Name: Agpaoa, Ma.Diane J.	**Date Performed:** 10/4/2022
Course/Section: CPE232-CPE31S22	**Date Submitted:** 10/5/2022
Instructor: Dr. Jonathan Taylar	**Semester and SY:** 1st Sem 2022-2023

Activity 6: Targeting Specific Nodes and Managing Services

1. **Objectives:**

1.1 Individualize hosts

1.2 Apply tags in selecting plays to run

1.3 Managing Services from remote servers using playbooks

2. **Discussion**:

In this activity, we try to individualize hosts. For example, we don't want apache on all our servers, or maybe only one of our servers is a web server, or maybe we have different servers like database or file servers running different things on different categories of servers and that is what we are going to take a look at in this activity.

We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.

Requirement:

In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command *ssh-copy-id* to copy the public key to Server 3. Verify if you can successfully SSH to Server 3.

Task 1: Targeting Specific Nodes

1. Create a new playbook and named it site.yml. Follow the commands as shown in the image below. Make sure to save the file and exit.

```
---

- hosts: all
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "CentOS"
```

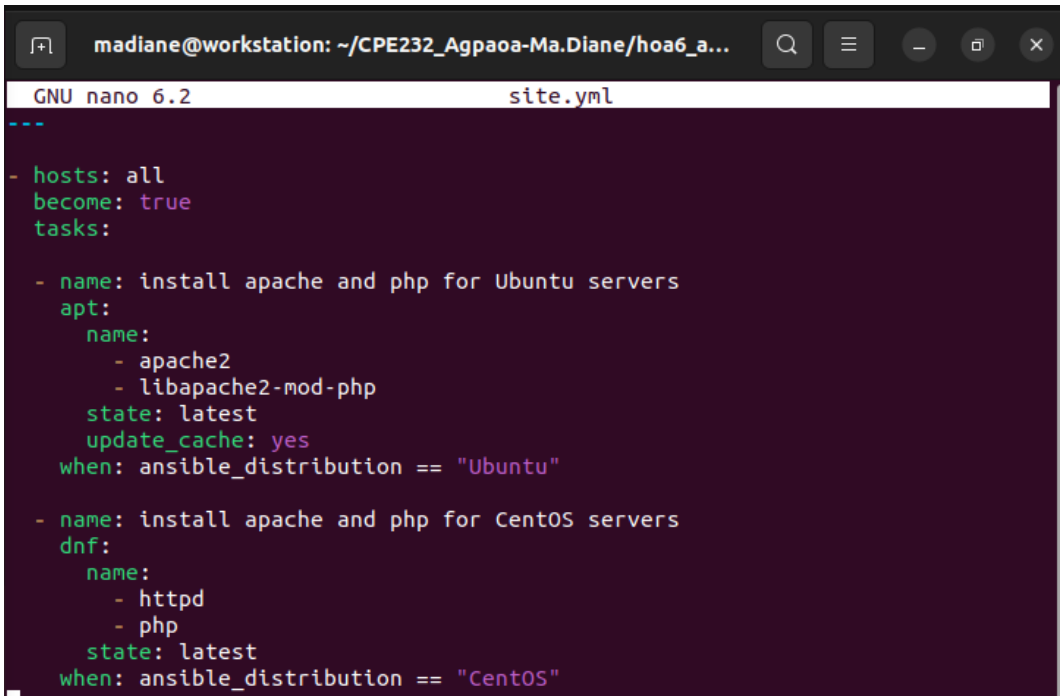

Figure 1.1 Creating the new playbook named as site.yml

2. Edit the inventory file. Remove the variables we put in our last activity and group according to the image shown below:

```
[web_servers]
192.168.56.120
192.168.56.121

[db_servers]
192.168.56.122


[file_servers]
192.168.56.123
```

Make sure to save the file and exit.

Figure 1.2 Grouping the remote servers according to the image shown in Task 1 step 2

Right now, we have created groups in our inventory file and put each server in its own group. In other cases, you can have a server be a member of multiple groups, for example you have a test server that is also a web server.

3. Edit the *site.yml* by following the image below:

```
---

- hosts: all
  become: true
  pre_tasks:

  - name: install updates (CentOS)
    dnf:
      update_only: yes
      update_cache: yes
    when: ansible_distribution == "CentOS"

  - name: install updates (Ubuntu)
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"


- hosts: web_servers
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "CentOS"
```

Make sure to save the file and exit.

