Name: Bacong, El Cid A.	**Date Performed:** 8/15/25
Course/Section: CPE212 - CPE31S4	**Date Submitted:** 8/15/25
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 2025-2026

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.

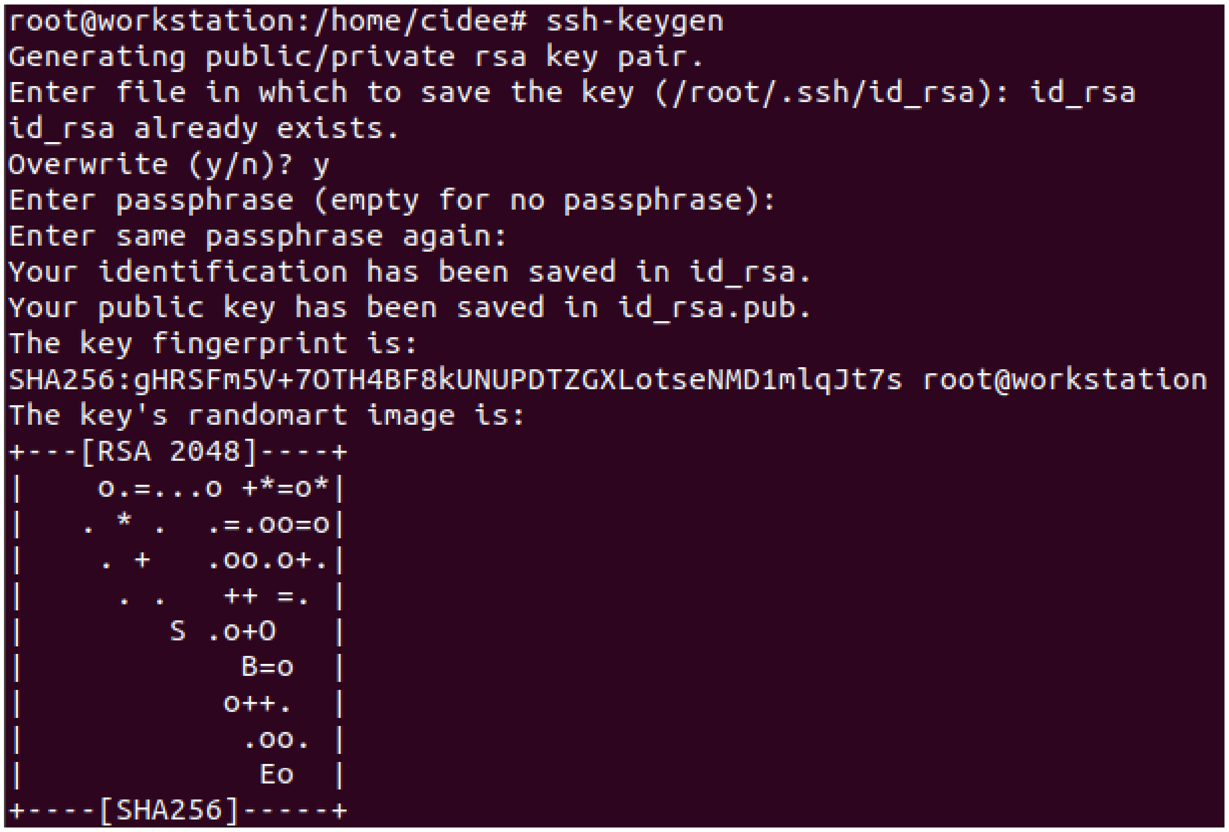
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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

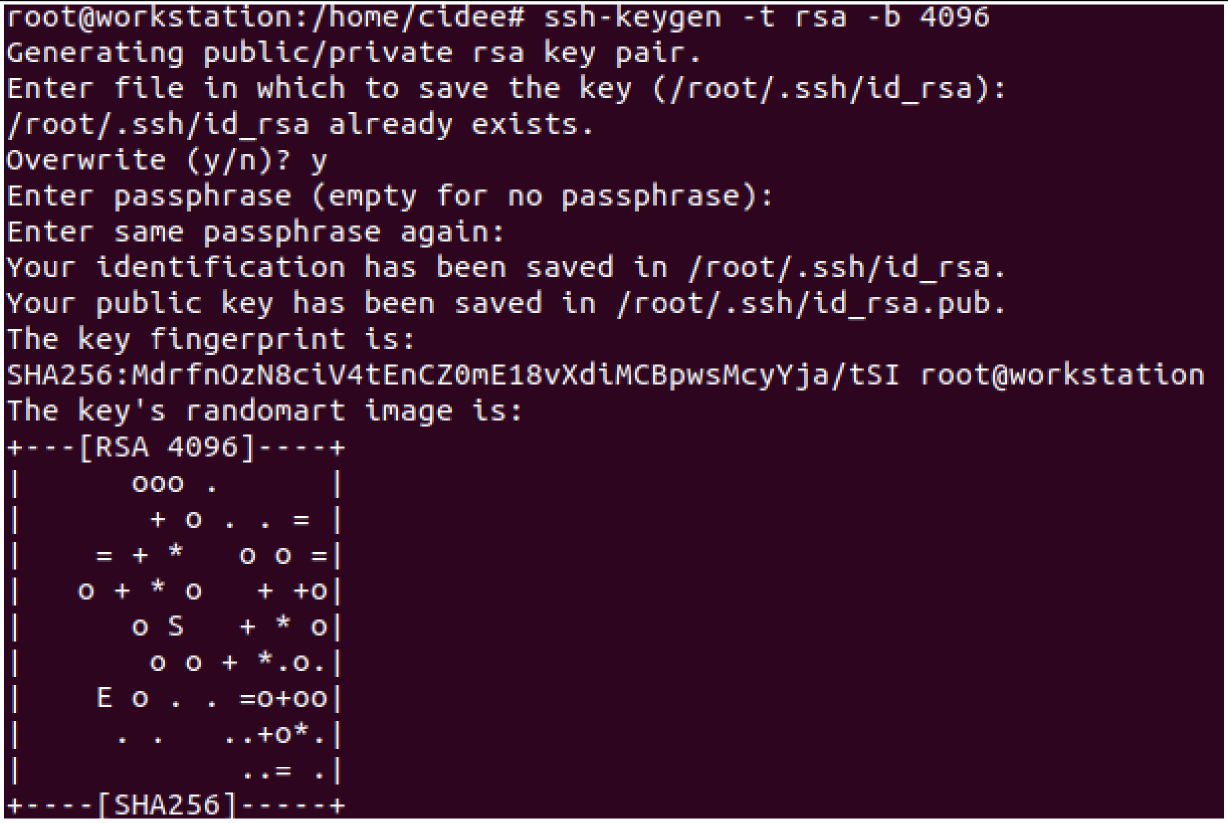
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First,

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*.



