Name: Mamaril, Justin Kenneth I.	Date Performed: 09/03/2025
Course/Section: CPE212-CPE31S2	Date Submitted: 09/05/2025
Instructor: Engr Robin Valenzuela	Semester and SY: 1st Year 25-26

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages in the remote machines. We will also create a playbook that will be used for automations.

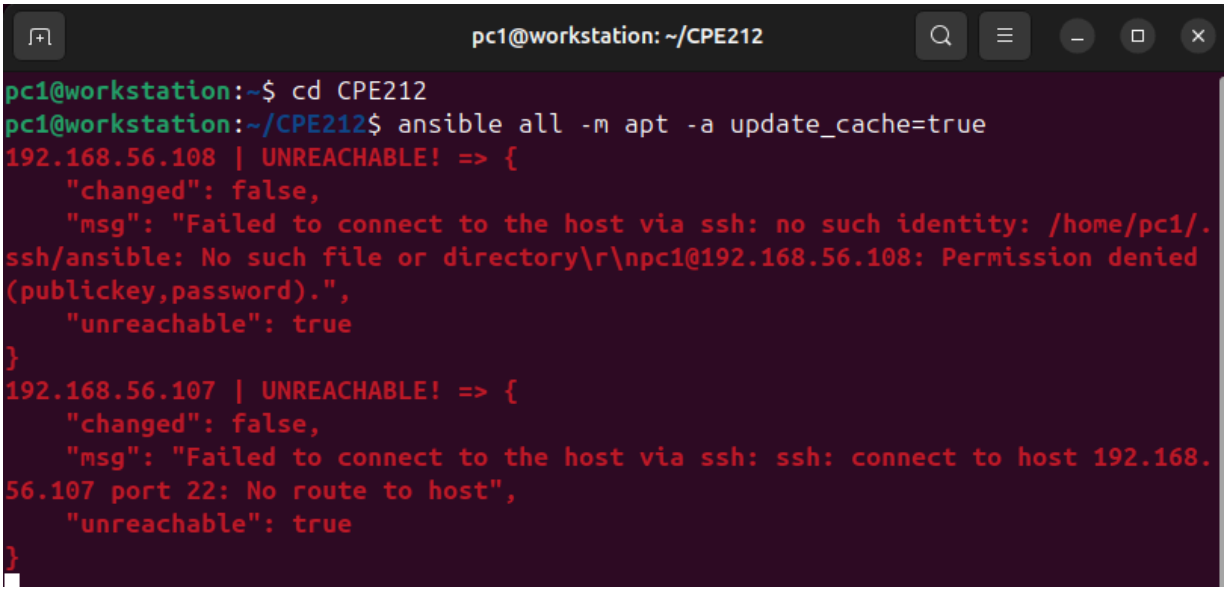
Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. You can check this documentation if you want to learn more about playbooks. Working with playbooks — Ansible Documentation

Task 1: Run elevated ad hoc commands

1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package information from the Internet. It is useful to get info on an updated version of packages or their dependencies. We can only run

an apt update command in a remote machine. Issue the following command:

