

Name: Recto, Jon Jeous J.	Date Performed: Jan. 22, 2024
Course/Section: CPE232-CPE31S1	Date Submitted: Jan. 23, 2024
Instructor: Dr. Jonathan V. Taylor	Semester and SY: 2nd Sem. 23-24
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 <p>It is assumed that you are already done with the last Activity (Activity 1: Configure Network using Virtual Machines). <i>Provide screenshots for each task.</i></p> <p>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.</p> <p>What is ssh-keygen?</p> <p>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.</p> <p>SSH Keys and Public Key Authentication</p> <p>The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.</p> <p>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.</p> <p>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.</p>	
Task 1: Create an SSH Key Pair for User Authentication <ul style="list-style-type: none"> 1. The simplest way to generate a key pair is to run <i>ssh-keygen</i> 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`.

```
jonjeous@localmachine-VirtualBox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jonjeous/.ssh/id_rsa): /home/jonjeou
s/.ssh/id_rsa
/home/jonjeous/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jonjeous/.ssh/id_rsa.
Your public key has been saved in /home/jonjeous/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:nIf0/BdBMMJq1ocjvwtSVVZrEkfh/Prb8tyZc+U1bQk jonjeous@localmachine-Virtua
lBox
The key's randomart image is:
+---[RSA 2048]---+
|      .. *+.=.      |
|      .+ *..      |
|      o...=      |
|      =o+o.o.E      |
|      o.oSo. .o o|
|      . +.. .. o=|
|      . . +. .. o+|
|      . . . . .+*|
```

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.

```
jonjeous@localmachine-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jonjeous/.ssh/id_rsa): /home/jonjeou
s/.ssh/id_rsa
/home/jonjeous/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jonjeous/.ssh/id_rsa.
Your public key has been saved in /home/jonjeous/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:etAPdnbpWtX66rNDthOrJ+OWvy6IDSQZ81uroxGj/o jonjeous@localmachine-Virtua
lBox
The key's randomart image is:
+---[RSA 4096]---+
|      . o      |
|      o o o . . |
|      + .. o o  |
|      o.oS o + . |
|      .. ++.=+. . .|
|      . o..o.+.=o ..|
|      oo.o.o *.=. .|
|      .+E o . .==Bo=. .|
```

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`.

```

jonjeous@localmachine-VirtualBox:~$ ls -la .ssh
total 20
drwx----- 2 jonjeous jonjeous 4096 Jan 23 17:10 .
drwxr-xr-x 15 jonjeous jonjeous 4096 Jan 23 16:47 ..
-rw----- 1 jonjeous jonjeous 3243 Jan 23 17:10 id_rsa
-rw-r--r-- 1 jonjeous jonjeous 758 Jan 23 17:10 id_rsa.pub
-rw-r--r-- 1 jonjeous jonjeous 666 Jan 22 20:16 known_hosts
jonjeous@localmachine-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.

```

jonjeous@localmachine-VirtualBox:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n] [-i [identity_file]] [-p port] [[-o <
ssh -o options>] ...] [user@]hostname
    -f: force mode -- copy keys without trying to check if they are already
    installed
    -n: dry run -- no keys are actually copied
    -h|-?: print this help
jonjeous@localmachine-VirtualBox:~$

```

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

```

jonjeous@localmachine-VirtualBox:~$ ssh-copy-id -i ~/.ssh/id_rsa jonjeous@local
machine
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jonjeous/.
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: Could not resolve hostname localmachine: Name
or service not known

```

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.

```

jonjeous@localmachine-VirtualBox:~$ ssh-copy-id -i ~/.ssh/id_rsa jonjeous@serve
r1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jonjeous/.
ssh/id_rsa.pub"
The authenticity of host 'server1 (192.168.56.113)' can't be established.
ECDSA key fingerprint is SHA256:hyi5/zJf3URyWl703e93TSQJ1rnSDyICXSCjCbA9i/Q.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
The authenticity of host 'server1 (192.168.56.113)' can't be established.
ECDSA key fingerprint is SHA256:hyi5/zJf3URyWl703e93TSQJ1rnSDyICXSCjCbA9i/Q.
Are you sure you want to continue connecting (yes/no)? yes
/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
jonjeous@server1's password:

Number of key(s) added: 1

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

```

```

jonjeous@localmachine-VirtualBox:~$ ssh-copy-id -i ~/.ssh/id_rsa jonjeous@server2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jonjeous/.ssh/id_rsa.pub"
The authenticity of host 'server2 (192.168.56.115)' can't be established.
ECDSA key fingerprint is SHA256:PitI+aSEwFkS+SV0JYndttMgi42d6Bpb+ciOLF202s.
Are you sure you want to continue connecting (yes/no)? 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
jonjeous@server2's password:

Number of key(s) added: 1

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

```

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

```

jonjeous@localmachine-VirtualBox:~$ ssh jonjeous@server1
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.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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: Mon Jan 22 20:14:49 2024 from 192.168.56.113
jonjeous@server1-VirtualBox:~$

```