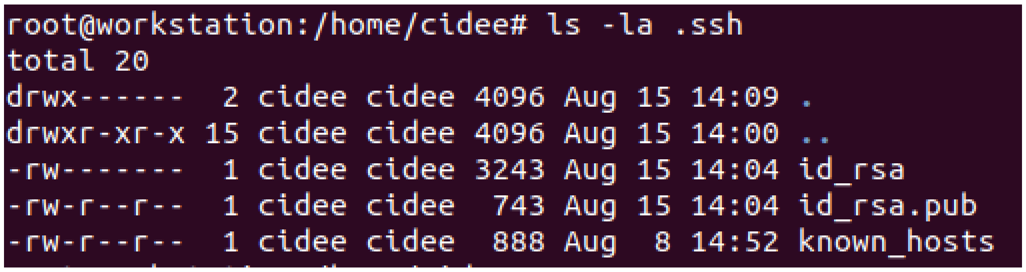
```
root@workstation:/home/cidee# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): id_rsa
id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:gHRSFm5V+7OTH4BF8kUNUPDTZGXLotseNMD1mlqJt7s root@workstation
The key's randomart image is:
+---[RSA 2048]----+
|    o.=...o +*=o*|
|   . * .  .=.oo=o|
|    . +   .oo.o+.|
|     . .   ++ =. |
|        S .o+O   |
|            B=o  |
|           o++.  |
|            .oo. |
|             Eo  |
+----[SHA256]-----+
```

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.



```
root@workstation:/home/cidee# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.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 /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:MdrfnOzN8ciV4tEnCZ0mE18vXdiMCBpwsMcyYja/tSI root@workstation
The key's randomart image is:
+---[RSA 4096]----+
|       ooo .     |
|        + o . . =|
|     = + *   o o =|
|    o + * o   + +o|
|       o S   + * o|
|        o o + *.o.|
|     E o . . =o+oo|
|      . .   ..+o*.|
|             ..= .|
+----[SHA256]-----+
```

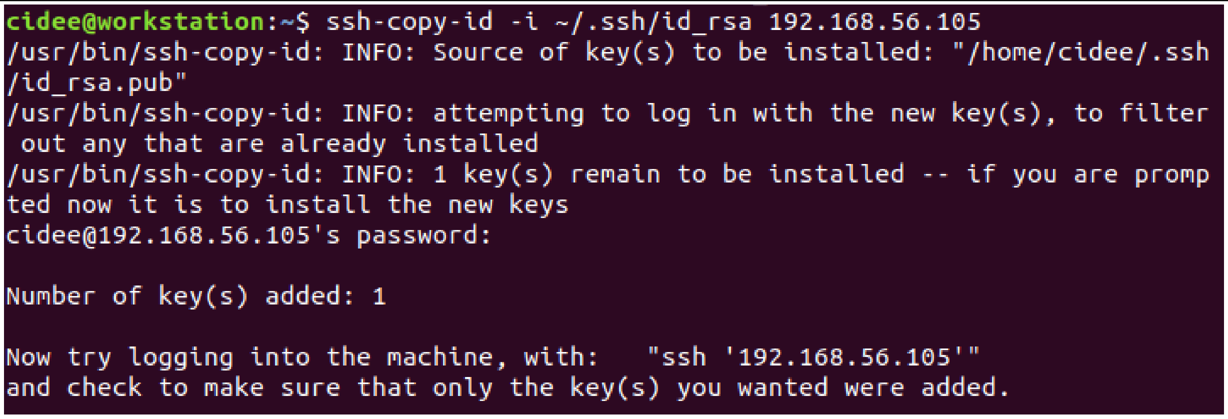
3. When asked for a passphrase, just press enter. The passphrase is used for encrypting the key, so that it cannot be used even if someone obtains the private key file. The passphrase should be cryptographically strong.
4. Verify that you have created the key by issuing the command *ls -la .ssh.* The command should show the .ssh directory containing a pair of keys. For example, id_rsa.pub and id_rsa.



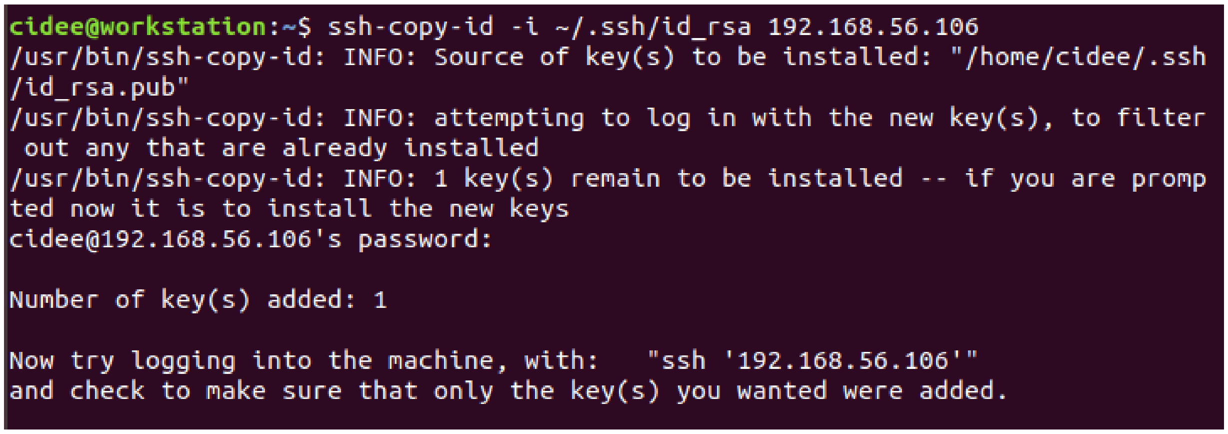
```
root@workstation:/home/cidee# ls -la .ssh
total 20
drwx------  2 cidee cidee 4096 Aug 15 14:09 .
drwxr-xr-x 15 cidee cidee 4096 Aug 15 14:00 ..
-rw-------  1 cidee cidee 3243 Aug 15 14:04 id_rsa
-rw-r--r--  1 cidee cidee  743 Aug 15 14:04 id_rsa.pub
-rw-r--r--  1 cidee cidee  888 Aug  8 14:52 known_hosts
```

