Name: John Renzo L. Mendoza	**Date Performed:** August 26, 2023
Course/Section: CPE31S5	**Date Submitted:** August 29, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** First 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 require 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 passwords stored in files and eliminated the possibility of a compromised server stealing the user's password.

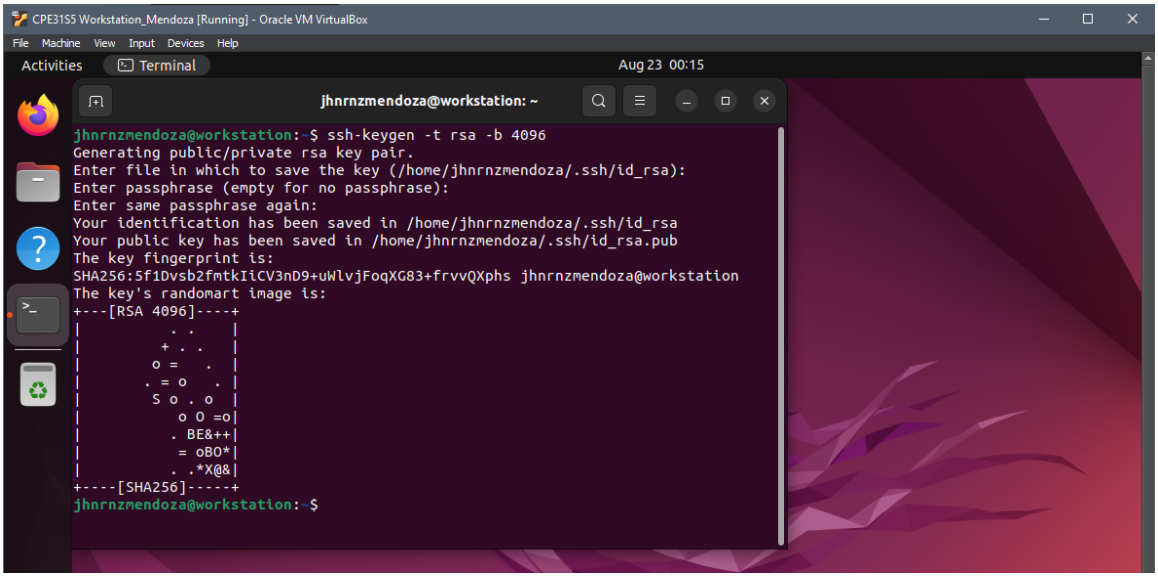
However, SSH keys are authentication credentials just like passwords. Thus, they must be managed somewhat analogously to usernames and passwords. They should have a proper termination process so that keys are removed when no longer needed.

Task 1: Create an SSH Key Pair for User Authentication

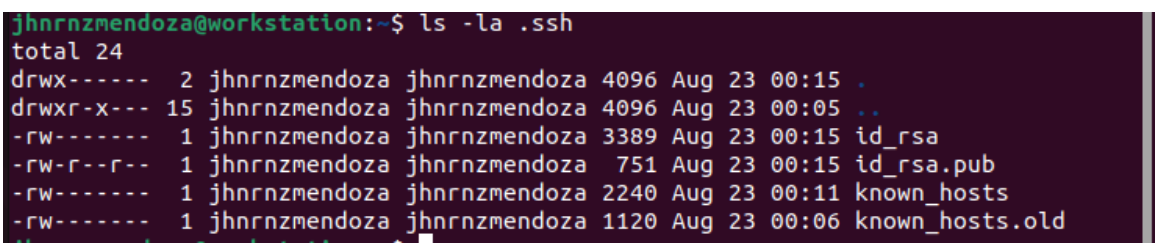
1. The simplest way to generate a key pair is to run *ssh-keygen* without arguments. In this case, it will prompt for the file in which to store keys. First, the tool asked where to save the file. SSH keys for user authentication are usually stored in the users .ssh directory under the home directory. 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.

