

<b>Name:</b> Clarence B. Chavez	<b>Date Performed:</b> Sept. 4, 2024
<b>Course/Section:</b> CPE31S2	<b>Date Submitted:</b> Sept. 4, 2024
<b>Instructor:</b> Engr. Robin Valenzuela	<b>Semester and SY:</b> 1st Sem (2024 - 2025)

### **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**

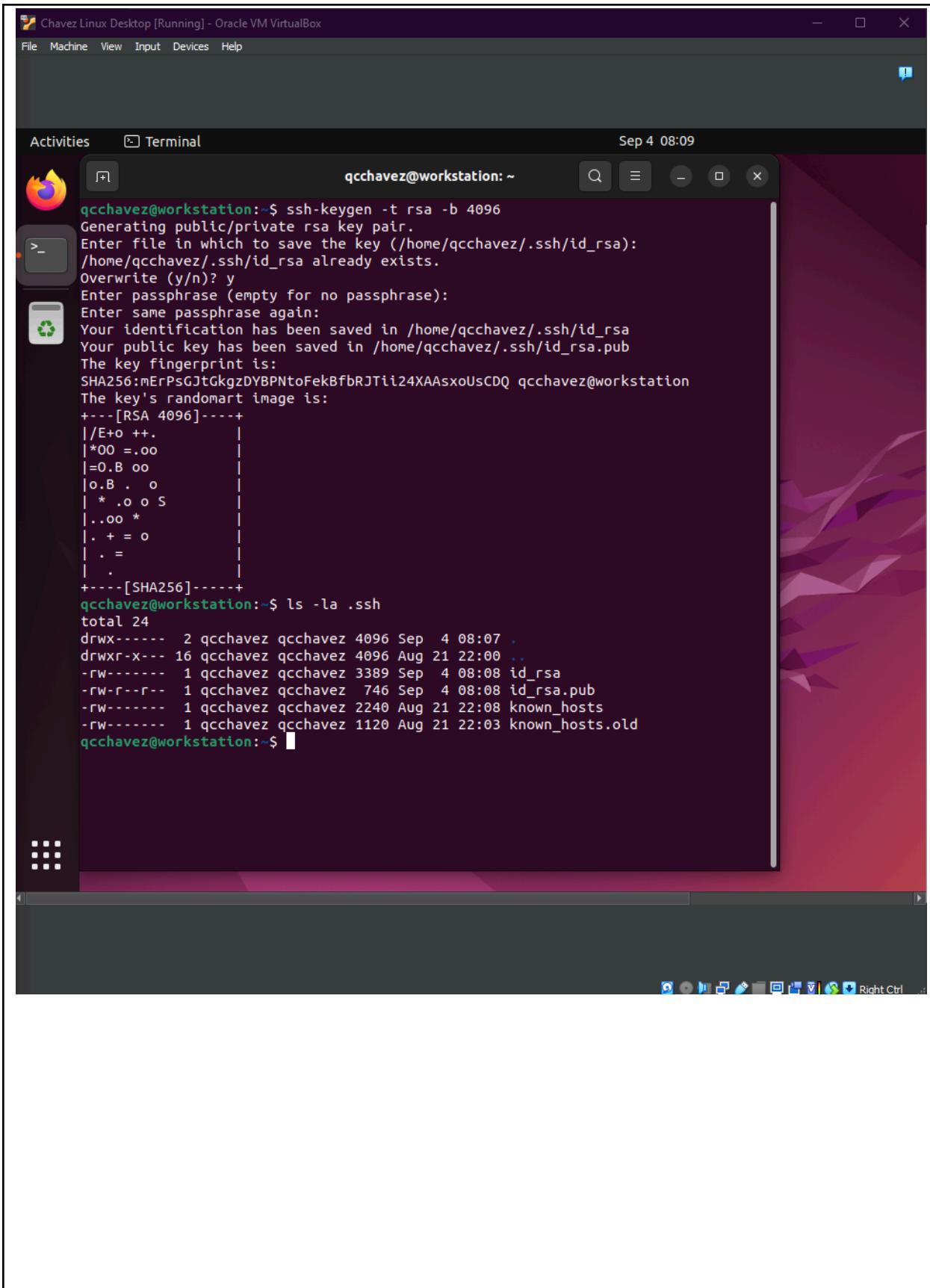
The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

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***

**server1**

```
qcchavez@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qcchavez@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qcchavez/.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 prompted now it is to
install the new keys
qcchavez@server1's password:

Number of key(s) added: 1

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

qcchavez@workstation:~$
```

## server2

```
qcchavez@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qcchavez@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qcchavez/.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 prompted now it is to
install the new keys
qcchavez@server1's password:

Number of key(s) added: 1

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

qcchavez@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qcchavez@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qcchavez/.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: ERROR: ssh: connect to host server2 port 22: No route to host

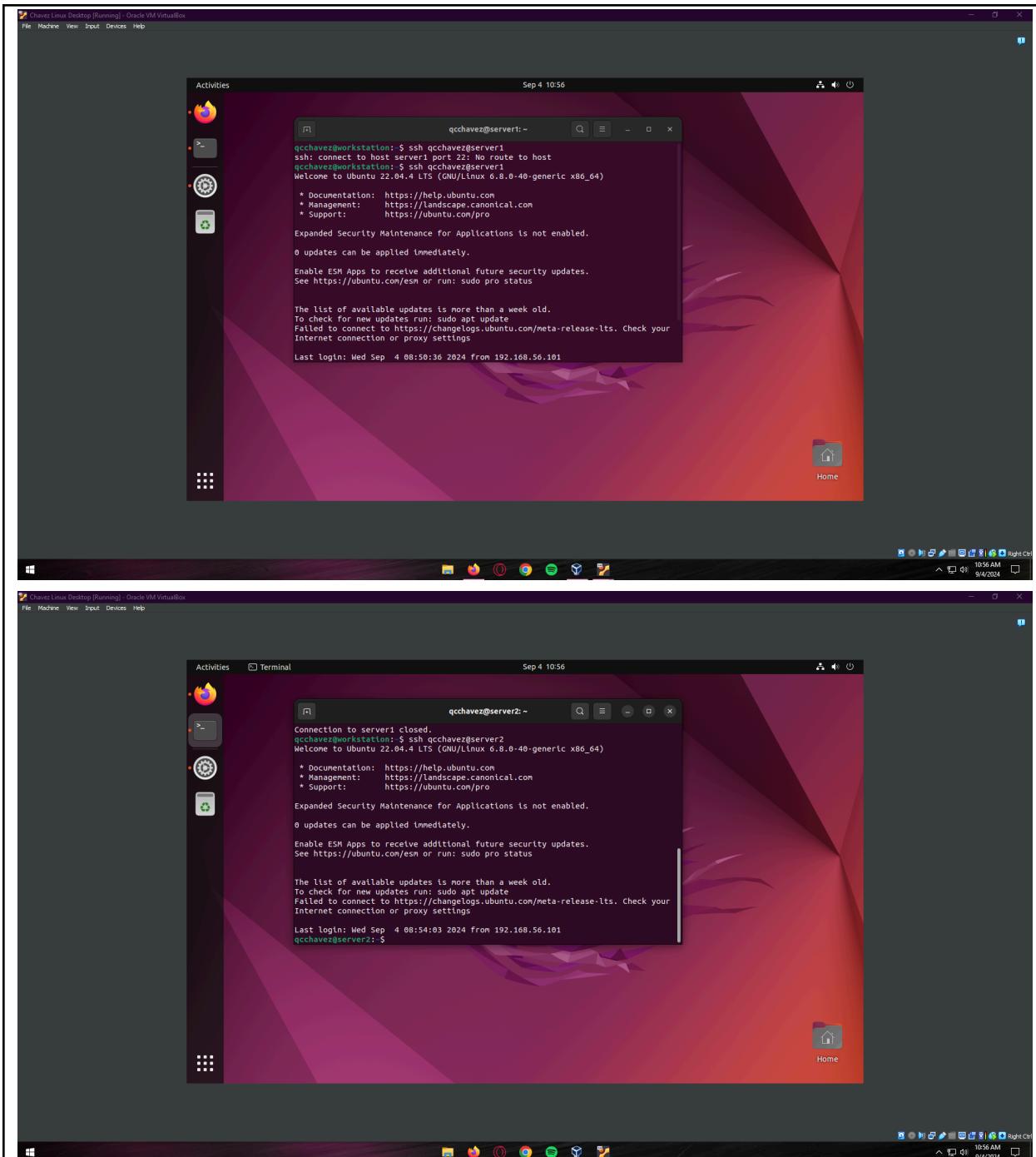
qcchavez@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa qcchavez@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/qcchavez/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: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

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'qcchavez@server2'"
and check to make sure that only the key(s) you wanted were added.

qcchavez@workstation:~$
```

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?



