Name: Mamaril, Justin Kenneth I.	Date Performed: 09/12/25
Course/Section: CPE 212-CPE31S2	Date Submitted: 09/12/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2025-2026

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

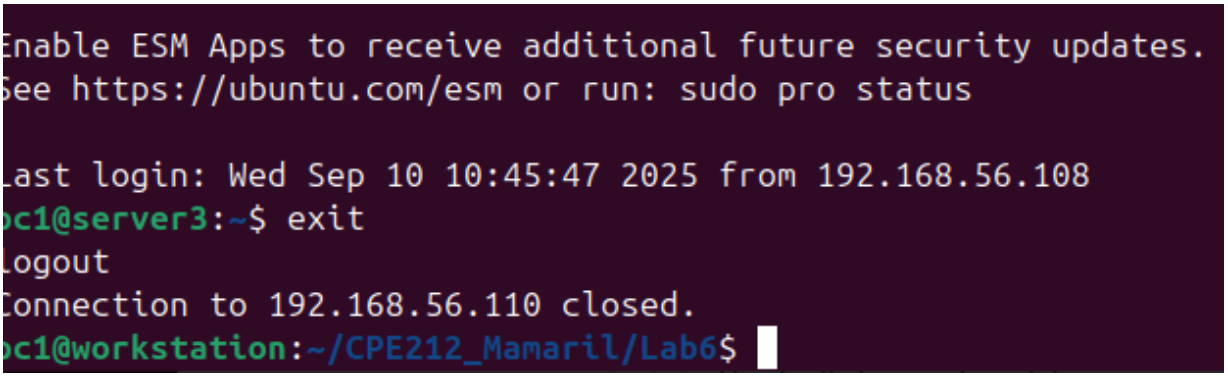
2. **Discussion**:

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.