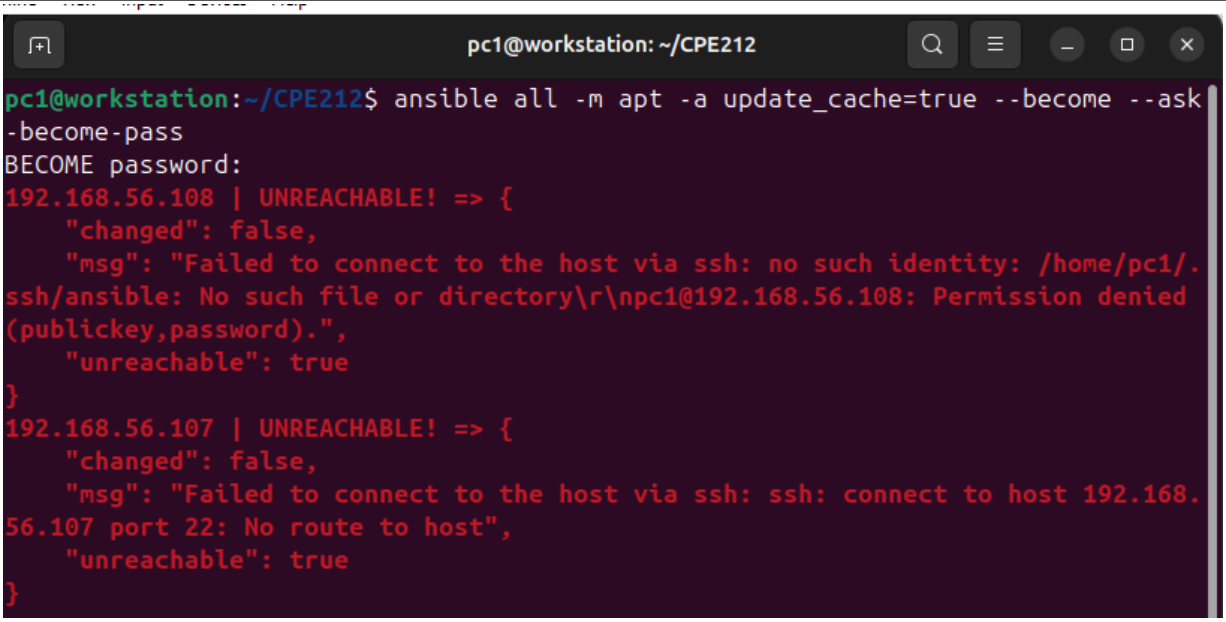
ansible all -m apt -a update_cache=true



What is the result of the command? Is it successful?

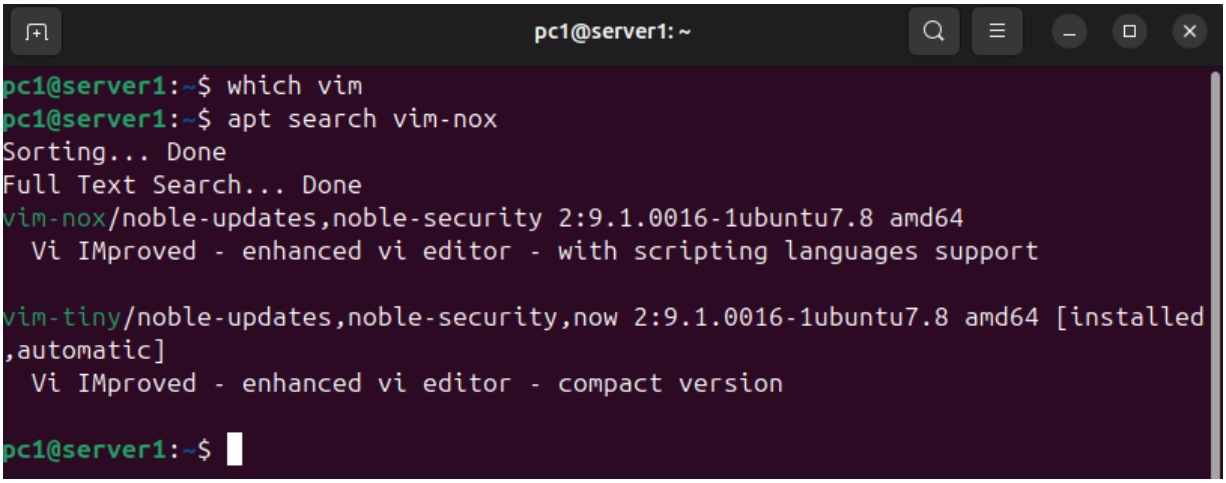
- It failed with a error message “failed to lock apt for exclusive operation”

Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same thing as running *sudo apt update*. The --become command elevate the privileges and the *--ask-become-pass* asks for the password. For now, even if we only have changed the packaged index, we were able to change something on the remote server.