- It is because from the main desktop (the workstation host), I did not enter any passphrase, hence, the connection did not ask for a password.

### Reflections:

Answer the following:

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

- **ssh-program** allows clients to access remote servers securely by providing a secure way to connect to a remote computer, transfer files, and even execute commands on it. It also uses strong encryption to protect the data that will be transmitted between the client and server.
2. How do you know that you already installed the public key to the remote servers?
- **Successful Login Attempt**, the terminal will give you a message that you have successfully logged in to the remote servers.
  - **Checking Authorized Keys File**, by executing the command `.ssh/authorized_keys`
  - **ssh-keygen command**, by executing this command with the combination of the necessary prefixes, you'll be able to find out if it is successfully installed.

## 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**).

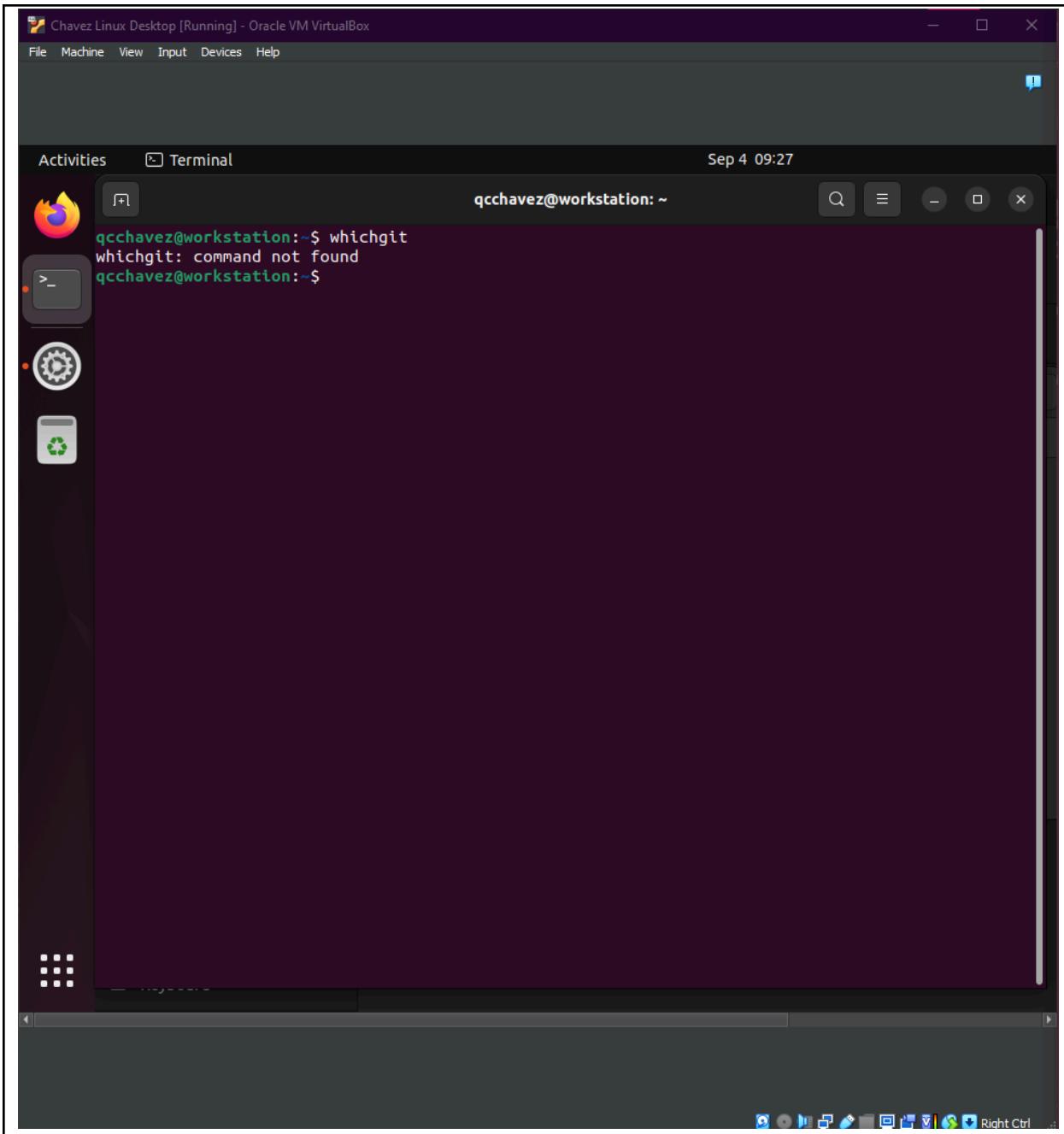
### Set up Git

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

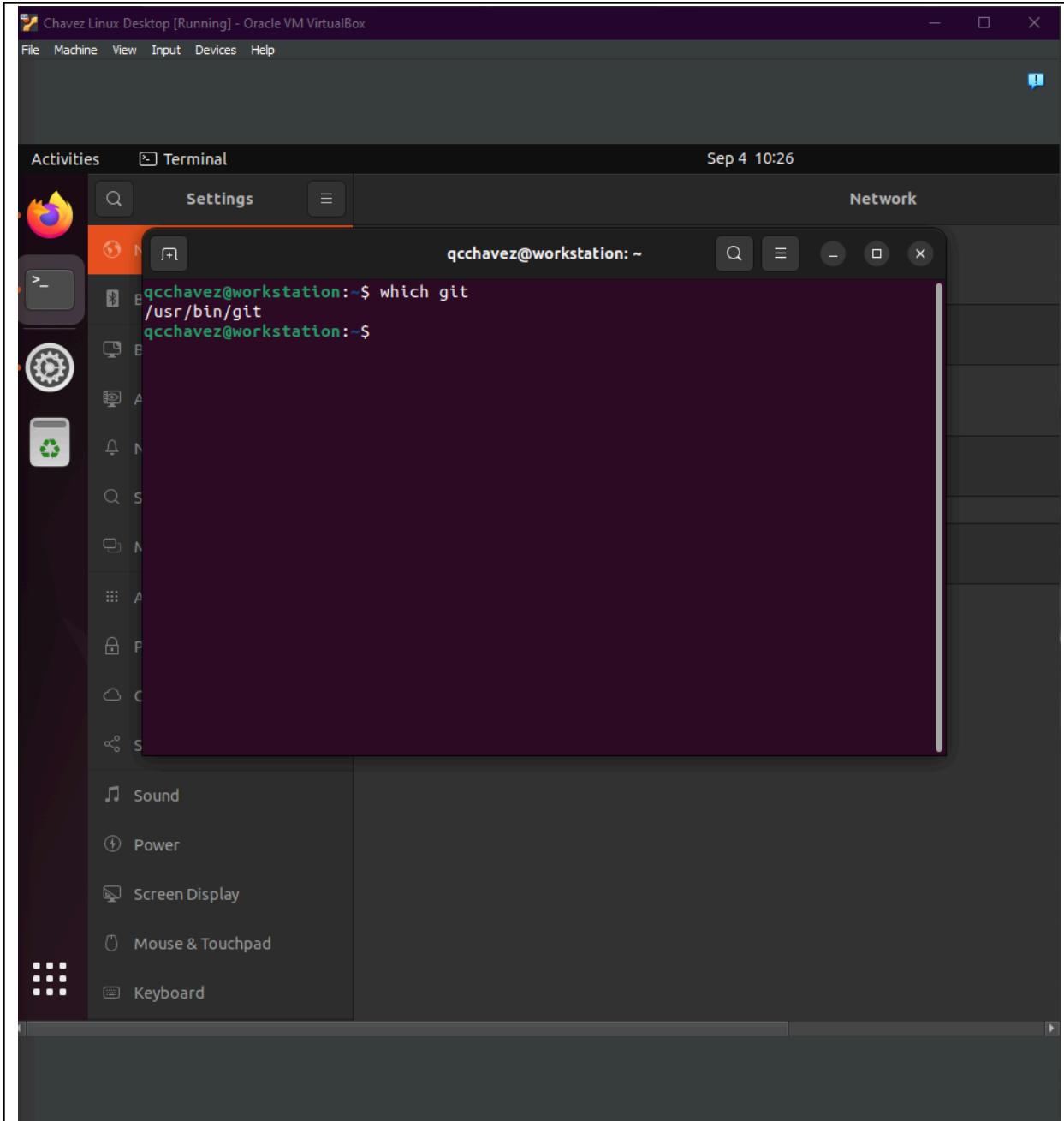
1. On the local machine, verify the version of your git using the command `which git`. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: `sudo apt install git`



