Name: Castillo Joshua L.	**Date Performed: march 4, 2024**
Course/Section: CPE31S1	**Date Submitted: march 25, 2024**
Instructor: Dr. Jonathan Tylar	**Semester and SY:**

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**:

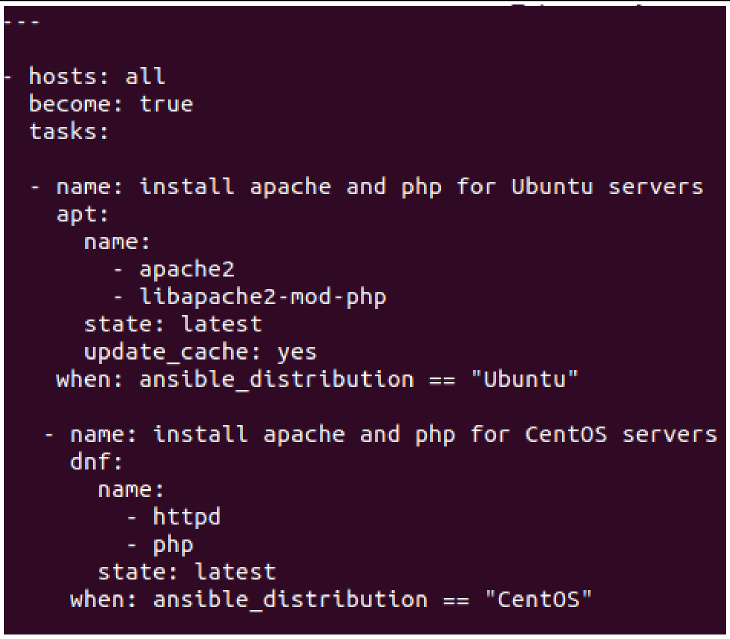
In this activity, we try to individualize hosts. For example, we don't want apache on all our servers, or maybe only one of our servers is a web server, or maybe we have different servers like database or file servers running different things on different categories of servers and that is what we are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

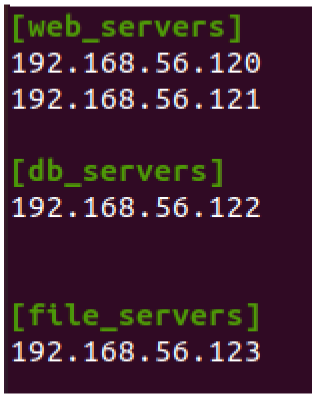
Task 1: Targeting Specific Nodes

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"
```

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

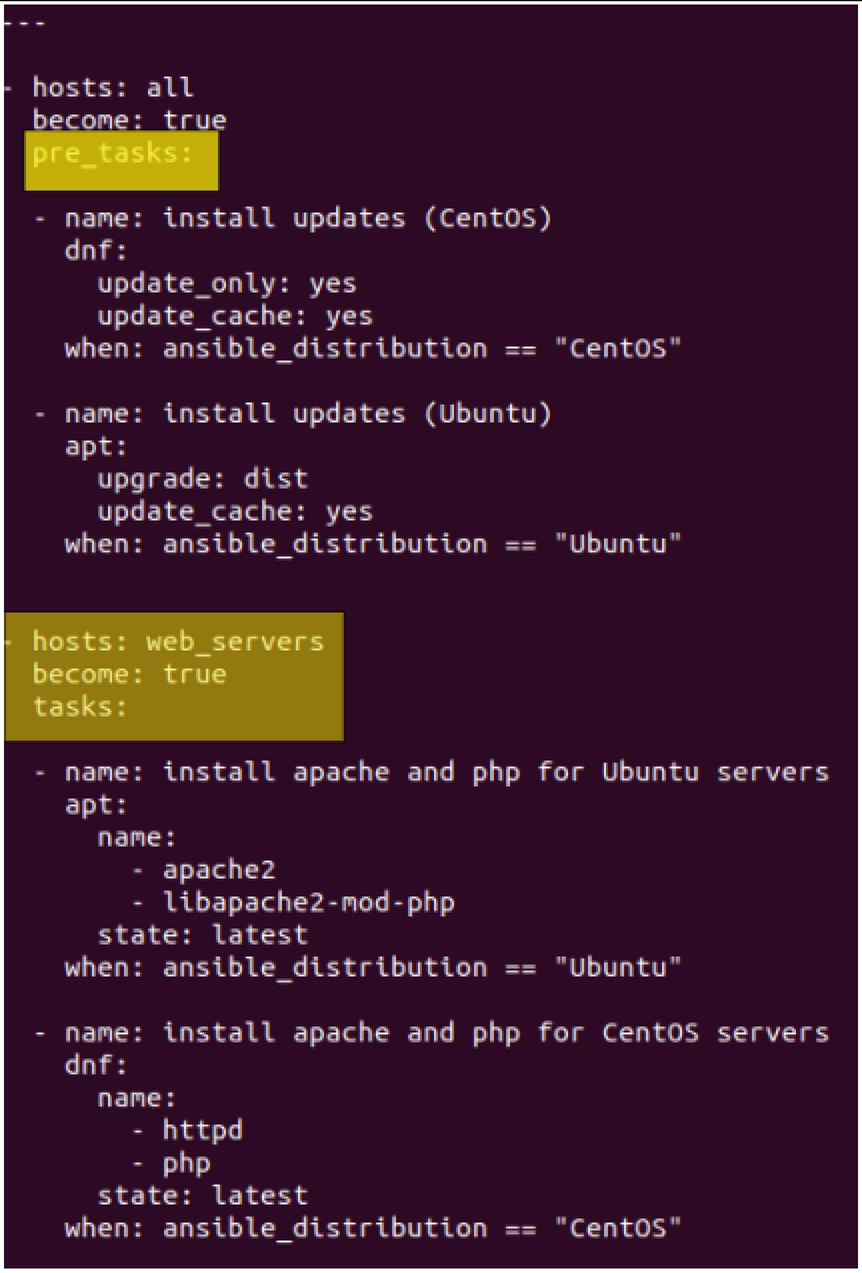


```
[web_servers]
192.168.56.120
192.168.56.121

[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.