You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction. Here is the command: *ansible all -m apt -a name=vim-nox --become --ask-become-pass.* The command would take some time after typing the password because the local machine instructed the remote servers to actually install the package.
   2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?



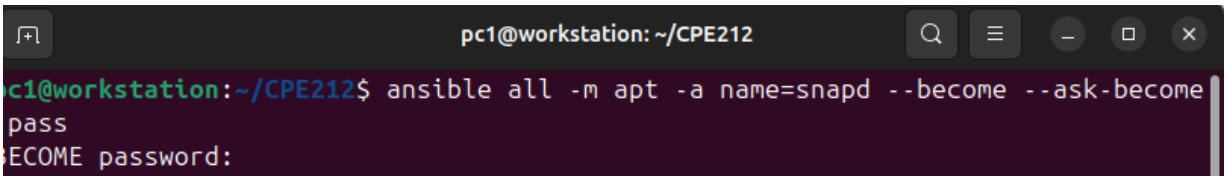
- The command is successful and vim-nox was installed in the server1

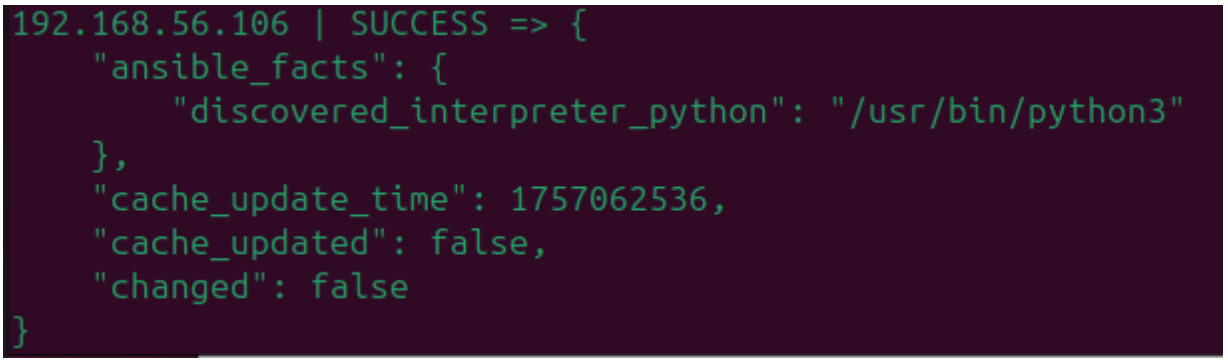
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.

3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

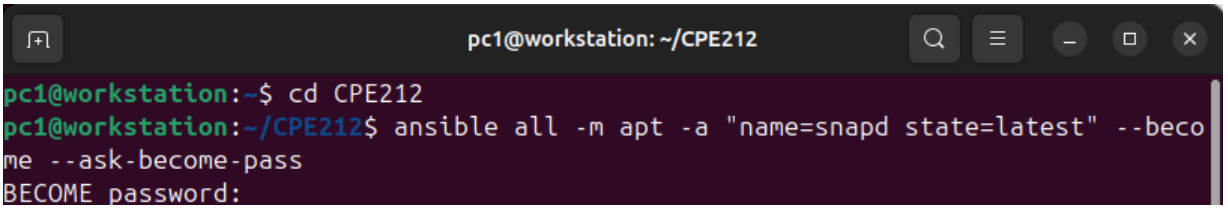
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*
   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?

- The result was success on server1 ip address which is 192.168.56.106 and it says the snapd package was successfully installed in the remote server