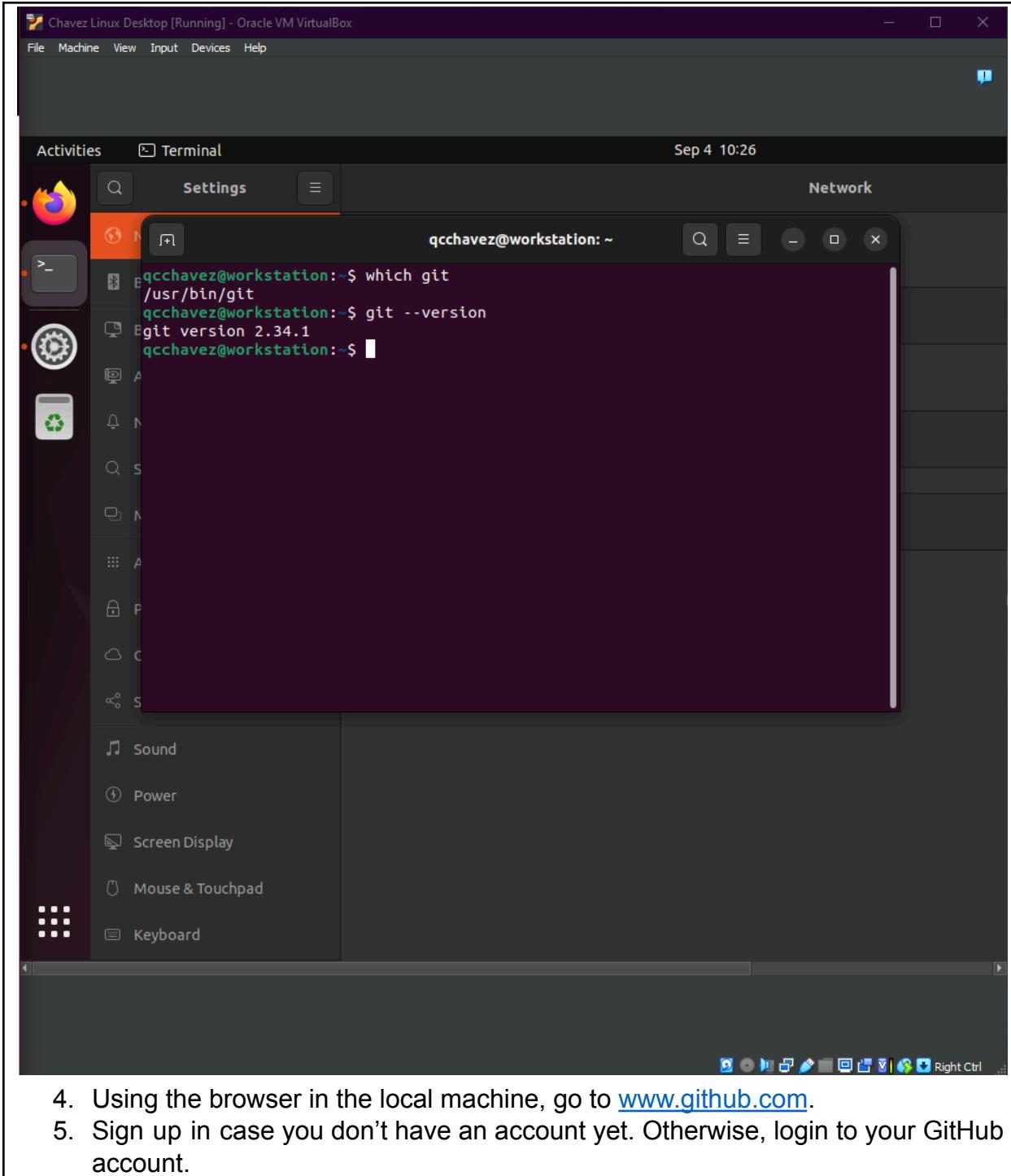
The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open, showing the command `sudo apt install git` being run by the user `qcchavez` at the root prompt. The terminal output indicates that the package is being installed from the `jammy` repository. The desktop interface includes a dock with icons for Sound, Power, Screen Display, Mouse & Touchpad, and Keyboard.

```
qcchavez@workstation:~$ su root
Password:
root@workstation:/home/qcchavez# sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
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 3 not upgraded.
Need to get 4146 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
S:2.34.1-1ubuntu1.11 [955 kB]
```