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

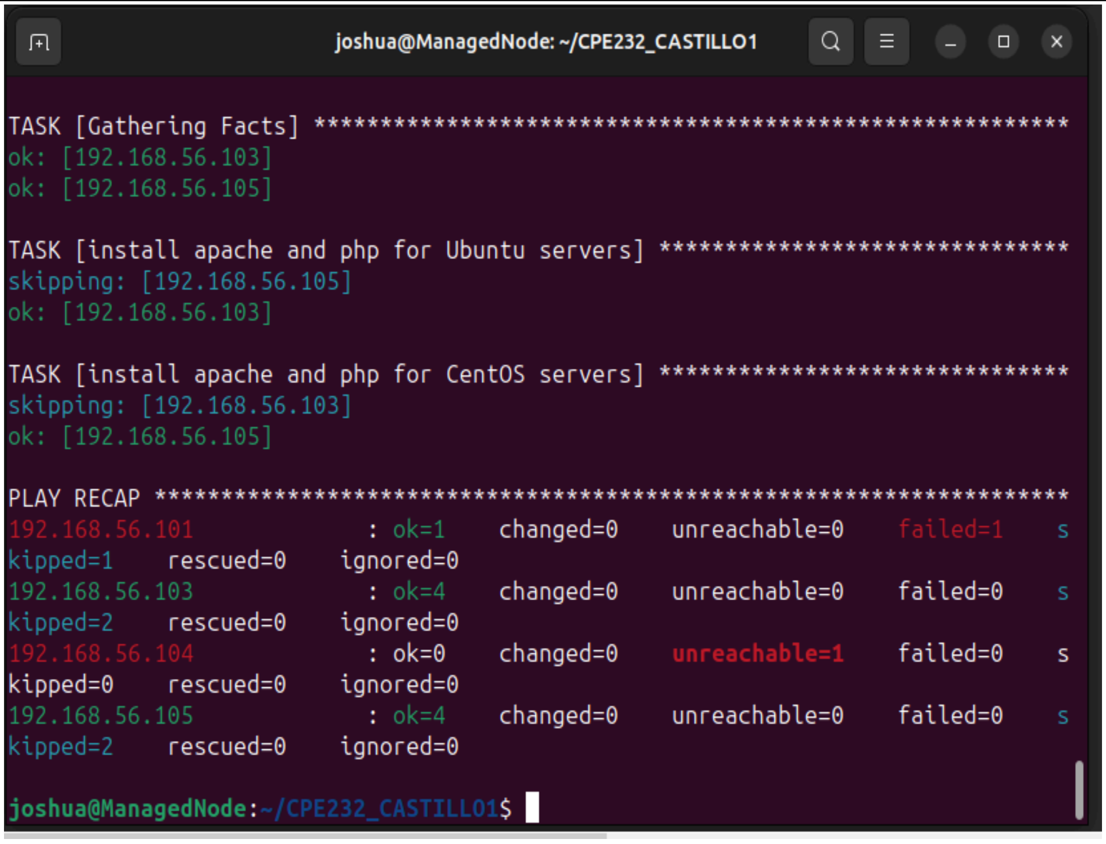


```
---

- 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.

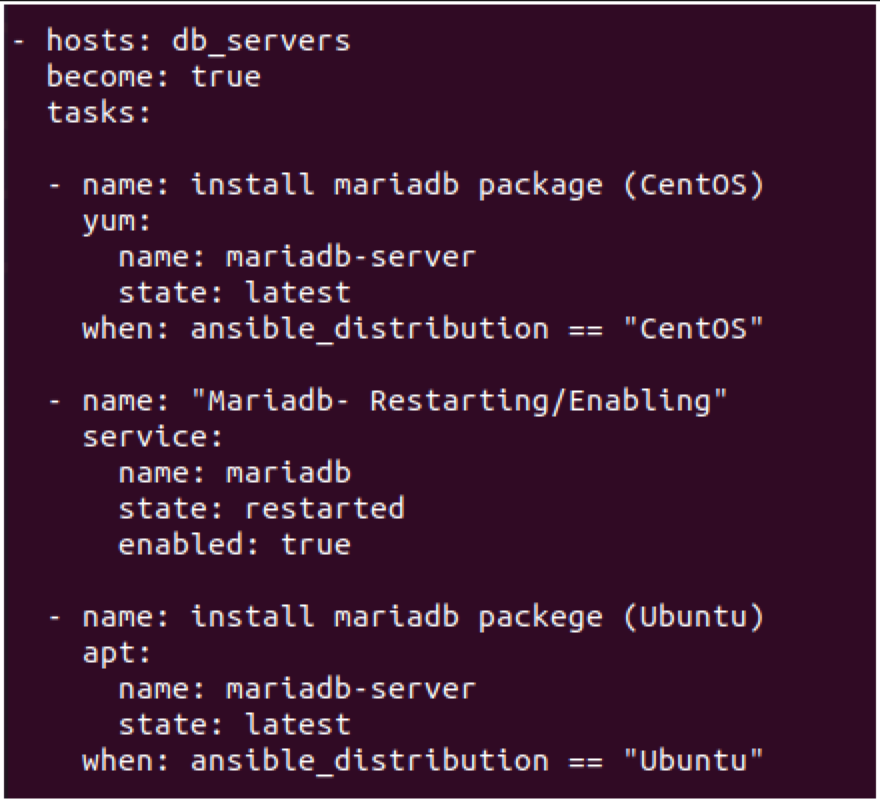
Run the *site.yml* file and describe the result.



```
TASK [Gathering Facts] *********************************************************
ok: [192.168.56.103]
ok: [192.168.56.105]

TASK [install apache and php for Ubuntu servers] *******************************
skipping: [192.168.56.105]
ok: [192.168.56.103]

TASK [install apache and php for CentOS servers] *******************************
skipping: [192.168.56.103]
ok: [192.168.56.105]

PLAY RECAP *********************************************************************
192.168.56.101             : ok=1    changed=0    unreachable=0    failed=1    s
kipped=1    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

site.yml's purpose is to target only the web_servers and as shown on the result my two addresses listed on the web_servers got the updates.