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*

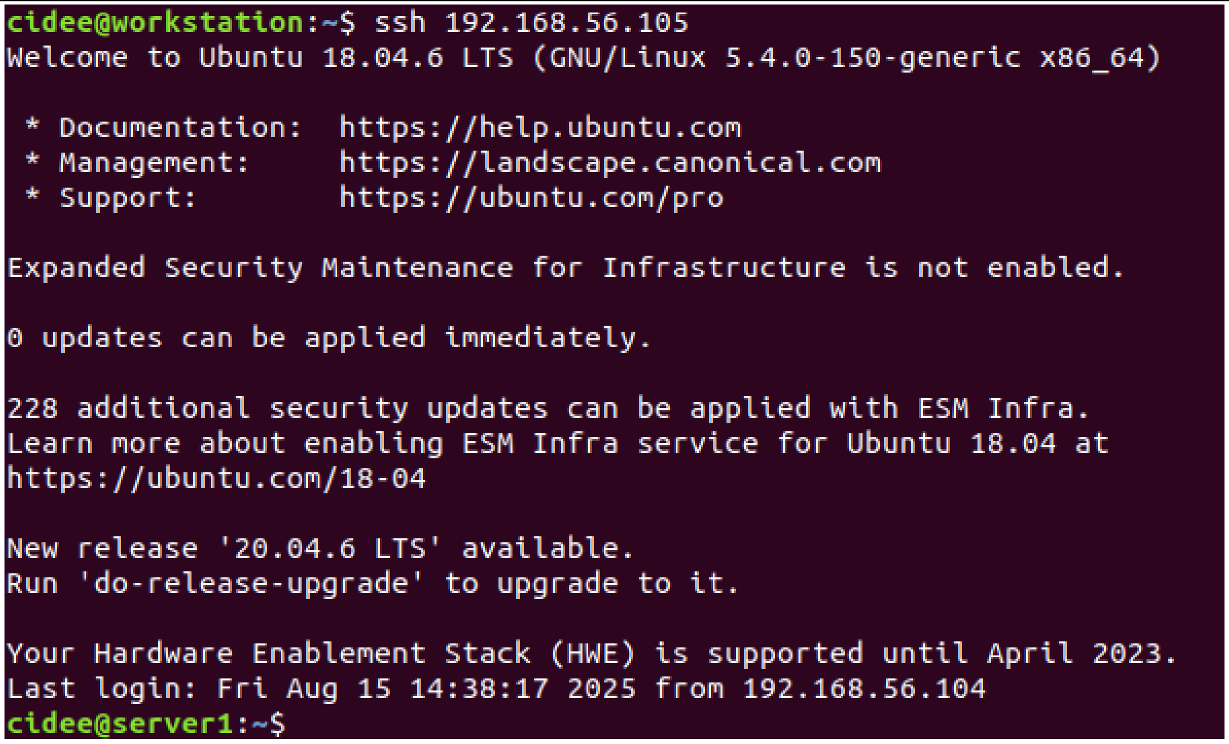


```
cidee@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa 192.168.56.105
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/cidee/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
cidee@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.56.105'"
and check to make sure that only the key(s) you wanted were added.
```

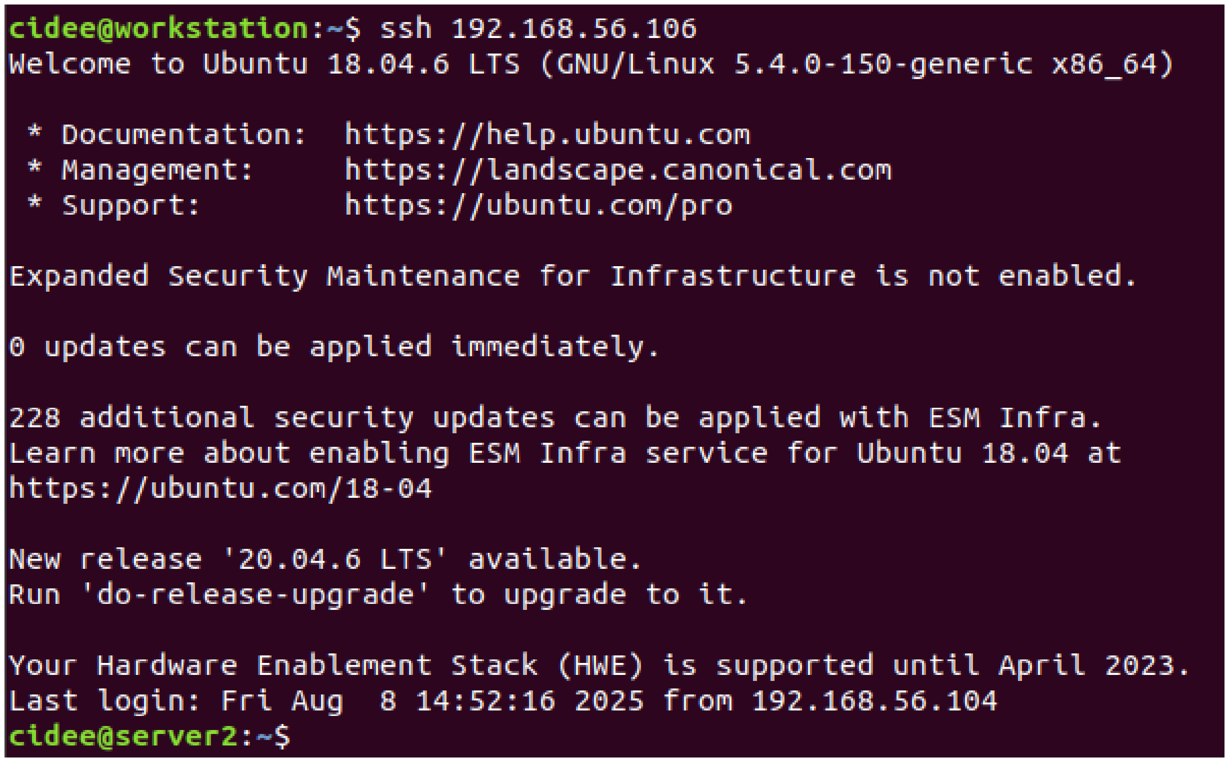


```
cidee@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa 192.168.56.106
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/cidee/.ssh
/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
cidee@192.168.56.106's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.56.106'"
and check to make sure that only the key(s) you wanted were added.
```