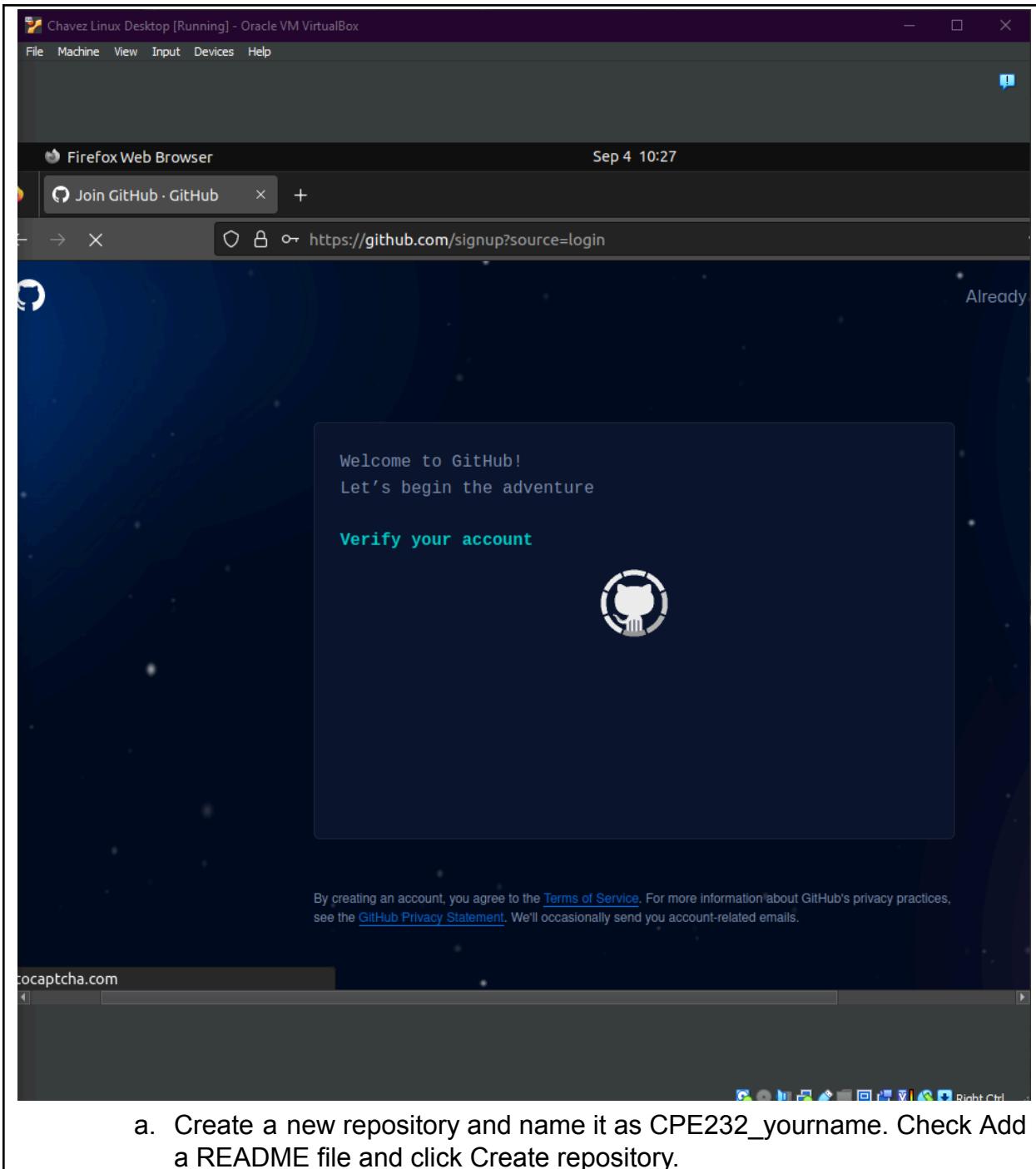
2. After the installation, issue the command **which git** again. The directory of git is usually installed in this location: **user/bin/git**.

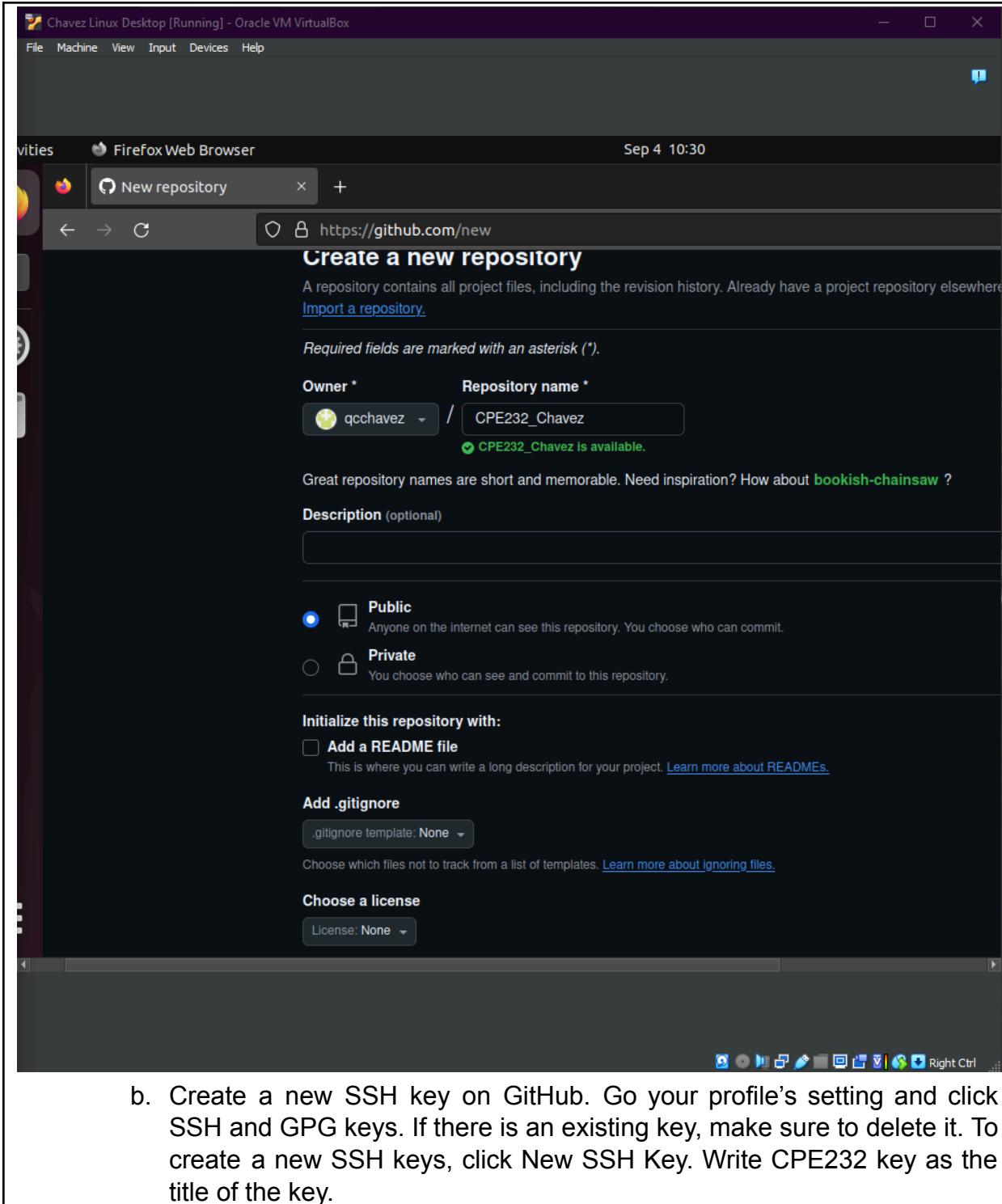


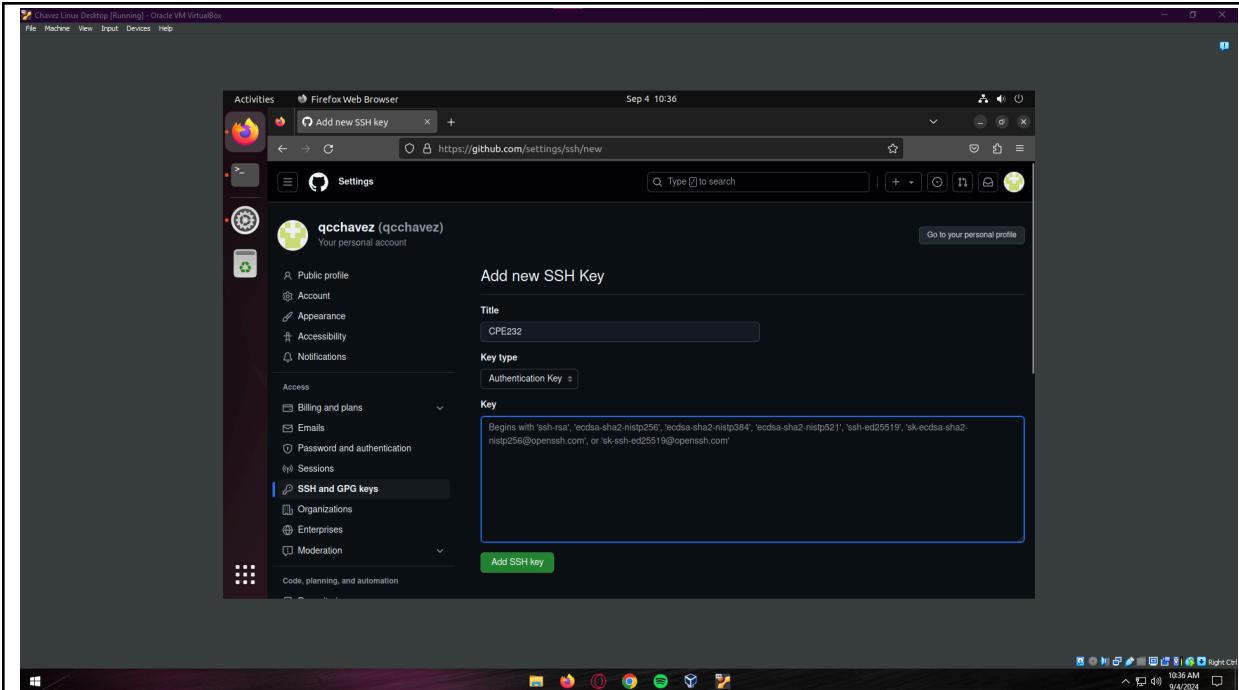
3. The version of git installed in your device is the latest. Try issuing the command **git --version** to know the version installed.



