Course Code: CPE 212	**Program:** BSCPE
Course Title: AUTOMATING SERVER MANAGEMENT	**Date Performed:** 09/19/2025
Section: CPE31S4	**Date Submitted:** 09/19/2025
Name: Hazel Aillson T. Manuel	**Instructor:** Engr. Robin Valenzuel

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.

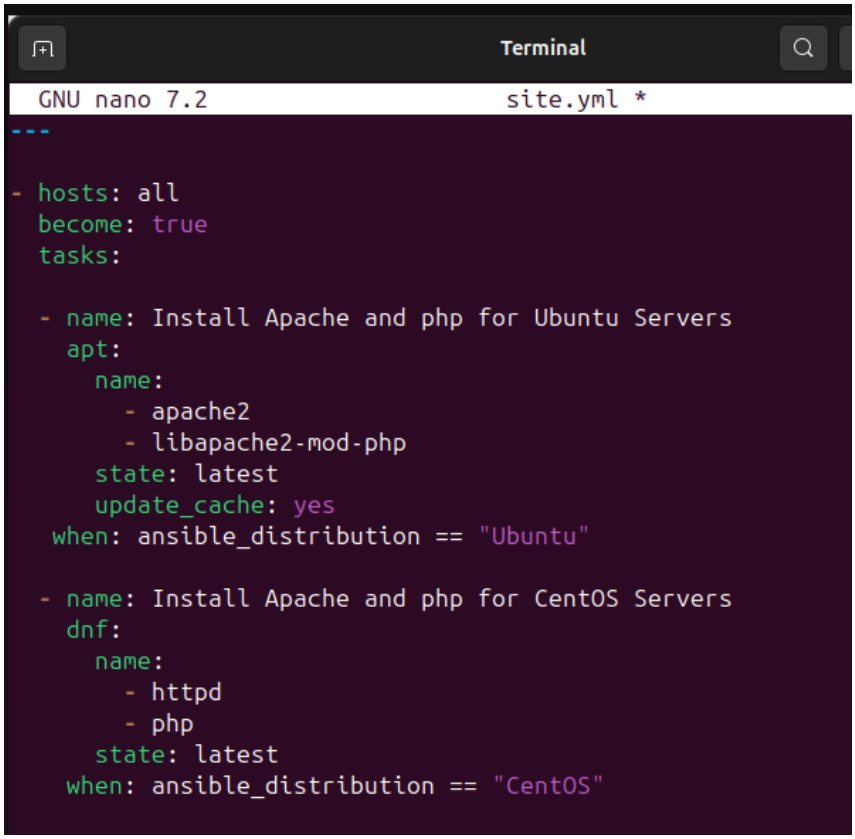
Requirement:

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"
```



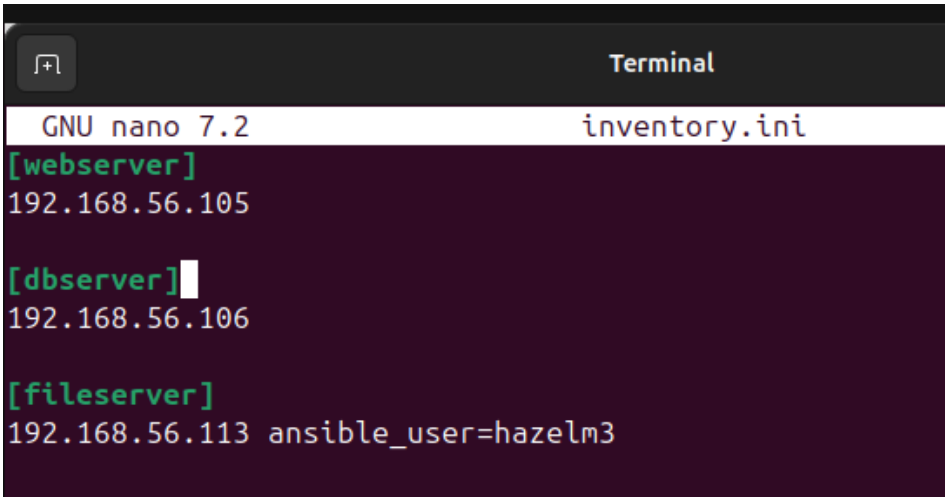
Creating a new playbook file called site.yml for installing apache and php. This will install to both Ubuntu and CentOS. Additionally, it ensures that the package is updated.

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.

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.



Modifying the inventory file where the webserver contains the Manage Node 1, dbserver contains the Manage node 2, and fileserver contains the CentOS 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.