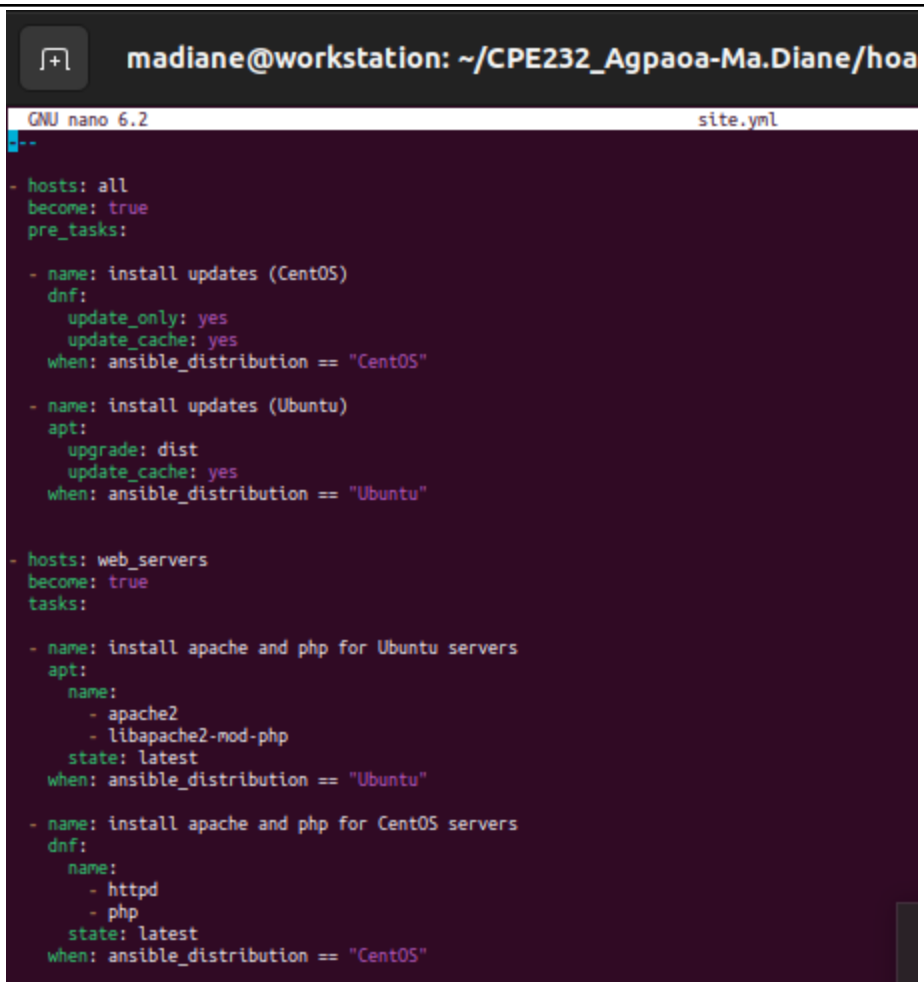

Figure 1.3 Changed the site.yml based on the shown image in Task 1 step 3

The *pre-tasks* command tells the ansible to run it before any other thing. In the *pre-tasks*, CentOS will install updates while Ubuntu will upgrade its distribution package. This will run before running the second play, which is targeted at *web_servers*. In the second play, apache and php will be installed on both Ubuntu servers and CentOS servers.

Run the *site.yml* file and describe the result.

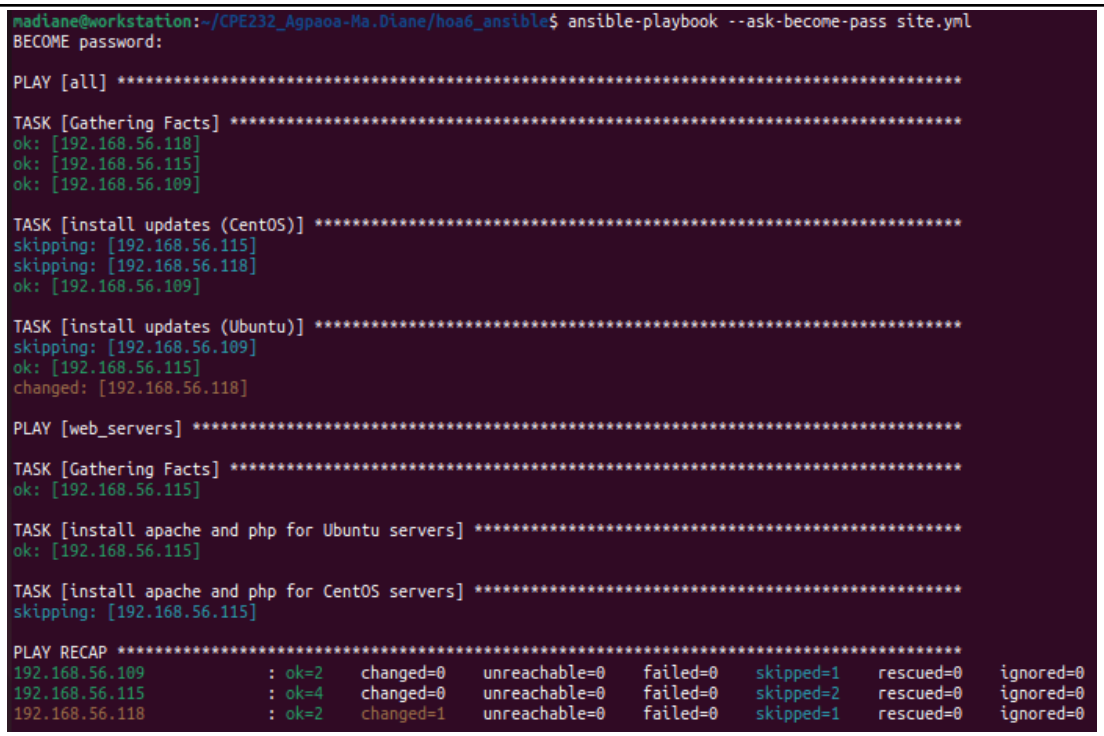

Figure 1.4 Executing the playbook site.yml

I used 3 remote servers to manage so I can open it all at the same time without causing lag to the device. Based on the result shown in the figure, the first task skipped the remote servers without the operating system of CentOS (192.168.56.115 and 192.168.56.118).The second task skipped the remote server with the operating system of CentOS (192.168.56.109) and updated the remote servers with the operating system of Ubuntu, however it didn't show any changes to the remote server of Ubuntu live server (192.168.56.115). The play in web_servers group has also 2 tasks, the first one is to install the apache and php for Ubuntu servers, although the remote server with the operating system of Ubuntu live server (192.168.56.118) fit the requirements it wasn't grouped with the remote server with the Ubuntu desktop (192.168.56.115). The second task was to install apache and php for CentOS server, it only included the remote server with Ubuntu desktop (192.168.56.115) and since it didn't fit the condition it was skipped. In addition, although the task is for the update of CentOS, the remote server with Ubuntu desktop (192.168.56.115) is the only remote server in the group of web_servers, however if I use 4 remote servers the CentOS would not be skipped and will have an ok state.

4. Let's try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db_servers* by adding it on

the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3.

```
- hosts: db_servers
  become: true
  tasks:

  - name: install mariadb package (CentOS)
    yum:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true

  - name: install mariadb packege (Ubuntu)
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "Ubuntu"
```

Make sure to save the file and exit.