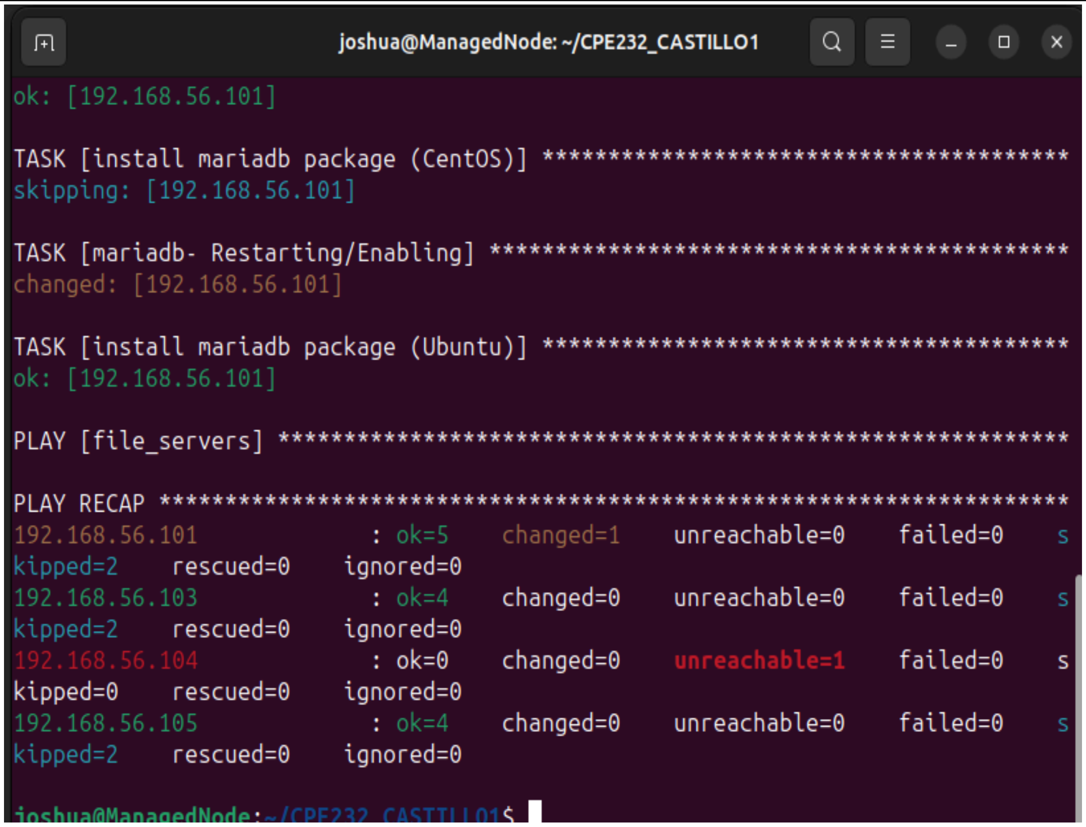
4. Let's try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db_servers* by adding it on the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3.



```
- 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.

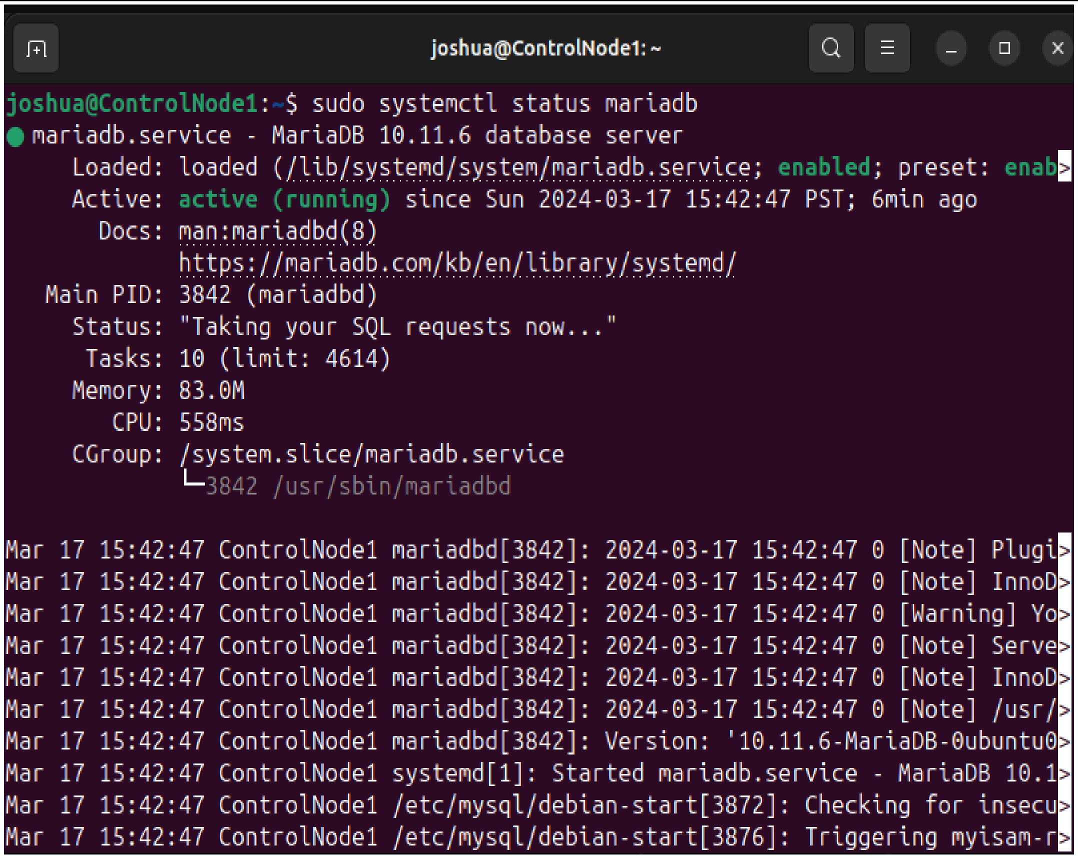
Run the *site.yml* file and describe the result.



```
ok: [192.168.56.101]

TASK [install mariadb package (CentOS)] ****************************************
skipping: [192.168.56.101]

TASK [mariadb- Restarting/Enabling] ********************************************
changed: [192.168.56.101]

TASK [install mariadb package (Ubuntu)] ****************************************
ok: [192.168.56.101]

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

PLAY RECAP *********************************************************************
192.168.56.101             : ok=5    changed=1    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

this time i installed mariadb on db_servers and it was successful.

