

Name: Solis, Paul Vincent M.	Date Performed: 8/15/25
Course/Section: CPE 212 - CPE31S2	Date Submitted: 8/15/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2025-2026
Activity 2: SSH Key-Based Authentication and Setting up Git	
1. Objectives: <ul style="list-style-type: none"> 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.

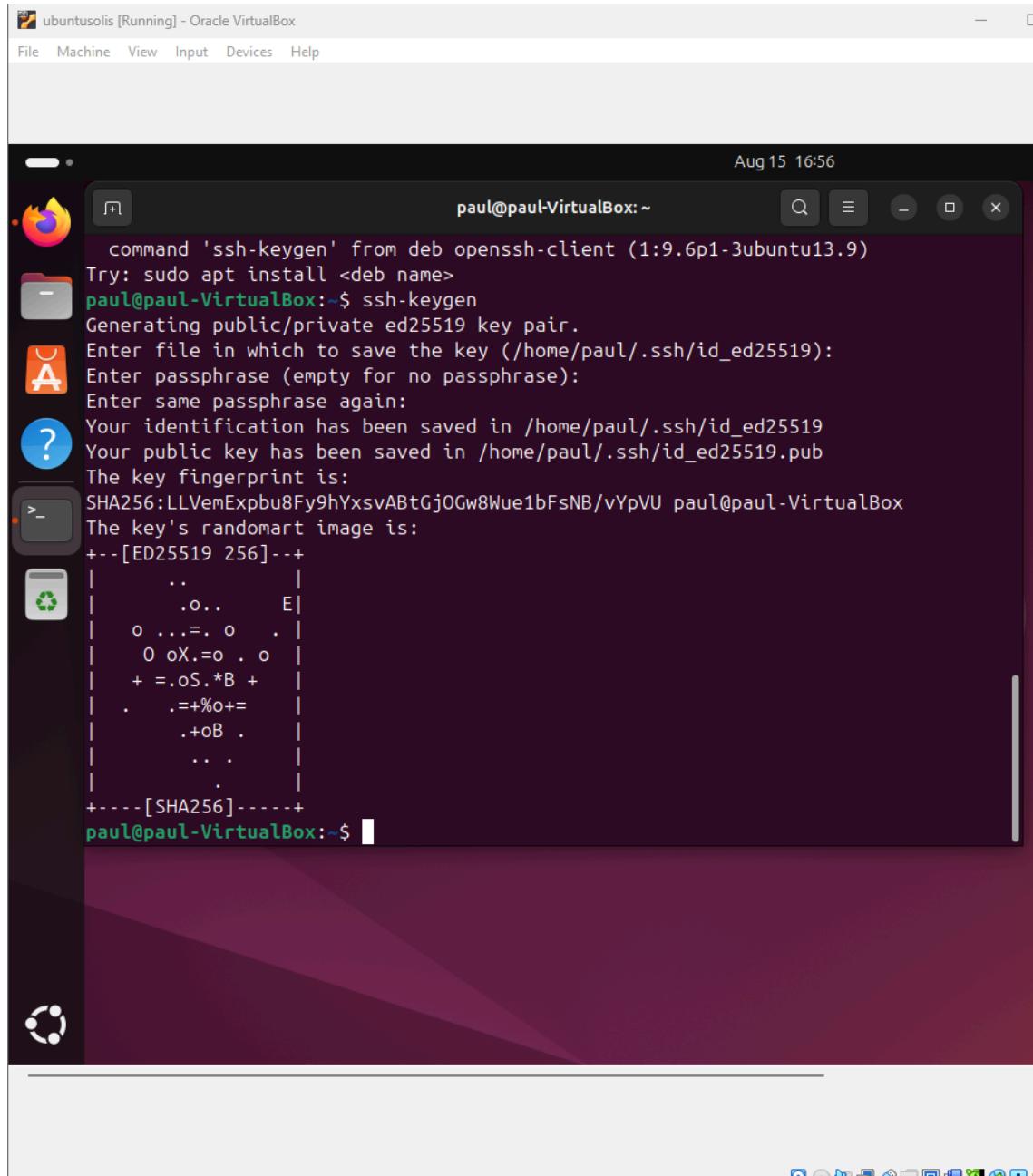
SSH introduced public key authentication as a more secure alternative to the older .rhosts authentication. It improved security by avoiding the need to have password stored in files and eliminated the possibility of a compromised server stealing the user's password.