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?



```
cidee@workstation:~$ ssh 192.168.56.105
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

228 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Aug 15 14:38:17 2025 from 192.168.56.104
cidee@server1:~$
```



```
cidee@workstation:~$ ssh 192.168.56.106
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

228 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Fri Aug  8 14:52:16 2025 from 192.168.56.104
cidee@server2:~$
```

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
   SSH (Secure Shell) is a secure network protocol that allows users, especially system administrators, to safely access and manage remote computers over

insecure networks like the internet. It encrypts all communications to prevent unauthorized access to sensitive data.

2. How do you know that you already installed the public key to the remote servers?
   You can confirm it by attempting an SSH login. If you were able to access it directly then it means the public key is recognized by the server.

Part 2: Discussion

Provide screenshots for each task.

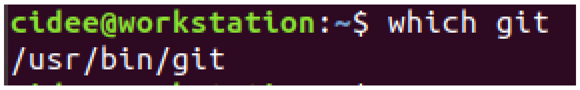
It is assumed that you are done with the last activity (**Activity 2: SSH Key-Based Authentication**).

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

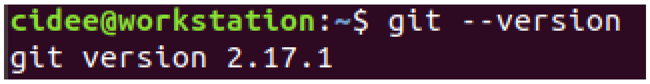
Task 3: Set up the Git Repository

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*



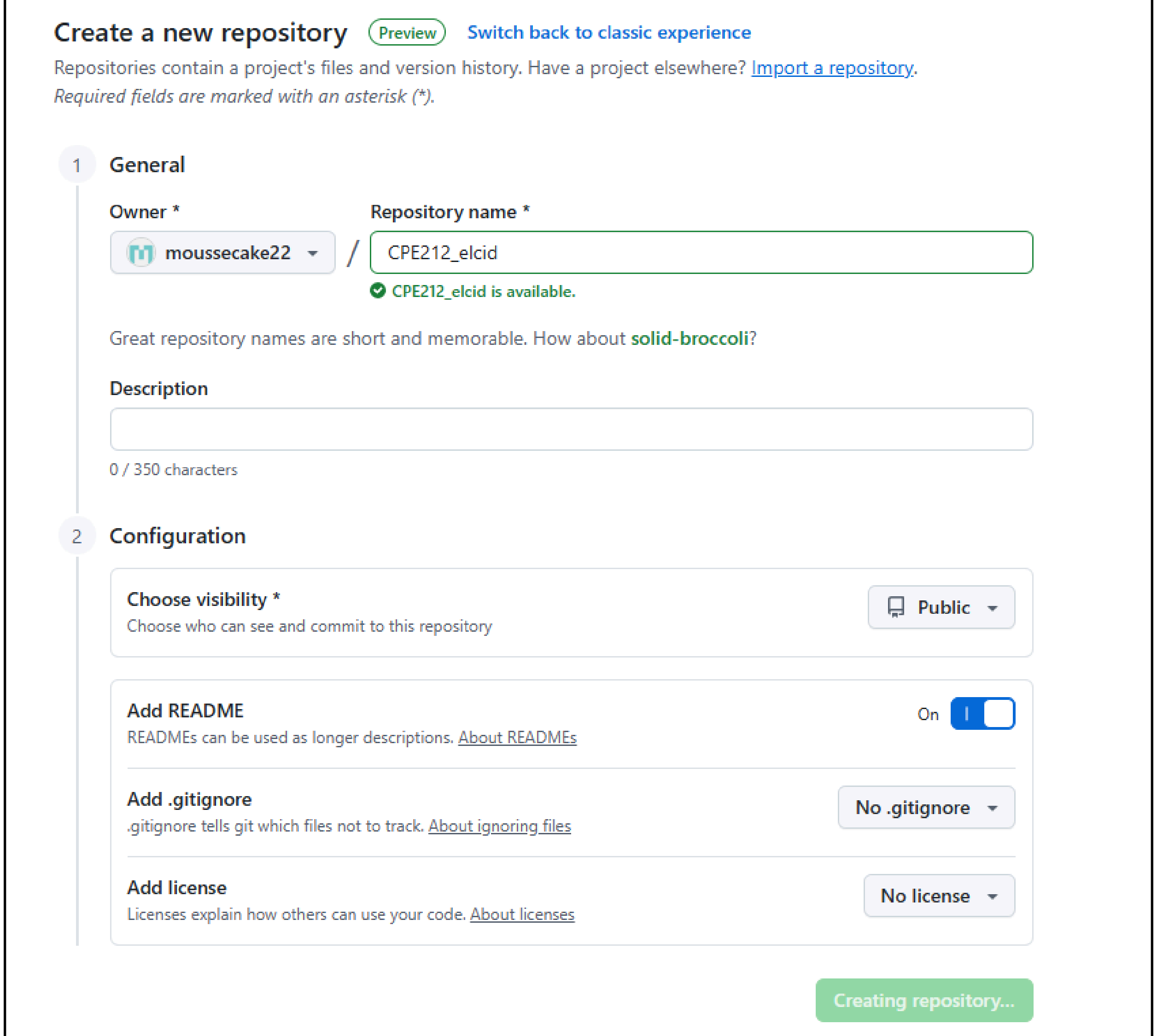
```
cidee@workstation:~$ which git
/usr/bin/git
```

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.



```
cidee@workstation:~$ git --version
git version 2.17.1
```

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.


Create a new repository Preview Switch back to classic experience

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner * moussecake22 / Repository name * CPE212_elcid

CPE212_elcid is available.