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.



```
joshua@ControlNode1:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.6 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enab>
     Active: active (running) since Sun 2024-03-17 15:42:47 PST; 6min ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 3842 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 10 (limit: 4614)
     Memory: 83.0M
        CPU: 558ms
     CGroup: /system.slice/mariadb.service
             └─3842 /usr/sbin/mariadbd

Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Note] Plugi>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Note] InnoD>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Warning] Yo>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Note] Serve>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Note] InnoD>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: 2024-03-17 15:42:47 0 [Note] /usr/>
Mar 17 15:42:47 ControlNode1 mariadbd[3842]: Version: '10.11.6-MariaDB-0ubuntu0>
Mar 17 15:42:47 ControlNode1 systemd[1]: Started mariadb.service - MariaDB 10.1>
Mar 17 15:42:47 ControlNode1 /etc/mysql/debian-start[3872]: Checking for insecu>
Mar 17 15:42:47 ControlNode1 /etc/mysql/debian-start[3876]: Triggering myisam-r>
```

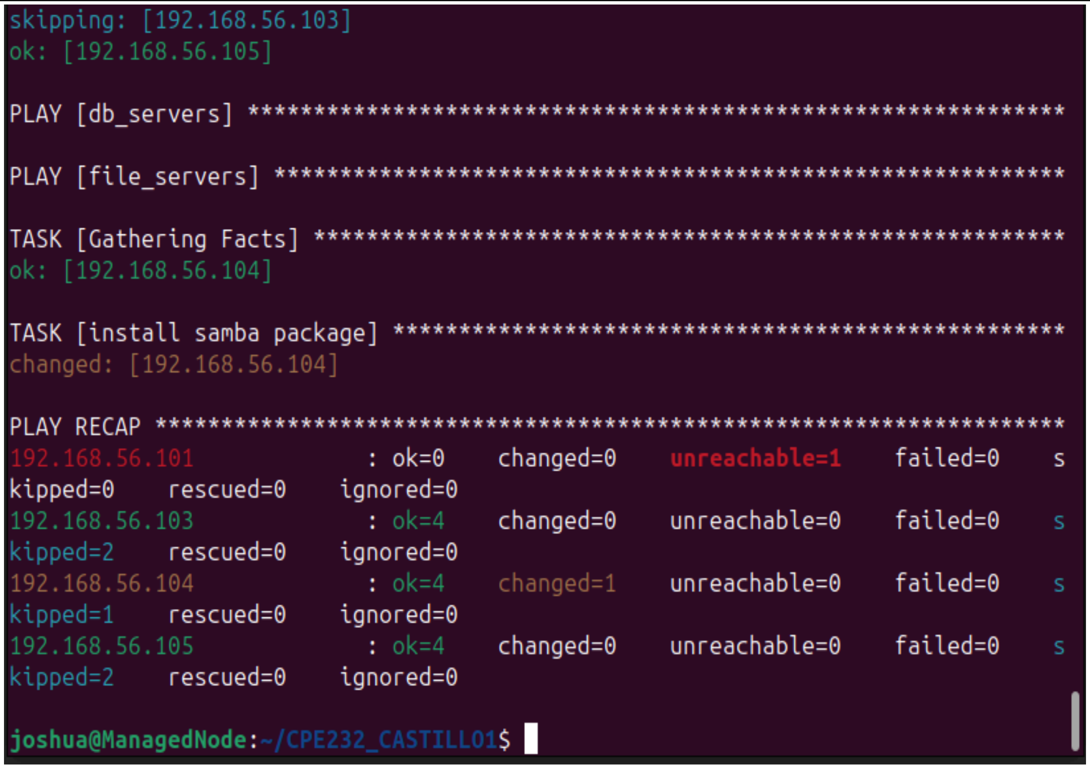
Describe the output.

