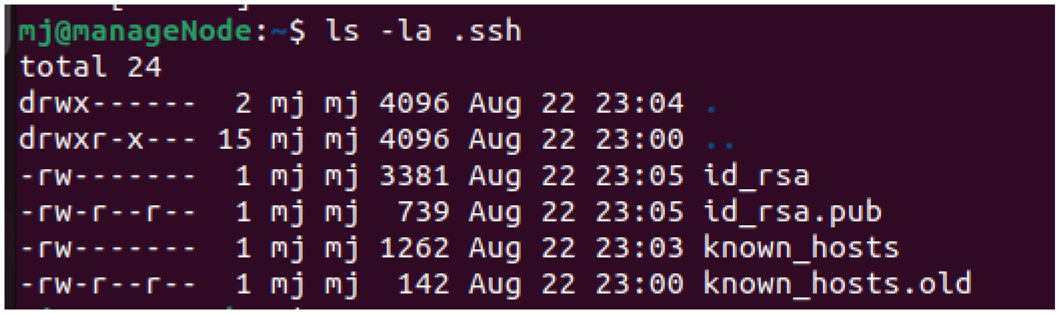
```
mj@manageNode:~$ ssh-keygen -t rsa -b 4096
```

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.



```
mj@manageNode:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mj/.ssh/id_rsa):
/home/mj/.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/mj/.ssh/id_rsa
Your public key has been saved in /home/mj/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:vLEFUy0feF82Gp2gIGdvwnRW/AzbhtknFAqlwCkMTrY mj@manageNode
The key's randomart image is:
+---[RSA 4096]----+
|    +o o.*o*+o+..|
|   + .o OoBo==.+o|
|    E  .oo.B.+%o.|
|       . oo .=o*.|
|        S .   ...|
|         =       |
|        o        |
|                 |
|                 |
+----[SHA256]-----+
```

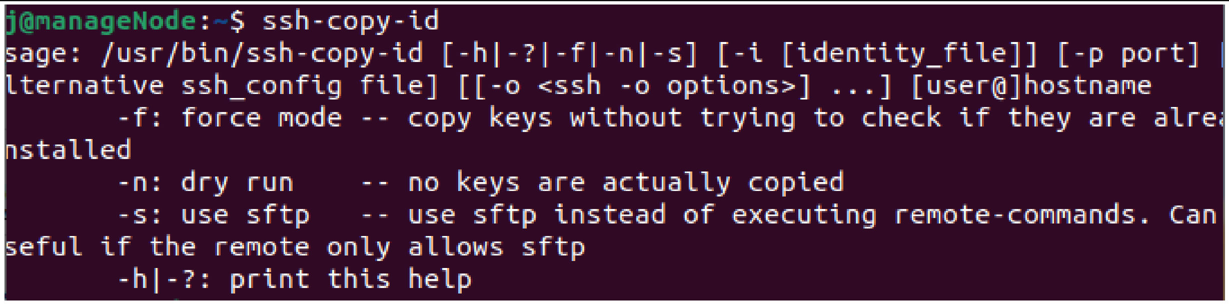
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



```
mj@manageNode:~$ ls -la .ssh
total 24
drwx------  2 mj mj 4096 Aug 22 23:04 .
drwxr-x--- 15 mj mj 4096 Aug 22 23:00 ..
-rw-------  1 mj mj 3381 Aug 22 23:05 id_rsa
-rw-r--r--  1 mj mj  739 Aug 22 23:05 id_rsa.pub
-rw-------  1 mj mj 1262 Aug 22 23:03 known_hosts
-rw-r--r--  1 mj mj  142 Aug 22 23:00 known_hosts.old
```

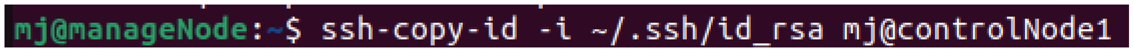
Task 2: Copying the Public Key to the remote servers

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.



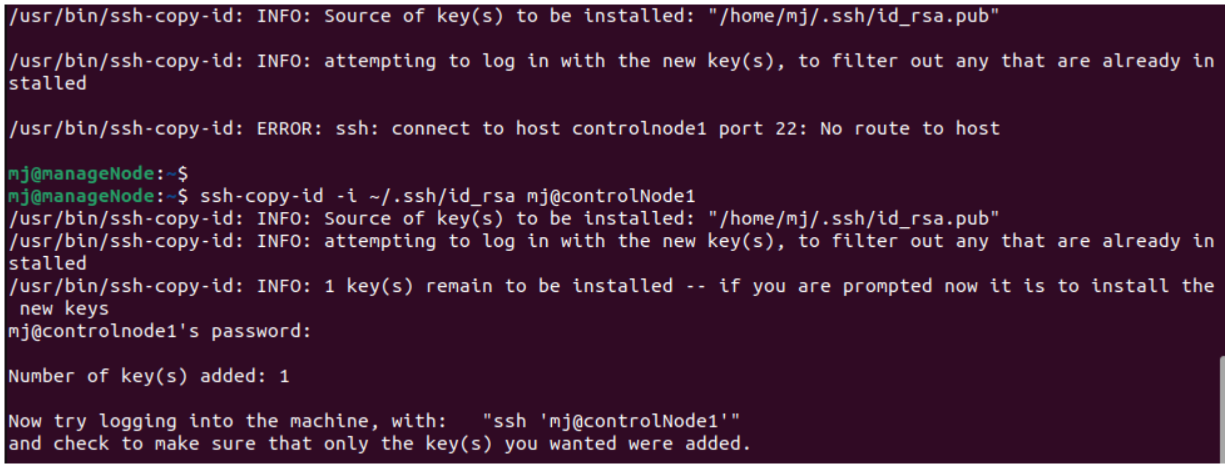
```
j@manageNode:~$ ssh-copy-id
sage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [
lternative ssh_config file] [[-o <ssh -o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are alre
nstalled
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can
seful if the remote only allows sftp
        -h|-?: print this help
```

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



```
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode1
```