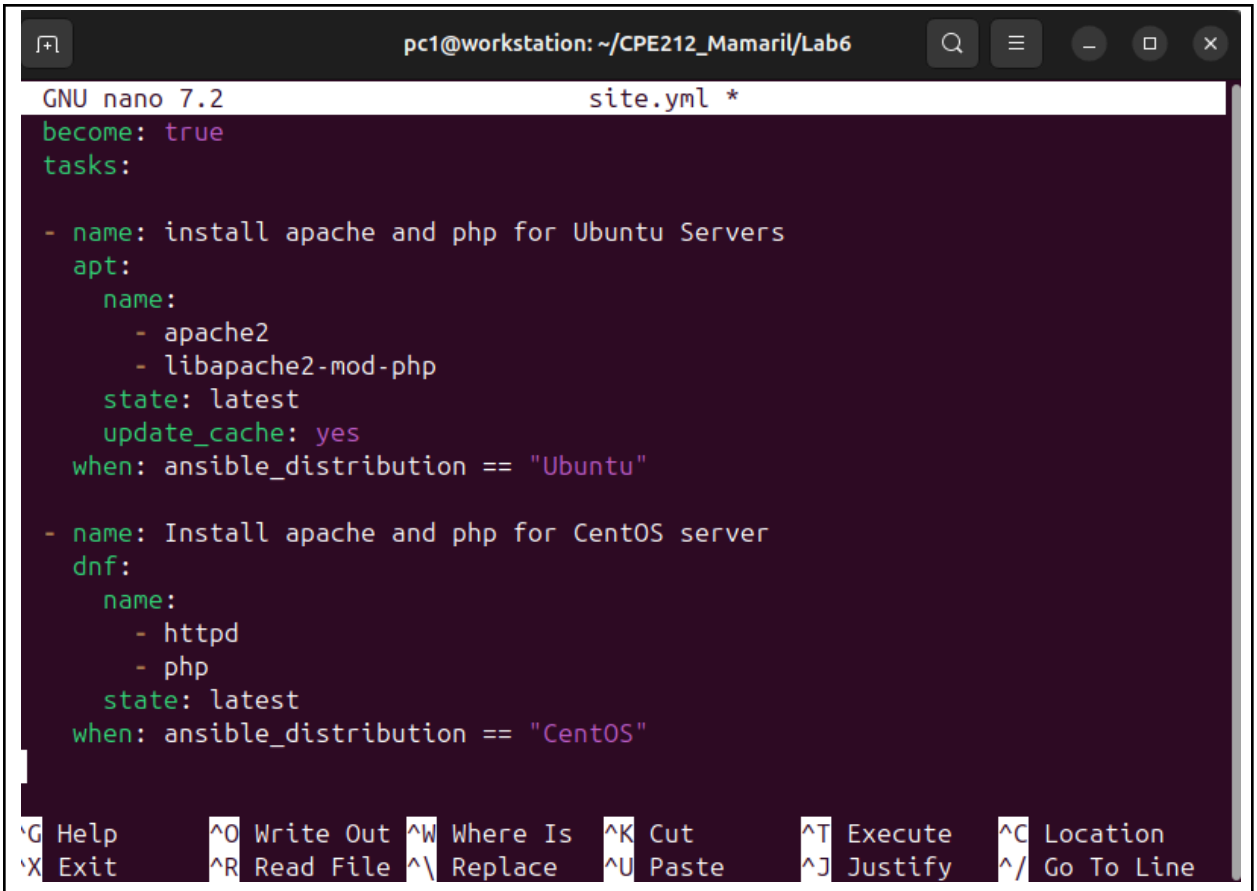


Task 1: Targeting Specific Nodes

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.

```
pc1@workstation:~/CPE212_Mamaril$ mkdir Lab6
pc1@workstation:~/CPE212_Mamaril$ cd Lab6
pc1@workstation:~/CPE212_Mamaril/Lab6$ sudo nano site.yml
[sudo] password for pc1:
```

```
---

- 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"
```



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

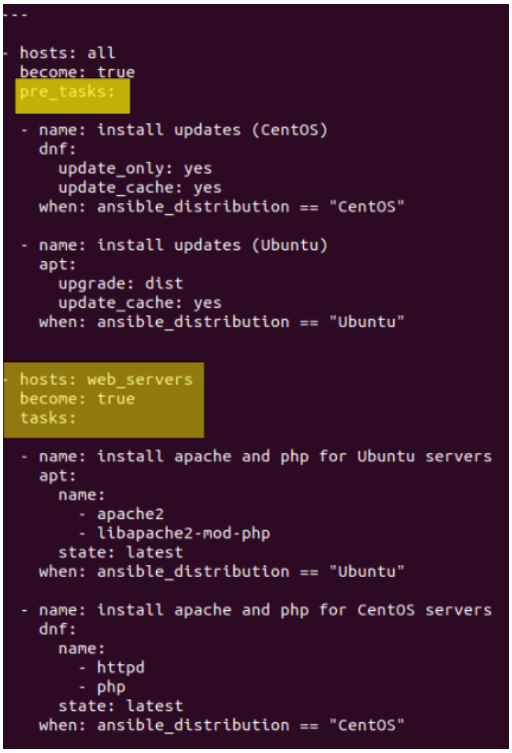
```
[web_servers]
192.168.56.120
192.168.56.121

[db_servers]
192.168.56.122


[file_servers]
192.168.56.123
```

Make sure to save the file and exit.

```
pc1@workstation: ~/CPE212_Mamaril/Lab6
  GNU nano 7.2                                inventory.yml
[web_servers]
192.168.56.108
192.168.56.110

[db_servers]
192.168.56.106

[file_servers]
192.168.56.107
```