Great repository names are short and memorable. How about solid-broccoli?

Description

0 / 350 characters

2 Configuration

Choose visibility * — Choose who can see and commit to this repository — Public

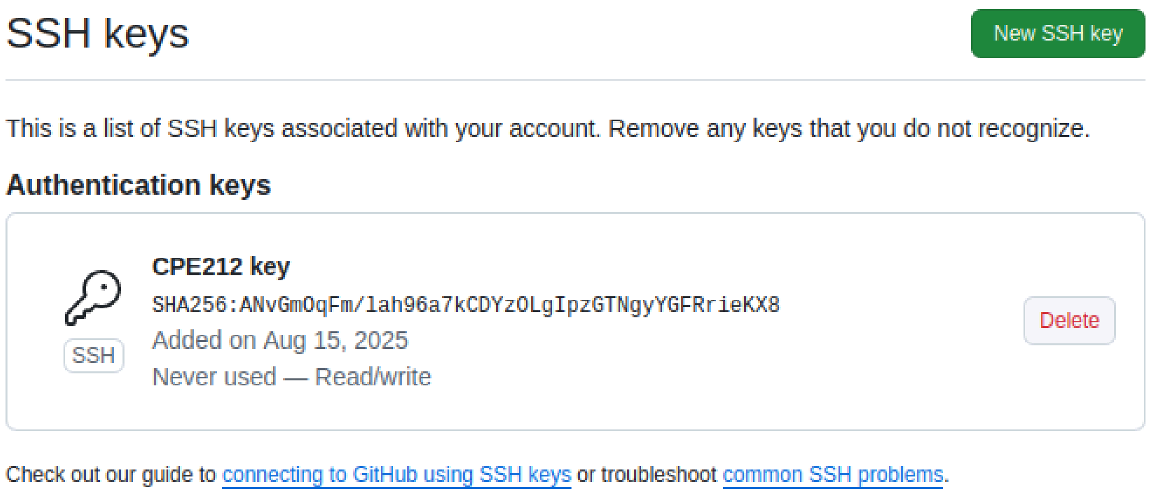
Add README — READMEs can be used as longer descriptions. About READMEs — On

Add .gitignore — .gitignore tells git which files not to track. About ignoring files — No .gitignore

Add license — Licenses explain how others can use your code. About licenses — No license

Creating 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.



SSH keys

New SSH key

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

Authentication keys

SSH

CPE212 key

`SHA256:ANvGmOqFm/lah96a7kCDYzOLgIpzGTNgyYGFRrieKX8`

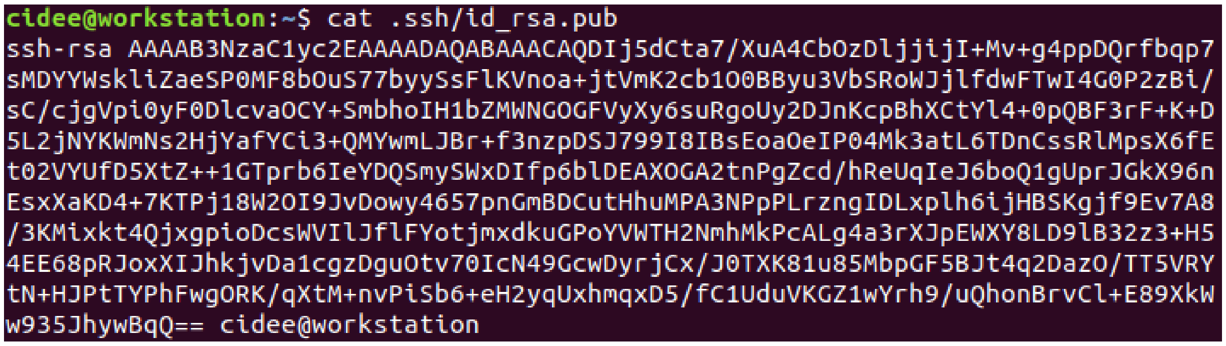
Added on Aug 15, 2025

Never used — Read/write

Delete

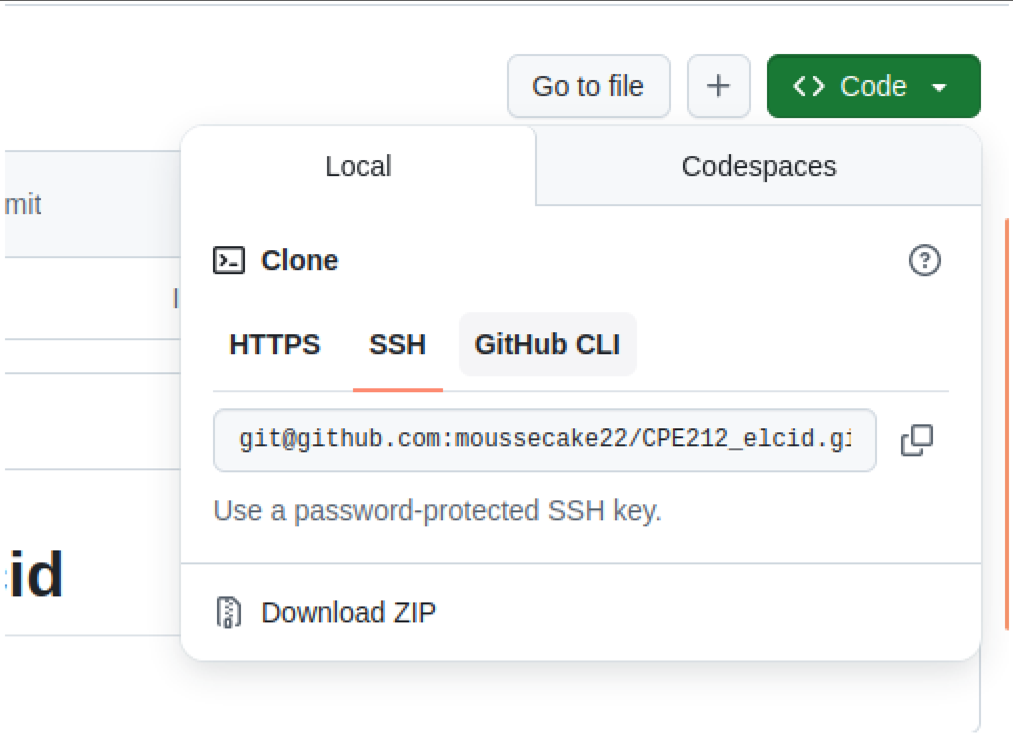
Check out our guide to connecting to GitHub using SSH keys or troubleshoot common SSH problems.