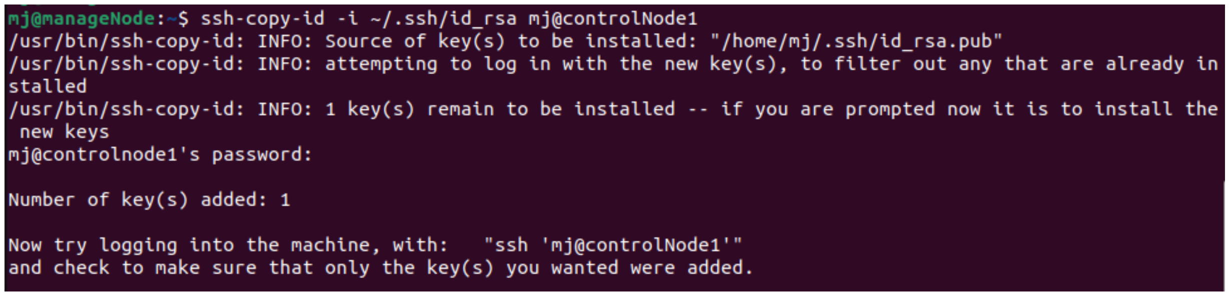
3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proves possession of the private key by digitally signing the key exchange.



```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.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 in
stalled

/usr/bin/ssh-copy-id: ERROR: ssh: connect to host controlnode1 port 22: No route to host

mj@manageNode:~$
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.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 in
stalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
 new keys
mj@controlnode1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mj@controlNode1'"
and check to make sure that only the key(s) you wanted were added.
```

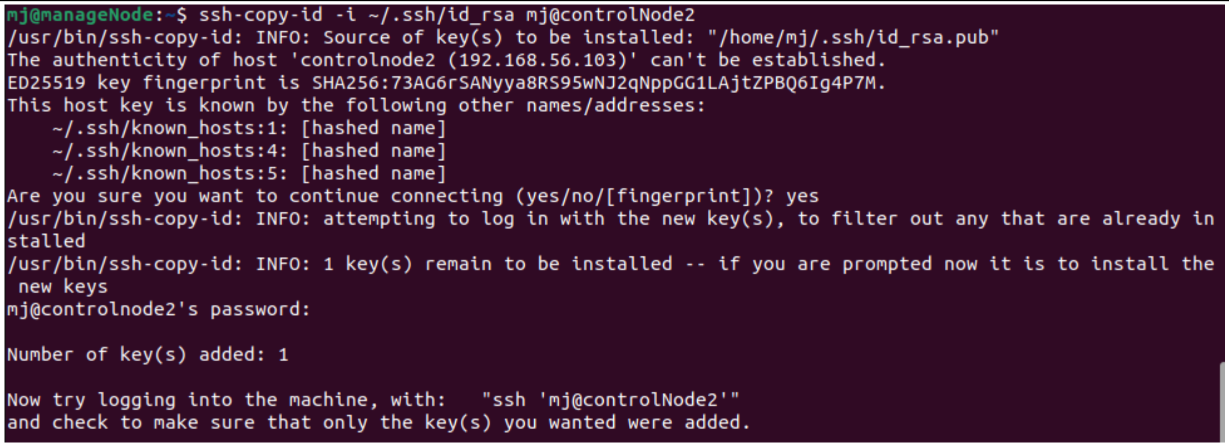
4. On the local machine, verify that you can SSH with Server 1 and Server 2. What did you notice? Did the connection ask for a password? If not, why?

Server 1:



```
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.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 in
stalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
 new keys
mj@controlnode1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mj@controlNode1'"
and check to make sure that only the key(s) you wanted were added.
```

Server 2:



```
mj@manageNode:~$ ssh-copy-id -i ~/.ssh/id_rsa mj@controlNode2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/mj/.ssh/id_rsa.pub"
The authenticity of host 'controlnode2 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:73AG6rSANyya8RS95wNJ2qNppGG1LAjtZPBQ6Ig4P7M.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
    ~/.ssh/known_hosts:4: [hashed name]
    ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already in
stalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
 new keys
mj@controlnode2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mj@controlNode2'"
and check to make sure that only the key(s) you wanted were added.
```

I noticed that using the command “ssh-copy-id” is very helpful when accessing servers because it doesn’t prompt you with the username and password every time you try to access servers. The command is useful as it can recognize the user as the authorized person who can freely access the server. It works because we added a key to the server, and the only one who can access is it freely is the main server or the local machine.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?

SSH in general is very useful in handling and securing connections to other servers remotely. We can manipulate and manage servers in such ways that is convenient to us. SSH is essential for data transfer, system upkeep, and secure remote server administration. When working with enterprise-level servers and systems, it offers a wide range of capabilities that make it an essential tool for system administrators and IT specialists. Overall, SSH is a great tool for system administrators in maintaining enterprise servers, as it provides encryption, access and authentication to users

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

We already know that we installed the public key to the remote servers if our input is correct and it prompted us that a key is added to the specific server. We can also distinguish it if we try to access a remote server and it doesn’t prompt us to input a username or a password each time we try to access a server. In addition, once connected to a remote network, we can also check if a public key is in the remote server, as it can be listed in the ~/.ssh/authorized_keys file. This file contains a list of public keys that are allowed to connect to the server using your account.

Part 2: Discussion

Provide screenshots for each task.