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.



```
skipping: [192.168.56.103]
ok: [192.168.56.105]

PLAY [db_servers] ***************************************************************

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

TASK [Gathering Facts] **********************************************************
ok: [192.168.56.104]

TASK [install samba package] ****************************************************
changed: [192.168.56.104]

PLAY RECAP **********************************************************************
192.168.56.101             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.104             : ok=4    changed=1    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.105             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

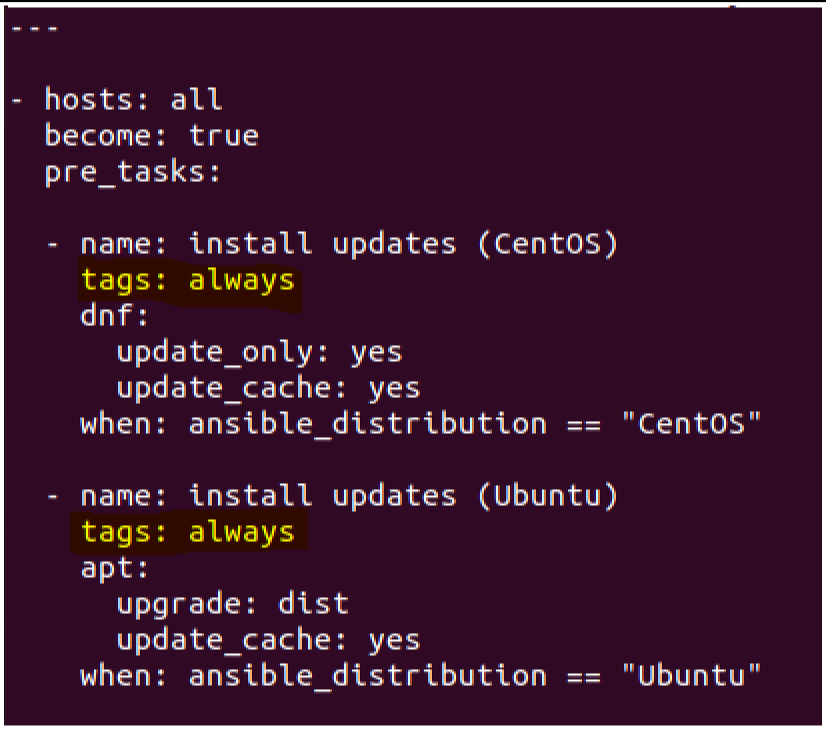
It installs samba package only on the ip address where it is listed to the file_servers

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.

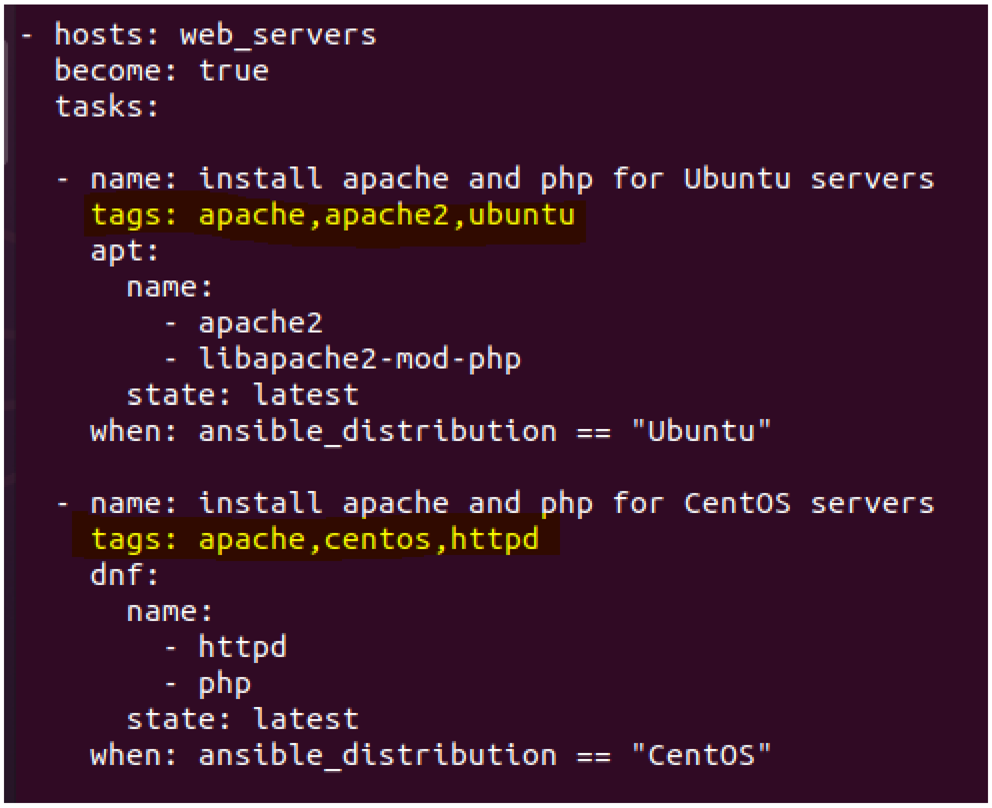
Task 2: Using Tags in running playbooks

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.

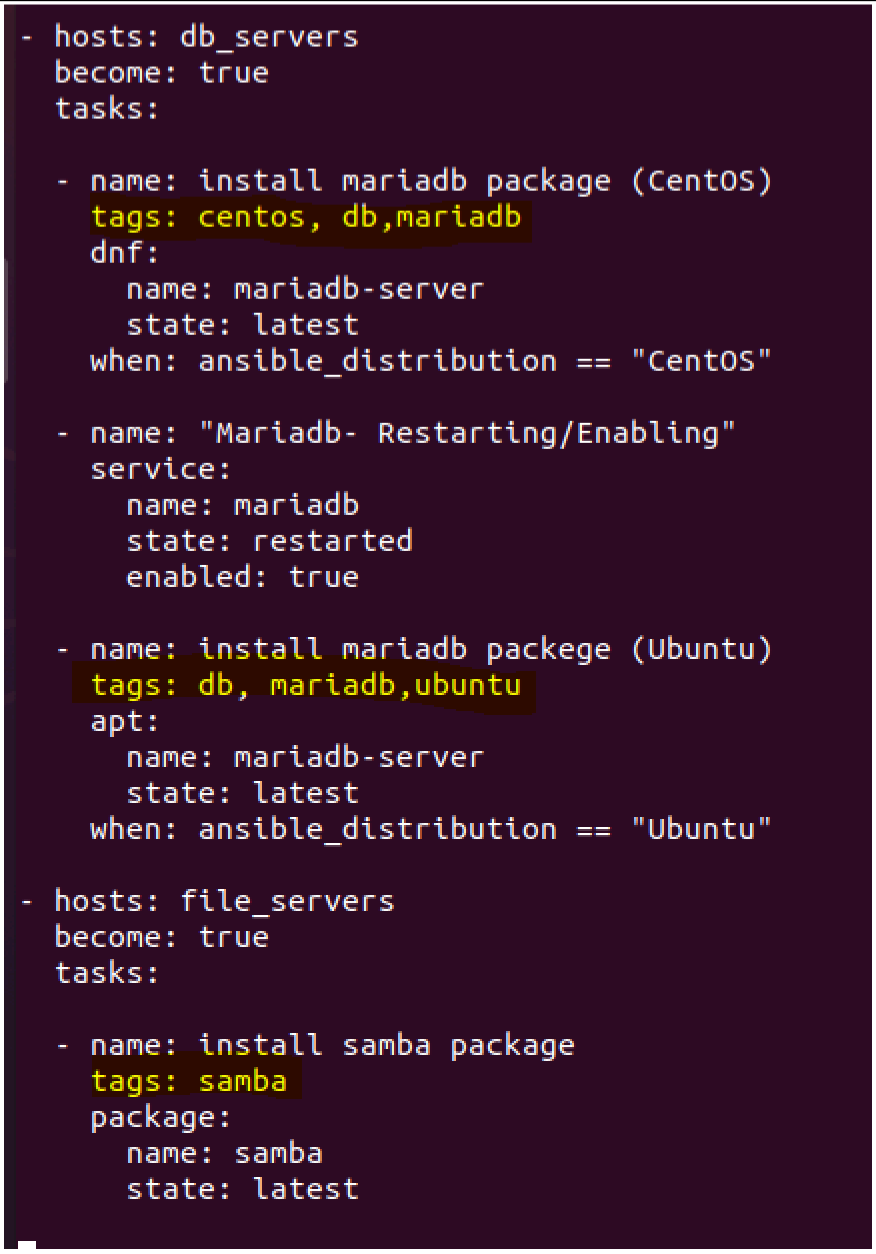
1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.