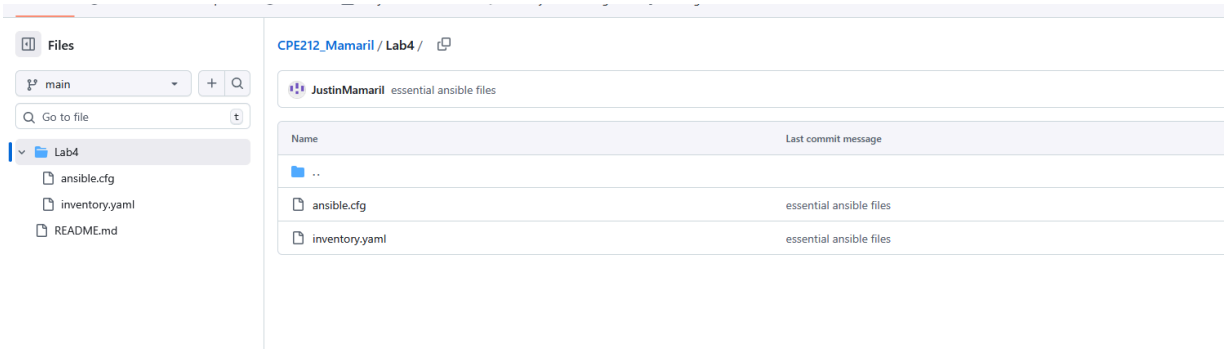
3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*
Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.



4. At this point, make sure to commit all changes to GitHub.

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ git add ansible.cfg
pc1@workstation:~/CPE212_Mamaril/Lab4$ git add inventory.yaml
pc1@workstation:~/CPE212_Mamaril/Lab4$ 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)
        new file:   ansible.cfg
        new file:   inventory.yaml
```

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ git commit -m "essential ansible files"
[main 3c8c4ec] essential ansible files
 2 files changed, 13 insertions(+)
 create mode 100644 Lab4/ansible.cfg
 create mode 100644 Lab4/inventory.yaml
pc1@workstation:~/CPE212_Mamaril/Lab4$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 5 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 508 bytes | 508.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:JustinMamaril/CPE212_Mamaril.git
   68b7f53..3c8c4ec  main -> main
```



Task 2: Writing our First Playbook

1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and the place or commands that ansible will carry out to bring to that state. You can use an editor to create a playbook. Before we proceed, make sure that you are in the directory of the repository that we use in the previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. This will create a playbook file called *install_apache.yml*. The .yml is the basic standard extension for playbook files.

   When the editor appears, type the following:

```
  GNU nano 4.8                  install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2
```

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ sudo nano install_apache.yml
[sudo] password for pc1:
```

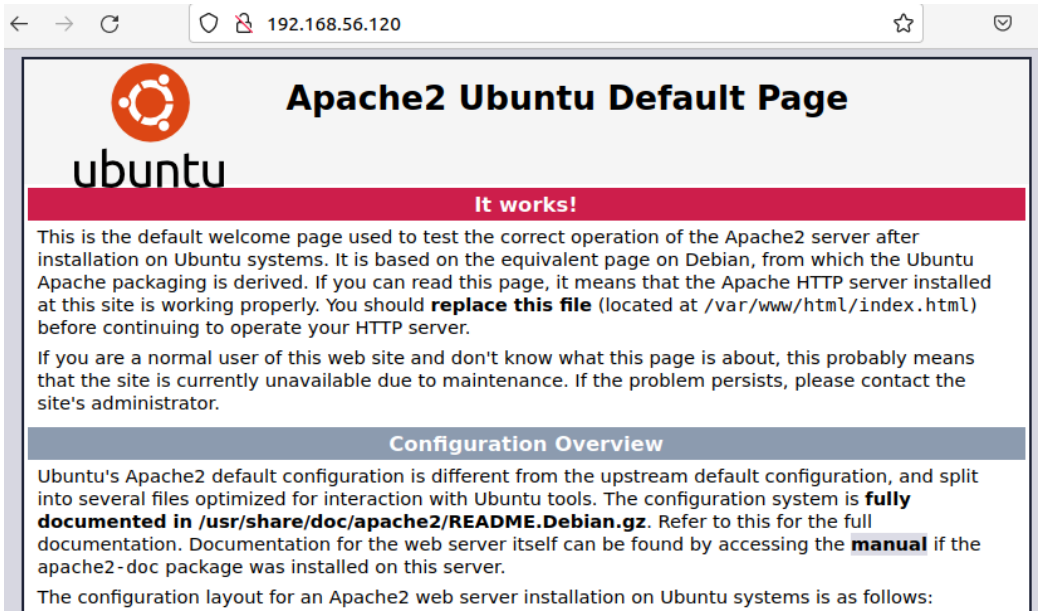
```
pc1@workstation: ~/CPE212_Mamaril/Lab4
  GNU nano 7.2                                  install_apache.yml *
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2
```

   Make sure to save the file. Take note also of the alignments of the texts.

2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [webserver] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.56.107]
ok: [192.168.56.106]

TASK [install apache2 package] *************************************************************************************
ok: [192.168.56.106]
ok: [192.168.56.107]

PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.107             : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.



4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?