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.

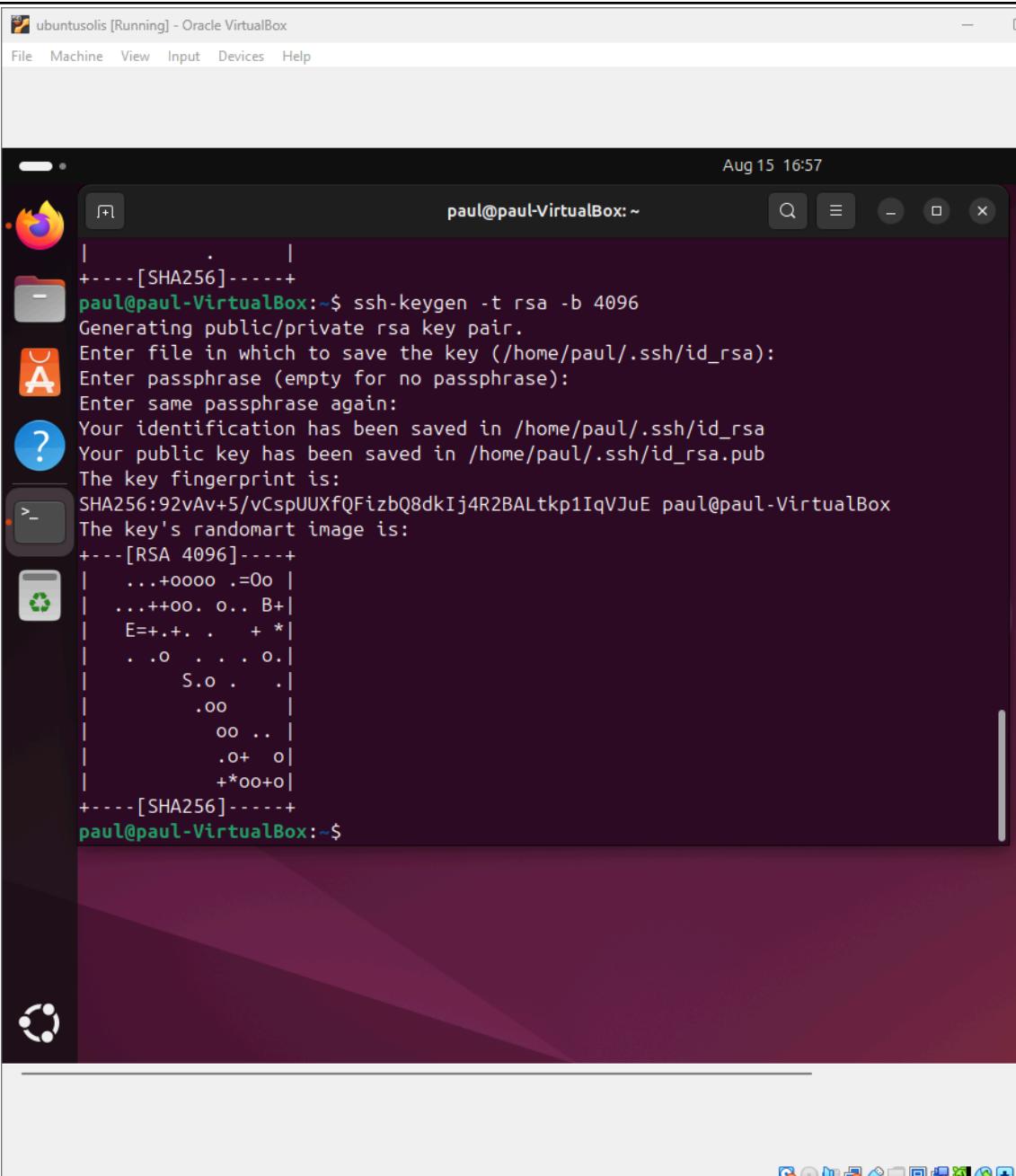
Task 1: Create an SSH Key Pair for User Authentication

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*.