```
---

- hosts: all
  become: true
  pre_tasks:

  - name: Install Updates (CentOS)
    dnf:
      update_only: yes
      update_cahce: yes
    when: ansible_distribution == "CentOS"

  - name: Install Updates (Ubuntu)
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

- hosts: webservers
  become: true
  tasks:
```

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

TASK [Gathering Facts] ************
ok: [192.168.56.105]
ok: [192.168.56.106]
ok: [192.168.56.114]
ok: [192.168.56.113]

TASK [Install Updates (CentOS)] ***
skipping: [192.168.56.105]
skipping: [192.168.56.114]
skipping: [192.168.56.106]
ok: [192.168.56.113]

TASK [Install Updates (Ubuntu)] ***
skipping: [192.168.56.113]
ok: [192.168.56.106]
ok: [192.168.56.114]
ok: [192.168.56.105]
```

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

TASK [Gathering Facts] ****************************
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for Ubuntu Servers] ***
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for CentOS Servers] ***
skipping: [192.168.56.105]
skipping: [192.168.56.114]
```

Updating and Running the ansible playbook. This creates a new task section for pre_tasks where it will install the packages in ubuntu and centos then proceeds to do the main task. However, as we set the task with only webserver nodes, this just executes the tasks with the IP addresses that are within the group. It skips when it starts doing the installation for CentOS Servers as the group does not have a CentOS Server in it.

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"
```

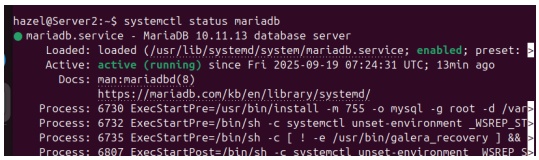
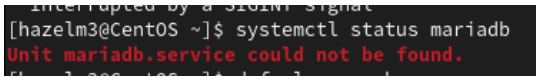
Make sure to save the file and exit.
Run the *site.yml* file and describe the result.

```
PLAY [dbserver] ********************************

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

TASK [Install MariaDB Package (CentOS)] ********
skipping: [192.168.56.106]

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

TASK [Install MariaDB package (Ubuntu)] ********
ok: [192.168.56.106]
```

Running the updated ansible-playbook shows that it skips the CentOS as the dbserver group only contains one IP address for Ubuntu Servers. Moreover, the second task worked as we didn't set any constraint like when: which means that this task did execute but in the IP address in dbserver. Lastly, no issues to the third one as it the server is Ubuntu and satisfies the condition when:.

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.

dbserver	CentOS server
	
This allow us to see that despite starting the mariadb in a local machine, it is unable to start it. Moreover, in the CentOS server, it shows that it can't be found as the IP address of it is not within the dbserver which skips the installation and starting.	

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.

   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.

```
PLAY [fileserver] ****************

TASK [Gathering Facts] ***********
ok: [192.168.56.113]

TASK [Install samba packagge] ****
changed: [192.168.56.113]
```

This shows that we are able to install a samba package successfully as we have no constraints on its distribution. This means all the servers within fileserver will execute this.