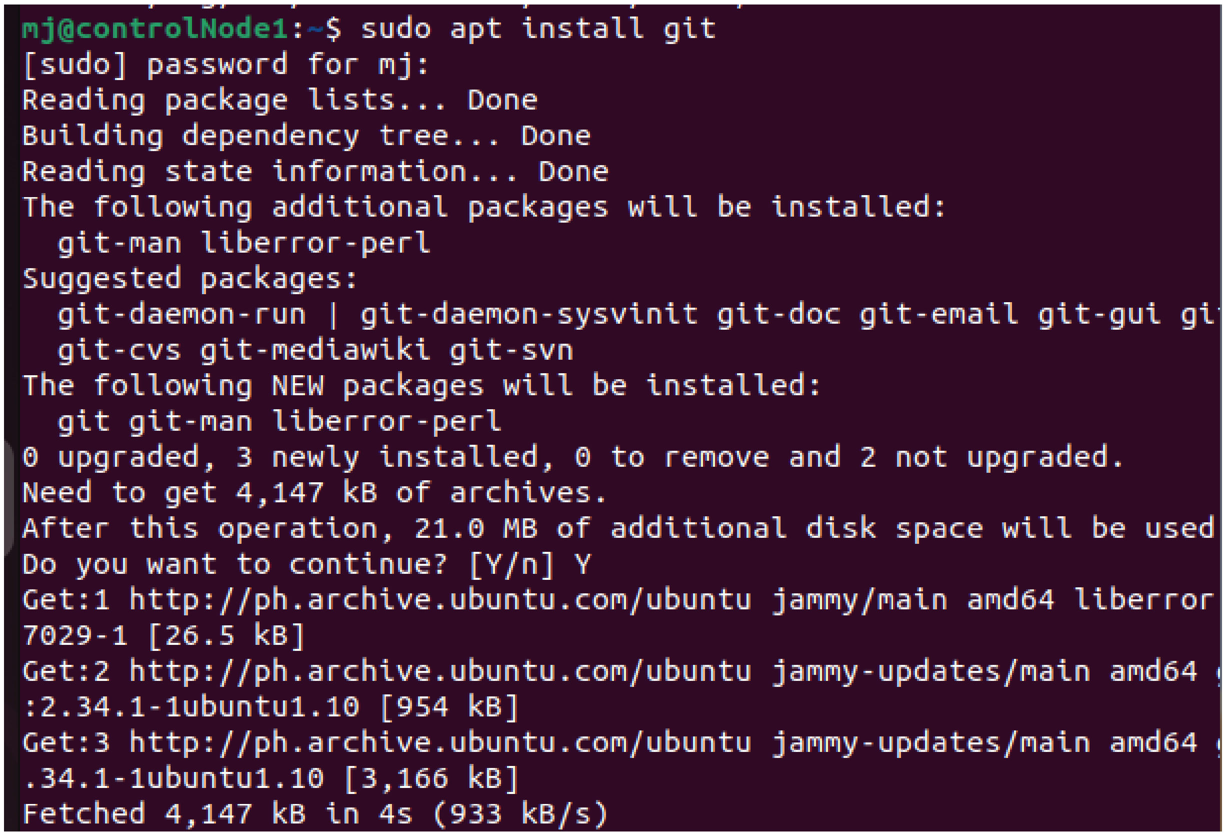
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

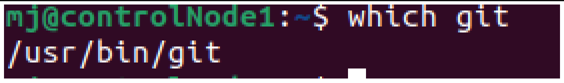
Task 3: Set up the Git Repository

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*



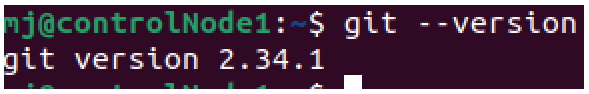
```
mj@controlNode1:~$ sudo apt install git
[sudo] password for mj:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gi
  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 2 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
7029-1 [26.5 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64
:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64
.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 4s (933 kB/s)
```

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



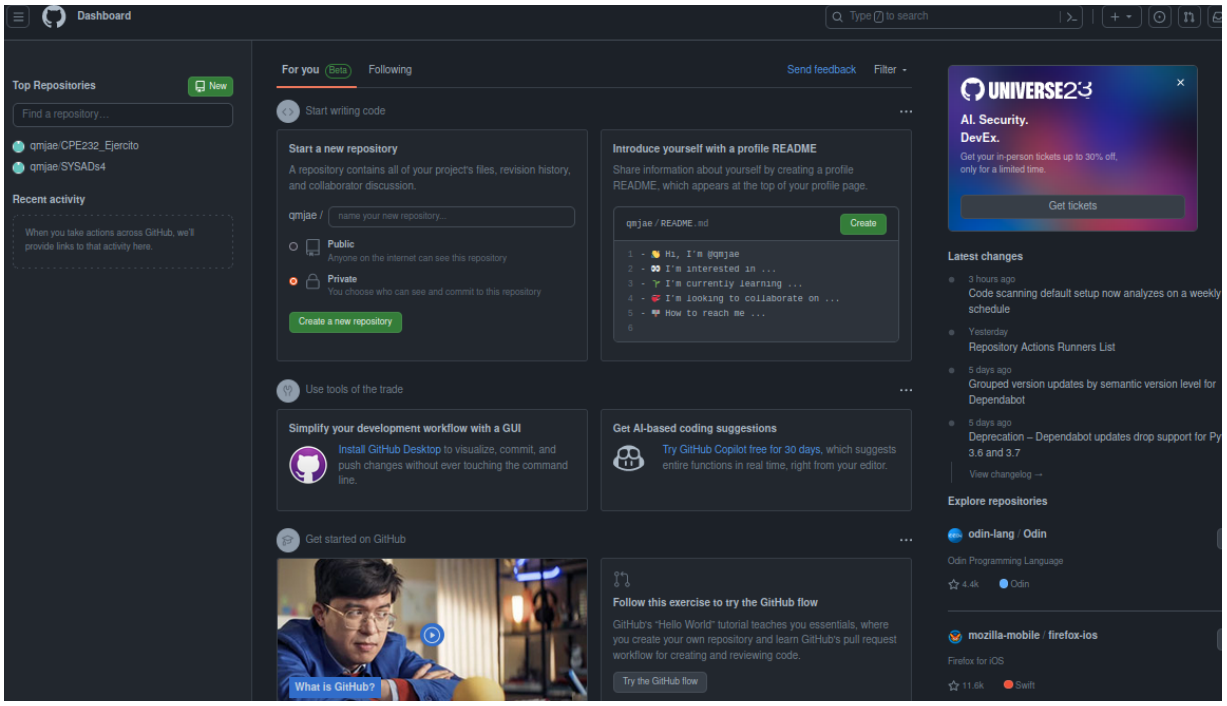
```
mj@controlNode1:~$ which git
/usr/bin/git
```

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



```
mj@controlNode1:~$ git --version
git version 2.34.1
```

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



When you take actions across GitHub, we'll provide links to that activity here.

Start a new repository

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

qmjae / name your new repository...

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

Share information about yourself by creating a profile README, which appears at the top of your profile page.

```
qmjae / README.md
- 👋 Hi, I'm @qmjae
- 👀 I'm interested in ...
- 🌱 I'm currently learning ...
- 💞️ I'm looking to collaborate on ...
- 📫 How to reach me ...
```

Simplify your development workflow with a GUI

Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

Get AI-based coding suggestions

Try GitHub Copilot free for 30 days, which suggests entire functions in real time, right from your editor.