```
GNU nano 7.2
---
- hosts: webserver
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: legendslang
```

```
TASK [install apache2 package] *************************************************************************
fatal: [192.168.56.107]: FAILED! => {"changed": false, "msg": "No package matching 'legendslang' is available"}
fatal: [192.168.56.106]: FAILED! => {"changed": false, "msg": "No package matching 'legendslang' is available"}
```

- It is not recognizable

5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2
```

Save the changes to this file and exit.

```
  GNU nano 7.2
---
- hosts: webserver
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2
```

6. Run the playbook and describe the output. Did the new command change anything on the remote servers?

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [webserver] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.56.106]
ok: [192.168.56.107]

TASK [update repository index] *************************************************************************************
changed: [192.168.56.106]
changed: [192.168.56.107]

TASK [install apache2 package] *************************************************************************************
ok: [192.168.56.107]
ok: [192.168.56.106]

PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.107             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

- apt update and installation was successfully installed in the servers 1 and 2

7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

Save the changes to this file and exit.

```
  GNU nano 7.2
---
- hosts: webserver
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

8. Run the playbook and describe the output. Did the new command change anything on the remote servers?

```
pc1@workstation:~/CPE212_Mamaril/Lab4$ ansible-playbook --ask-become-pass install_apache.
BECOME password:

PLAY [webserver] ***************************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [192.168.56.107]
ok: [192.168.56.106]

TASK [update repository index] *************************************************************
changed: [192.168.56.107]
changed: [192.168.56.106]

TASK [install apache2 package] *************************************************************
ok: [192.168.56.106]
ok: [192.168.56.107]

TASK [add PHP support for apache] **********************************************************
changed: [192.168.56.106]
```

9. Finally, make sure that we are in sync with GitHub. Provide the link to your GitHub repository.

```
pc1@workstation:~/CPE212_Mamaril/Lab4$
pc1@workstation:~/CPE212_Mamaril/Lab4$ git add install_apache.yml
pc1@workstation:~/CPE212_Mamaril/Lab4$ git commit -m "first ansible playbook"
[main b516594] first ansible playbook
 1 file changed, 16 insertions(+)
 create mode 100644 Lab4/install_apache.yml
pc1@workstation:~/CPE212_Mamaril/Lab4$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 5 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 565 bytes | 565.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:JustinMamaril/CPE212_Mamaril.git
   3c8c4ec..b516594  main -> main
```

git@github.com:JustinMamaril/CPE212_Mamaril.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
    - Playbook enables us to simplify and automate the chores required when maintaining servers. As seen in this activity, manually typing commands in the terminal can be time-consuming and inefficient for maintaining servers. We may use playbooks to design tasks and even include particular conditions that must be executed alongside the task, as well as identify the hosts that the playbook will target. This simplifies server management and makes it much more efficient.
2. Summarize what we have done on this activity.

    - In this activity, I set up my nodes for Ansible automation by including the essential files, such as ansible.cfg and inventory.yaml. After creating the ansible file and adding the relevant IP addresses to the inventory file, we attempted to install packages on the distant servers by manually running the instructions in the terminal. Following that, I attempted an easier technique of conducting tasks directed to distant servers by creating a script. In my playbook, I was able to create numerous tasks and designate which hosts the playbook should be directed to, demonstrating the benefits of using.yml files for server management and automation. After completing the activity, I committed and pushed all of the produced files to my github repository.