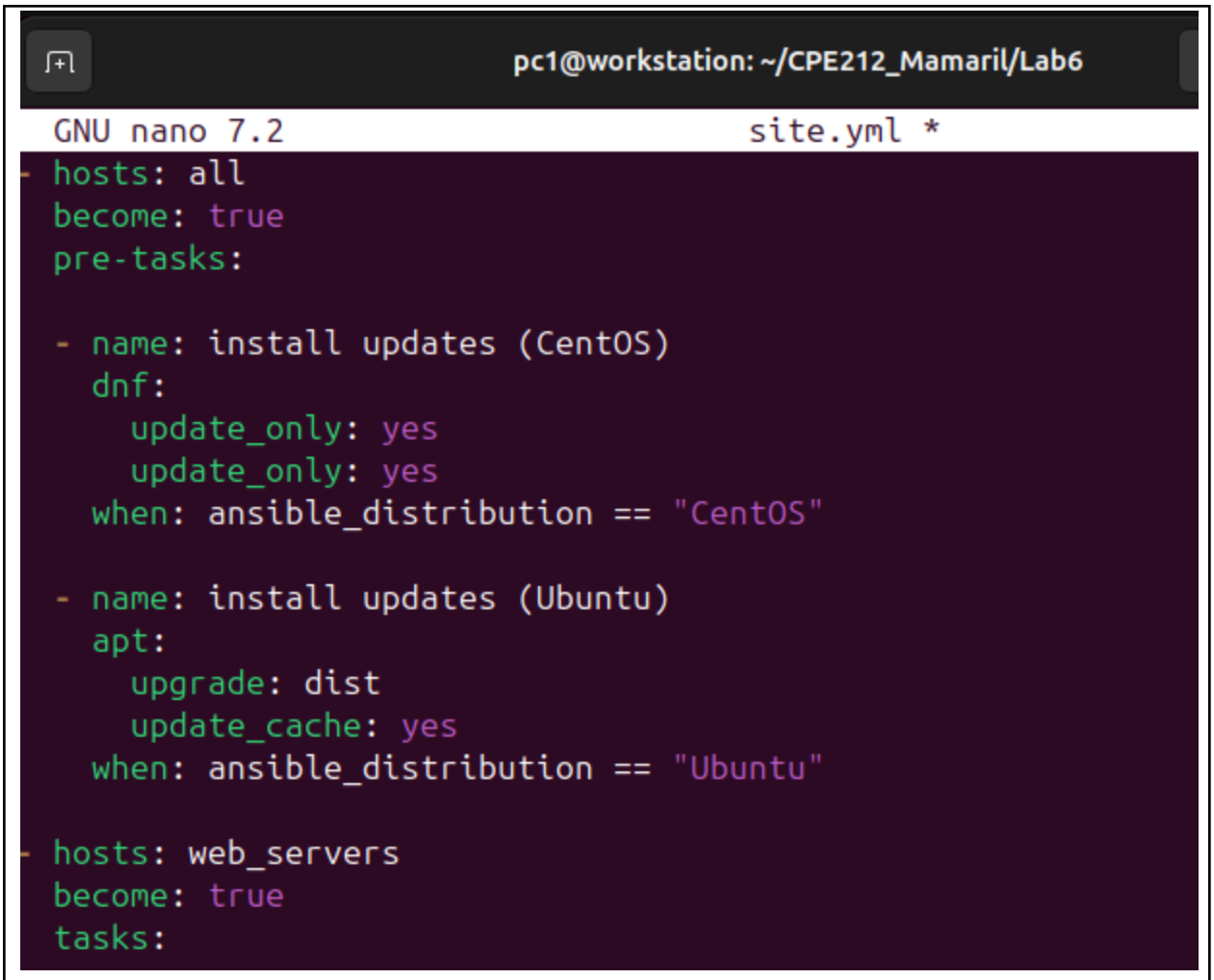
Right now, we have created groups in our inventory file and put each server in its own group. In other cases, you can have a server be a member of multiple groups, for example you have a test server that is also a web server.