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

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

TASK [Gathering Facts] *************
ok: [192.168.56.106]
ok: [192.168.56.113]
ok: [192.168.56.114]
ok: [192.168.56.105]

TASK [Install Updates (CentOS)] ****
skipping: [192.168.56.105]
skipping: [192.168.56.114]
skipping: [192.168.56.106]
ok: [192.168.56.113]

TASK [Install Updates (Ubuntu)] ****
skipping: [192.168.56.113]
ok: [192.168.56.105]
ok: [192.168.56.106]
ok: [192.168.56.114]
```

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

TASK [Gathering Facts] ***************************
ok: [192.168.56.114]
ok: [192.168.56.105]

TASK [Install Apache and php for Ubuntu Servers] **
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for CentOS Servers] **
skipping: [192.168.56.114]
skipping: [192.168.56.105]
```

```
PLAY [dbserver] **********************************

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

TASK [Install MariaDB Package (CentOS)] *********
skipping: [192.168.56.106]

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

TASK [Install MariaDB package (Ubuntu)] *********
ok: [192.168.56.106]
```

```
PLAY [fileserver] ****************************

TASK [Gathering Facts] ***********************
ok: [192.168.56.113]

TASK [Install samba package] *****************
ok: [192.168.56.113]
```

Updating the ansible playbook again with tags, there is no visible changes upon running it as we did not specify anything yet.

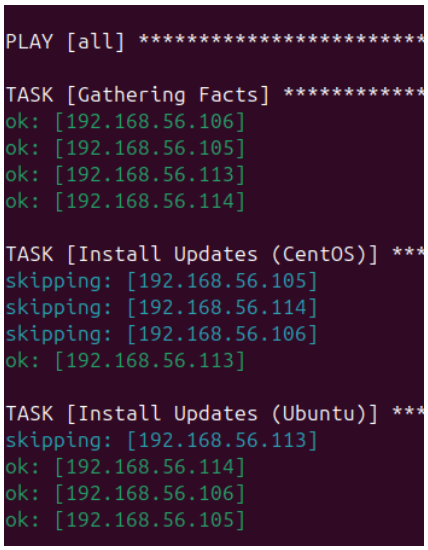
2. On the local machine, try to issue the following commands and describe each result:

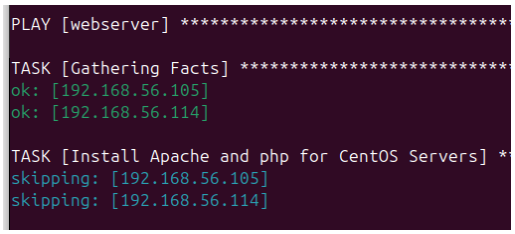
   2.1 *ansible-playbook --list-tags site.yml*

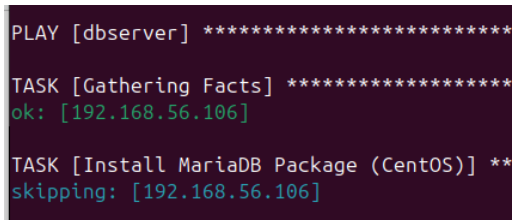
```
hazel@LocalMachine:~/CPE232_Manuel/Activity6$ ansible-playbook --list-tags site.yml

playbook: site.yml

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

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

  play #3 (dbserver): dbserver  TAGS: []
      TASK TAGS: [centos, db, mariadb, ubuntu]

  play #4 (fileserver): fileserver      TAGS: []
      TASK TAGS: [samba]
```

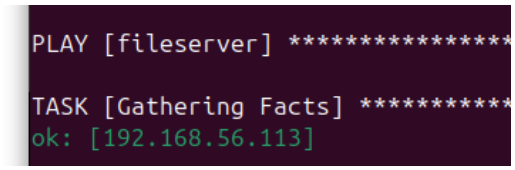
This shows just the list of tags in site.yml playbook. This does not execute the tasks in it.