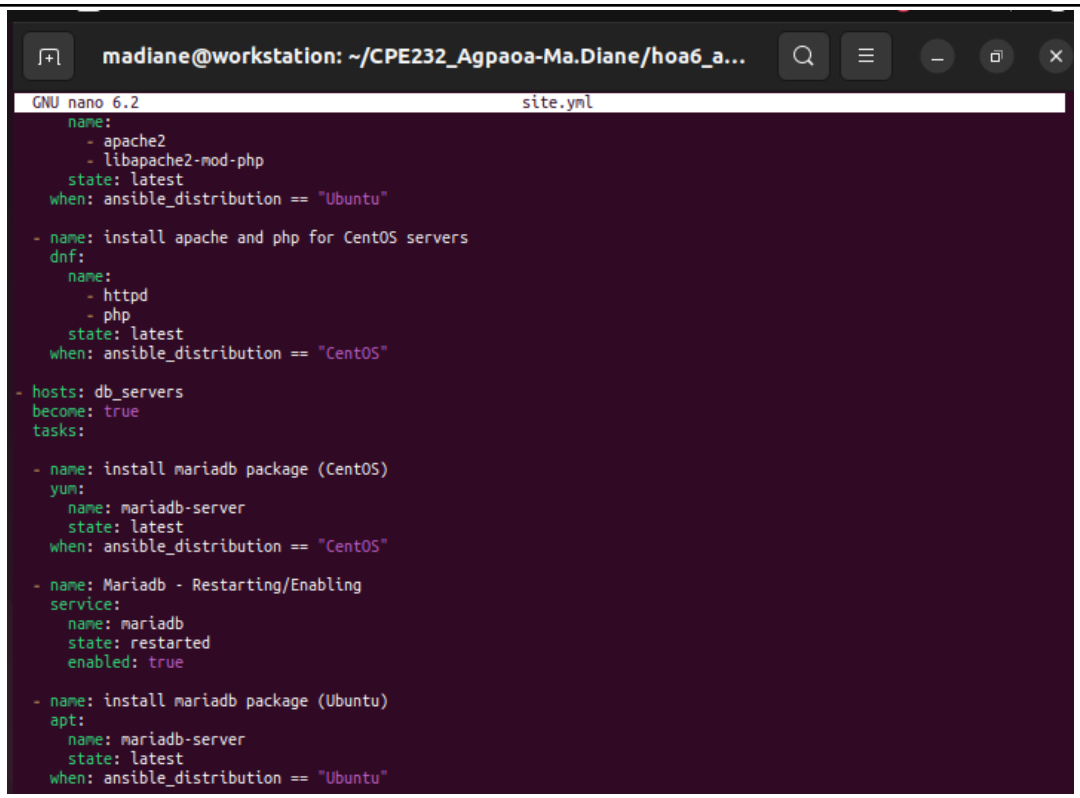

Figure 1.5 Adding the code to the site.yml shown in the Task 1 Step 3

Run the *site.yml* file and describe the result.

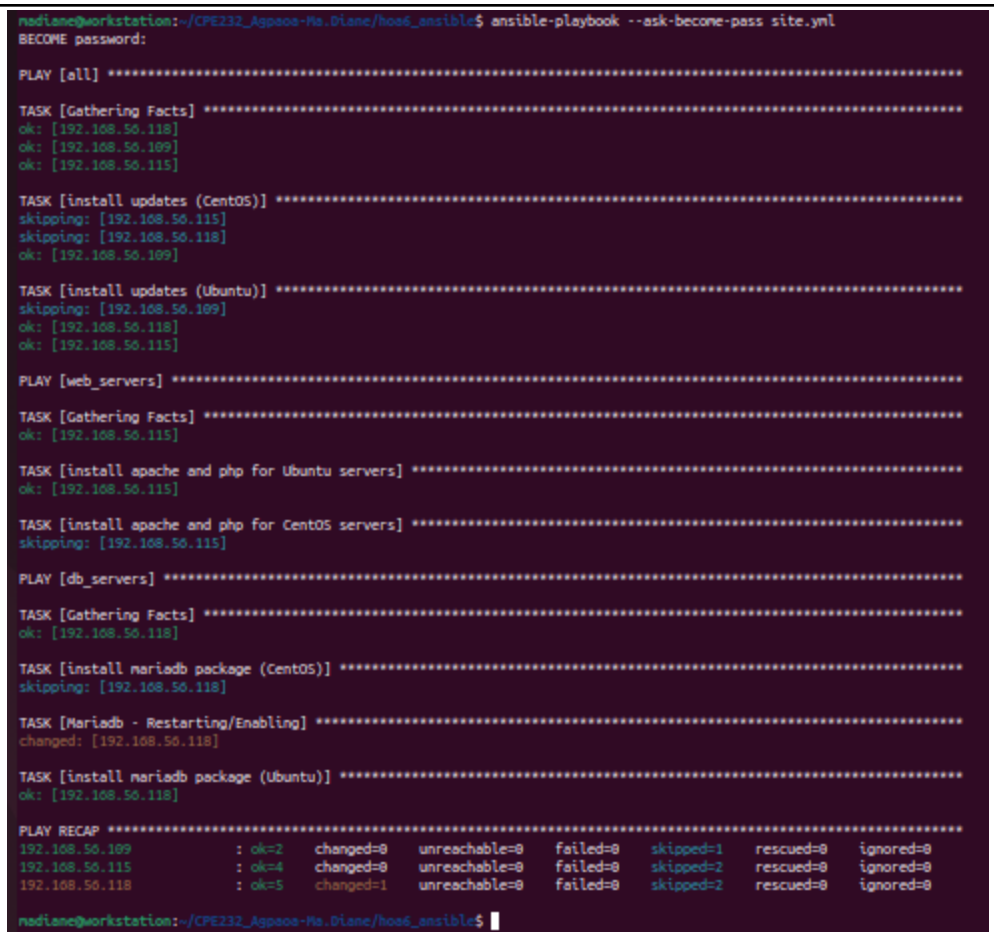

Figure 1.6 Executing the playbook site.yml

Based on the results, the first task on play of db_servers is the Mariadb - Restarting/Enabling, the remote server in the group have completed its task on the remote server (192.168.56.118), the second task of installing the mariadb package in CentOS skipped the remote server (192.168.56.118) because it didn't meet the condition, the thirds task install mariadb package in Ubuntu has an ok state. We could notice that the only remote server in the these tasks is the remote server 192.168.56.118 which is the remote server within the group db_servers.

5. Go to the remote server (Ubuntu) terminal that belongs to the db_servers group and check the status for mariadb installation using the command: *systemctl status mariadb.* Do this on the CentOS server also.

   Describe the output.