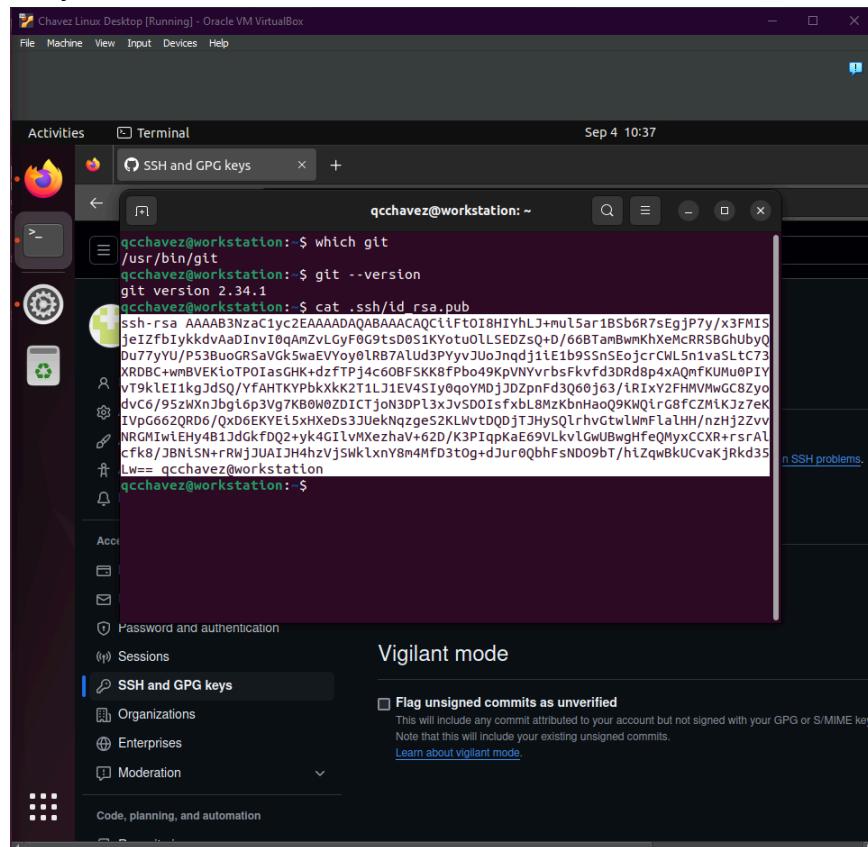
4. Using the browser in the local machine, go to [www.github.com](http://www.github.com).
5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.