A screenshot of a Linux desktop environment, likely Ubuntu, running in a virtual machine. The desktop has a dark theme with a purple and black gradient background. A terminal window is open in the foreground, showing the command 'ssh-keygen' being run. The terminal output shows the creation of an Ed25519 key pair, with the public key saved to /home/paul/.ssh/id_ed25519 and the private key to /home/paul/.ssh/id_ed25519.pub. The key fingerprint is SHA256:LLVemExpbu8Fy9hYxsvABtGj0Gw8Wue1bFsNB/vYpVU. The terminal window title is 'paul@paul-VirtualBox: ~'. The desktop interface includes a dock with icons for Home, Dash, Applications, and others, and a system tray at the bottom.

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.



The screenshot shows a terminal window titled "ubuntusolis [Running] - Oracle VirtualBox". The terminal session is as follows:

```
Aug 15 16:57
paul@paul-VirtualBox: ~
| . . .
+---[SHA256]---+
paul@paul-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/paul/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/paul/.ssh/id_rsa
Your public key has been saved in /home/paul/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:92vAv+5/vCspUUXfQFizbQ8dkIj4R2BALtkp1IqVJuE paul@paul-VirtualBox
The key's randomart image is:
+---[RSA 4096]---+
| ...+oooo .=Oo |
| ...++oo. o... B+|
| E=+.+. . + *|
| . .o . . . o.|
| S.o . . .|
| .OO . . .|
| .OO .. .|
| .O+ . O.|
| +*OO+O|
```

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`.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "paul@paul-VirtualBox: ~". The date and time at the top right of the terminal window are "Aug 15 16:58". The terminal window contains the following text:

```
The key's randomart image is:  
+---[RSA 4096]---+  
| ...+oooo .=Oo |  
| ...++oo. o... B+|  
| E=+.+. . + *|  
| ..o . . . o.|  
| S.o . . |  
| .oo |  
| oo .. |  
| .o+ o|  
| +*oo+o|  
+---[SHA256]---+  
paul@paul-VirtualBox:~$ ls -la .ssh  
total 32  
drwx----- 2 paul paul 4096 Aug 15 16:57 .  
drwxr-x--- 16 paul paul 4096 Aug 8 17:49 ..  
-rw----- 1 paul paul 0 Aug 8 17:39 authorized_keys  
-rw----- 1 paul paul 411 Aug 15 16:56 id_ed25519  
-rw-r--r-- 1 paul paul 102 Aug 15 16:56 id_ed25519.pub  
-rw----- 1 paul paul 3389 Aug 15 16:57 id_rsa  
-rw-r--r-- 1 paul paul 746 Aug 15 16:57 id_rsa.pub  
-rw----- 1 paul paul 978 Aug 8 18:25 known_hosts  
-rw-r--r-- 1 paul paul 142 Aug 8 18:24 known_hosts.old  
paul@paul-VirtualBox:~$
```

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*

ubuntusolis [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Aug 15 17:09

paul@paul-VirtualBox:~

```
-s: use sftp    -- use sftp instead of executing remote-commands. Can be useful if the remote only allows sftp
-x: debug      -- enables -x in this shell, for debugging
-h|-?: print this help
paul@paul-VirtualBox:~$ ssh-copy-id -i ~/.ssh/id_rsa paul@192.168.56.110
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/paul/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.110 (192.168.56.110)' can't be established.
ED25519 key fingerprint is SHA256:NTVZlQIVkq0D10ukr1qA+eCQs/T1kaAXuYOIsClQZ8U.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [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 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
paul@192.168.56.110's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'paul@192.168.56.110'"
and check to make sure that only the key(s) you wanted were added.

paul@paul-VirtualBox:~$
```