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



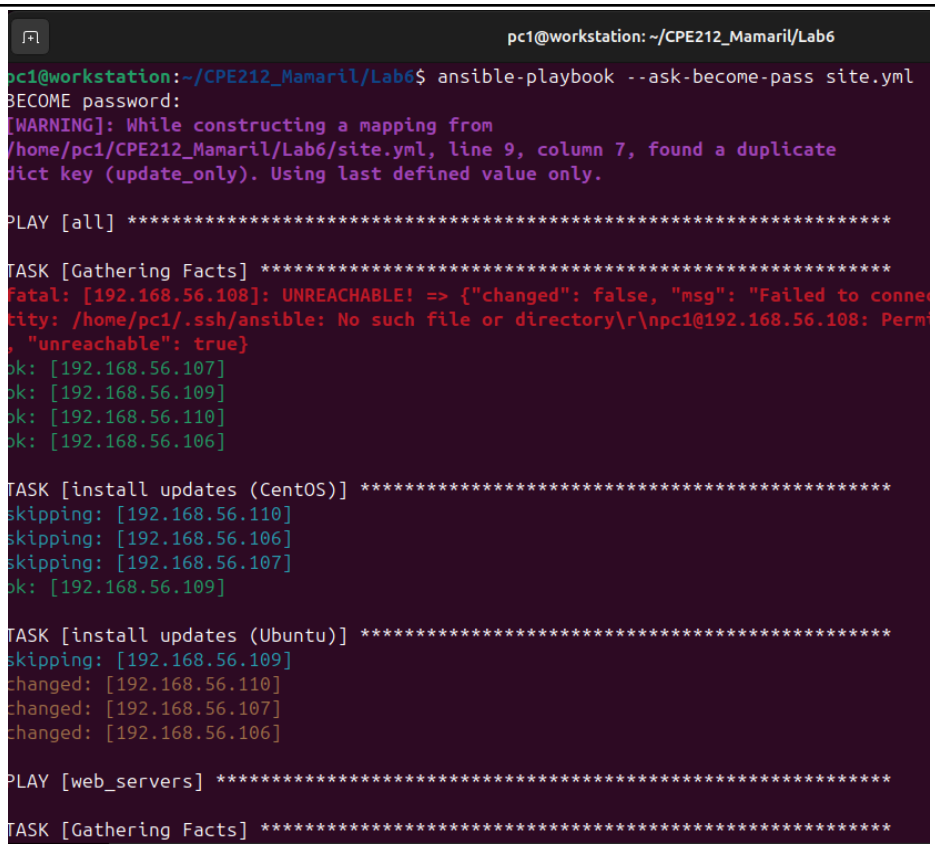
```
---

- 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.



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.



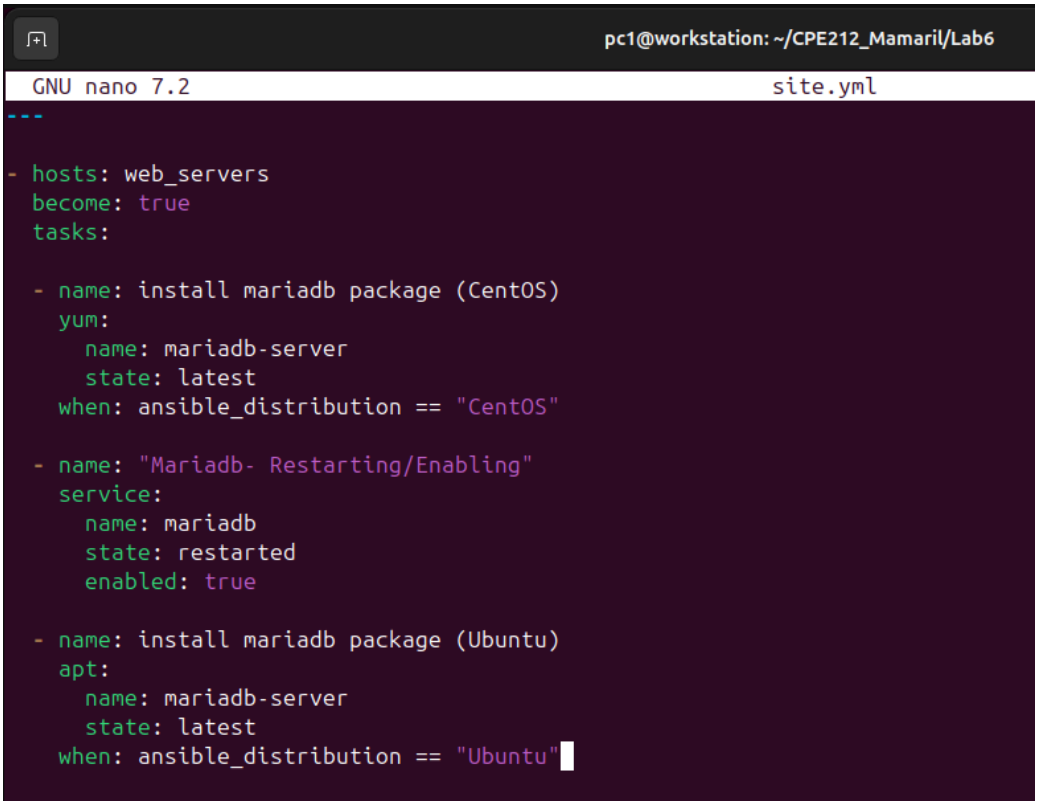