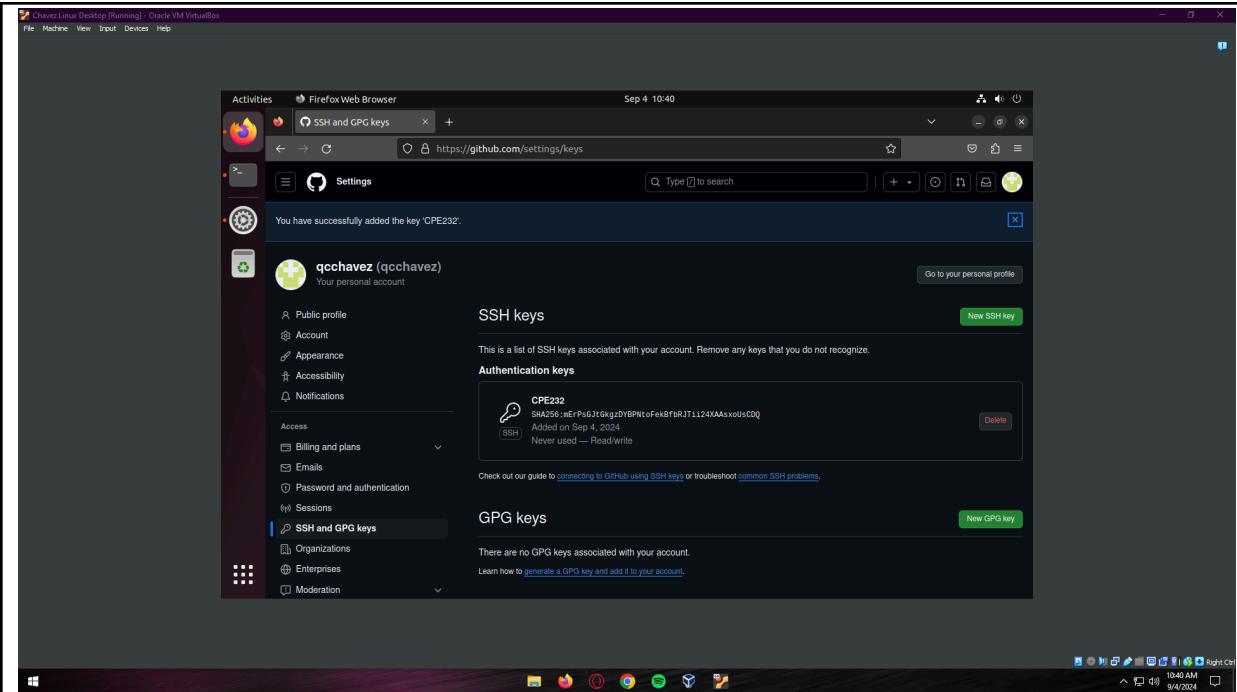




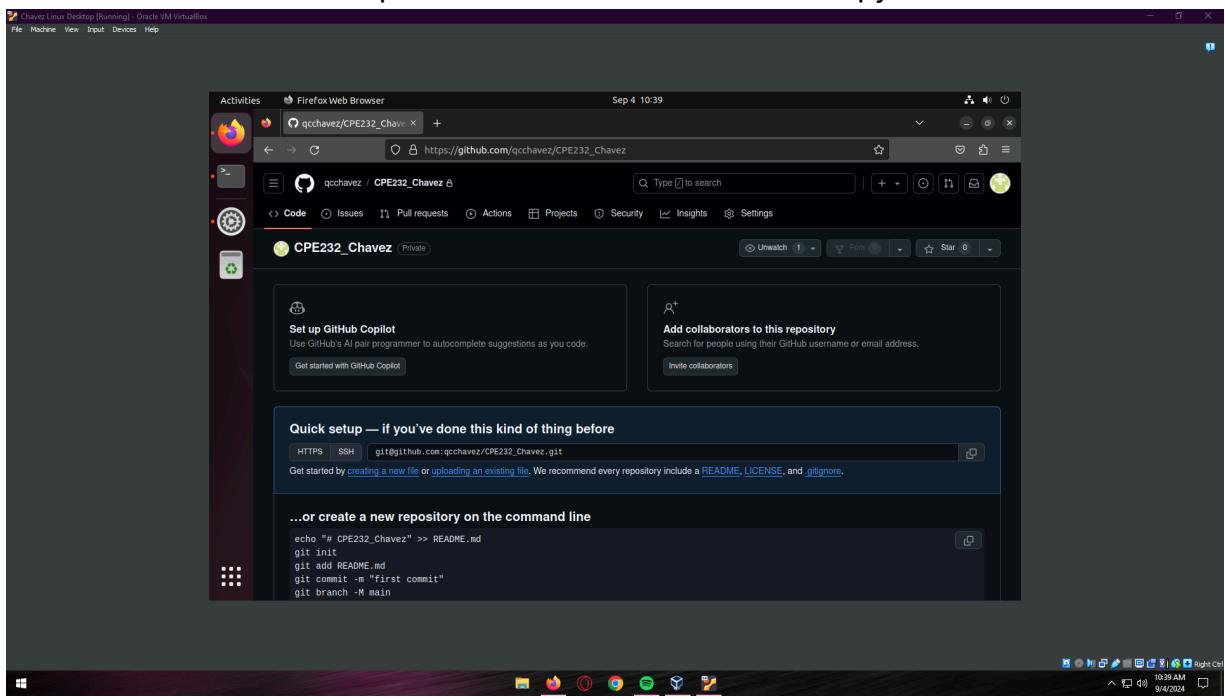


- c. On the local machine's terminal, issue the command `cat .ssh/id_rsa.pub` and copy the public key. Paste it on the GitHub key and press Add SSH key.





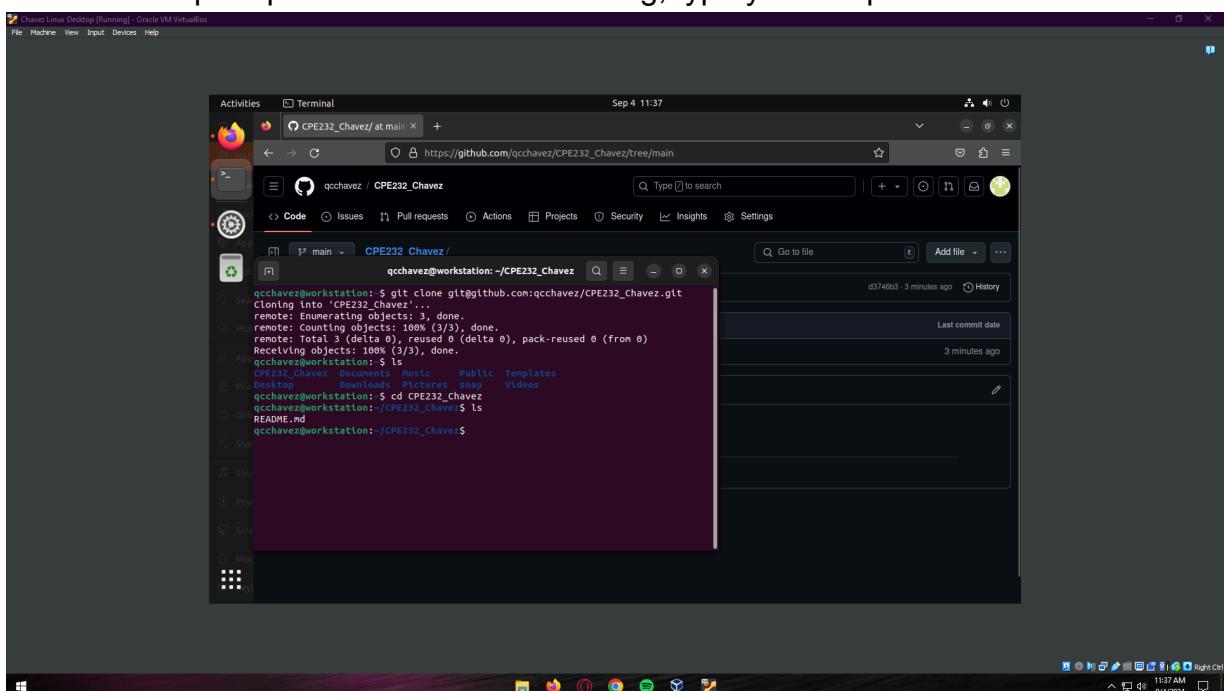
- d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



## Example:

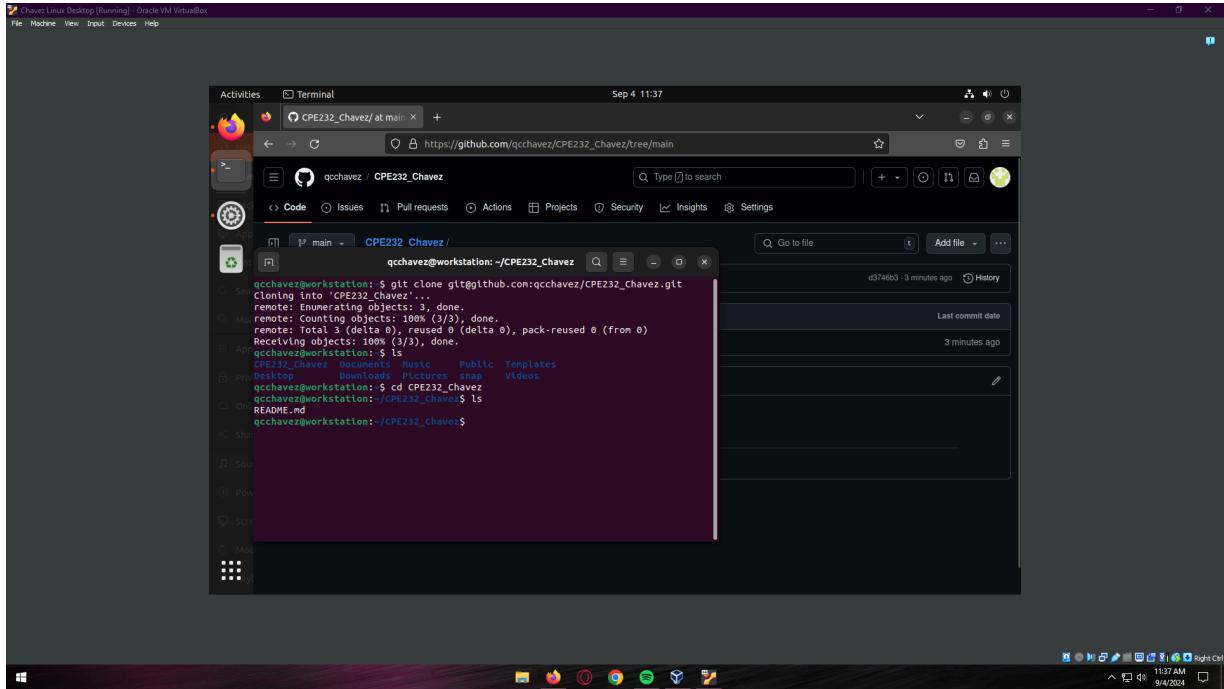
The screenshot shows a GitHub repository page for 'jvtaylor-cpe / CPE302\_yourname'. The 'Code' dropdown menu is open, and the 'Clone' option is highlighted with a yellow circle. Below it, the 'HTTPS SSH GitHub CLI' link and the copied URL 'git@github.com:jvtaylor-cpe/CPE302\_yourname' are visible. Other options like 'Download ZIP' are also shown.

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



- f. To verify that you have cloned the GitHub repository, issue the command `ls`. Observe that you have the CPE232\_yourname in the list of

your directories. Use CD command to go to that directory and LS command to see the file README.md.