Join GitHub Education

Home

Ask Copilot

Suggest next steps for an issue Summarize a

if

The screenshot shows a Linux desktop environment with a terminal window open in a window titled "ubuntusolis [Running] - Oracle VirtualBox". The terminal window has a dark background and displays the following command-line session:

```
paul@paul-VirtualBox:~$ /usr/bin/ssh-copy-id: ERROR: ssh: Could not resolve hostname server1: Temporary failure in name resolution
paul@paul-VirtualBox:~$ rm -./.ssh/id_rsa paul@192.168.56.110
rm: invalid option -- '/'
Try 'rm --help' for more information.
paul@paul-VirtualBox:~$ rmdir -./.ssh/id_rsa paul@192.168.56.110
rmdir: invalid option -- '/'
Try 'rmdir --help' for more information.
paul@paul-VirtualBox:~$ ssh-copy-id -i ~/ssh/id_rsa paul@192.168.56.111
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/paul/.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
paul@192.168.56.111's password:
Number of key(s) added: 1

Now try logging into the machine, with:    "ssh 'paul@192.168.56.111'"
and check to make sure that only the key(s) you wanted were added.
```

Below the terminal window, there is a GitHub Copilot sidebar with the following options:

- Simple cloud
- Copilot: Turn natural suggestions
- Heroku: Build, run, and deploy in the cloud
- Microso: Access to Microsoft learning

The terminal window also includes a "Join GitHub Education" button and a "Home" section with a "Ask Copilot" button.

On the desktop, there are several icons visible in the dock, including a terminal, file manager, browser, and system tray icons.

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?

- If SSH didn't ask for a password when connecting to Server 1 and Server 2, it means your public key is already stored in their authorized_keys files. This allows secure, passwordless login using key-based authentication.

Reflections:

Answer the following:

1. How will you describe the ssh-program? What does it do?
SSH is a secure protocol that lets you remotely access and control another computer over a network.
2. How do you know that you already installed the public key to the remote servers?

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`

```
paul@paul-VirtualBox:~$ sudo apt install git -y
[sudo] password for paul:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libgl1-amber-dri libglapi-mesa libllvm17t64 python3-netifaces
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 2 not upgraded.
Need to get 4,806 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
Get:1 http://ph.archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl all 0.1
7029-2 [25.6 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 git-man all 1
:2.43.0-1ubuntu7.3 [1,100 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2
.43.0-1ubuntu7.3 [3,680 kB]
```