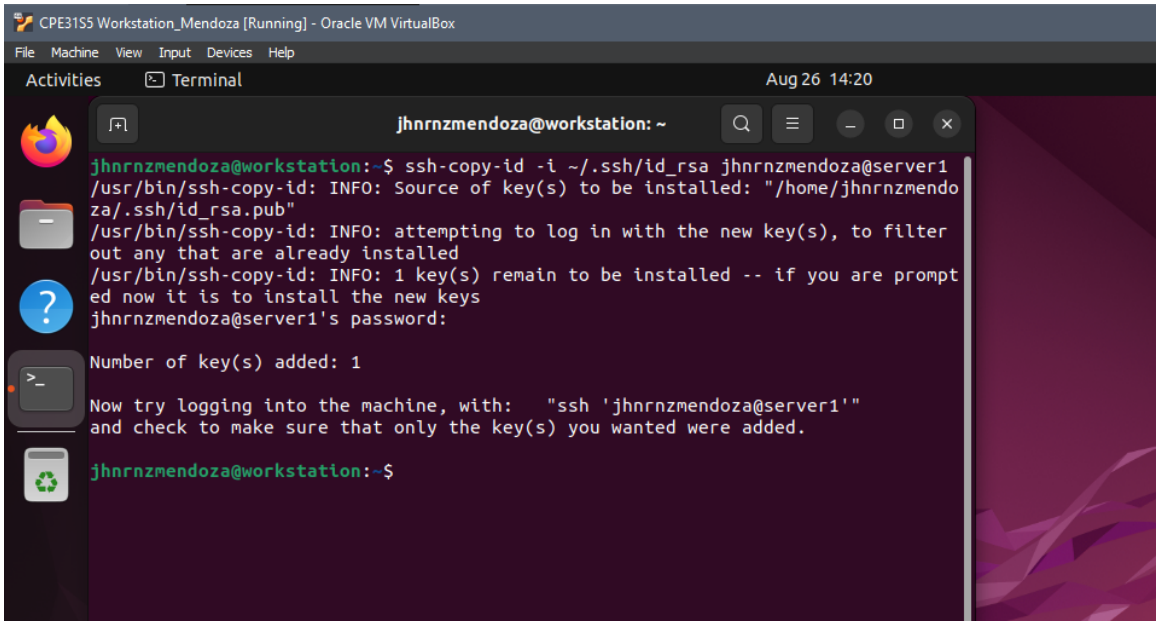


Task 2: Copying the Public Key to the remote servers

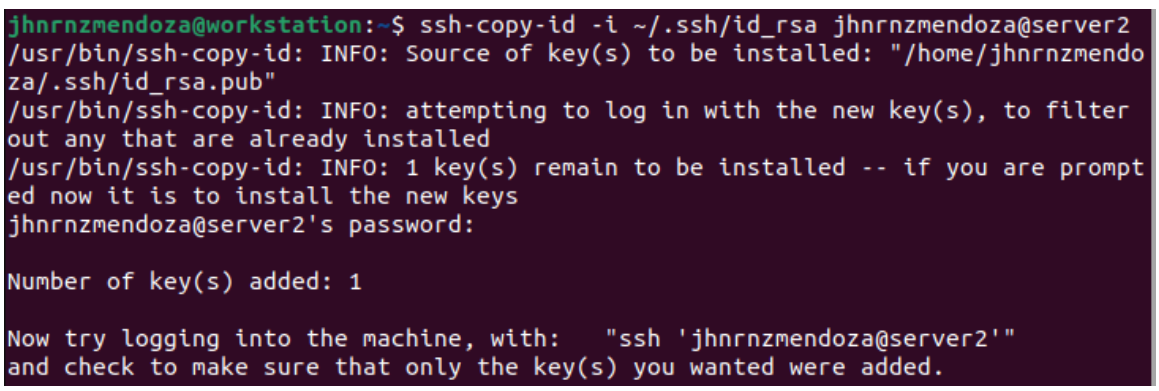
1. To use public key authentication, the public key must be copied to a server and installed in an *authorized_keys* file. This can be conveniently done using the *ssh-copy-id* tool.

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

Server 1



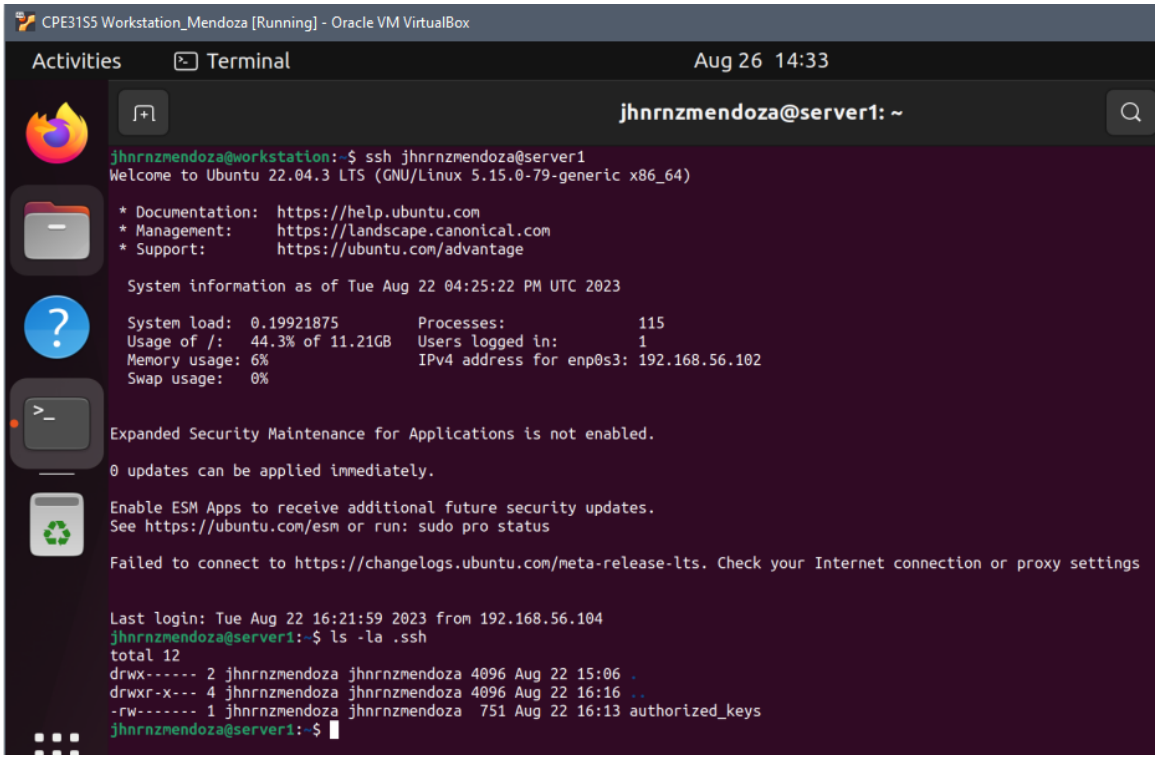
Server 2



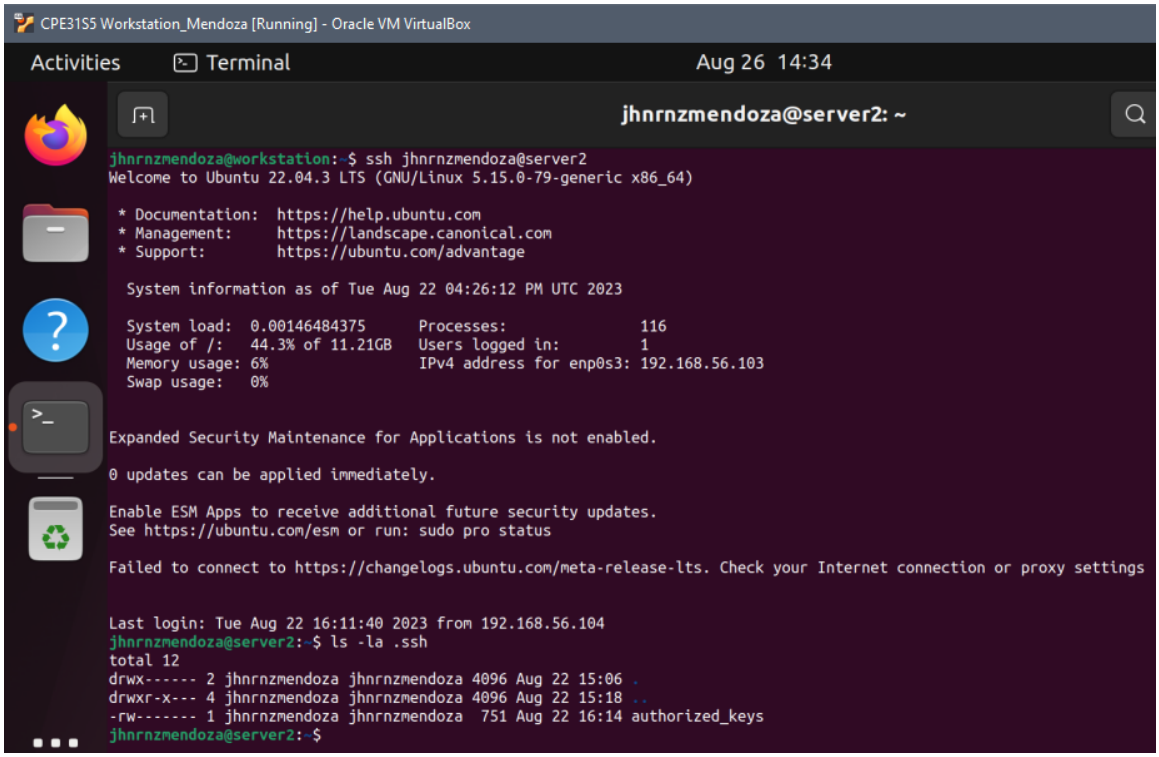
3. Once the public key has been configured on the server, the server will allow any connecting user that has the private key to log in. During the login process, the client proofs possession of the private key by digitally signing the key exchange.