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



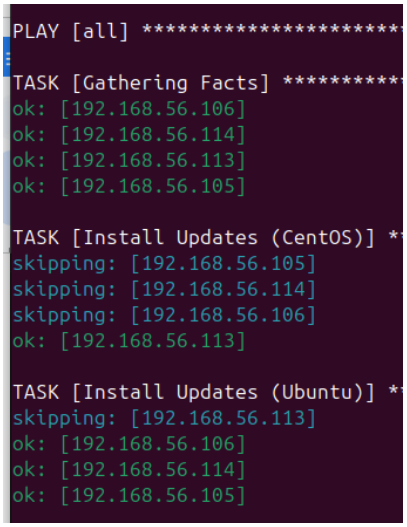


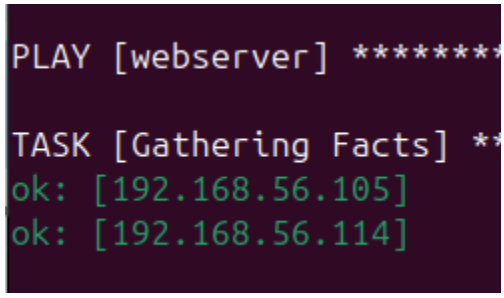


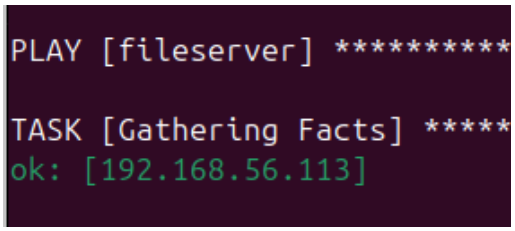


This just executes the tasks with a centos tag in it.

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

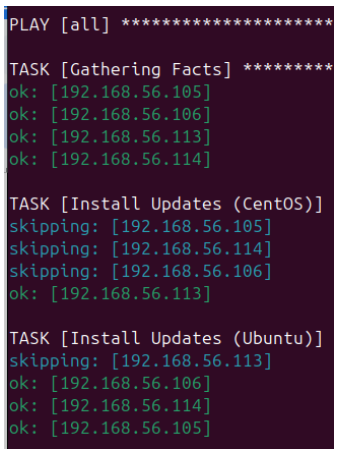


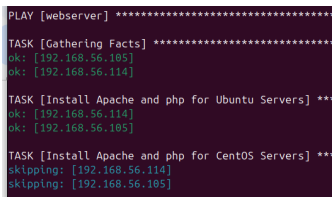




This just executes the tasks with a db tag in it.

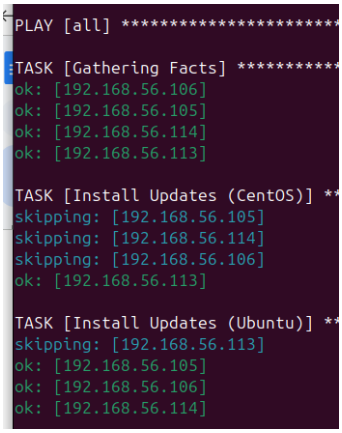
2.4 *ansible-playbook --tags apache --ask-become-pass site.yml*

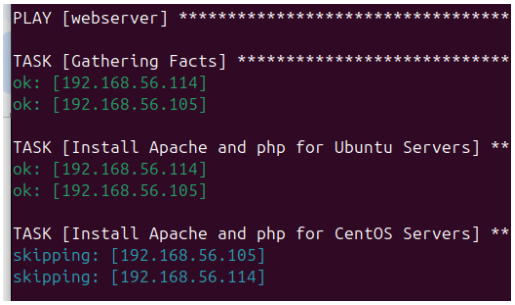




This just executes the tasks with a apache tag in it.

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





```
PLAY [dbserver] ***********************************

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

TASK [Install MariaDB Package (CentOS)] ************
skipping: [192.168.56.106]

TASK [Install MariaDB package (Ubuntu)] ************
ok: [192.168.56.106]
```

```
PLAY [fileserver] ***************

TASK [Gathering Facts] **********
ok: [192.168.56.113]
```

This just executes the tasks with both apache and db tag in it.

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.