Follow this exercise to try the GitHub flow

GitHub's "Hello World" tutorial teaches you essentials, where you create your own repository and learn GitHub's pull request workflow for creating and reviewing code.

AI. Security. DevEx.

Get your in-person tickets up to 30% off, only for a limited time.

3 hours ago — Code scanning default setup now analyzes on a weekly schedule

Yesterday — Repository Actions Runners List

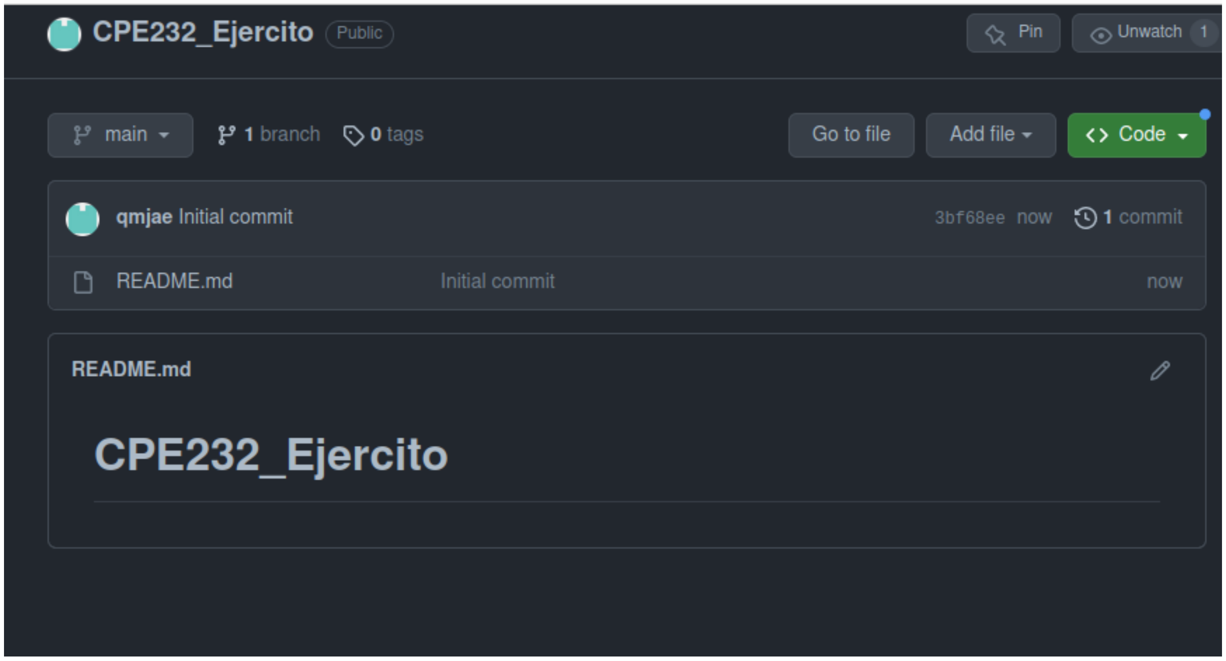
5 days ago — Grouped version updates by semantic version level for Dependabot

5 days ago — Deprecation – Dependabot updates drop support for Py 3.6 and 3.7

odin-lang / Odin — Odin Programming Language — 4.4k — Odin

mozilla-mobile / firefox-ios — Firefox for iOS — 11.6k — Swift

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.



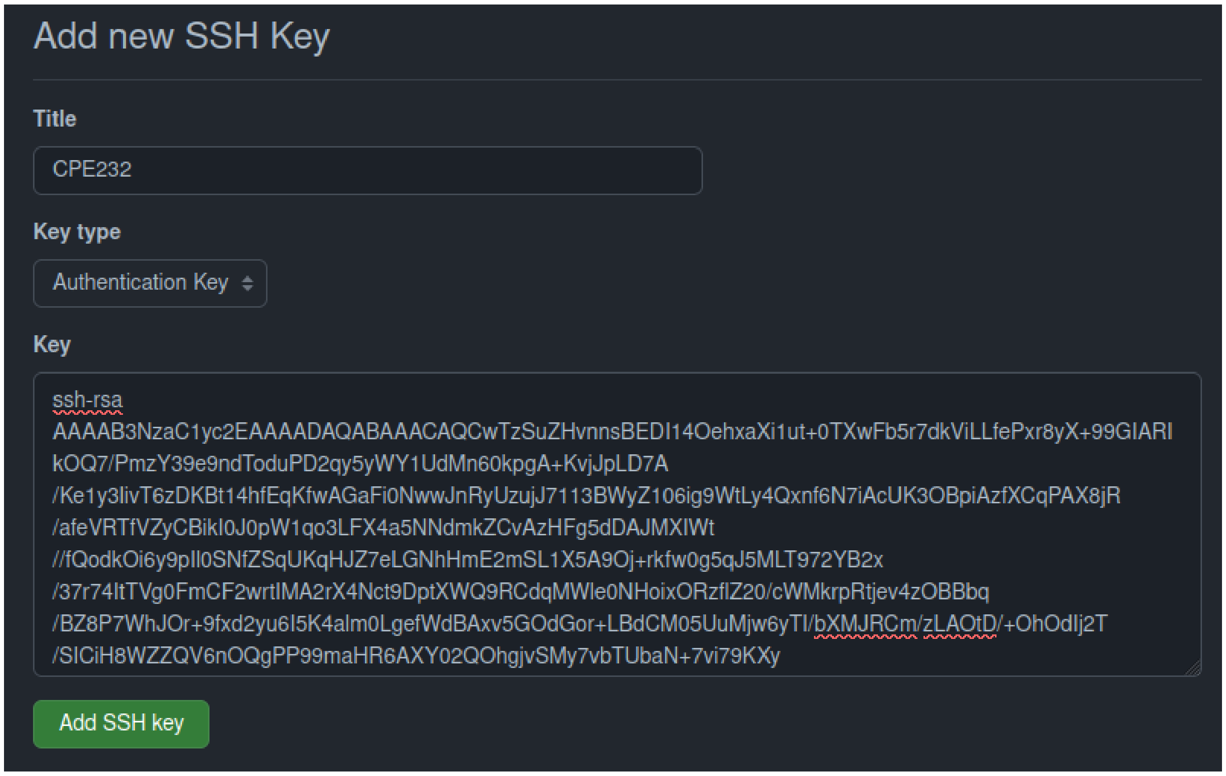
qmjae Initial commit 3bf68ee now 1 commit

README.md Initial commit now

README.md

CPE232_Ejercito

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.