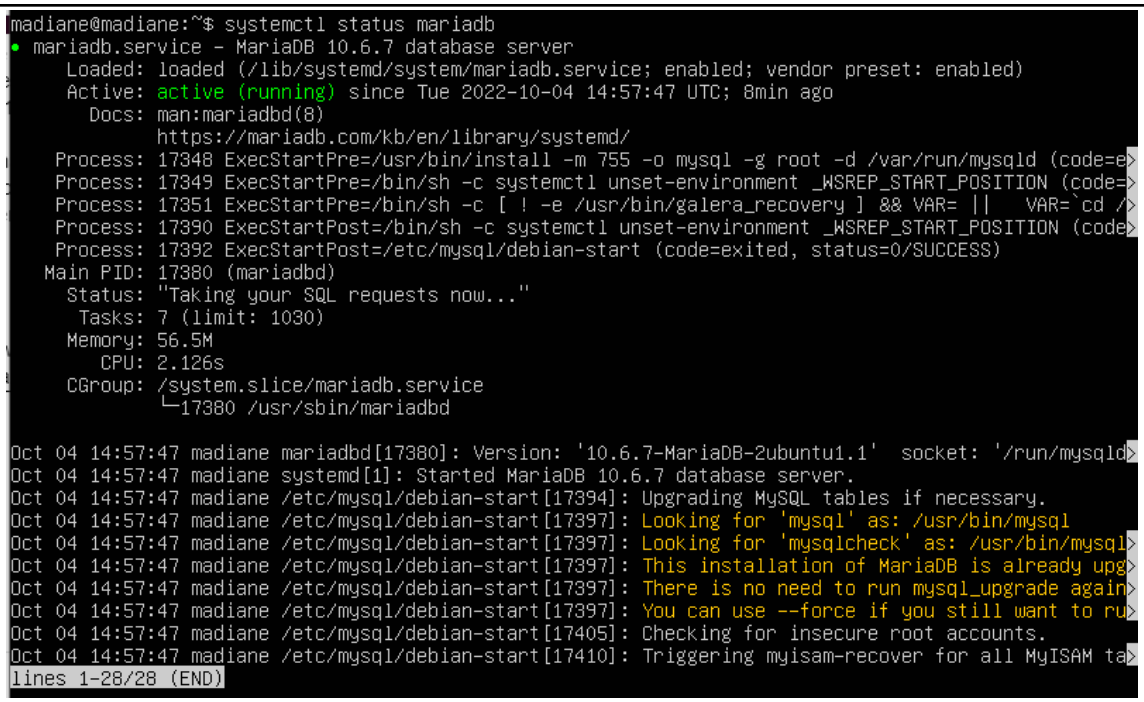

Figure 1.7 Executing the command systemctl status mariadb on a remote servers that belong to the db_servers

```
[madiane@localhost ~]$ systemctl status mariadb
Unit mariadb.service could not be found.
```

Figure 1.8 Executing the command systemctl status mariadb on CentOS

6. Edit the *site.yml* again. This time we will append the code to configure installation on the *file_servers* group. We can add the following on our file.

```
- hosts: file_servers
  become: true
  tasks:

  - name: install samba package
    package:
      name: samba
      state: latest
```

Make sure to save the file and exit.

Run the *site.yml* file and describe the result.

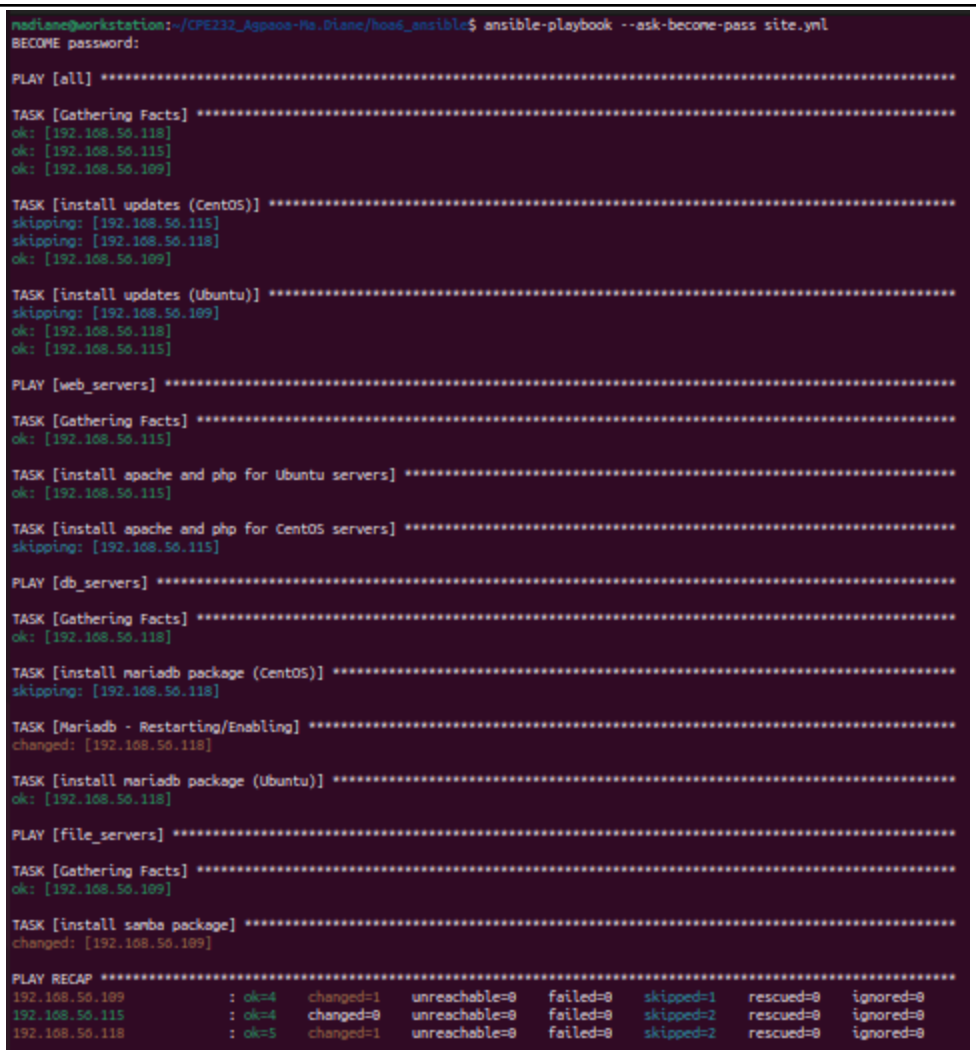

Figure 1.9 Executing the playbook site.yml

The samba package would be installed in the remote server that belongs to the file_servers group which is the remote server with the ip address of 192.168.56.109.

The testing of the *file_servers* is beyond the scope of this activity, and as well as our topics and objectives. However, in this activity we were able to show that we can target hosts or servers using grouping in ansible playbooks.

Task 2: Using Tags in running playbooks

In this task, our goal is to add metadata to our plays so that we can only run the plays that we want to run, and not all the plays in our playbook.

1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.

```
---

- hosts: all
  become: true
  pre_tasks:

  - name: install updates (CentOS)
    tags: always
    dnf:
      update_only: yes
      update_cache: yes
    when: ansible_distribution == "CentOS"

  - name: install updates (Ubuntu)
    tags: always
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"
```

```
- hosts: web_servers
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    tags: apache,apache2,ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    tags: apache,centos,httpd
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "CentOS"
```

```
- hosts: db_servers
  become: true
  tasks:

  - name: install mariadb package (CentOS)
    tags: centos, db,mariadb
    dnf:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true

  - name: install mariadb packege (Ubuntu)
    tags: db, mariadb,ubuntu
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "Ubuntu"

- hosts: file_servers
  become: true
  tasks:

  - name: install samba package
    tags: samba
    package:
      name: samba
      state: latest
```

Make sure to save the file and exit.