Join GitHub Education

Home

Ask Copilot

Suggest next steps for an issue Summarize a

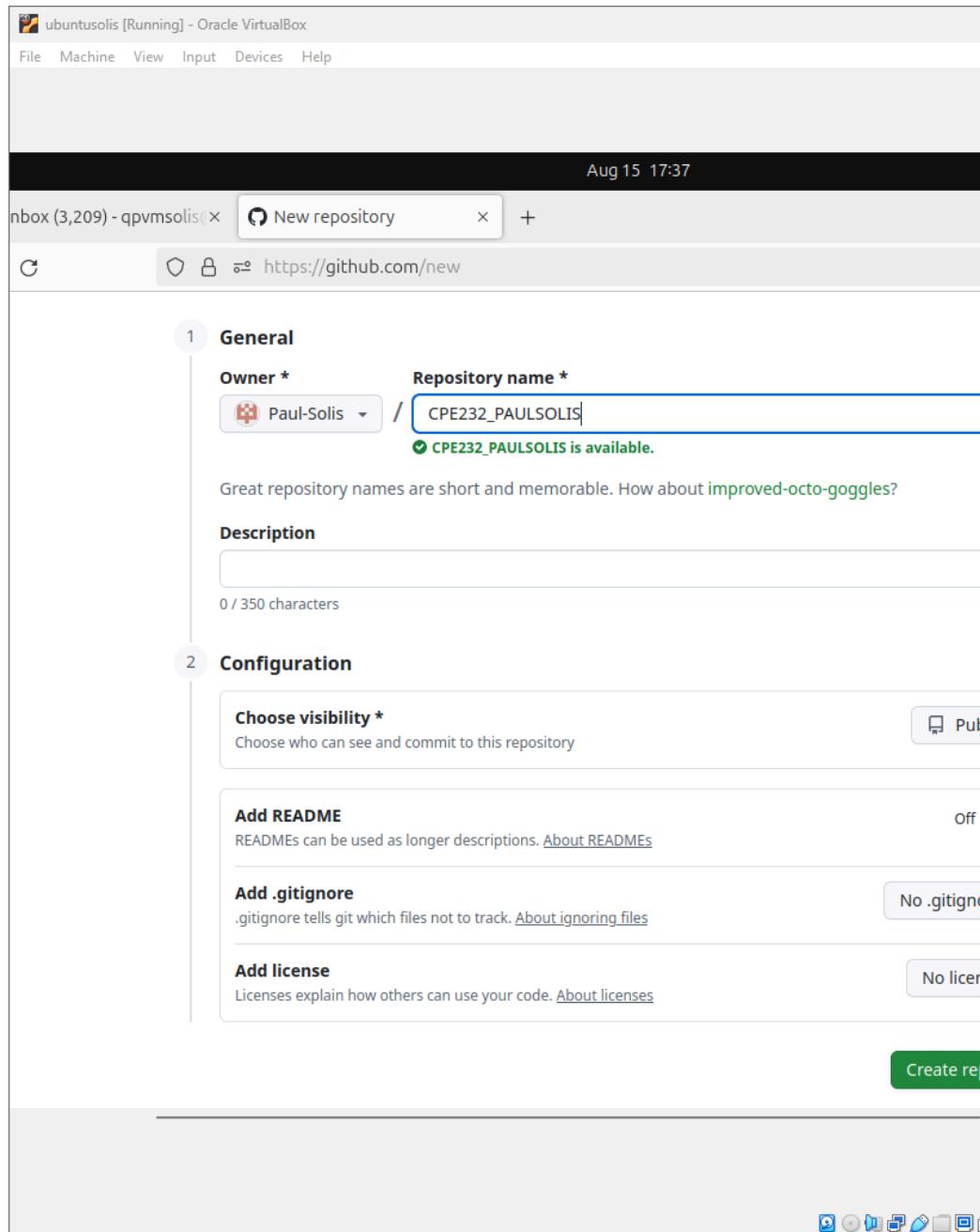
2. After the installation, issue the command **which git** again. The directory of git is usually installed in this location: **user/bin/git**.

```
paul@paul-VirtualBox:~$ 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.