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.



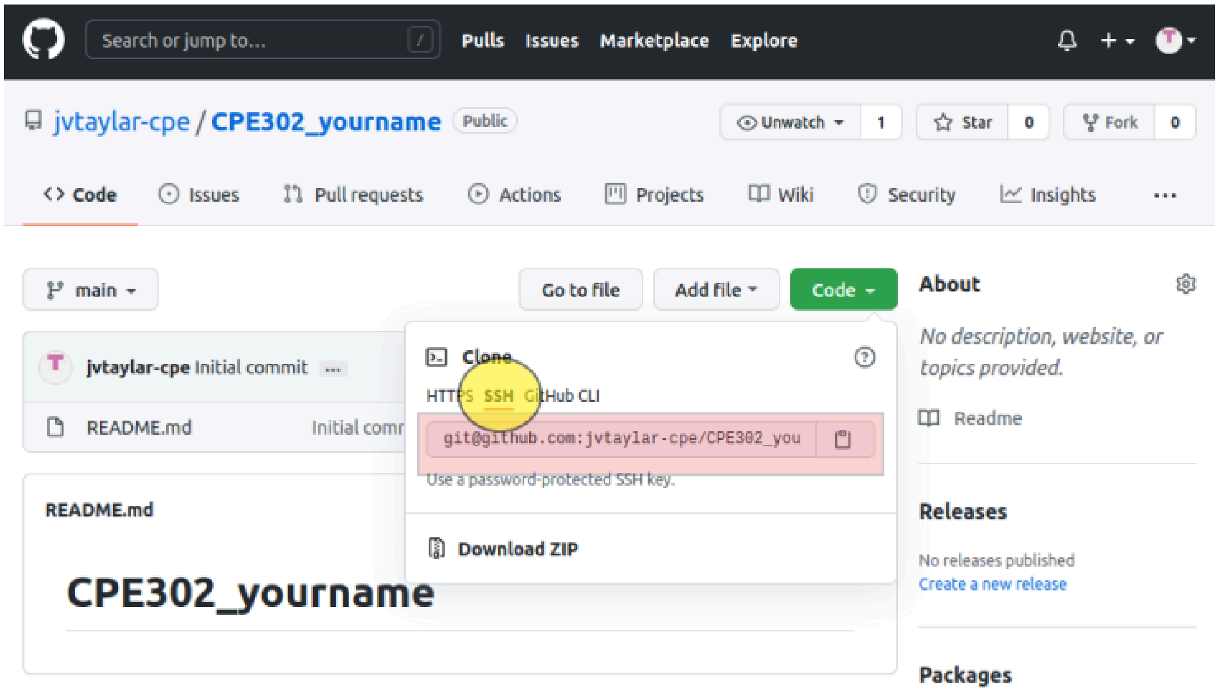
```
cidee@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDIj5dCta7/XuA4CbOzDljjijI+Mv+g4ppDQrfbqp7
sMDYYWskliZaeSP0MF8bOuS77byySsFlKVnoa+jtVmK2cb1O0BByu3VbSRoWJjlfdwFTwI4G0P2zBi/
sC/cjgVpi0yF0DlcvaOCY+SmbhoIH1bZMWNGOGFVyXy6suRgoUy2DJnKcpBhXCtYl4+0pQBF3rF+K+D
5L2jNYKWmNs2HjYafYCi3+QMYwmLJBr+f3nzpDSJ799I8IBsEoaOeIP04Mk3atL6TDnCssRlMpsX6fE
t02VYUfD5XtZ++1GTprb6IeYDQSmySWxDIfp6blDEAXOGA2tnPgZcd/hReUqIeJ6boQ1gUprJGkX96n
EsxXaKD4+7KTPj18W2OI9JvDowy4657pnGmBDCutHhuMPA3NPpPLrzngIDLxplh6ijHBSKgjf9Ev7A8
/3KMixkt4QjxgpioDcsWVIlJflFYotjmxdkuGPoYVWTH2NmhMkPcALg4a3rXJpEWXY8LD9lB32z3+H5
4EE68pRJoxXIJhkjvDa1cgzDguOtv70IcN49GcwDyrjCx/J0TXK81u85MbpGF5BJt4q2DazO/TT5VRY
tN+HJPtTYPhFwgORK/qXtM+nvPiSb6+eH2yqUxhmqxD5/fC1UduVKGZ1wYrh9/uQhonBrvCl+E89XkW
w935JhywBqQ== cidee@workstation
```

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



```
git@github.com:moussecake22/CPE212_elcid.gi
```

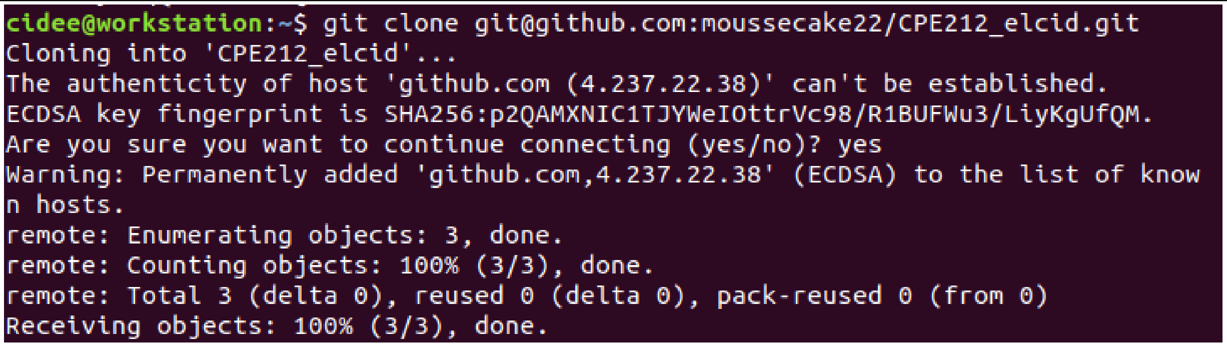
Use a password-protected SSH key.



```
git@github.com:jvtaylar-cpe/CPE302_you
```