```
---

- 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"
```



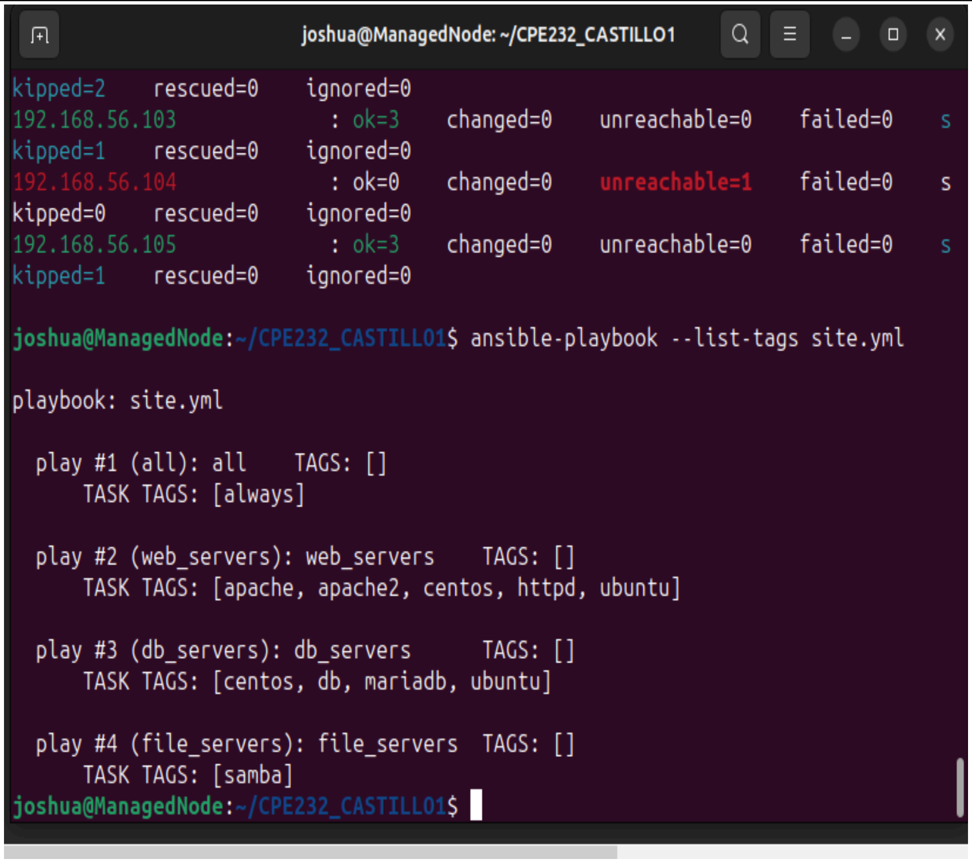
```
- hosts: web_servers
  become: true
  tasks:

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

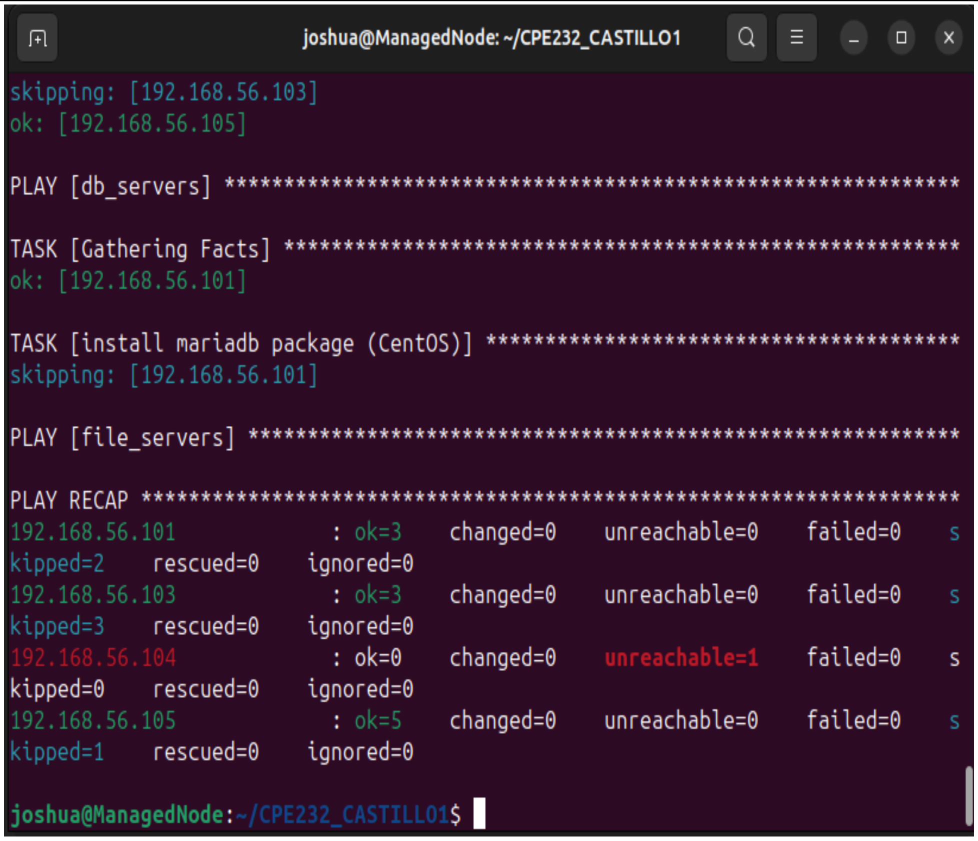
2. On the local machine, try to issue the following commands and describe each result:
   2.1 *ansible-playbook --list-tags site.yml*



```
joshua@ManagedNode: ~/CPE232_CASTILLO1
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$ ansible-playbook --list-tags site.yml

playbook: site.yml

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

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

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

  play #4 (file_servers): file_servers  TAGS: []
      TASK TAGS: [samba]
joshua@ManagedNode:~/CPE232_CASTILLO1$
```

it specifies the task which is by listing only the tags you put in the site.yml.