```
paul@paul-VirtualBox:~$ git --version
git version 2.43.0
```

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 to 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 key, click New SSH Key. Write CPE232 key as the

title of the key.

The screenshot shows a Linux desktop environment with a window titled "SSH and GPG keys". The URL in the address bar is "https://github.com/settings/keys". The main content area displays the "Authentication keys" section, which lists a single key named "CPE232 KEY". This key is identified as an SSH key, was added on Aug 15, 2025, and has never been used. A "Delete" button is visible next to the key entry. Below this, there is a note about connecting to GitHub using SSH keys and troubleshooting common SSH problems. The "GPG keys" section is shown below, indicating there are no GPG keys associated with the account. A "New SSH key" button is located at the top right of the "SSH keys" section, and a "New GPG key" button is located at the top right of the "GPG keys" section. The desktop interface includes a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help".

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.

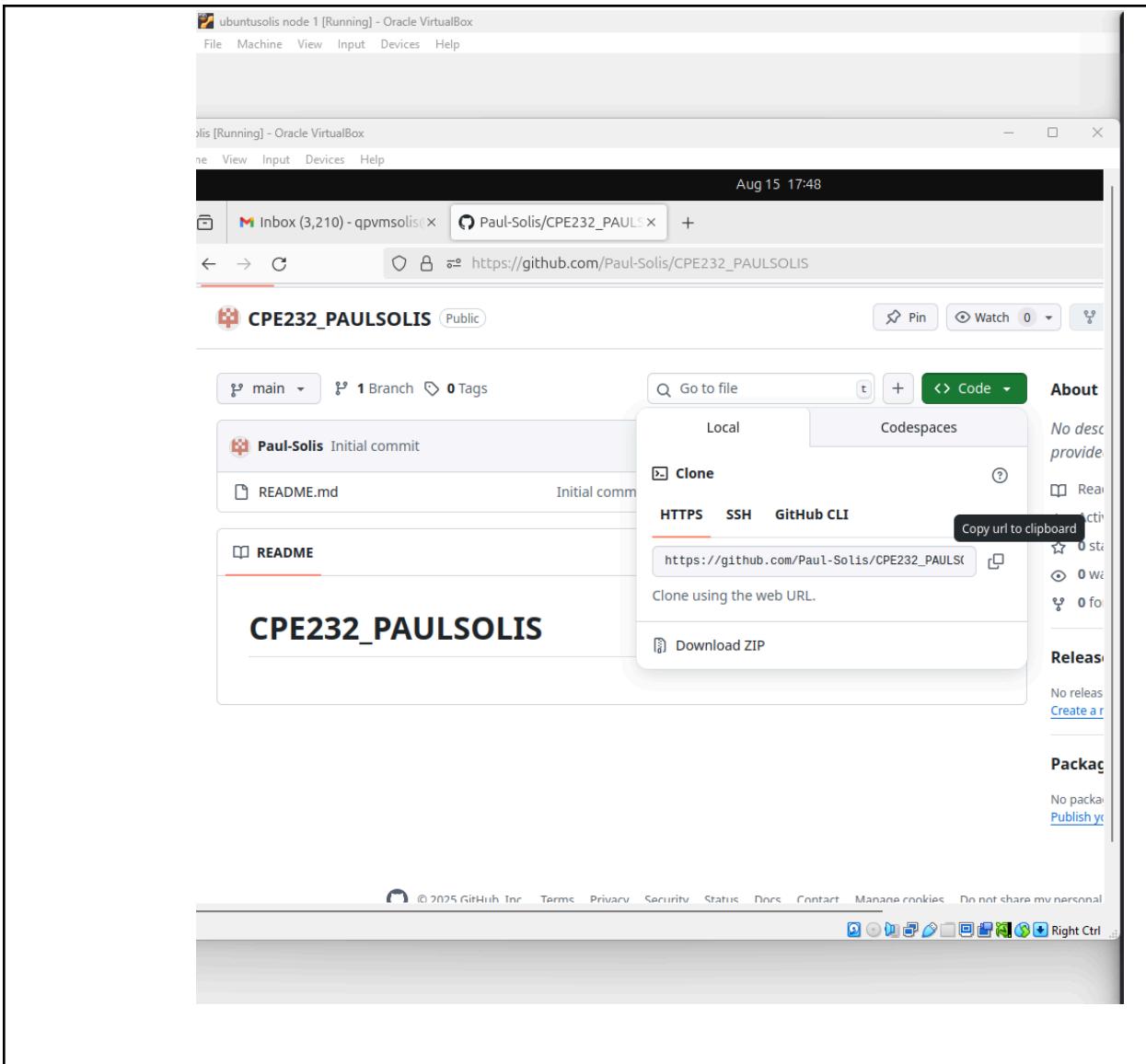
```
paul@paul-VirtualBox:~$ 
d now it is to install the new keys
paul@192.168.56.111's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'paul@192.168.56.111'"
and check to make sure that only the key(s) you wanted were added.