Figure 2.1. Adding tags in the site.yml

Run the *site.yml* file and describe the result.

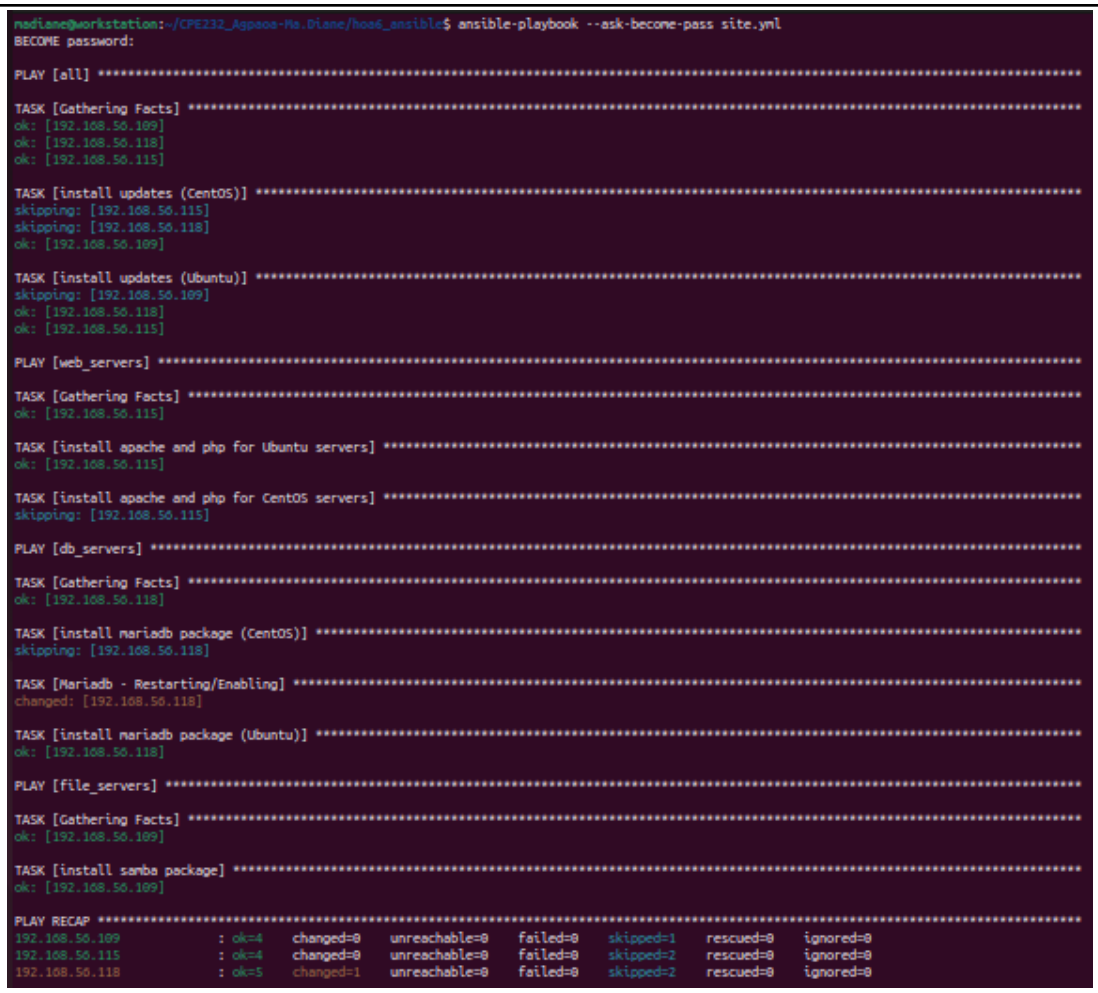

Figure 2.2. Executing the site.yml

The samba would be installed in the remote servers in the file_servers group. In this case, the remote server is 192.168.56.109.

2. On the local machine, try to issue the following commands and describe each result:

   2.1 ansible-playbook --list-tags site.yml

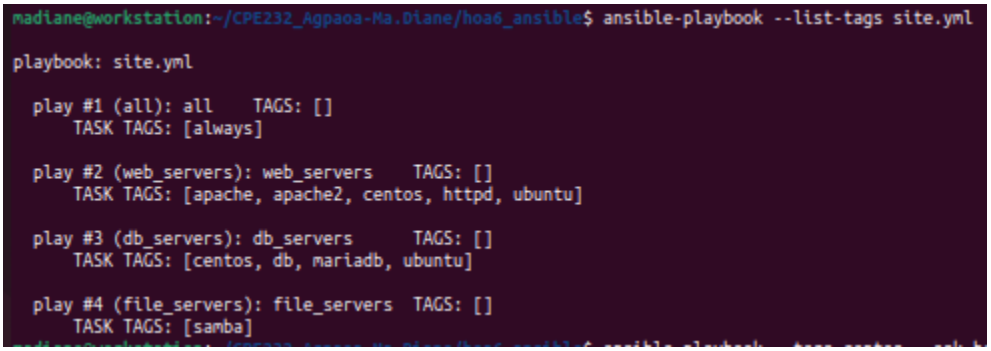

Figure 2.3. Executing the command ansible-playbook –list-tags site.yml

This command will list all the play with tasks and the tags it contains.