```

jonjeous@localmachine-VirtualBox:~$ ssh jonjeous@server2
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.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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: Mon Jan 22 20:15:13 2024 from 192.168.56.115
jonjeous@server2-VirtualBox:~$

```

I noticed that the connection did not ask for the password, I think it is because we are already in the local machine.

Reflections:

Answer the following:

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

The SSH program is a secure tool that lets you control a computer or server from a distance. It ensures that your interactions are protected from unauthorized access or tampering, making it a safe way to manage systems and transfer data over the internet. It's commonly used for tasks like remote server management and secure file transfers.

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

To check if your public key is installed on a remote server, try connecting to it using SSH. If you don't get asked for a password during the connection, your public key is likely installed correctly. On the server, you can look in a file called "authorized_keys" to make sure your public key is listed there. This file should match the key on your local machine. If everything aligns, your public key is set up on the remote 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*

```

jonjeous@localmachine-VirtualBox:~$ which git
jonjeous@localmachine-VirtualBox:~$ sudo apt install git
[sudo] password for jonjeous:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el 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 0 not upgraded.
Need to get 4,817 kB of archives.
After this operation, 34.3 MB of additional disk space will be used.

```

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

```

jonjeous@localmachine-VirtualBox:~$ which git
/usr/bin/git
jonjeous@localmachine-VirtualBox:~$

```

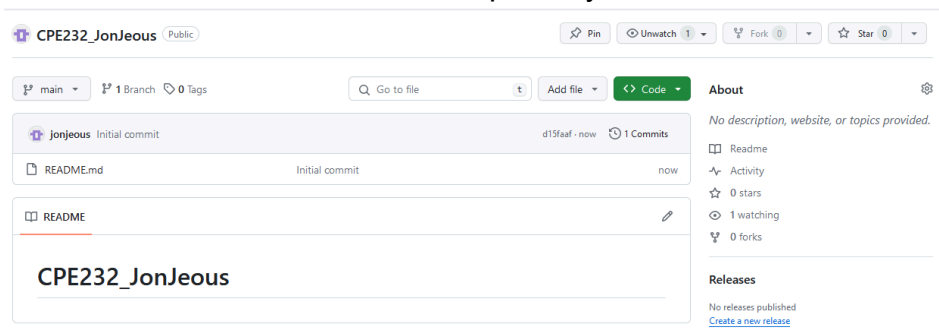
3. The version of git installed in your device is the latest. Try issuing the command *git --version* to know the version installed.

```

jonjeous@localmachine-VirtualBox:~$ git --version
git version 2.17.1
jonjeous@localmachine-VirtualBox:~$

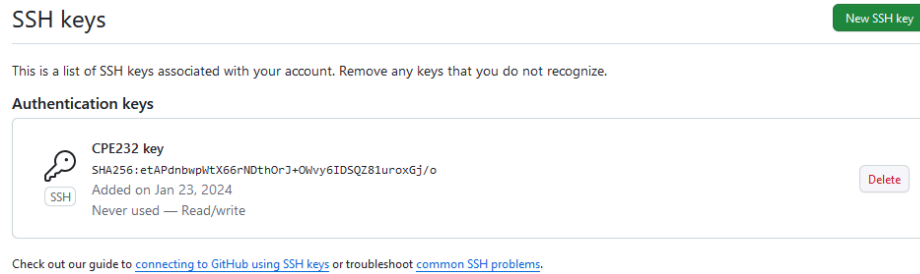
```

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.

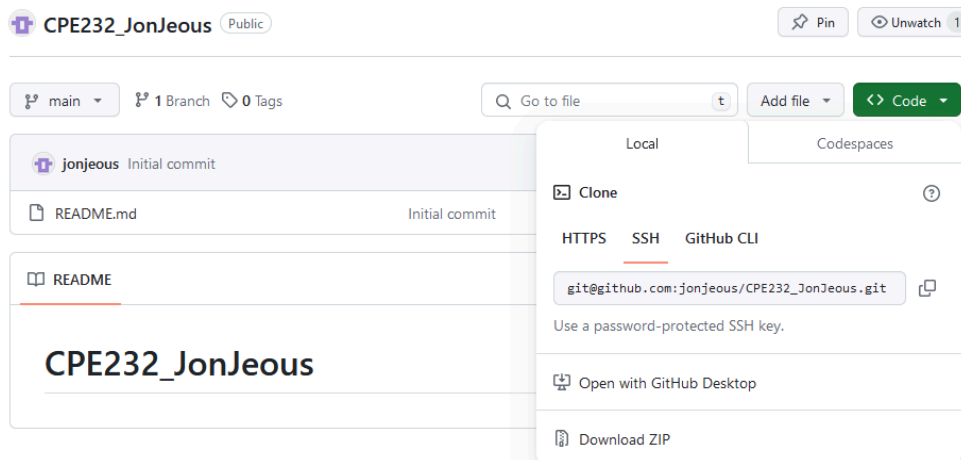


- b. Create a new SSH key on GitHub. Go your profile's setting and click SSH and GPG keys. If there is an existing key, make sure to delete it. To create a new SSH keys, click New SSH Key. Write CPE232 key as the title of the key.

- 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.



- 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.