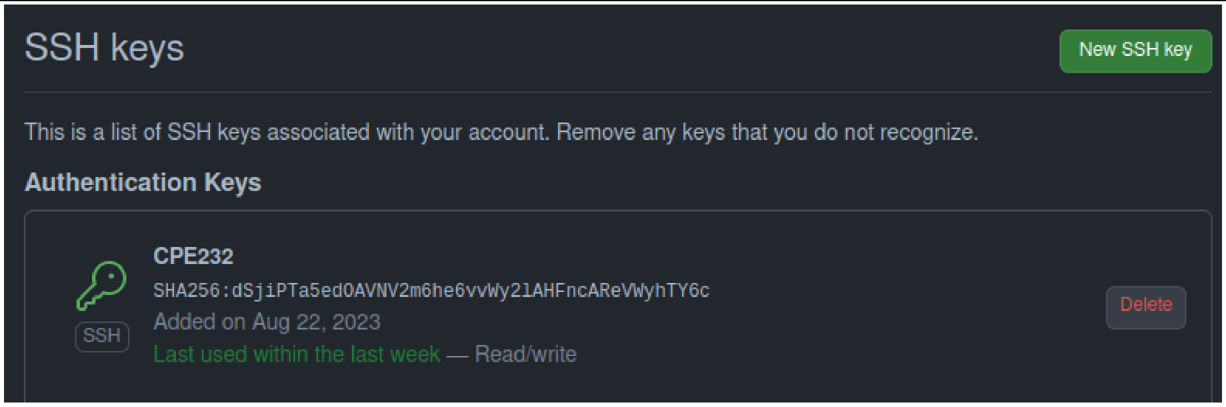
Add new SSH Key

Title: CPE232

Key type: Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCwTzSuZHvnnsBEDI14OehxaXi1ut+0TXwFb5r7dkViLLfePxr8yX+99GIARI
kOQ7/PmzY39e9ndToduPD2qy5yWY1UdMn60kpgA+KvjJpLD7A
/Ke1y3livT6zDKBt14hfEqKfwAGaFi0NwwJnRyUzujJ7113BWyZ106ig9WtLy4Qxnf6N7iAcUK3OBpiAzfXCqPAX8jR
/afeVRTfVZyCBikl0J0pW1qo3LFX4a5NNdmkZCvAzHFg5dDAJMXIWt
//fQodkOi6y9pIl0SNfZSqUKqHJZ7eLGNhHmE2mSL1X5A9Oj+rkfw0g5qJ5MLT972YB2x
/37r74ItTVg0FmCF2wrtIMA2rX4Nct9DptXWQ9RCdqMWle0NHoixORzfIZ20/cWMkrpRtjev4zOBBbq
/BZ8P7WhJOr+9fxd2yu6I5K4alm0LgefWdBAxv5GOdGor+LBdCM05UuMjw6yTI/bXMJRCm/zLAOtD/+OhOdIj2T
/SICiH8WZZQV6nOQgPP99maHR6AXY02QOhgjvSMy7vbTUbaN+7vi79KXy
```

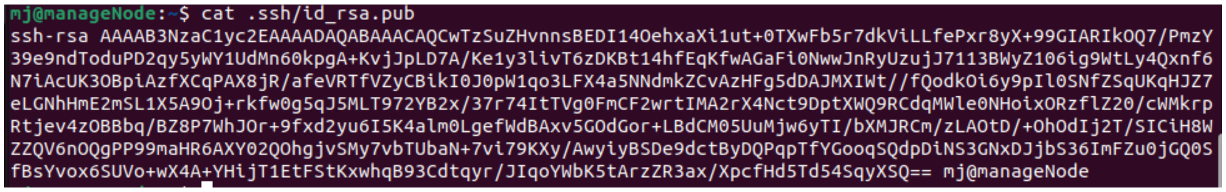


This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

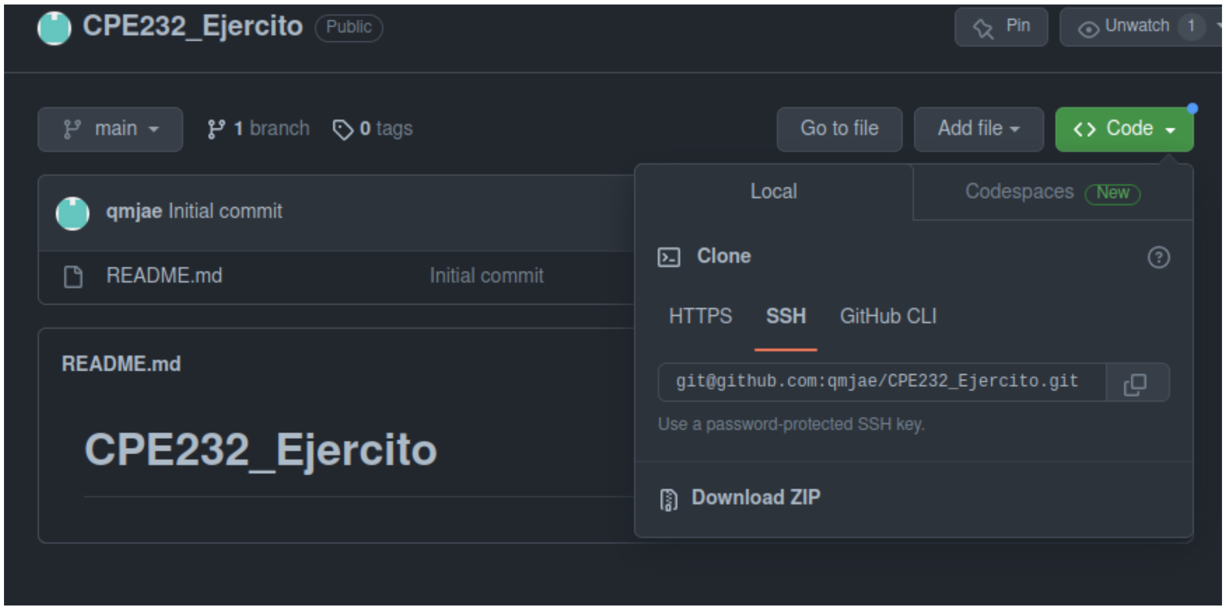
CPE232
SHA256:dSjiPTa5edOAVNV2m6he6vvWy2lAHFncAReVWyhTY6c
Added on Aug 22, 2023
Last used within the last week — Read/write

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.