2.2 ansible-playbook --tags centos --ask-become-pass site.yml

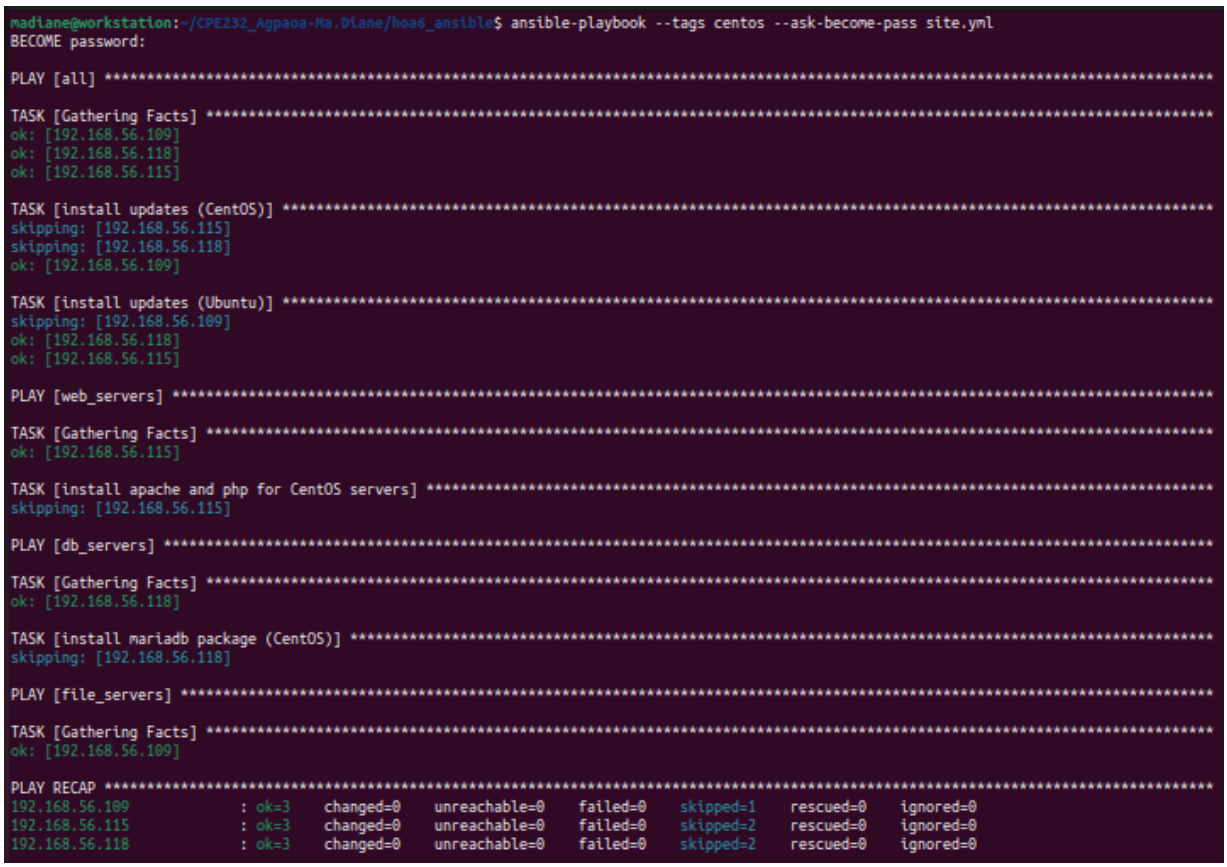

Figure 2.4 Executing the command ansible-playbook –tags centos –ask-become-pass site.yml

This command will execute the tasks with tags of centos. In addition, the pre_tasks will still run because it was configured to always run.

2.3 ansible-playbook --tags db --ask-become-pass site.yml

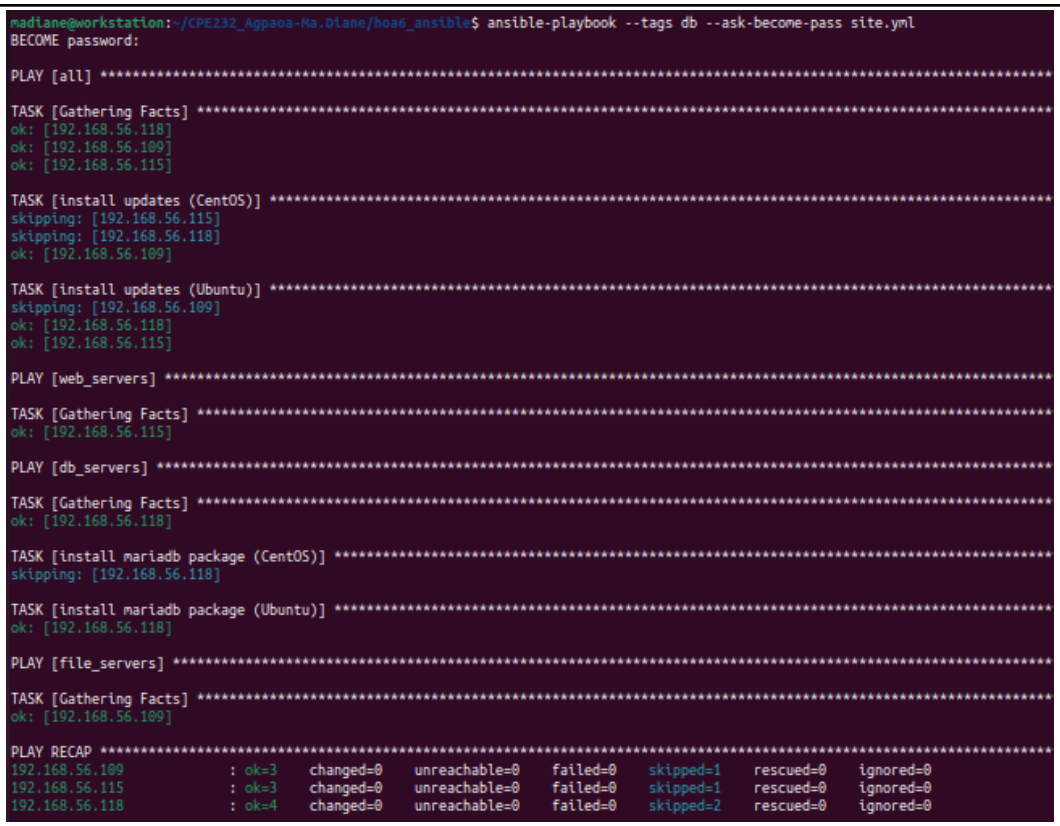

Figure 2.5 Executing the command ansible-playbook –tags db –ask-become-pass site.yml

This command will execute the tasks with tags of db. In addition, the pre_tasks will still run because it was configured to always run.