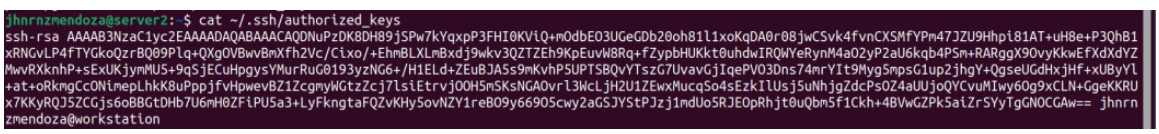
4. On the local machine, verify that you can SSH with Server 1 and Server 2.

SSH on Server 1



SSH on Server 2





What did you notice? Did the connection ask for a password? If not, why?

The remote access to the servers (1 and 2) did not ask for a password / passphrase as I did not enter any password / passphrase when initializing the private key. Moreover, the SSH command to the servers directly allowed me to access it as no password / passphrase is required to access them.

Additional: Add a passphrase on server 1 for proof.

Generating key, now with a passphrase.

```
jhnrnzmendoza@workstation:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jhnrnzmendoza/.ssh/id_rsa):
/home/jhnrnzmendoza/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jhnrnzmendoza/.ssh/id_rsa
Your public key has been saved in /home/jhnrnzmendoza/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1ksj0+KJoMKNHBfbb0SHHIe98+rL+Es2VVNwbyqP7lE jhnrnzmendoza@workstation
The key's randomart image is:
+---[RSA 4096]----+
|        .o.   ..o |
|       ..+.    o .|
|    .   + ..  o  o|
|     + . .=  . .o |
| . o.. .S B.. E  |
|o =. .o+ *.+ =   |
|.+..  .oo+o o .  |
| .    . =... .   |
|       .o*o.o    |
+----[SHA256]-----+
```

```
jhnrnzmendoza@workstation:~$ ls -la .ssh
total 28
drwx------  3 jhnrnzmendoza jhnrnzmendoza 4096 Aug 26 14:21 .
drwxr-x--- 15 jhnrnzmendoza jhnrnzmendoza 4096 Aug 23 00:05 ..
-rw-------  1 jhnrnzmendoza jhnrnzmendoza 3434 Aug 26 14:43 id_rsa
-rw-r--r--  1 jhnrnzmendoza jhnrnzmendoza  751 Aug 26 14:43 id_rsa.pub
-rw-------  1 jhnrnzmendoza jhnrnzmendoza 2240 Aug 23 00:11 known_hosts
-rw-------  1 jhnrnzmendoza jhnrnzmendoza 1120 Aug 23 00:06 known_hosts.old
drwx------  2 jhnrnzmendoza jhnrnzmendoza 4096 Aug 26 14:18 ssh-copy-id.dt7deD4Cvv
```

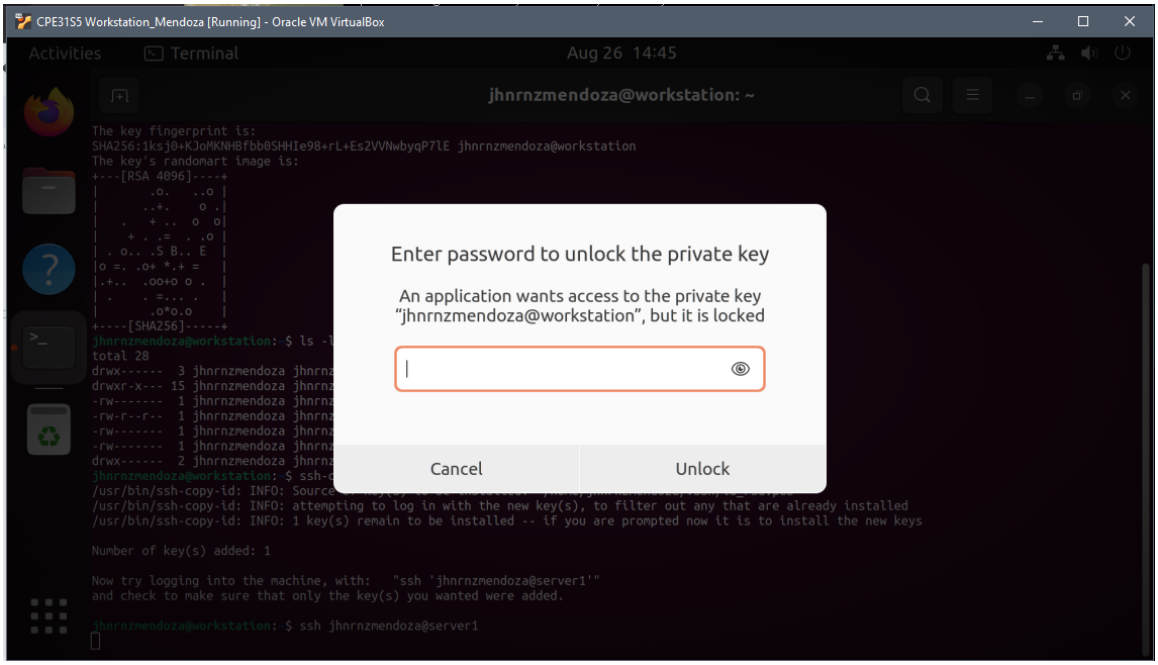
Copying key file to server1

```
jhnrnzmendoza@workstation:~$ ssh-copy-id -i ~/.ssh/id_rsa jhnrnzmendoza@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jhnrnzmendoza/.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

Number of key(s) added: 1

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

Remotely accessing Server 1, now requiring the passphrase.



Reflections:

Answer the following:

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

   The Secure Socket Shell allows the system administrator to remotely configure a device such as a server. In this laboratory activity, the students were tasked to generate a ssh-key which will be required when remotely accessing the server. With the use of this ssh-key, it makes the access secure since only those who know the password or passphrase will be able to access the remote server.

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

   By observing the files on the servers (1 and 2), I have observed that the file *~/.ssh/authorized_keys* is present whenever we have finished copying the file containing the private key from the local machine to the server.