```
mj@manageNode:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCwTzSuZHvnnsBEDI14OehxaXi1ut+0TXwFb5r7dkViLLfePxr8yX+99GIARIkOQ7/PmzY
39e9ndToduPD2qy5yWY1UdMn60kpgA+KvjJpLD7A/Ke1y3livT6zDKBt14hfEqKfwAGaFi0NwwJnRyUzujJ7113BWyZ106ig9WtLy4Qxnf6
N7iAcUK3OBpiAzfXCqPAX8jR/afeVRTfVZyCBikI0J0pW1qo3LFX4a5NNdmkZCvAzHFg5dDAJMXIWt//fQodkOi6y9pIl0SNfZSqUKqHJZ7
eLGNhHmE2mSL1X5A9Oj+rkfw0g5qJ5MLT972YB2x/37r74ItTVg0FmCF2wrtIMA2rX4Nct9DptXWQ9RCdqMWle0NHoixORzflZ20/cWMkrp
Rtjev4zOBBbq/BZ8P7WhJOr+9fxd2yu6I5K4alm0LgefWdBAxv5GOdGor+LBdCM05UuMjw6yTI/bXMJRCm/zLAOtD/+OhOdIj2T/SICiH8W
ZZQV6nOQgPP99maHR6AXY02QOhgjvSMy7vbTUbaN+7vi79KXy/AwyiyBSDe9dctByDQPqpTfYGooqSQdpDiNS3GNxDJjbS36ImFZu0jGQ0S
fBsYvox6SUVo+wX4A+YHijT1EtFStKxwhqB93Cdtqyr/JIqoYWbK5tArzZR3ax/XpcfHd5Td54SqyXSQ== mj@manageNode
```

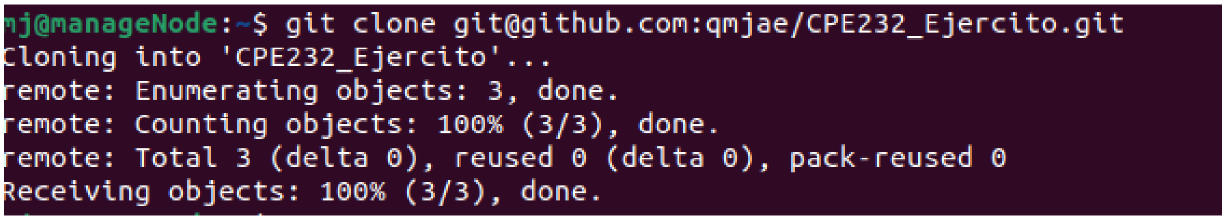
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



```
git@github.com:qmjae/CPE232_Ejercito.git
```

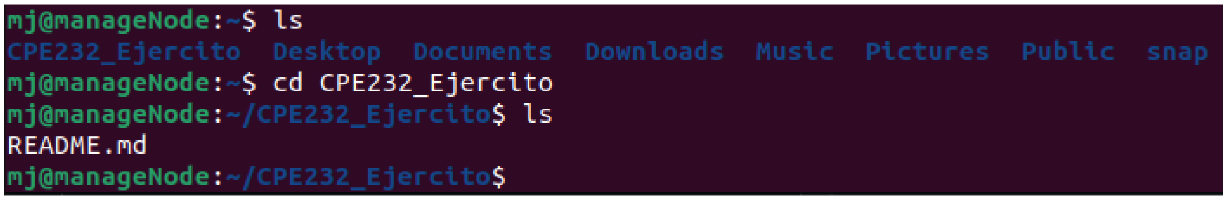
Use a password-protected SSH key.

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.



```
mj@manageNode:~$ git clone git@github.com:qmjae/CPE232_Ejercito.git
Cloning into 'CPE232_Ejercito'...
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.
```

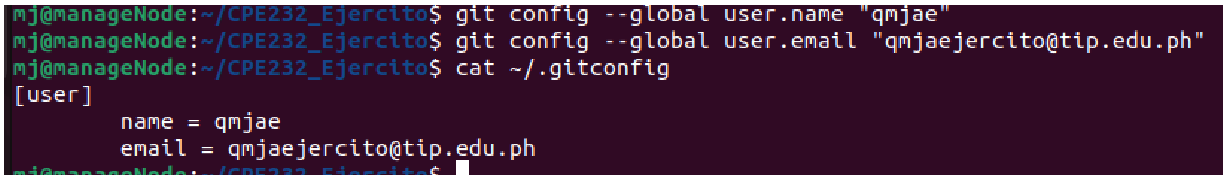
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
mj@manageNode:~$ ls
CPE232_Ejercito  Desktop  Documents  Downloads  Music  Pictures  Public  snap
mj@manageNode:~$ cd CPE232_Ejercito
mj@manageNode:~/CPE232_Ejercito$ ls
README.md
mj@manageNode:~/CPE232_Ejercito$
```