paul@paul-VirtualBox:~$ which git
/usr/bin/git
paul@paul-VirtualBox:~$ git --version
git version 2.43.0
paul@paul-VirtualBox:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQC6Y/bjbjYcfkIZjp/+nw5BXjQ6G18eTg
9xrJKsyRSTIZJyEo9h4mTJE7hRXyrKufsA0seB+UrCACroUJn0Sov0ebrsW0JH1DuSH020
lCxLVRQKVdvBQNPj5zF93NOCPrL6HYcu45lSG6PWPVBslFFMt6xbaYn43dZ1mRczqi49Ll
7nnQ0tJzH96znVtPE5yyTKmQFpyd+Jp0Zmy8bGLcpw/mAXLvKQeVc7nFGNc5XY8eEM23Bc
LHnm725CRjNwoZmMlj2GGDjlJmiZ8rSh4mbPcWsJws8zlb4SJmHiNDf17NF9miEBQXTi0F
QZMAe8fJHhKAs56BlhqBPNL85TTYw28tnw/hqAQHb4diPqA1oBaTQ4NmIyjs95yJgRX3ei
OaDuV0E8W7mSKEZ+bwNV1R++30yVSxbPhLQq6SLWr9bSlnm3uXIIdHrExcMTyqjxqB/Zhqd
ZwtzqPcoV0BCFcuXly4VoSsL/sx/oq2iJB+GwlJIjeV5Tv5F2MAUxDbYT3dFBlewpHtXCL
0KLcU594+cfvKWsDlHRKsVhUXNbeGB01FZPtyzNCeLXAIk1hKmRW4XFoUQaC1ylBTbDUr
@paul-VirtualBox:~$
```

- 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:jvtaylor-cpe/CPE232_yourname.git`. When prompted to continue connecting, type yes and press enter.

```
paul@paul-VirtualBox:~$ git clone https://github.com/Paul-Solis/CPE232
git
Cloning into 'CPE232_PAULSOLIS'...
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.
paul@paul-VirtualBox:~$
```

- 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.

```
paul@paul-VirtualBox:~$ ls
CPE232_PAULSOLIS  Documents  Music      Public   Templates
Desktop           Downloads  Pictures    snap     Videos
paul@paul-VirtualBox:~$
```

```
paul@paul-VirtualBox:~$ cd CPE232_PAULSOLIS
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ ls
README.md
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$
```

- 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*

```
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git config --global user.name "Paul Solis"
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git config --global user.email qpvmSolis@tip.edu.ph
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ cat ~/.gitconfig
[user]
    name = Paul Solis
    email = qpvmSolis@tip.edu.ph
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$
```

- 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.
- 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.