```
TASK [Gathering Facts] *********************************************************
ok: [192.168.56.110]

TASK [install apache and php for Ubuntu Servers] *******************************
ok: [192.168.56.110]

TASK [Install apache and php for CentOS server] ********************************
skipping: [192.168.56.110]

PLAY RECAP *********************************************************************
192.168.56.106             : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.107             : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.108             : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.109             : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.110             : ok=4    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

- The tasks that are within the all server worked. The task that is only for CentOS worked for centOS and Ubuntu for only Ubuntu. Then the other task that have the server web_server did what they had to do

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.



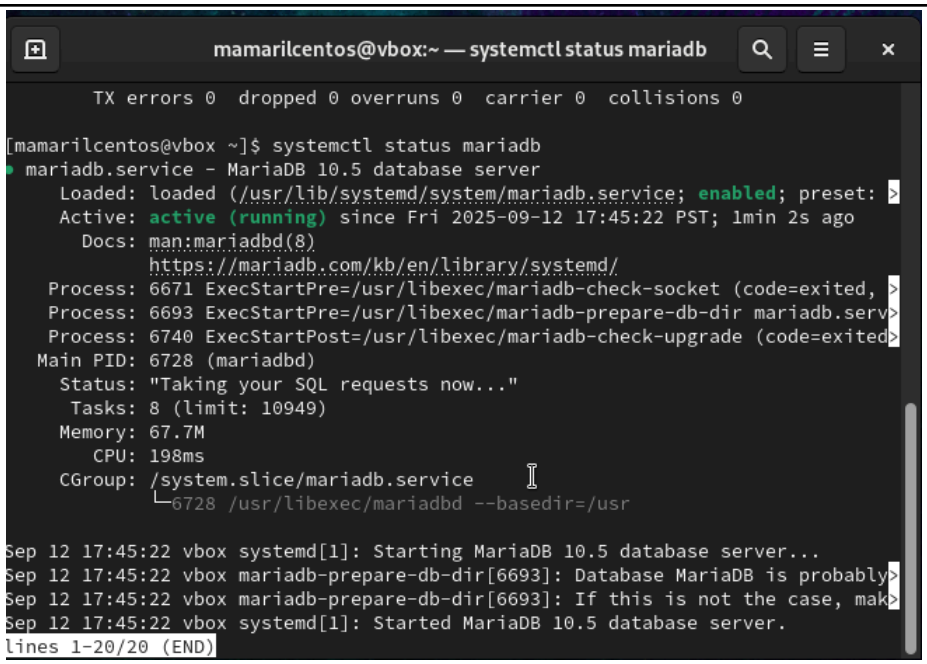
Run the *site.yml* file and describe the result.

```
pc1@workstation:~/CPE212_Mamaril/Lab6$ ansible-playbook --ask-become-pass site.yml
BECOME password:

PLAY [web_servers] ************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [192.168.56.106]
ok: [192.168.56.109]

TASK [install mariadb package (CentOS)] ***************************************************************************
skipping: [192.168.56.106]
ok: [192.168.56.109]

TASK [install mariadb package (Ubuntu)] ***************************************************************************
skipping: [192.168.56.109]
changed: [192.168.56.106]

TASK [Mariadb- Restarting/Enabling] *******************************************************************************
changed: [192.168.56.109]
changed: [192.168.56.106]

PLAY RECAP ********************************************************************************************************
192.168.56.106             : ok=3    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.109             : ok=3    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

- The mariadb was successfully installed on my Ubuntu and CentOS nodes and we can confirm it by continuing on number 5.

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.