```
jonjeous@localmachine-VirtualBox:~$ git clone git@github.com:jonjeous/CPE232_JonJeous.git
Cloning into 'CPE232_JonJeous'...
The authenticity of host 'github.com (20.205.243.166)' 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,20.205.243.166' (ECDSA) to the list of known 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
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.


```

jonjeous@localmachine-VirtualBox:~$ ls
CPE232_JonJeous  Documents  examples.desktop  id_rsa.pub  Pictures  Templates
Desktop          Downloads  id_rsa           Music       Public    Videos
jonjeous@localmachine-VirtualBox:~$ cd CPE232_JonJeous
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ ls
README.md
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$

```

- 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`

```

jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ git config --global user.name "Jon Jeous"
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ git config --global user.email qjjjrecto@tip.edu.ph
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ cat ~/.gitconfig
cat: /home/jonjeous/.gitconfig: No such file or directory
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ cat ~/.gitconfig
[user]
    name = Jon Jeous
    email = qjjjrecto@tip.edu.ph
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$

```

- 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.

```

jonjeous@localmachine-VirtualBox: ~/CPE232_JonJeous
File Edit View Search Terminal Help
GNU nano 2.9.3 README.md
# CPE232_JonJeous
Hello!

```

- 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?

```

jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ 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 checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$

```

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


```
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ git add README.md
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$
```

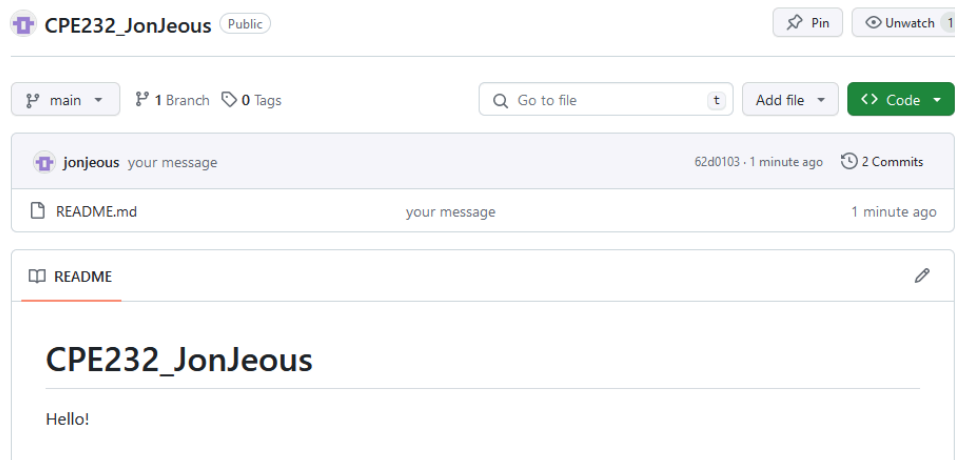
- k. Use the *git commit -m "your message"* to create a snapshot of the staged changes along the timeline of the Git projects history. The use of this command is required to select the changes that will be staged for the next commit.

```
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ git commit -m "your message"
[main 62d0103] your message
1 file changed, 2 insertions(+), 1 deletion(-)
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$
```

- 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*.

```
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$ git push origin main
Counting objects: 3, done.
Writing objects: 100% (3/3), 268 bytes | 268.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:jonjeous/CPE232_JonJeous.git
d15faaf..62d0103 main -> main
jonjeous@localmachine-VirtualBox:~/CPE232_JonJeous$
```

- 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.



The screenshot shows the GitHub interface for the repository 'CPE232_JonJeous'. At the top, it indicates the repository is 'Public'. Below this, there are buttons for 'Pin' and 'Unwatch'. The main section shows the commit history with a single commit by 'jonjeous' titled 'your message' at hash '62d0103', committed '1 minute ago'. Below the commit list, the 'README' file is shown, which contains the text 'CPE232_JonJeous' and 'Hello!'.

Reflections:

Answer the following:

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

We connected our managedNode to our github account. To test its connectivity we put texts inside the README.md file to see if it will reflect on my github account.

4. How important is the inventory file?

The inventory file is crucial for keeping track of servers and organizing tasks. It helps in managing configurations, executing tasks on specific servers, and adapting to changes in the infrastructure. By categorizing hosts and providing a structured overview, it simplifies the process of automation and serves as a useful form of documentation for understanding the network's layout. Overall, the inventory file plays a vital role in efficient server management and organization.

Conclusions/Learnings:

In conclusion, accomplishing these objectives has equipped me with the skills to establish secure connections, efficiently manage code, and administer remote servers effectively. Using SSH keys enhances security, Git repositories support collaborative development, and running commands on remote servers provides flexibility in server management.