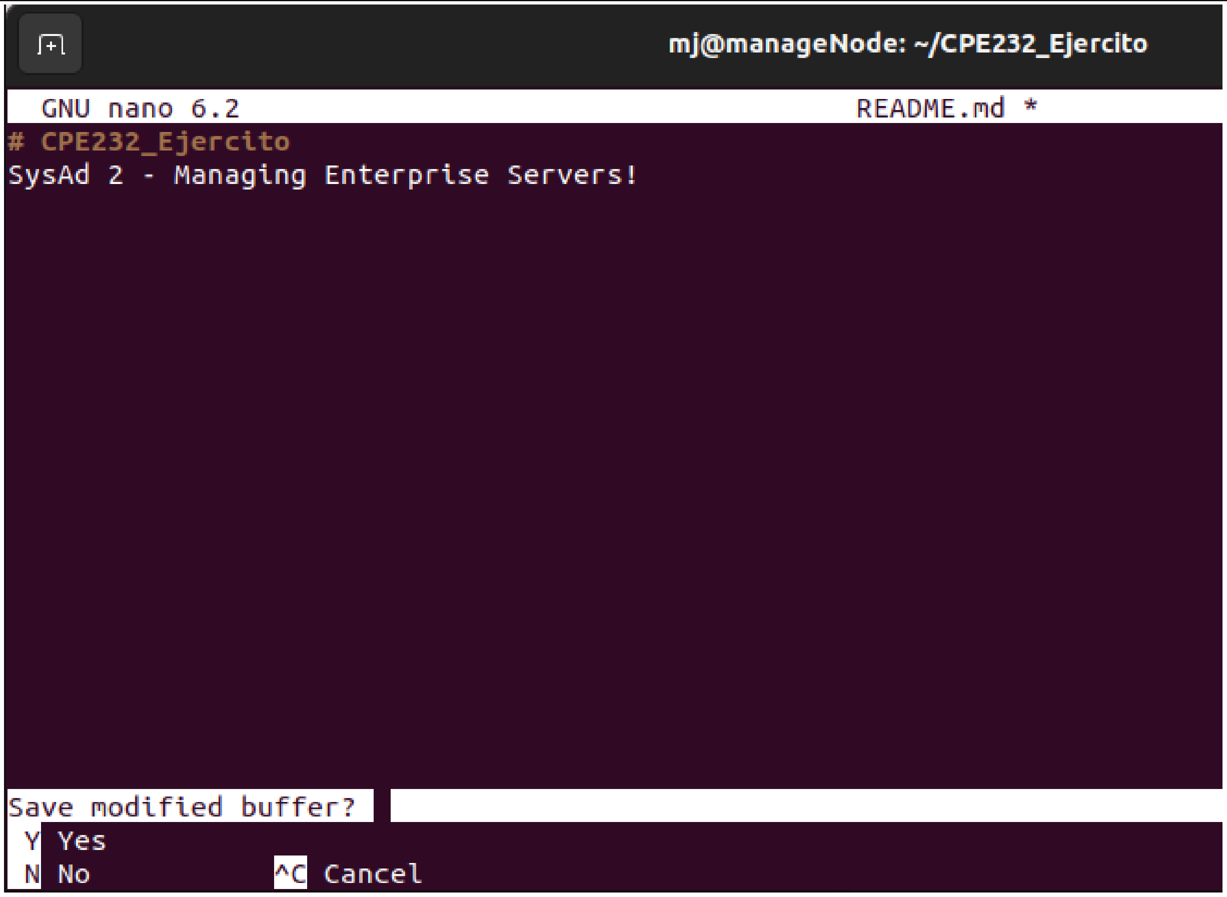
g. Use the following commands to personalize your git.

- *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*



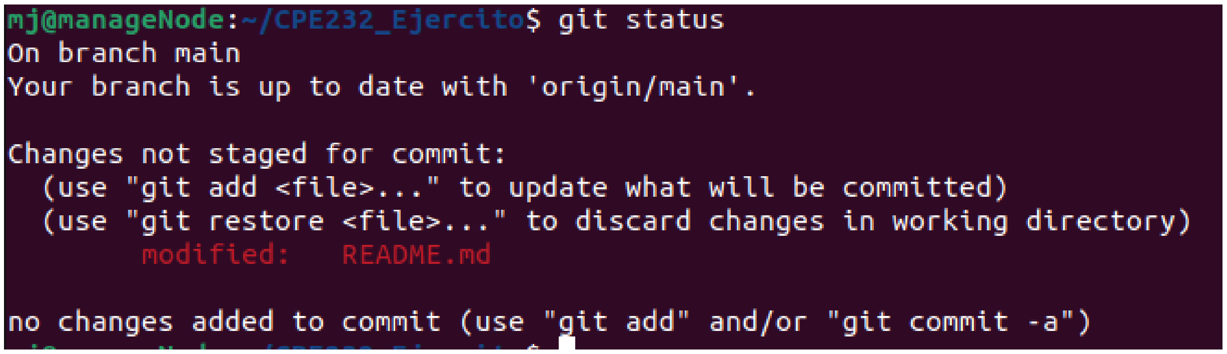
```
mj@manageNode:~/CPE232_Ejercito$ git config --global user.name "qmjae"
mj@manageNode:~/CPE232_Ejercito$ git config --global user.email "qmjaejercito@tip.edu.ph"
mj@manageNode:~/CPE232_Ejercito$ cat ~/.gitconfig
[user]
        name = qmjae
        email = qmjaejercito@tip.edu.ph
```

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.



```
mj@manageNode: ~/CPE232_Ejercito
  GNU nano 6.2                                              README.md *
# CPE232_Ejercito
SysAd 2 - Managing Enterprise Servers!



Save modified buffer?
 Y Yes
 N No              ^C Cancel
```

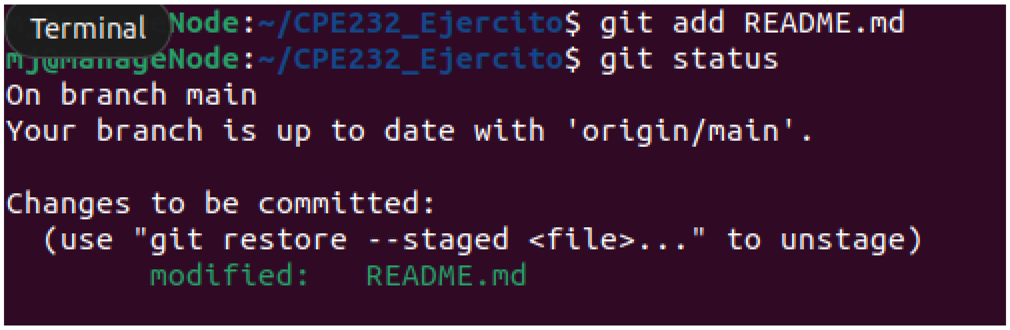
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What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



```
mj@manageNode:~/CPE232_Ejercito$ 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")
```

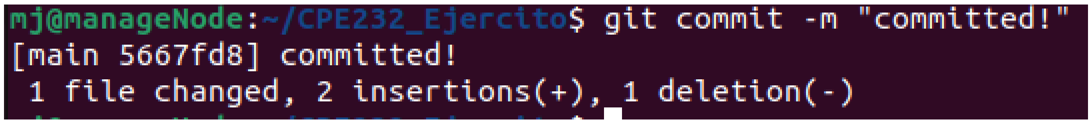
The results of issuing this command shows the files that are on stage, or that are ready for commit. As shown in the output above, the red highlight shows that the changes made in the "README.md" file have not yet been fully changed. We need to add it to the stage, then commit it and push it so that the changes will transfer to the original file, which is in the github page.