```
pc1@server1:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.11.13 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: >
     Active: active (running) since Fri 2025-09-12 09:49:57 UTC; 6min ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 9473 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var>
    Process: 9476 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_ST>
    Process: 9478 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && >
    Process: 9553 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_S>
    Process: 9555 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/>
   Main PID: 9538 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 11 (limit: 30383)
     Memory: 78.6M (peak: 82.3M)
        CPU: 765ms
     CGroup: /system.slice/mariadb.service
```



Describe the output.

- Both of the mariadb servers were installed because it shows that it is active and running.

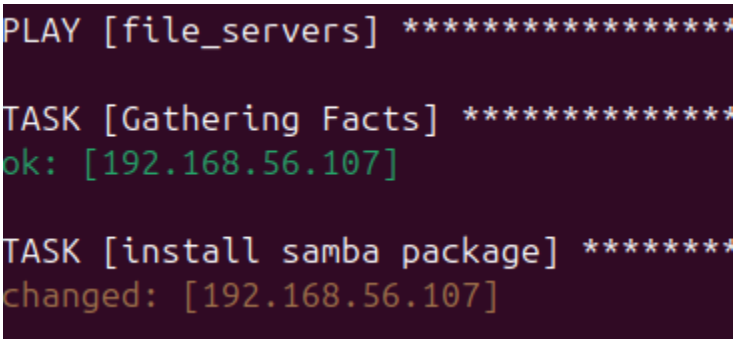
6. Edit the *site.yml* again. This time we will append the code to configure installation on the *file_servers* group. We can add the following on our file.

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

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

Make sure to save the file and exit.

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

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

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



- It successfully installed the samba package at my 2nd node

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.

```yaml
---

- 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"
```

```yaml
- 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
```

```
- name: install updates (CentOS)
  tags: always
```

```
- name: install updates (Ubuntu)
  tags: always
```

```
- name: install apache and php for Ubuntu servers
  tags: apache,apache2,ubuntu
  apt:
```

```
- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
```

```
- name: install mariadb package (CentOS)
  tags: centos, db,mariadb
  yum:
```

```
- name: install mariadb package (Ubuntu)
  tags: db, mariadb, ubuntu
  apt:
```

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

Make sure to save the file and exit.

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

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

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

```
pc1@workstation:~/CPE212_Mamaril/Lab6$ ansible-playbook --list-tags site.yml
[WARNING]: While constructing a mapping from /home/pc1/CPE212_Mamaril/Lab6/si
duplicate dict key (update_only). Using last defined value only.

playbook: site.yml

  play #1 (all): all    TAGS: []
      TASK TAGS: [always]

  play #2 (web_servers): web_servers    TAGS: []
      TASK TAGS: [apache, apache2, centos, db, httpd, mariadb, ubuntu]

  play #3 (file_servers): file_servers  TAGS: []
      TASK TAGS: [samba]
```

- *It shows each tag that we input in the yml file*

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

```
pc1@workstation:~/CPE212_Mamaril/Lab6$ ansible-playbook --tags centos --ask-become-pass site.yml
BECOME password:
[WARNING]: While constructing a mapping from /home/pc1/CPE212_Mamaril/Lab6/site.yml, line 10, column 7, found a
duplicate dict key (update_only). Using last defined value only.