2.4 ansible-playbook --tags apache --ask-become-pass site.yml

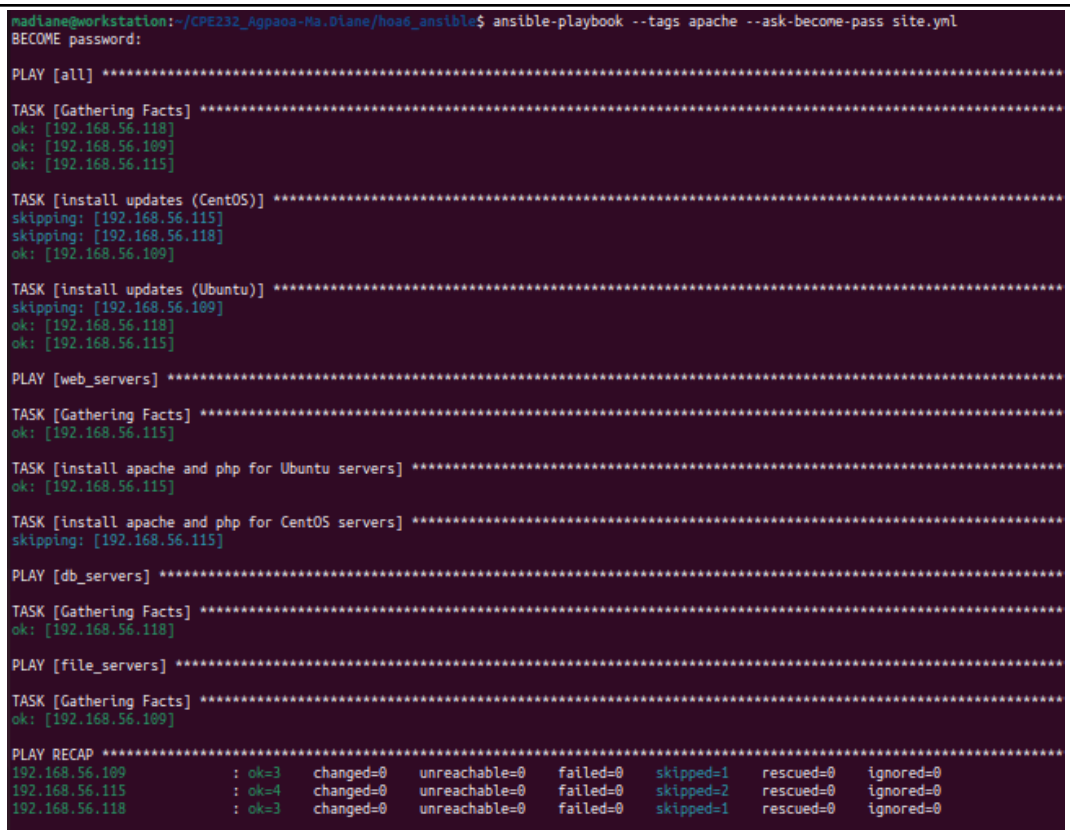

Figure 2.6 Executing-playbook–tags apache –ask-become-pass site.yml

This command will execute the tasks with tags of apache. In addition, the pre_tasks will still run because it was configured to always run.

2.5 *ansible-playbook --tags “apache,db” --ask-become-pass site.yml*

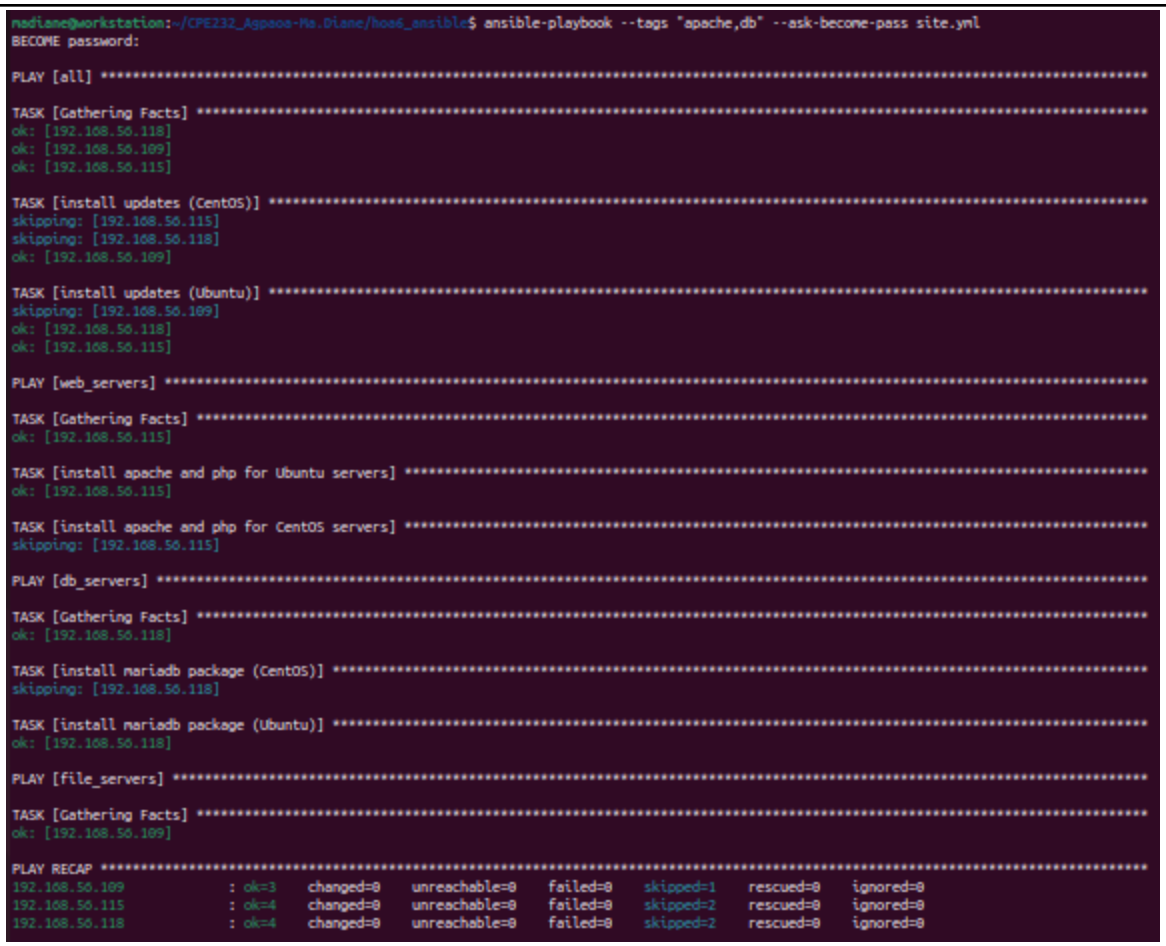

Figure 2.7 Executing the command ansible-playbook “apache,db” –ask-become-pass site.yml

This command will execute the tasks with apache and db. In addition, the pre_tasks will still run because it was configured to always run.

Task 3: Managing Services

1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.

```
- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"

- name: start httpd (CentOS)
  tags: apache, centos,httpd
  service:
    name: httpd
    state: started
  when: ansible_distribution == "CentOS"
```

Figure 3.1.1

Make sure to save the file and exit.

```
- name: start httpd (CentOS)
  tags: apache, centos, httpd
  service:
    name: httpd
    state: started
  when: ansible_distribution == "CentOS"
```

Figure 3.1. Adding a code in the site.yml based on the image shown in Task 3 Step 1

You would also notice from our previous activity that we already created a module that runs a service.

```
- hosts: db_servers
  become: true
  tasks:

  - name: install mariadb package (CentOS)
    tags: centos, db,mariadb
    dnf:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true
```

Figure 3.1.2

This is because in CentOS, installed packages' services are not run automatically. Thus, we need to create the module to run it automatically.