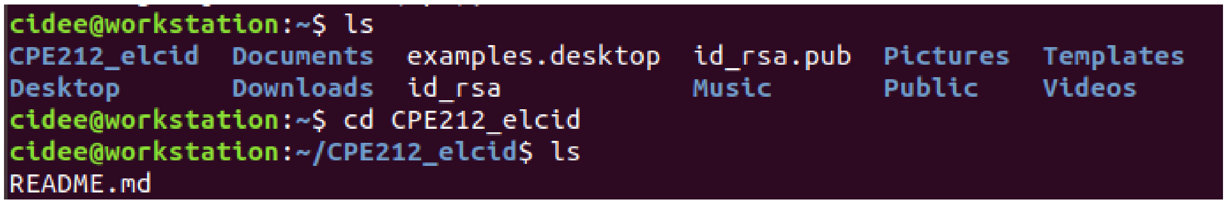
Use a password-protected SSH key.

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.



```
cidee@workstation:~$ git clone git@github.com:moussecake22/CPE212_elcid.git
Cloning into 'CPE212_elcid'...
The authenticity of host 'github.com (4.237.22.38)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,4.237.22.38' (ECDSA) to the list of know
n 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 (from 0)
Receiving objects: 100% (3/3), done.
```

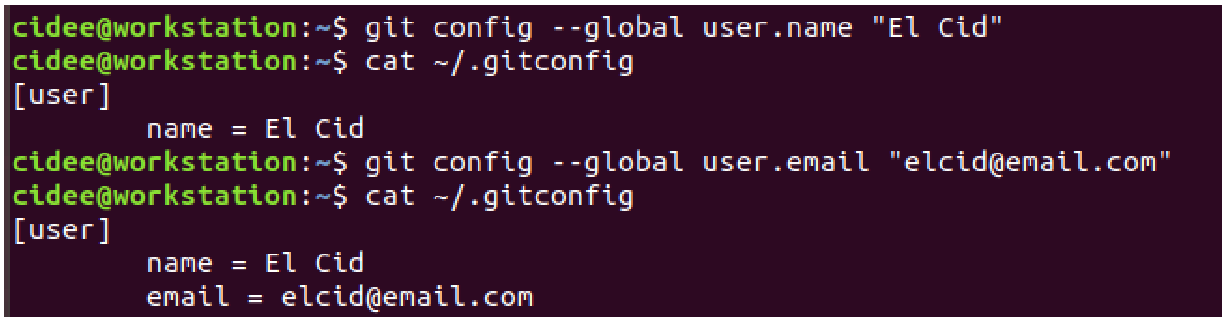
f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.



```
cidee@workstation:~$ ls
CPE212_elcid  Documents  examples.desktop  id_rsa.pub  Pictures  Templates
Desktop       Downloads  id_rsa            Music       Public    Videos
cidee@workstation:~$ cd CPE212_elcid
cidee@workstation:~/CPE212_elcid$ ls
README.md
```

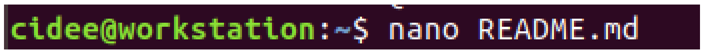
g. Use the following commands to personalize your git.

- *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*



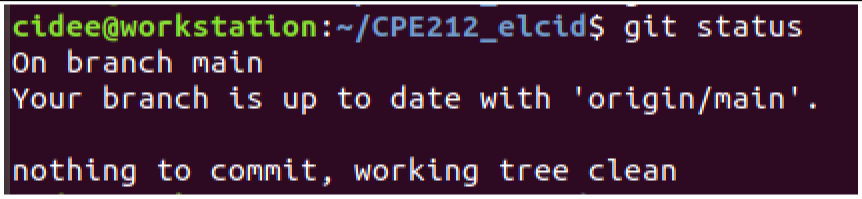
```
cidee@workstation:~$ git config --global user.name "El Cid"
cidee@workstation:~$ cat ~/.gitconfig
[user]
        name = El Cid
cidee@workstation:~$ git config --global user.email "elcid@email.com"
cidee@workstation:~$ cat ~/.gitconfig
[user]
        name = El Cid
        email = elcid@email.com
```

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.

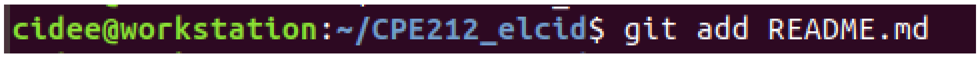


```
cidee@workstation:~$ nano README.md
```

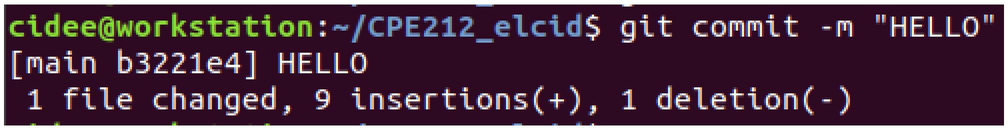
i. Use the *git status* command to display the state of the working directory and the staging area. This command shows which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show any information regarding the committed project history. What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


```
cidee@workstation:~/CPE212_elcid$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

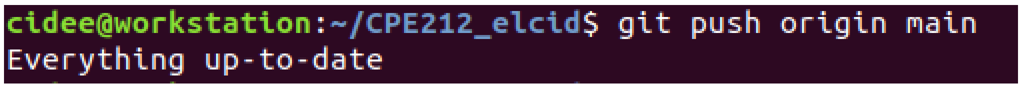
j. Use the command *git add README.md* to add the file into the staging area.