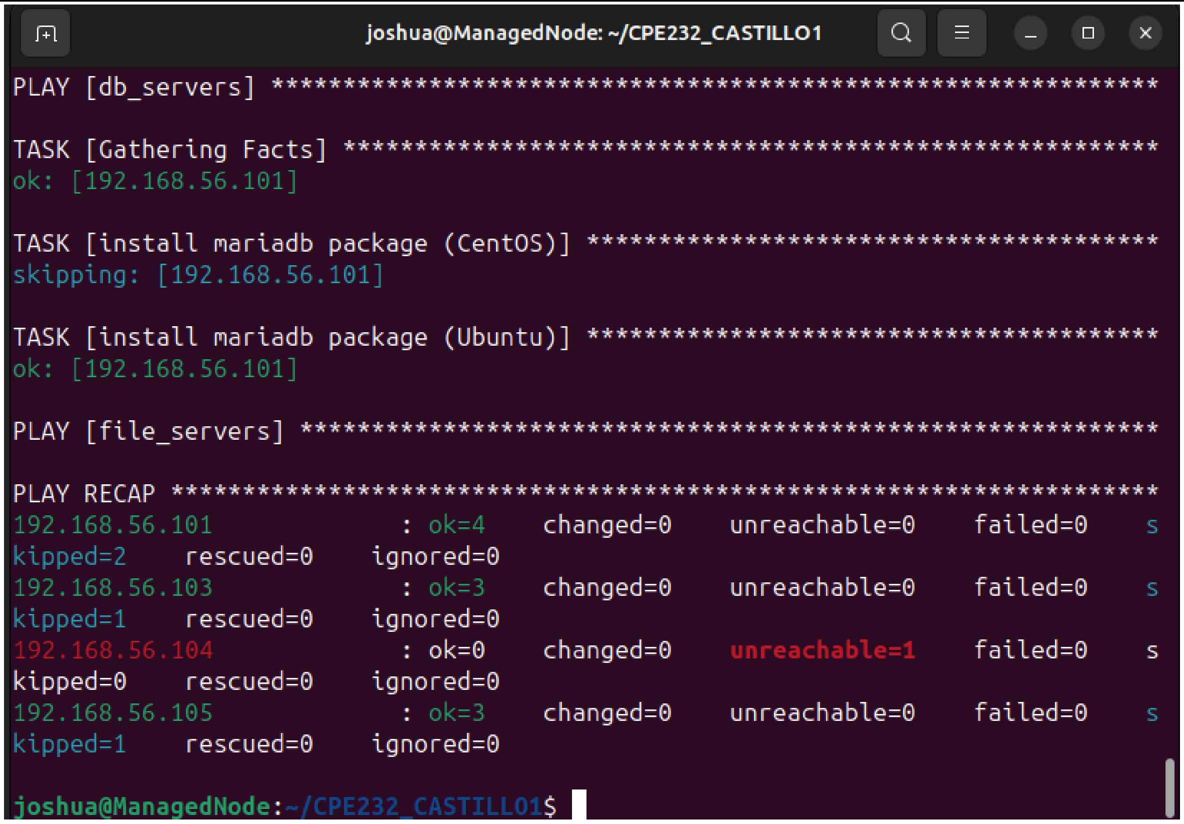
2.2 *ansible-playbook --tags centos --ask-become-pass site.yml*



```
skipping: [192.168.56.103]
ok: [192.168.56.105]

PLAY [db_servers] **************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.101]

TASK [install mariadb package (CentOS)] ****************************************
skipping: [192.168.56.101]

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

PLAY RECAP *********************************************************************
192.168.56.101             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=3    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=5    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

it specifies the tasks to only with centos from the site.yml.

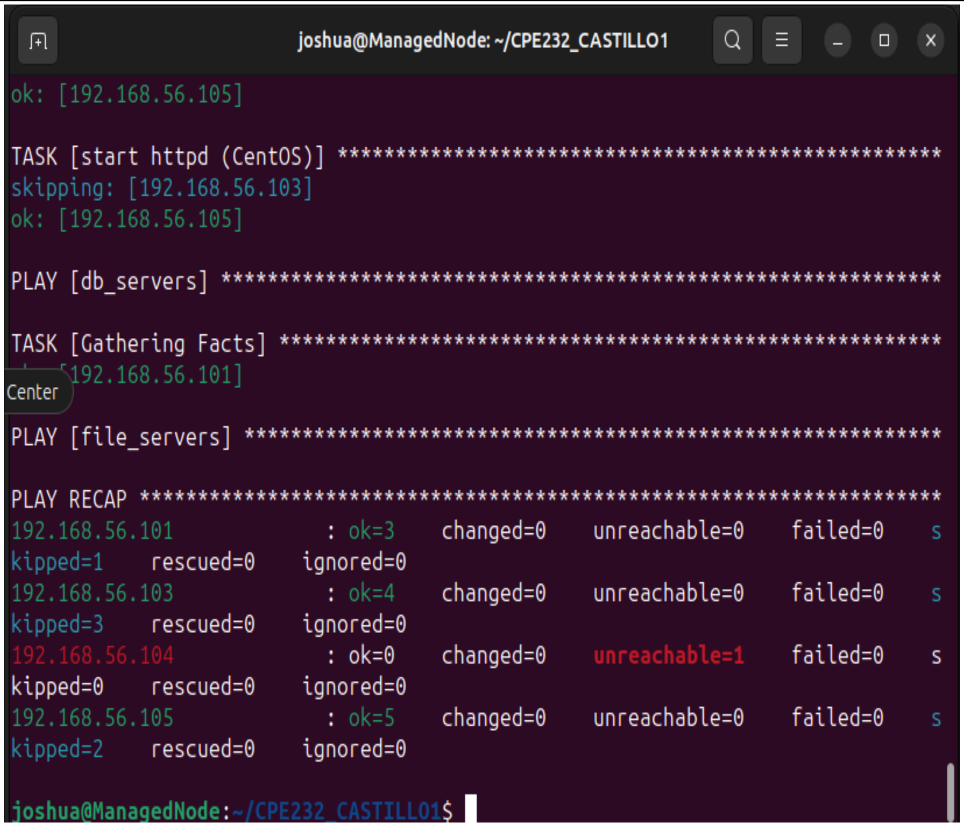
2.3 *ansible-playbook --tags db --ask-become-pass site.yml*



```
PLAY [db_servers] **************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.101]

TASK [install mariadb package (CentOS)] ****************************************
skipping: [192.168.56.101]

TASK [install mariadb package (Ubuntu)] ****************************************
ok: [192.168.56.101]

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

PLAY RECAP *********************************************************************
192.168.56.101             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

it specifies the task on ip addresses that is on db only.