2. To test it, before you run the saved playbook, go to the CentOS server and stop the currently running httpd using the command *sudo systemctl stop httpd.* When prompted, enter the sudo password. After that, open the browser and enter the CentOS server's IP address. You should not be getting a display because we stopped the httpd service already.

```
[madiane@localhost ~]$ sudo systemctl stop httpd
[sudo] password for madiane:
[madiane@localhost ~]$
```

Figure 3.2 Stopping the running httpd on CentOS

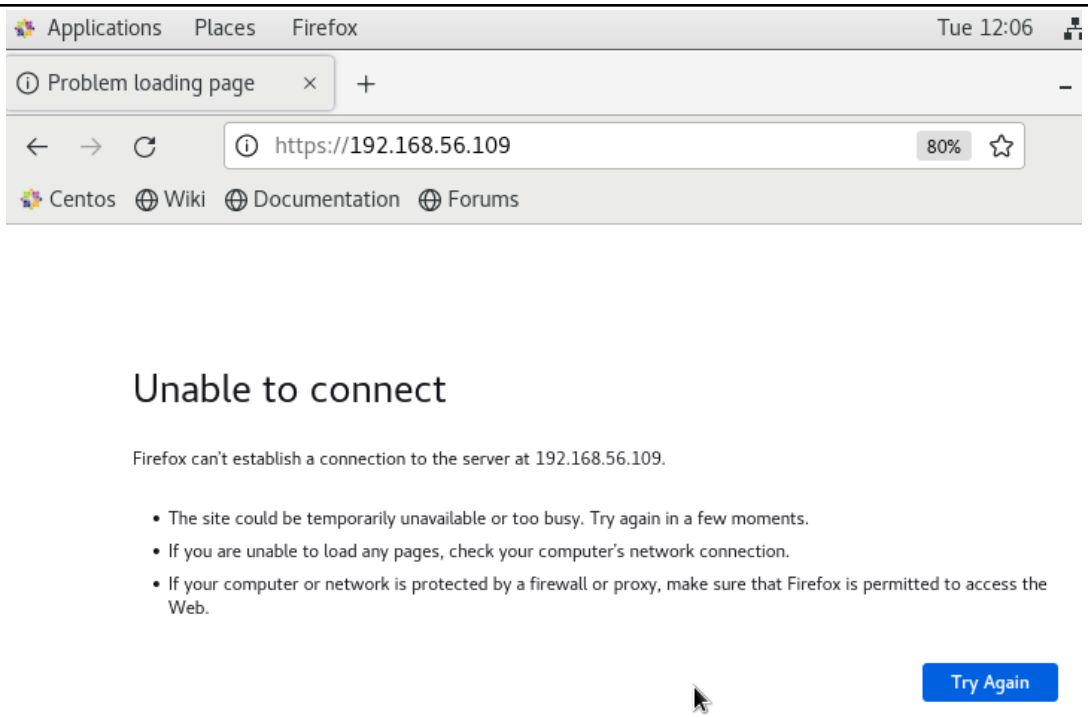

Figure 3.3 Verifying that the httpd is already stopped

3. Go to the local machine and this time, run the *site.yml* file. Then after running the file, go again to the CentOS server and enter its IP address on the browser. Describe the result.
   To automatically enable the service every time we run the playbook, use the command *enabled: true* similar to Figure 7.1.2 and save the playbook.

```
    when: ansible_distribution == "CentOS"

- name: start httpd (CentOS)
  tags: apache, centos, httpd
  service:
    name: httpd
    state: started
    enabled: true
  when: ansible_distribution == "CentOS"
```

Figure 3.4 Added a code based on the shown image in the Task 3 Step 3

Figure 3.5. Executing the site.yml

Based on the result, the activation of httpd did not work. This is because the code was added under the web_servers group. The remote server with the CentOS does not belong in that group, therefore running the httpd code will not work.

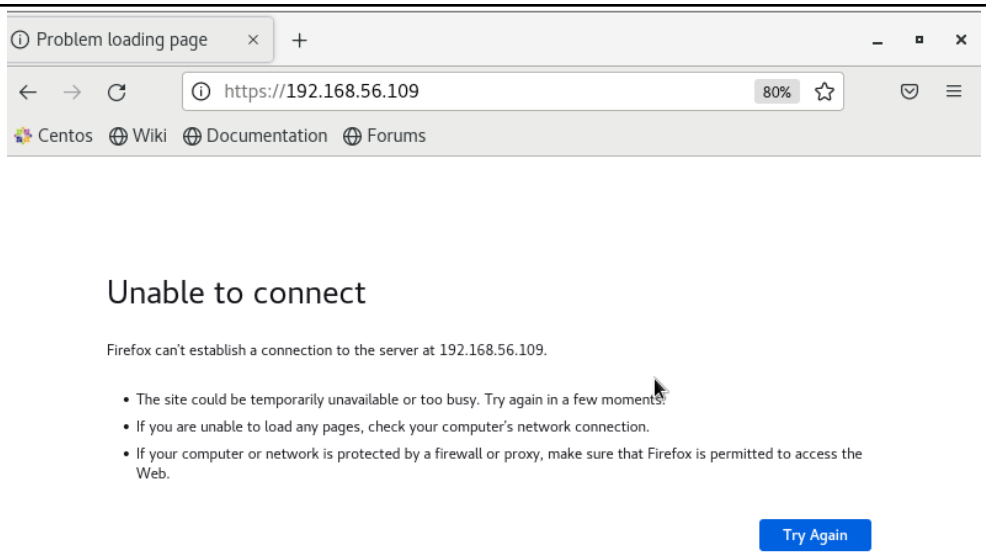

Figure 3.6 Verifying the if the httpd start running

Based on the result, the httpd will not start running using the code given in the Task 3 step 3.

Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?
   Grouping the remote servers is important in order to make custom changes based on their required configurations. It is also easier to debug and make changes because the remote servers that have the same configuration are in the same group.

2. What is the importance of tags in playbooks?
   The tags are important in playbooks because the administrator or the user can easily play a specific task based on the tags that the user assigned in each task.

3. Why do you think some services need to be managed automatically in playbooks?

   I think some services need to be managed automatically in playbooks because some of these services' configurations are repetitive, require a lot of effort and take a lot of time unnecessarily. Managing these services automatically using playbooks will increase work efficiency and it will be easier to debug errors because the source of error could be narrowed down from the playbooks that the user used.