```
jhnrnzmendoza@server2:~$ cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDU4pRC/TyDcZAMUpjuYm1m7CvdqazasxlBYqBb7B53
bTAqeAk8fCli3Oowq9Fgrq9qr7PHO/+be+auVJfDYmogDg5CPa+059icDtWXMSwbGA0+ecTkAPXsf1B/
ByZ5AOvOSof+a6JhGaINBh3HThM6eopSSZwdqjSc9XCcRuEL9bUzSOgiMOFj7JLRyqpECZqGgxy8MqSX
RL6ncF+sSKNWslz0uY6Ykm42dsZRVit5skIIATYBaILIHfv3EOS6No8f9YIyZy0+rk6BDc0lRLz5XT86
WkNewqZ98becjfuDvzprmPtLDfdgEH32CZSzDwCNQaKodYBj4QKYNdVSn0fTHw6VbFLS4Dvq3bbN5td1
+dnJY2A8N93mm92Ls1Ob6DZy3godhLB2TFtapPfPq/7A/hq3fRvODf3ilIsuHpGa0XIQk0VEuWpgy9X3
OawS4r19dsijzGoTzKV9jxL9YuCxmwk3F8pGTEDNQ1nN5AyUkh1vGKPTaSOlwbDSSN/y1PvVnx6OqoNL
ab2qXUkWyLTyr/wOzgVZGhkQEgMqPiubSpKsQTVqpUoP4UTVLixMXmJ8+XiQHVQnii6Qa+VipwEGC1nw
UYtfdLCHbAU9HIMYGrPFW6TD02BN9CA+baJcLCSoYOuphbiw1sSCT2DW21sVKkjmDiYXC+hE87KfLorf
Pw== jhnrnzmendoza@workstation
```

   By also concatenating the contents of the file, it also shows a relation with the local machine denoted by jhnrnzmendoza@workstation.

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**).

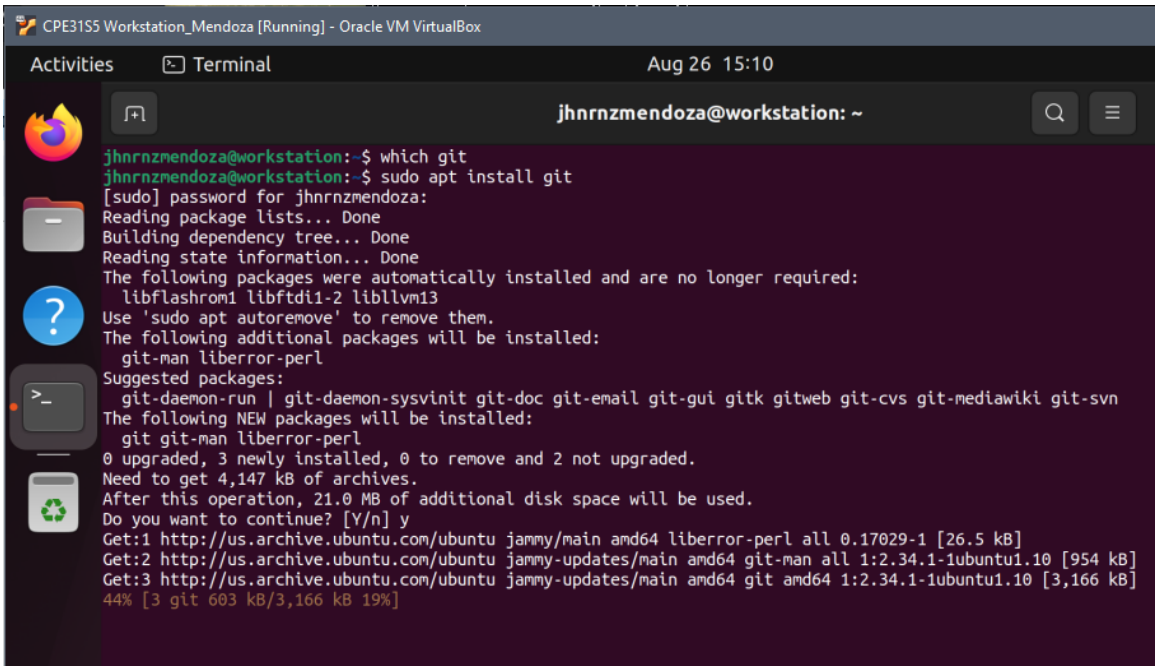
Set up Git

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:

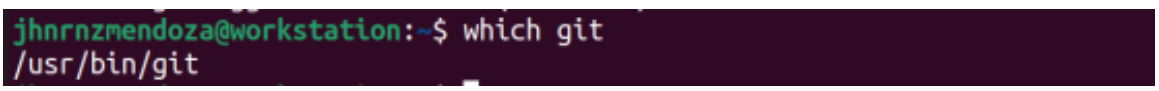
- Creating a repository
- Forking a repository
- Managing files
- Being social

Task 3: Set up the Git Repository

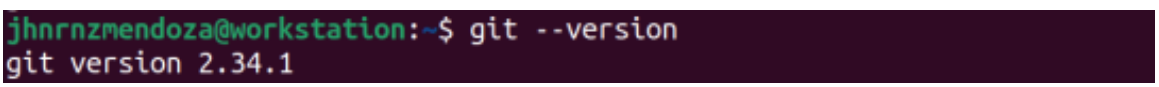