PLAY [all] *********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.56.107]
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [install updates (CentOS)] ************************************************************************************
skipping: [192.168.56.106]
skipping: [192.168.56.107]

PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=3    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.107             : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.109             : ok=5    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

- *It only runs the tasks with tag "centos"*

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

```
PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.107             : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.109             : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

- *it only runs the tasks with tag "db"*

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

```
PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.107             : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.109             : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

- *it only runs the tasks with tag "apache"*

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

```
PLAY RECAP *********************************************************************************************************
192.168.56.106             : ok=5    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.107             : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.109             : ok=5    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0

pc1@workstation:~/CPE212_Mamaril/Lab6$ ssh mamarilcentos@192.168.56.109
```

- *it runs the tasks with tags "apache,db"*

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"
```

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

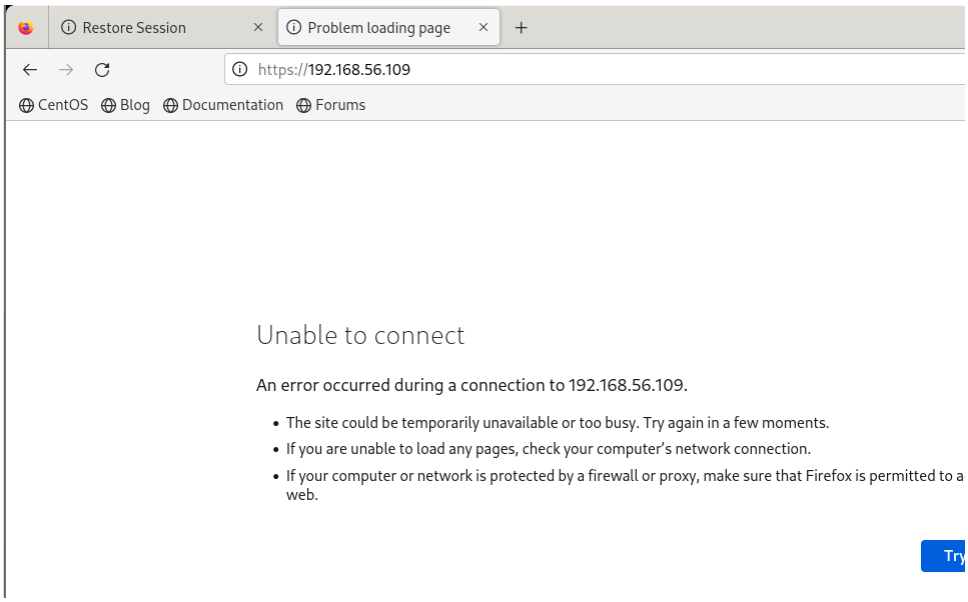
```
- hosts: db_servers
  become: true
  tasks:

  - 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.