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



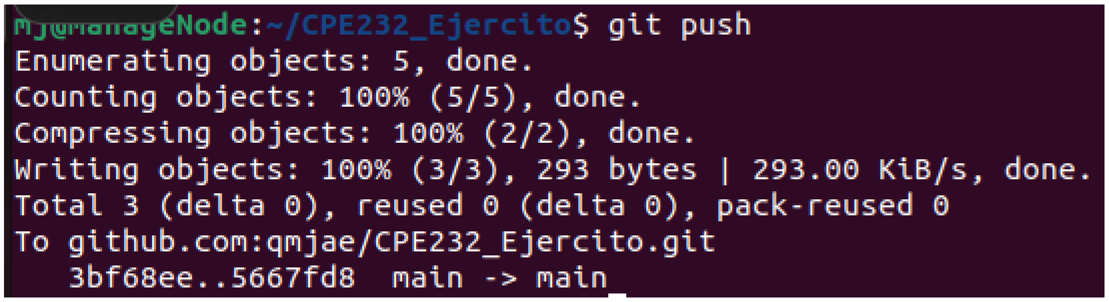
```
Terminal Node:~/CPE232_Ejercito$ git add README.md
mj@manageNode:~/CPE232_Ejercito$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

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.



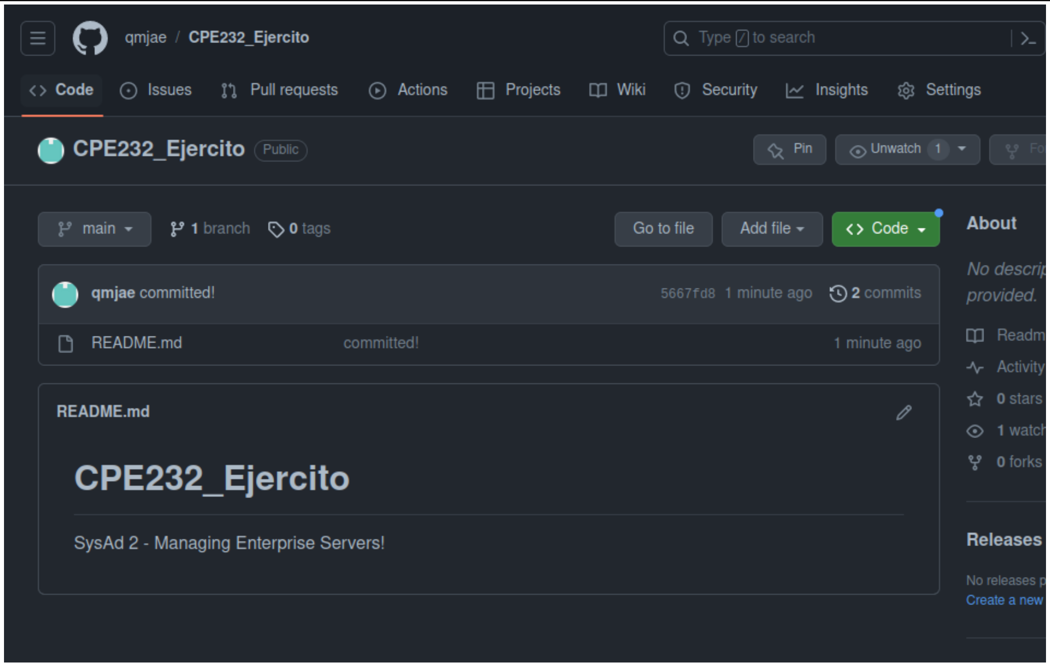
```
mj@manageNode:~/CPE232_Ejercito$ git commit -m "committed!"
[main 5667fd8] committed!
 1 file changed, 2 insertions(+), 1 deletion(-)
```

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.*



```
mj@manageNode:~/CPE232_Ejercito$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qmjae/CPE232_Ejercito.git
   3bf68ee..5667fd8  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.



qmjae committed! 5667fd8 1 minute ago 2 commits

README.md committed! 1 minute ago

README.md

CPE232_Ejercito

SysAd 2 - Managing Enterprise Servers!

Reflections:

Answer the following:

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

I have created an SSH Key Pair for User Authentication which in this case, I was able to use this encryption key to store, as well as copying the public key to the remote servers, which are controlNode1 and controlNode2. In addition to that, I was also able to copy the public key to the remote servers so that the main local machine will be authorized to use the other servers freely, without needing to prompt the user each time. Overall, we were able to access the remote servers using SSH, and we were able to manipulate, configure and verify connectivity using SSH-related commands

4. How important is the inventory file?

The inventory file serves as a crucial blueprint, outlining both individual hosts and host groups that interact with commands, modules, and tasks within a playbook. This file's format varies, contingent on the specific setup and plugins at play in your environment.

Conclusions/Learnings:

In this activity, I was able to learn how to use and create SSH keys, use it for remote servers, and I was also able to remotely login through SSH using a key without using a password. I was also able to create public and private keys, I was able to understand their concept, because the private keys reside in the main local machine, and public keys to remote servers. I was also able to understand and know the importance of encryption because it really helps us to secure and strengthen the protection of a server.

In addition, I was also able to learn how to use Git commands and the basics of GitHub. I was amazed that I can directly connect my computer to the website using only the CLI of my computer. I was able to add, commit and push files to the repository that I made in this activity which was fun and interesting.

In conclusion, I was able to attain the objectives of this activity, which is to configure remote and local machine to connect via SSH using a key instead of using a password, create a public key and private key, verify connectivity, setup Git repository using local and remote repositories, and configure and run ad hoc commands from local machine to remote server.