1. On the local machine, verify the version of your git using the command *which git*. If a directory of git is displayed, then you don't need to install git. Otherwise, to install git, use the following command: *sudo apt install git*



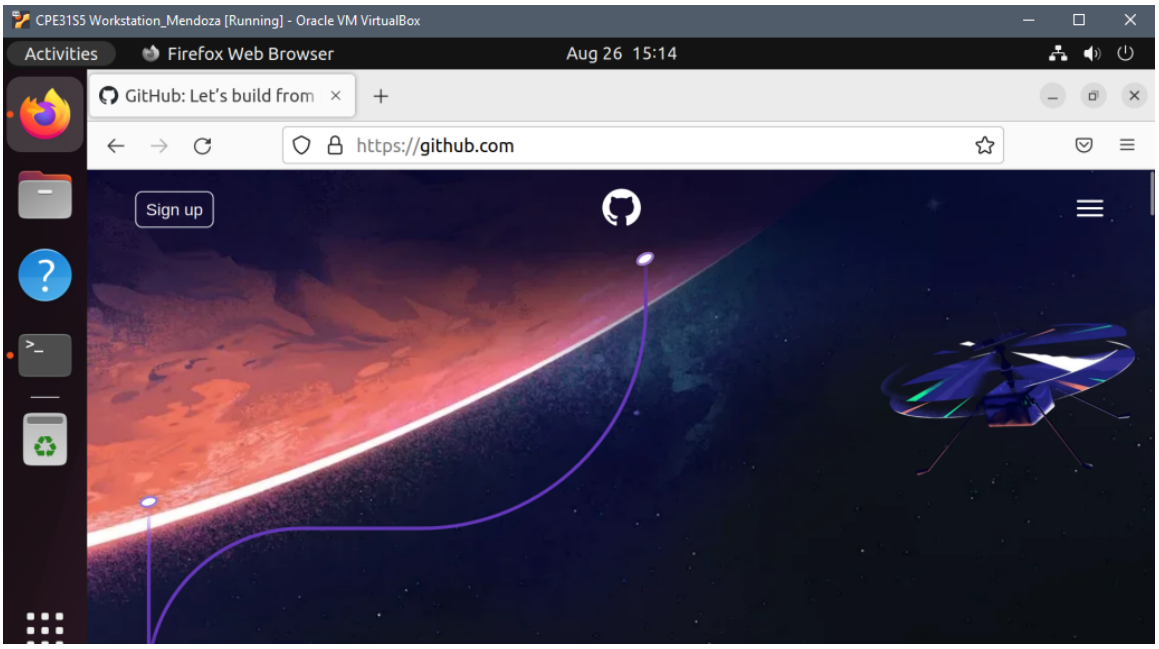
2. After the installation, issue the command *which git* again. The directory of git is usually installed in this location: *user/bin/git*.

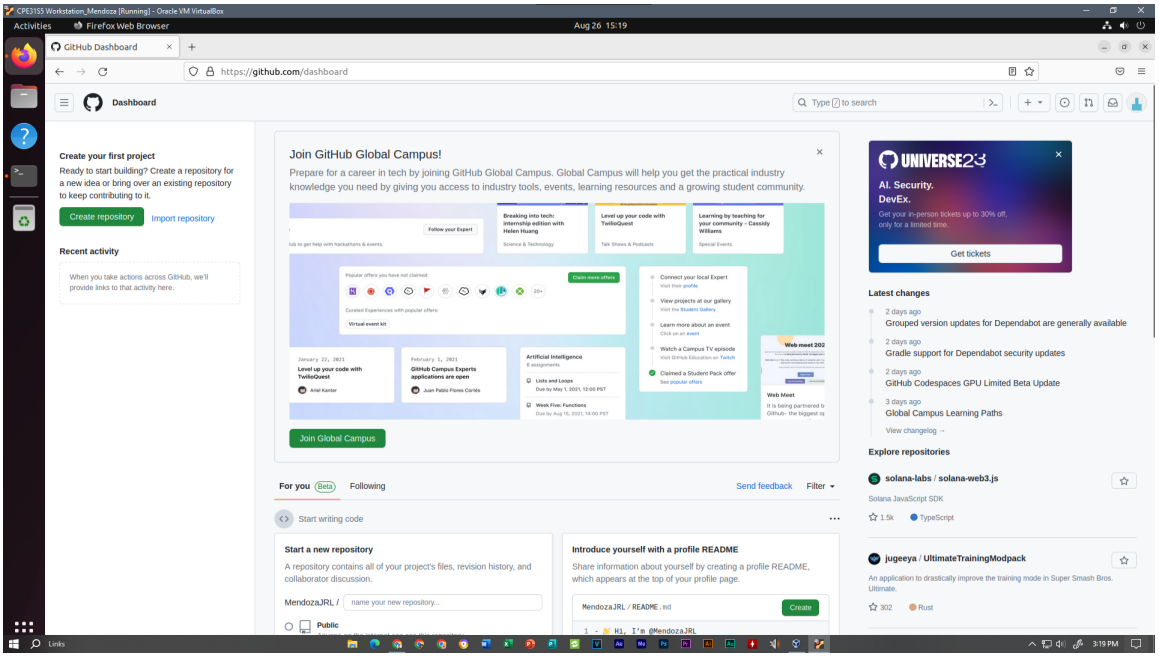


3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

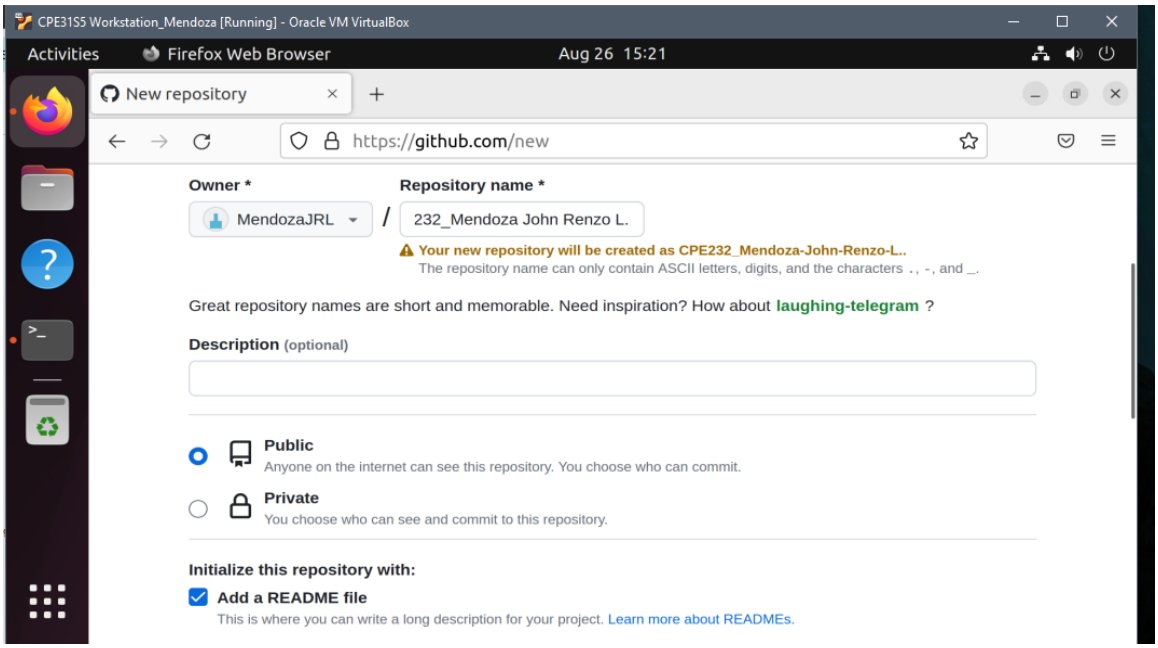


4. Using the browser in the local machine, go to www.github.com.

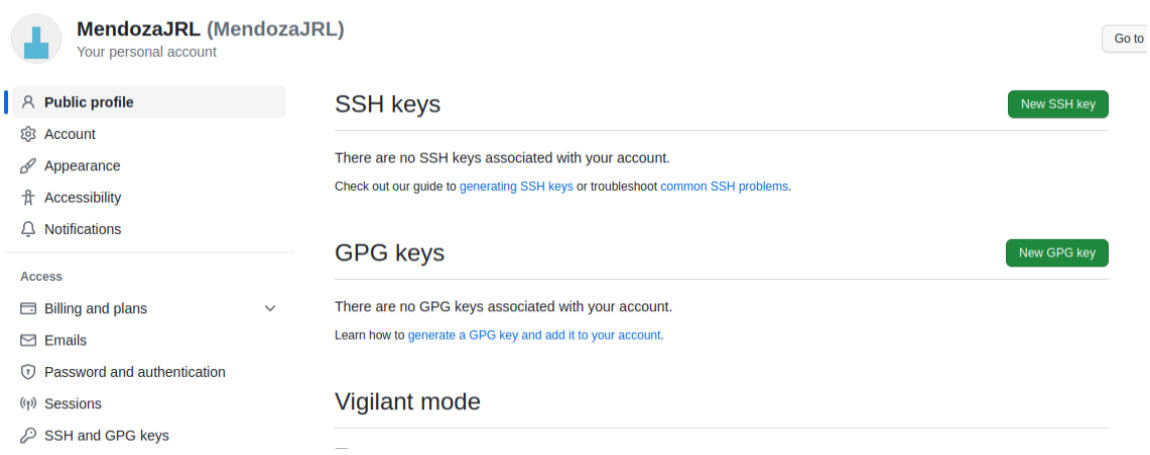


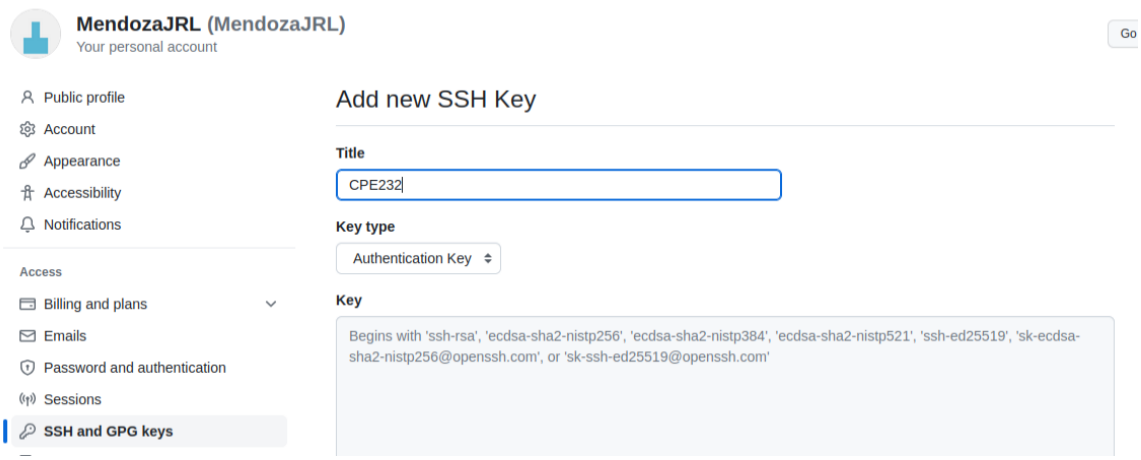


5. Sign up in case you don't have an account yet. Otherwise, login to your GitHub account.
   a. Create a new repository and name it as CPE232_yourname. Check Add a README file and click Create repository.



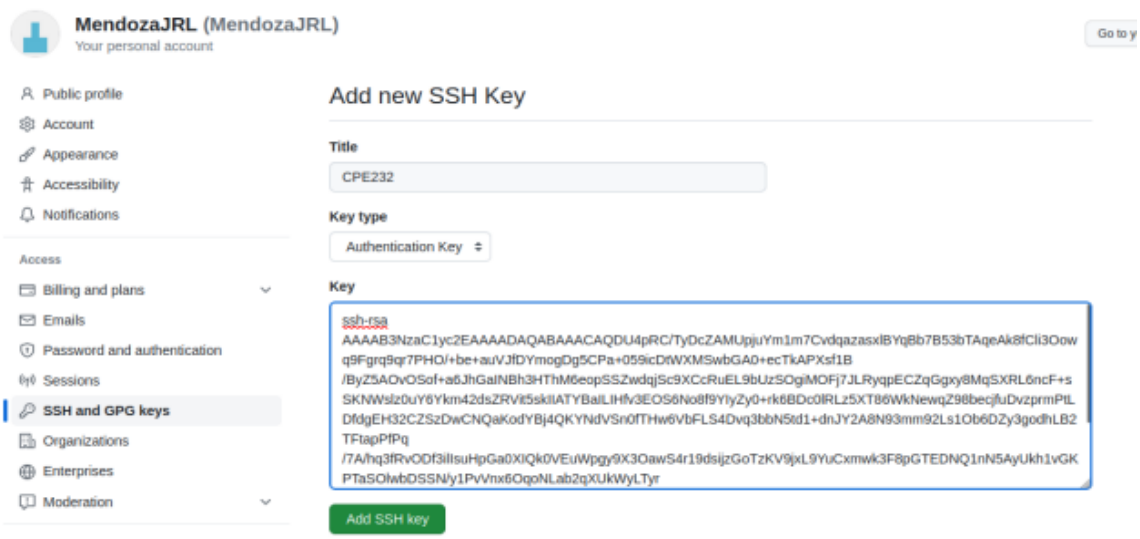
   b. Create a new SSH key on GitHub. Go to your profile's settings and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.



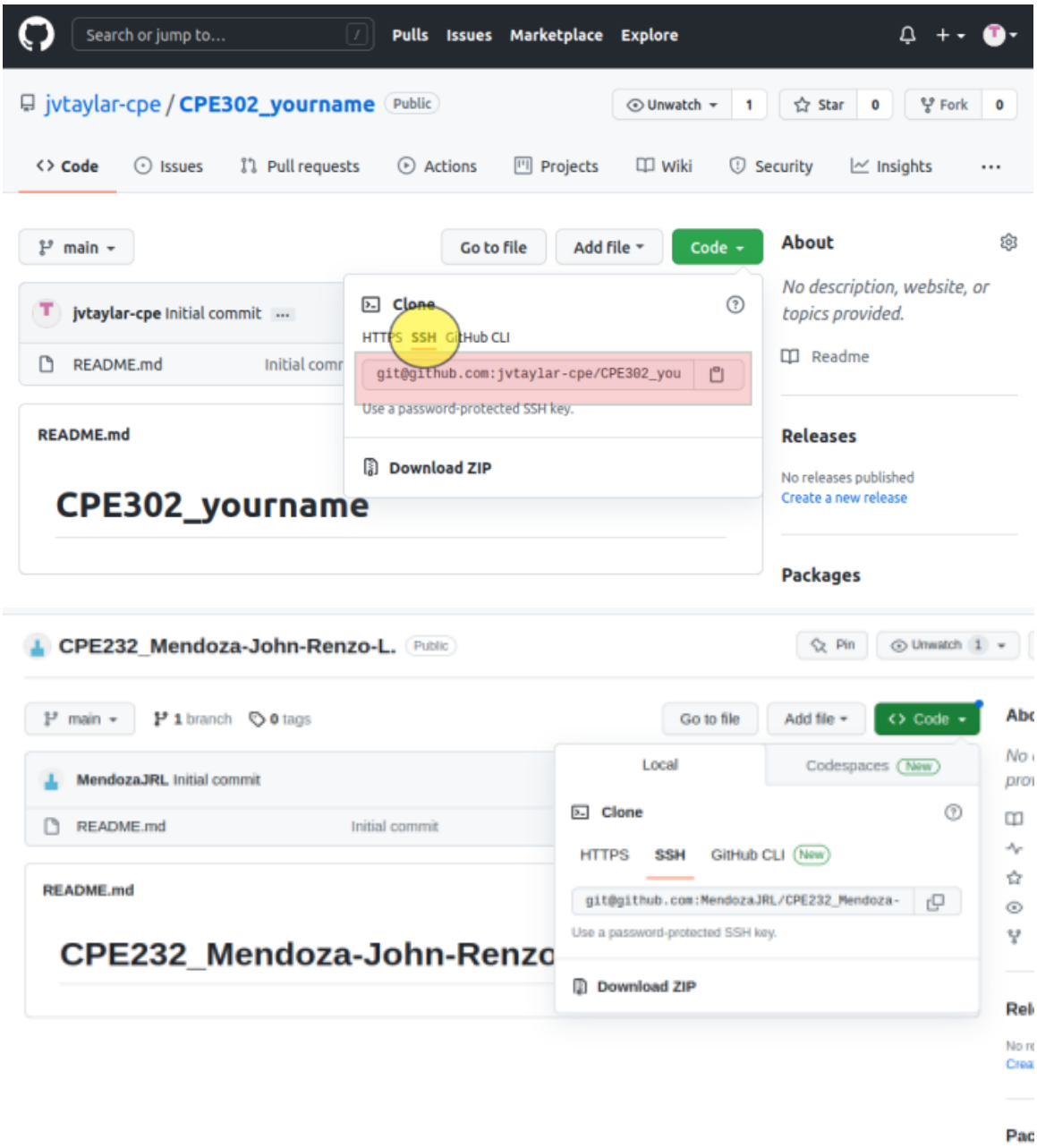


c. On the local machine's terminal, issue the command cat .ssh/id_rsa.pub and copy the public key. Paste it on the GitHub key and press Add SSH key.

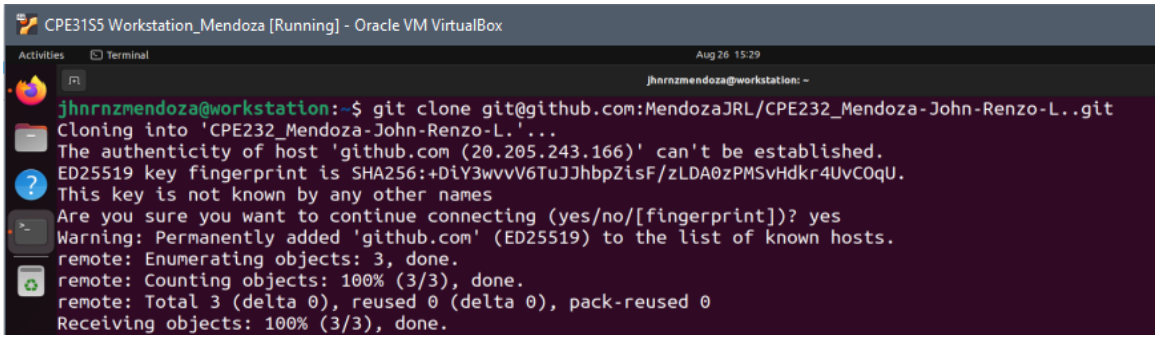
```
jhnrnzmendoza@workstation:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDU4pRC/TyDcZAMUpjuYm1m7Cvdqazasxl
BYqBb7B53bTAqeAk8fCli3Oowq9Fgrq9qr7PHO/+be+auVJfDYm