The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open, displaying the following command and its output:

```
git clone git@github.com:qcchavez/CPE232_Chavez.git
Cloning into 'CPE232_Chavez'...
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
qcchavez@workstation:~/CPE232_Chavez$ ls
CPE232_Chavez  Documents  Music  Public  Templates
Desktop  Downloads  Pictures  Videos
qcchavez@workstation:~/CPE232_Chavez$ cd CPE232_Chavez
qcchavez@workstation:~/CPE232_Chavez$ ls
README.md
qcchavez@workstation:~/CPE232_Chavez$
```

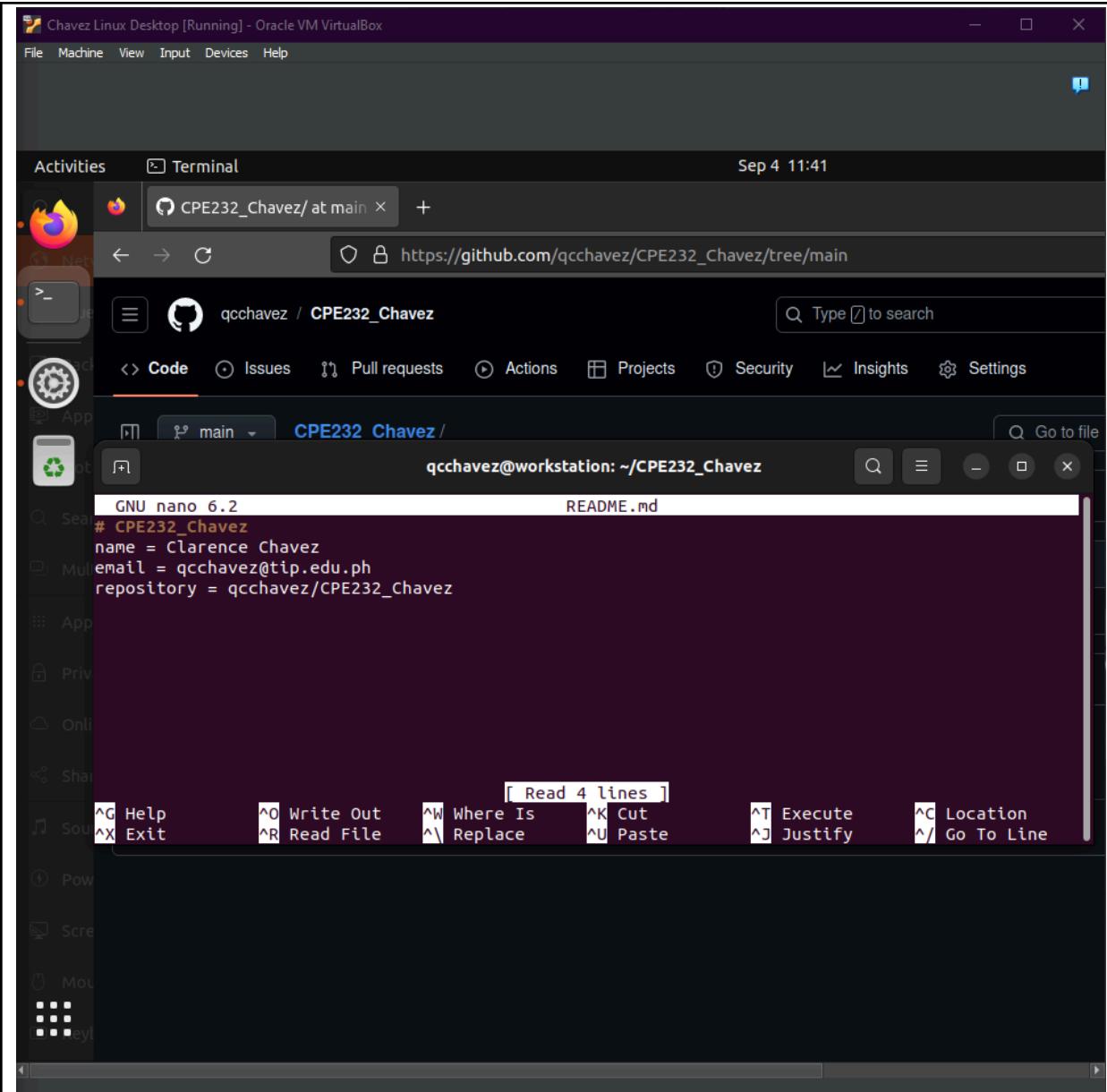
g. Use the following commands to personalize your git.

- *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*

The screenshot shows a Linux desktop environment with a dark theme. At the top, there's a menu bar with options like File, Machine, View, Input, Devices, and Help. Below the menu is a dock with icons for various applications, including a browser and a terminal. The terminal window is open and displays the following command-line session:

```
qcchavez@workstation:~/CPE232_Chavez$ git config --global user.name "Clarence Chavez"
qcchavez@workstation:~/CPE232_Chavez$ git config --global user.email qcchavez@tip.edu.ph
qcchavez@workstation:~/CPE232_Chavez$ cat ~/.gitconfig
[user]
    name = Clarence Chavez
    email = qcchavez@tip.edu.ph
qcchavez@workstation:~/CPE232_Chavez$
```

- 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?
  - By issuing this command, it says that the changes were not yet made to the commit. It also specifies the file that was modified which is the **README.md**

The screenshot shows a Linux desktop environment with a dark theme. At the top, there is a menu bar with options: File, Machine, View, Input, Devices, Help. Below the menu bar is a dock with several icons, including a browser icon and a terminal icon. The terminal window is open and displays the following command-line session:

```
qcchavez@workstation:~/CPE232_Chavez$ ls
README.md
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ 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")
qcchavez@workstation:~/CPE232_Chavez$
```

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

The screenshot shows a Linux desktop environment with a dark theme. At the top is a menu bar with options: File, Machine, View, Input, Devices, Help. Below the menu is a header bar with tabs: Activities, Terminal, and a date/time indicator: Sep 4 11:41. The main area is a terminal window titled 'CPE232\_Chavez/ at main'. The terminal shows the following command-line session:

```
email = qcchavez@tip.edu.ph
qcchavez@workstation:~/CPE232_Chavez$ ls
README.md
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ 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")
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ git add README.md
qcchavez@workstation:~/CPE232_Chavez$
```

The terminal window has a dark background with light-colored text. The file 'README.md' is being edited multiple times. The bottom of the terminal window shows a set of small icons for various applications.

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.

The screenshot shows a Linux desktop environment running in Oracle VM VirtualBox. The desktop interface includes a top bar with 'Chavez Linux Desktop [Running] - Oracle VM VirtualBox' and a menu bar with 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. A notification icon in the top right corner has a blue exclamation mark.

The desktop has a dark theme with a dock on the left containing icons for a terminal, file manager, browser, and system tools. An 'Activities' overview window is open, showing a list of applications including a terminal window titled 'CPE232\_Chavez/ at main'.

The terminal window displays the following command-line session:

```
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ 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")
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ git add README.md
qcchavez@workstation:~/CPE232_Chavez$ git commit -m "I modified README.md"
[main 3e4fb33] I modified README.md
 1 file changed, 3 insertions(+)
qcchavez@workstation:~/CPE232_Chavez$
```

- I. 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***.