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.

```
[hazelm3@CentOS ~]$ sudo systemctl stop httpd
[hazelm3@CentOS ~]$ sudo systemctl disable httpd
Removed "/etc/systemd/system/multi-user.target.wants/httpd.service".
[hazelm3@CentOS ~]$ systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: inactive (dead)
```

Stopping and disabling the httpd service in CentOS

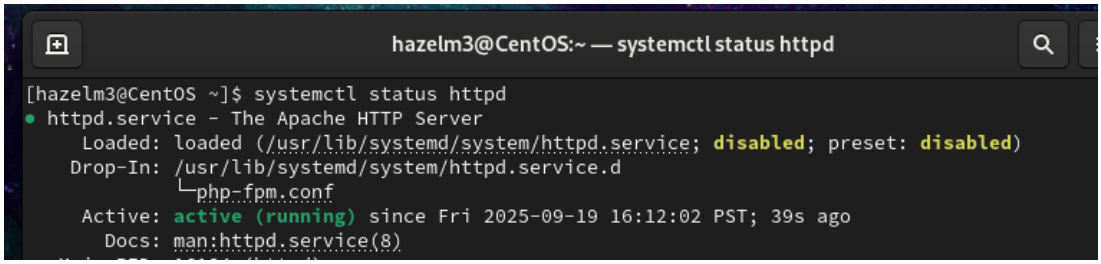
3. Go to the local machine and this time, run the *site.yml* file. Then after running the file, go again to the CentOS server and enter its IP address on the browser. Describe the result.

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

TASK [Gathering Facts] *****************************
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for Ubuntu Servers] ***
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for CentOS Servers] ***
skipping: [192.168.56.105]
skipping: [192.168.56.114]

TASK [Start httpd (CentOS)] ************************
skipping: [192.168.56.105]
skipping: [192.168.56.114]
```

This shows that it skips the tasks as the Servers are only in Ubuntu. To see whether our code works, we updated the inventory file by adding the IP address of the CentOS Server

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

TASK [Gathering Facts] *****************************
ok: [192.168.56.105]
ok: [192.168.56.113]
ok: [192.168.56.114]

TASK [Install Apache and php for Ubuntu Servers] ****
skipping: [192.168.56.113]
ok: [192.168.56.105]
ok: [192.168.56.114]

TASK [Install Apache and php for CentOS Servers] ****
skipping: [192.168.56.105]
skipping: [192.168.56.114]
ok: [192.168.56.113]

TASK [Start httpd (CentOS)] ************************
skipping: [192.168.56.105]
skipping: [192.168.56.114]
changed: [192.168.56.113]
```

Upon running it again, it now shows that there have been changes in the CentOS Server.

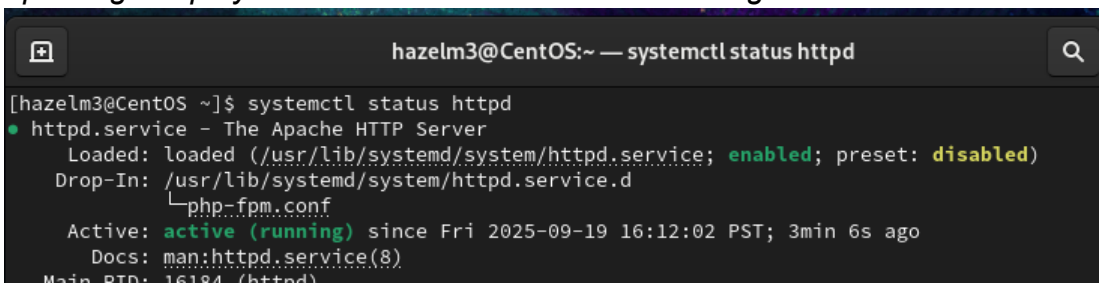


Checking the status of the httpd in the CentOS server, it now shows that it is active or running.

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.

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

Updating the playbook with the command and running it results to:



Now the httpd service is both active and enabled.

Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?
   Grouping our servers into groups is important as it organizes the infrastructure based on roles, environments, or functions. It allows us to target only specific servers without the need to affect other unrelated ones.
2. What is the importance of tags in playbooks?
   Tags kinda act like filters when running the playbooks. It allows us to selectively run, skip, and debug tasks that we only need. It allows you to execute the

playbook without having to execute all of the tasks in it. With tags, it allows us to quickly test, troubleshoot, and update specific components.

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

   This is because some services do not start or be enabled automatically after installation. Without the service running, it can result in a downtime or misconfigured system. So, playbooks ensure that certain services are not only installed but also started and enabled to run on boot. This ensures consistency among the servers and reduces the need for manual intervention.