ogDg5CPa+059icDtWXMSwbGA0+ecTkAPXsf1B/ByZ5AOvOSof+a6JhGaINBh3HThM6eopSS
ZwdqjSc9XCcRuEL9bUzSOgiMOFj7JLRyqpECZqGgxy8MqSXRL6n
cF+sSKNWslz0uY6Ykm42dsZRVit5skIIATYBaILIHfv3EOS6No8f9YIyZy0+rk6BDc0lRLz
5XT86WkNewqZ98becjfuDvzprmPtLDfdgEH32CZSzDwCNQaKodY
Bj4QKYNdVSn0fTHw6VbFLS4Dvq3bbN5td1+dnJY2A8N93mm92Ls1Ob6DZy3godhLB2TFtap
PfPq/7A/hq3fRvODf3ilIsuHpGa0XIQk0VEuWpgy9X3OawS4r19
dsijzGoTzKV9jxL9YuCxmwk3F8pGTEDNQ1nN5AyUkh1vGKPTaSOlwbDSSN/y1PvVnx6OqoN
Lab2qXUkWyLTyr/wOzgVZGhkQEgMqPiubSpKsQTVqpUoP4UTVLi
xMXmJ8+XiQHVQnii6Qa+VipwEGC1nwUYtfdLCHbAU9HIMYGrPFW6TD02BN9CA+baJcLCSoY
Ouphbiw1sSCT2DW21sVKkjmDiYXC+hE87KfLorfPw== jhnrnzm
endoza@workstation
```



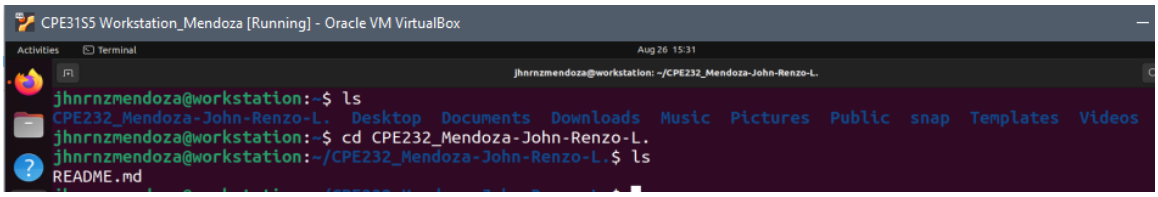
d. Clone the repository that you created. In doing this, you need to get the link from GitHub. Browse to your repository as shown below. Click on the Code drop down menu. Select SSH and copy the link.



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.

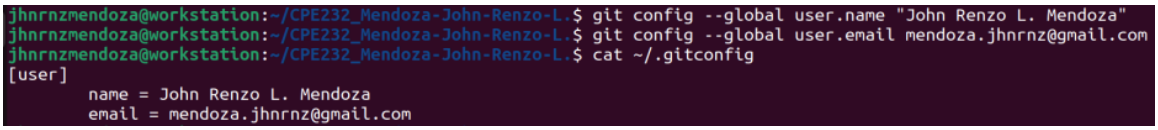


f. To verify that you have cloned the GitHub repository, issue the command *ls*. Observe that you have the CPE232_yourname in the list of your directories. Use CD command to go to that directory and LS command to see the file README.md.

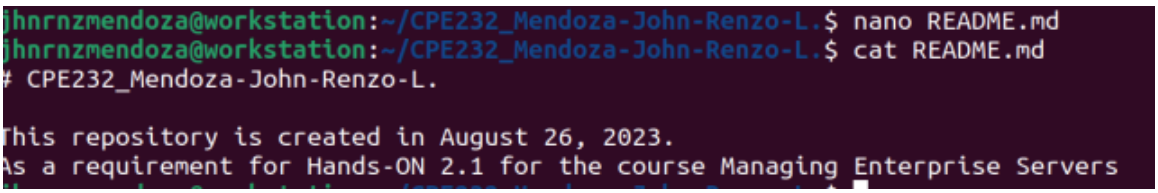


g. Use the following commands to personalize your git.

- *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*



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?