```
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ 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
```

What is the result of issuing this command?

It says my branch is up to date with origin/main and changes not staged for commit.

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

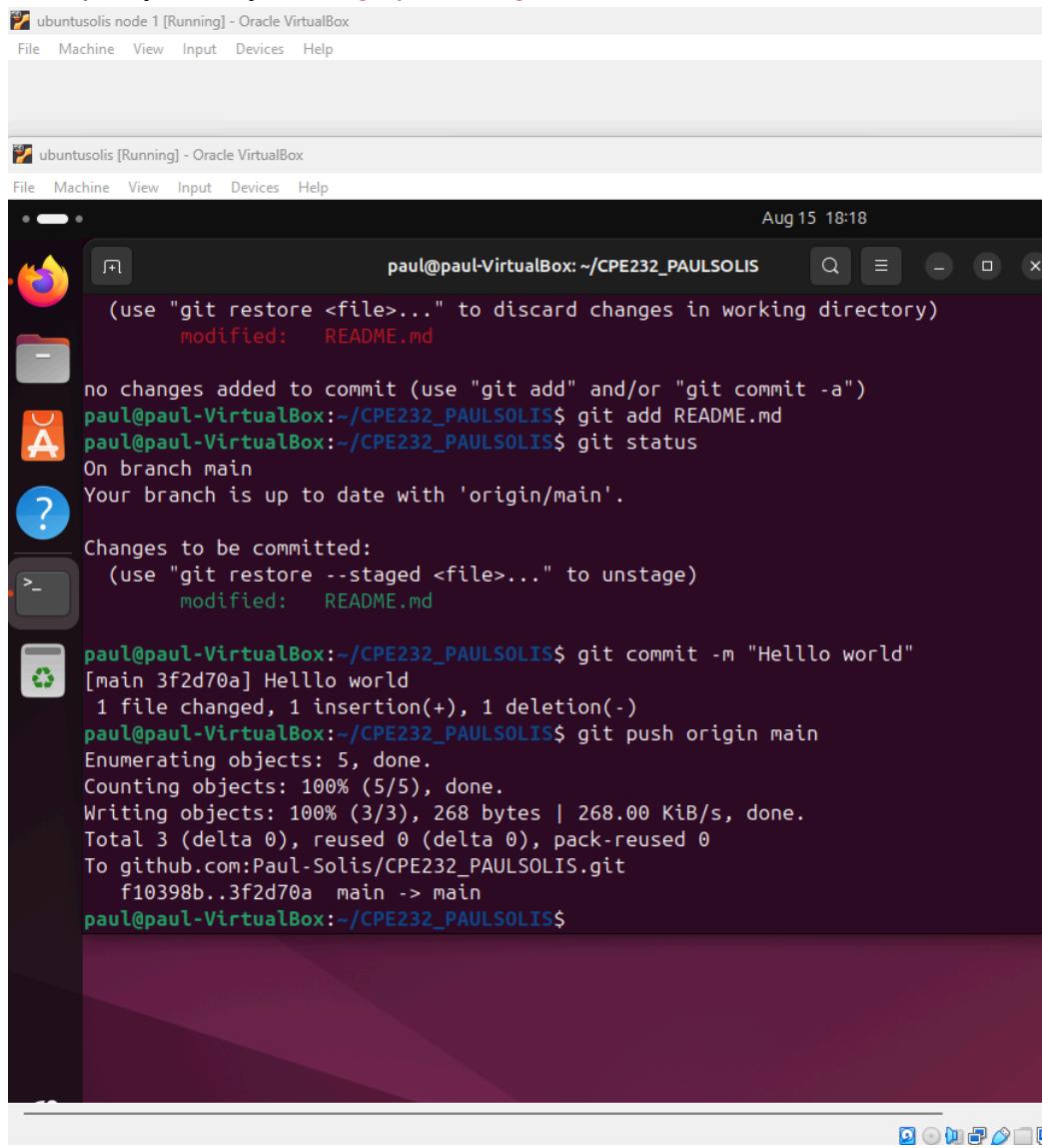
```
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git add README.md
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git status
On branch main
Your branch is up to date with 'origin/main'.

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.

```
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git commit -m "Hello World"
[main d6c6c41] Hello World
 1 file changed, 1 insertion(+), 1 deletion(-)
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$
```

- 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`.



The screenshot shows a Linux desktop environment with several windows open. In the foreground, a terminal window is active, displaying the following command-line session:

```
ubuntusolis node 1 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Aug 15 18:18
paul@paul-VirtualBox: ~/CPE232_PAULSOLIS
(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")
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git add README.md
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ 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

paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git commit -m "Hello world"
[main 3f2d70a] Hello world
 1 file changed, 1 insertion(+), 1 deletion(-)
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 268 bytes | 268.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Paul-Solis/CPE232_PAULSOLIS.git
  f10398b..3f2d70a  main -> main
paul@paul-VirtualBox:~/CPE232_PAULSOLIS$
```

The terminal window has a dark background and light-colored text. The desktop environment includes icons for a browser, file manager, and terminal in the dock at the bottom.

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?
Using Ansible commands, we've remotely installed packages, updated configurations, managed services, transferred files, and executed shell commands
4. How important is the inventory file?

The inventory file is crucial because it defines which servers Ansible will manage and how to connect to them. Without it, Ansible wouldn't know where to apply your commands or playbooks.

Conclusions/Learnings:

This activity demonstrated how SSH key-based authentication enhances security and efficiency in remote server management. By integrating Git and GitHub, we also learned how to version-control our work and collaborate through repositories. Overall, these foundational tools empower us to automate tasks and streamline development workflows in real-world environments