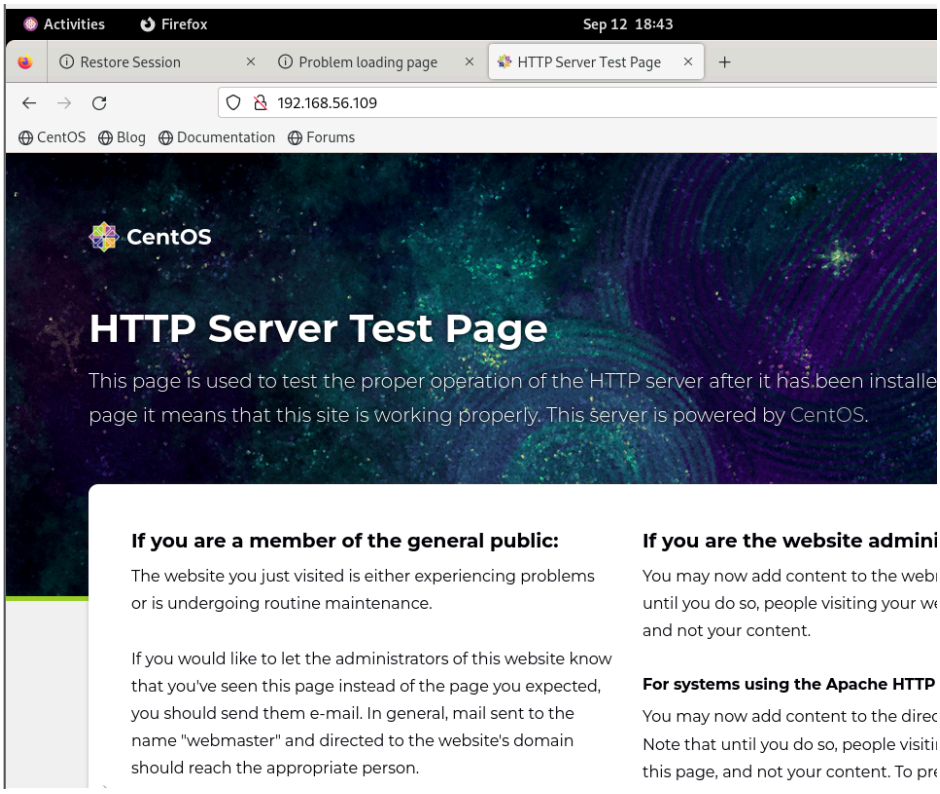
2. To test it, before you run the saved playbook, go to the CentOS server and stop the currently running httpd using the command *sudo systemctl stop httpd*. When prompted, enter the sudo password. After that, open the browser and enter the CentOS server's IP address. You should not be getting a display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

```
[mamarilcentos@vbox ~]$ sudo systemctl stop httpd
```



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.

```
TASK [start httpd (CentOS)] ****************************************************************
skipping: [192.168.56.106]
changed: [192.168.56.109]

TASK [install mariadb package (CentOS)] ****************************************************
skipping: [192.168.56.106]
ok: [192.168.56.109]

TASK [install mariadb package (Ubuntu)] ****************************************************
skipping: [192.168.56.109]
ok: [192.168.56.106]

TASK [Mariadb- Restarting/Enabling] ********************************************************
changed: [192.168.56.109]
changed: [192.168.56.106]

PLAY [file_servers] ************************************************************************

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

TASK [install samba package] ***************************************************************
ok: [192.168.56.107]

PLAY RECAP *********************************************************************************
192.168.56.106             : ok=6    changed=1    unreachable=0    failed=0    skipped=4
192.168.56.107             : ok=4    changed=0    unreachable=0    failed=0    skipped=1
192.168.56.109             : ok=7    changed=2    unreachable=0    failed=0    skipped=3
```



- The HTTP server test page now work because of the httpd automation service command we implemented

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.

Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?
   - The use of remote servers into groups is made simple because you can set configuration settings (such as port numbers or file paths) once to control a whole group of variables, thus eliminating redundancy and possible errors. Finally, server grouping can convert a complicated set of isolated machines into an easy to manage, organized, and audited system and automate your business more effectively and build a more reliable infrastructure.
2. What is the importance of tags in playbooks?
   - Ansible tags are an important element of controlling and managing activities. This will enable you to run or ignore specific parts of an enormous playbook by labeling individual tasks, or roles, or even entire plays with specific tags. This is so essential in enhancing efficiency especially in the process of development and debugging since you can test one thing at a time without necessarily watching the entire playbook run to completion.
3. Why do think some services need to be managed automatically in playbooks?
   - Playbooks require some services to be automatically managed as it is a core aspect of attaining idempotency and desired state configuration. Starting, stopping, or restarting services manually are likely to result in a human error, lack of uniformity across servers, and undesirable side effects.By using an automation tool like Ansible to manage a service, you declare the final state you want and the playbook's module will intelligently handle the necessary actions, such as starting a service only if it's currently stopped.