```
jhnrnzmendoza@workstation:~/CPE232_Mendoza-John-Renzo-L.$ 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")
```

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

```
jhnrnzmendoza@workstation:~/CPE232_Mendoza-John-Renzo-L.$ git add README.md
jhnrnzmendoza@workstation:~/CPE232_Mendoza-John-Renzo-L.$ 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
```

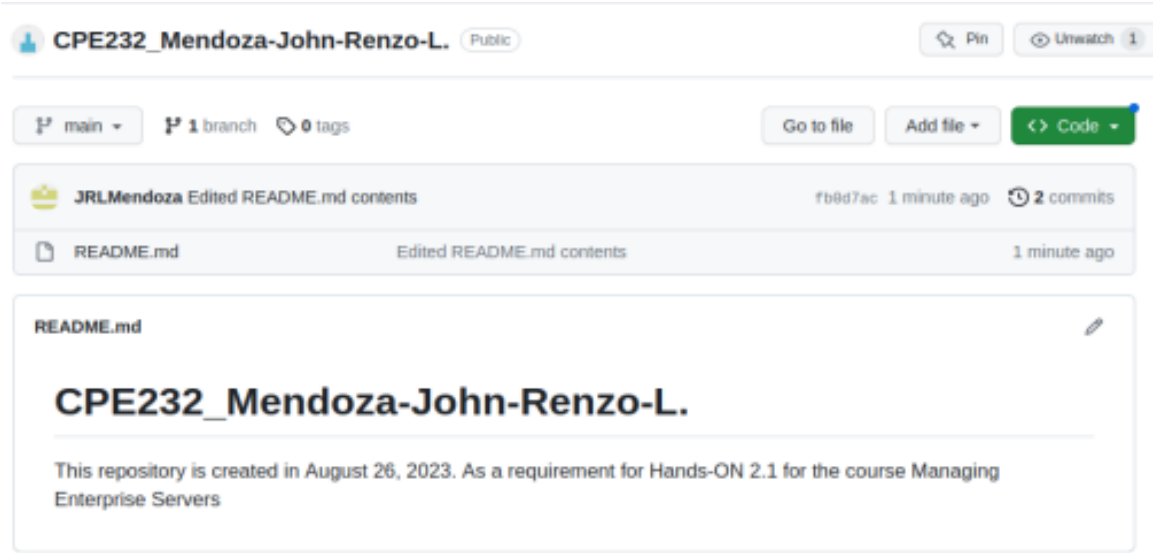
k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

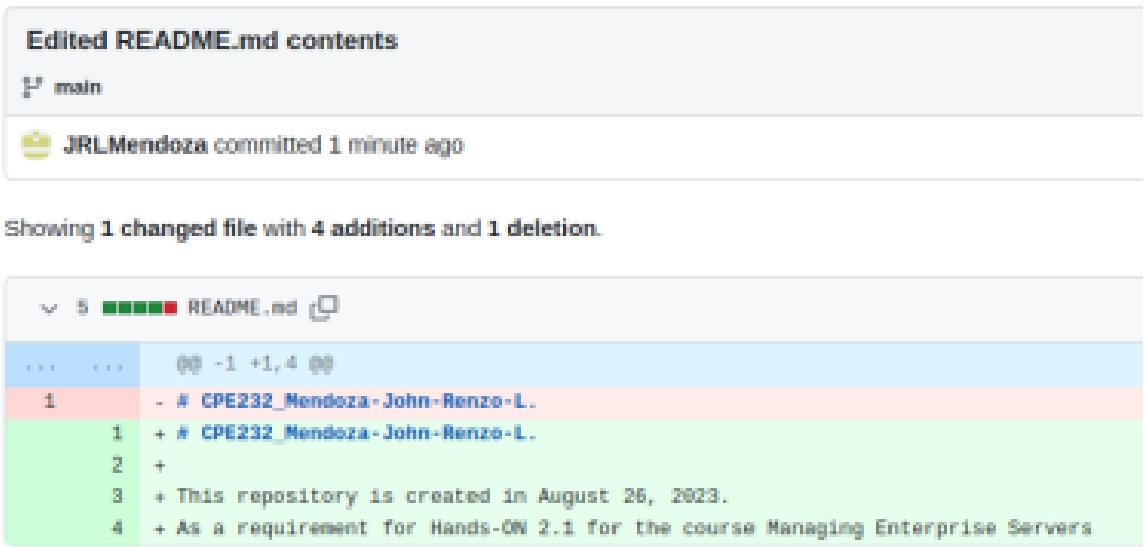
```
jhnrnzmendoza@workstation:~/CPE232_Mendoza-John-Renzo-L.$ git commit -m "Edited README.md contents"
[main fb0d7ac] Edited README.md contents
 1 file changed, 4 insertions(+), 1 deletion(-)
```

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.*