```
cidee@workstation:~/CPE212_elcid$ git add README.md
```

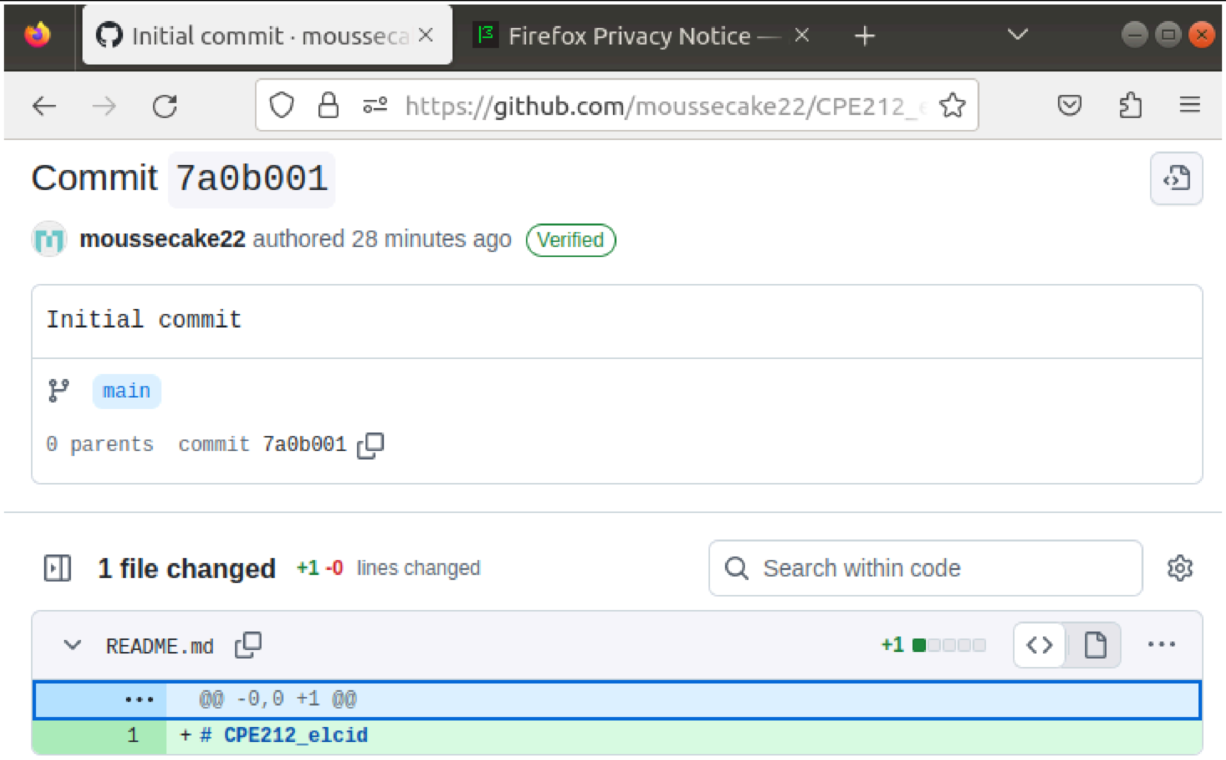
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.


```
cidee@workstation:~/CPE212_elcid$ git commit -m "HELLO"
[main b3221e4] HELLO
 1 file changed, 9 insertions(+), 1 deletion(-)
```

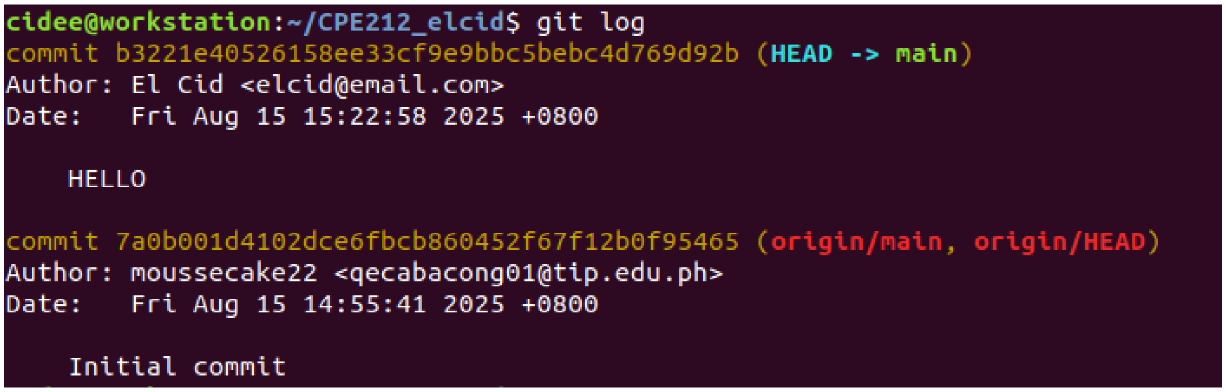
l. Use the command *git push <remote><branch>* to upload the local repository content to GitHub repository. Pushing means to transfer commits from the local repository to the remote repository. As an example, you may issue *git push origin main.*


```
cidee@workstation:~/CPE212_elcid$ git push origin main
Everything up-to-date
```

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.


Initial commit · mousseca
Firefox Privacy Notice —
https://github.com/moussecake22/CPE212_
Commit 7a0b001
moussecake22 authored 28 minutes ago
Verified
main
1 file changed +1 -0 lines changed
Search within code
README.md
+1

```
Initial commit

0 parents  commit 7a0b001

@@ -0,0 +1 @@
1   + # CPE212_elcid
```



```
cidee@workstation:~/CPE212_elcid$ git log
commit b3221e40526158ee33cf9e9bbc5bebc4d769d92b (HEAD -> main)
Author: El Cid <elcid@email.com>
Date:   Fri Aug 15 15:22:58 2025 +0800

    HELLO

commit 7a0b001d4102dce6fbcb860452f67f12b0f95465 (origin/main, origin/HEAD)
Author: moussecake22 <qecabacong01@tip.edu.ph>
Date:   Fri Aug 15 14:55:41 2025 +0800

    Initial commit
```

Reflections:

Answer the following:

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

   I used Ansible to automate remote server setup by configuring passwordless SSH access, testing connections, and preparing environments for future automation tasks.

4. How important is the inventory file?

   The inventory file is super important because it tells your automation tool which computers to work on. It's like a class list for your servers and without it, your tool wouldn't know where to go or what to do. It keeps everything organized and makes sure your tasks run on the right machines.

Conclusions/Learnings:

Doing this activity helped me learn how to connect to remote servers using SSH keys instead of passwords, which felt more secure and convenient. I created my own key pair, tested the connection, set up Git for both local and remote use, and ran ad hoc commands from my computer to the servers. It was such a fun activity although I did struggle a bit in the beginning. Overall, this was a cool way to see how everything works together.