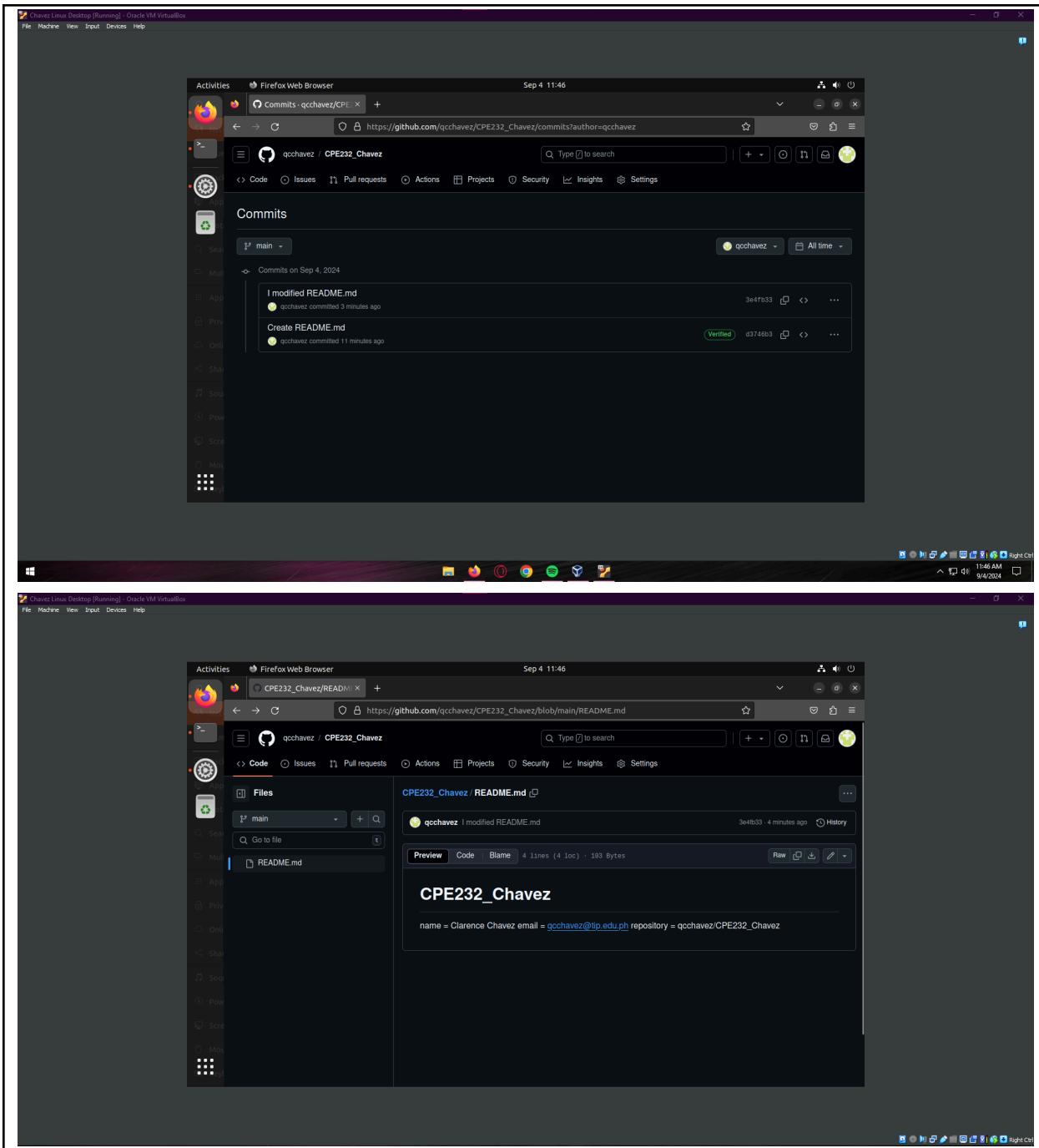
The screenshot shows a Linux desktop environment running in Oracle VM VirtualBox. The desktop interface includes a top bar with 'Chavez Linux Desktop [Running] - Oracle VM VirtualBox' and a menu bar with 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. A system tray icon with a blue exclamation mark is visible.

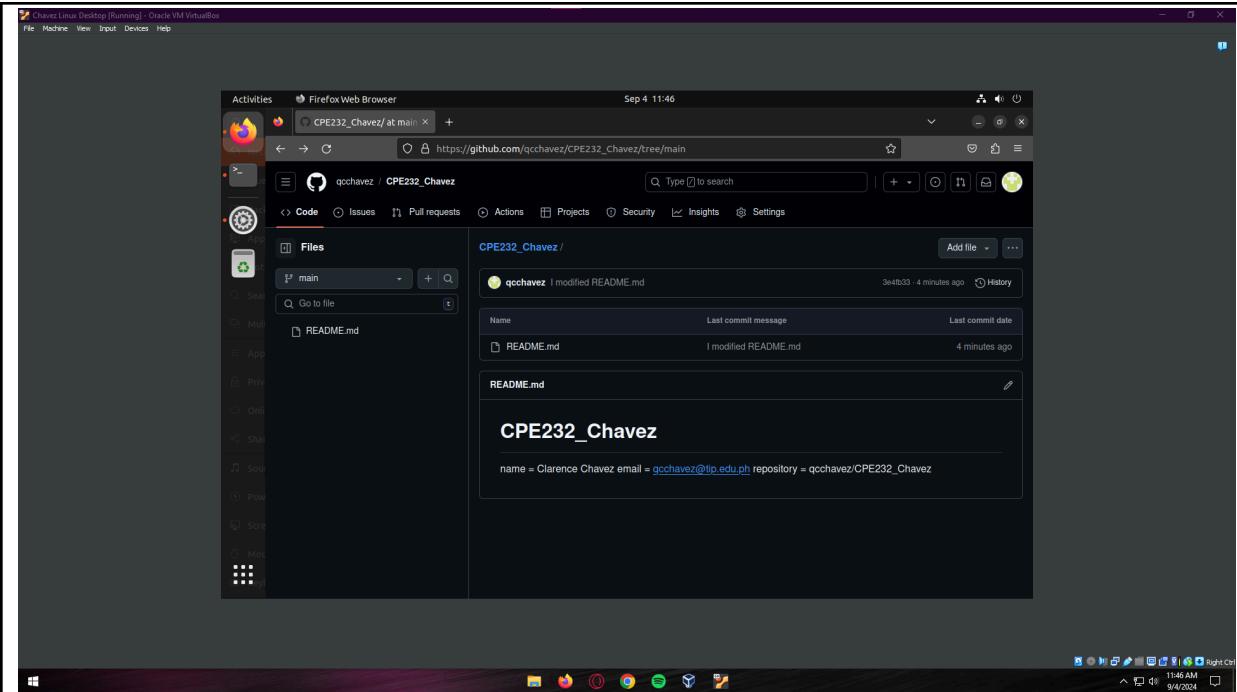
The main window contains an 'Activities' overview and a 'Terminal' application. The terminal window title is 'CPE232\_Chavez/ at main' and the URL is 'https://github.com/qcchavez/CPE232\_Chavez/tree/main'. The terminal content shows a user named 'qcchavez' performing the following actions:

```
modified: README.md
no changes added to commit (use "git add" and/or "git commit -a")
qcchavez@workstation:~/CPE232_Chavez$ nano README.md
qcchavez@workstation:~/CPE232_Chavez$ git add README.md
qcchavez@workstation:~/CPE232_Chavez$ git commit -m "I modified README.md"
[main 3e4fb33] I modified README.md
 1 file changed, 3 insertions(+)
qcchavez@workstation:~/CPE232_Chavez$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qcchavez/CPE232_Chavez.git
 d3746b3..3e4fb33 main -> main
qcchavez@workstation:~/CPE232_Chavez$
```

Below the terminal, there is a sidebar with various icons for system monitoring and configuration, including Power, Screen, Mouse, and Keyboard.

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?
  - In this activity, using ansible commands, we have installed, configured, and used the git repository to transfer files from the github website to the linux and vice-versa. I also used the SSH by installing it, copying the key from the linux to github. I also ensured that the connection between the two is secured.
4. How important is the inventory file?
  - Inventory file is important because it is where you store your files whenever you make changes or modifications.

### Conclusions/Learnings:

- In this activity, I have learned that GitHub is a very essential tool for tracking and making changes in a project. It allows you to transfer directly from Ubuntu Desktop to the GitHub repository, and vice-versa. Using also SSH for secure connection between clients and servers is also useful because it requires several authentication like username and passwords, depending on the administrator's preferences.