```
jhnrnzmendoza@workstation:~/CPE232_Mendoza-John-Renzo-L.$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 405 bytes | 405.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:MendozaJRL/CPE232_Mendoza-John-Renzo-L..git
   5ccfa01..fb0d7ac  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 how long the last commit is. 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?

   For the third task, the students were able to create a git repository in the github. By using ansible commands installed on the local machine, the students are able to import the created git from the website by using the SSH key in order to access it. Moreover, using the unique SSH key, the system administrator are able to access remote servers / directories. The imported git can also be modified using the terminal and the changes on it can be uploaded back to the github repository using ansible (git) commands.

4. How important is the inventory file?

   An inventory file stores the list of the connected servers or clients in a network. It also contains the commands or scripts stored in a playbook that can be imported on the remote servers or clients to be executed. By using the inventory file, the system administrator will be able to configure multiple devices to have the same configuration without manually configuring them one by one.

Conclusions/Learnings:

In this hands-on activity, the students were enlightened about the secured socket shell (SSH) basic configuration as well as setting up a Git in which connects to the remote servers or clients.

The SSH enables the system administrator to remotely access a server or client by using its IP address in order to link with them. As observed on past 1 of this activity, the students used the IP address of the servers (1 and 2) and used them to remotely connect with them. The students have also added a private key (public key in server) in order to authenticate the appropriate user to access the server. This was made possible by copying the file containing the private key to the remote server.

The Git is a control system which enables the system administrator to organize the network. As observed in this activity (part 2), the students have created a repository and imported it to the local machine using git commands in the terminal. In addition, the git commands can also be used to configure the repository in the local machine. In turn, git commands can also be used to push the changes to the cloud repository in GitHub.

Using these two new concepts, the students would be able to create cloud repositories in which they can import on their local machine to simplify or to make configuration much accessible and much faster.