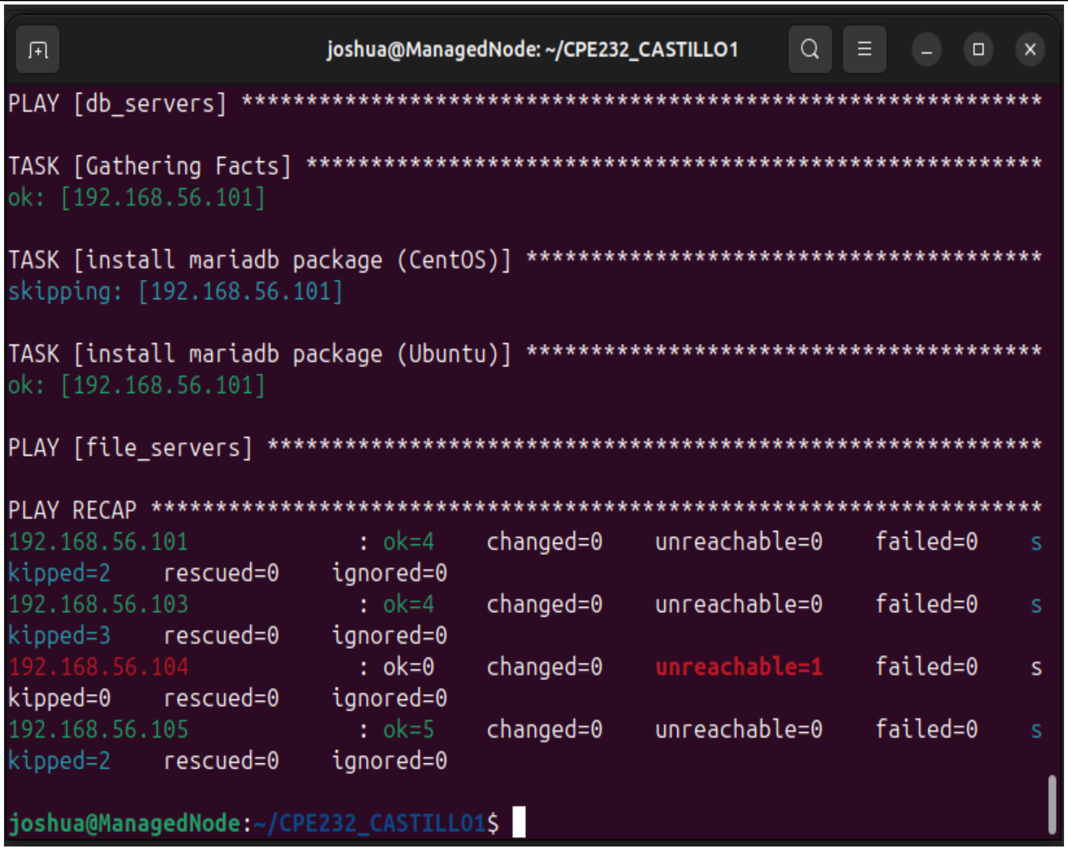
2.4 *ansible-playbook --tags apache --ask-become-pass site.yml*



```
joshua@ManagedNode: ~/CPE232_CASTILLO1
ok: [192.168.56.105]

TASK [start httpd (CentOS)] ***************************************************
skipping: [192.168.56.103]
ok: [192.168.56.105]

PLAY [db_servers] *************************************************************

TASK [Gathering Facts] ********************************************************
ok: [192.168.56.101]

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

PLAY RECAP ********************************************************************
192.168.56.101             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=3    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=5    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

it specifies the tasks to all the ip addresses that install apache.

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

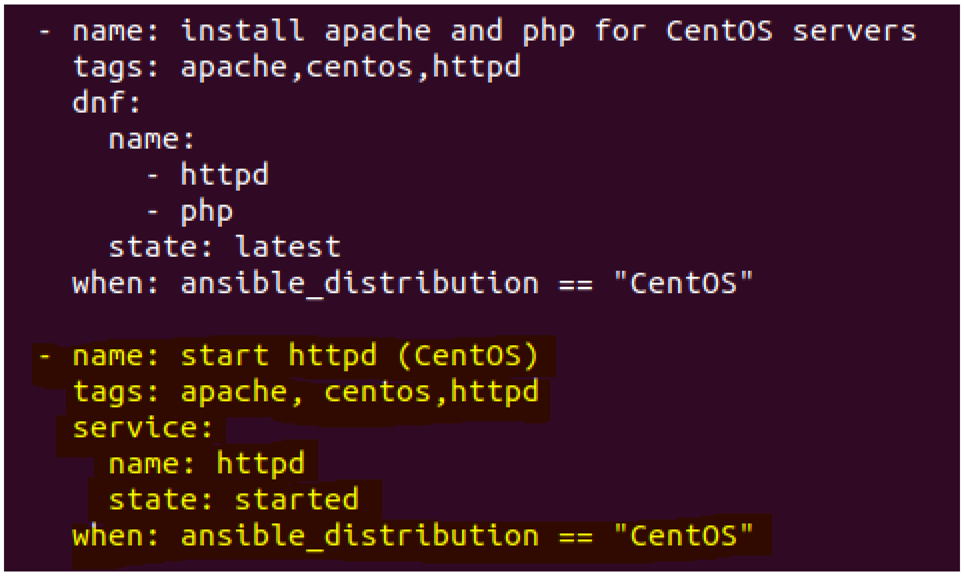


```
joshua@ManagedNode: ~/CPE232_CASTILLO1

PLAY [db_servers] *************************************************************

TASK [Gathering Facts] ********************************************************
ok: [192.168.56.101]

TASK [install mariadb package (CentOS)] ***************************************
skipping: [192.168.56.101]

TASK [install mariadb package (Ubuntu)] ***************************************
ok: [192.168.56.101]

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

PLAY RECAP ********************************************************************
192.168.56.101             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=3    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=5    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$ 
```

it specifies the task to the ip addresses that is on db and its apache.

Task 3: Managing Services

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

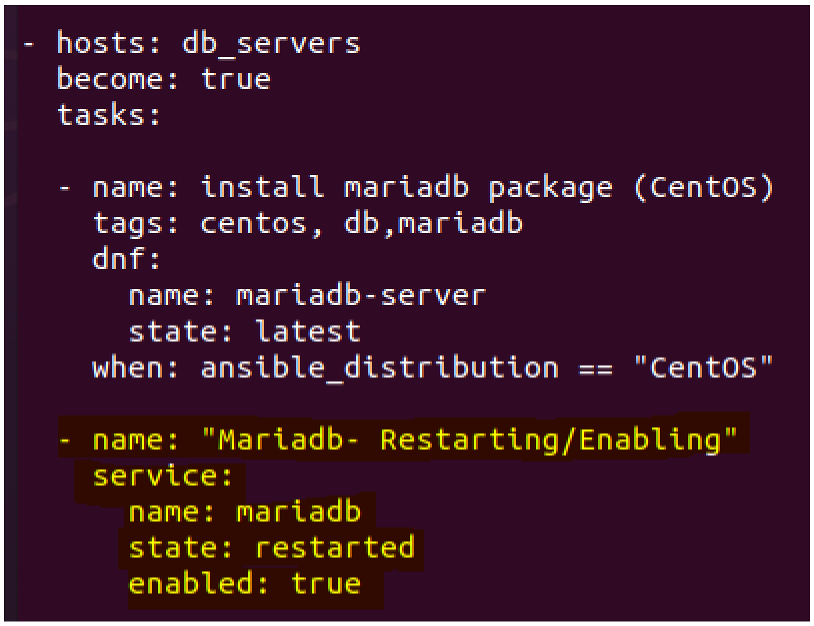


```
- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"

- 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.

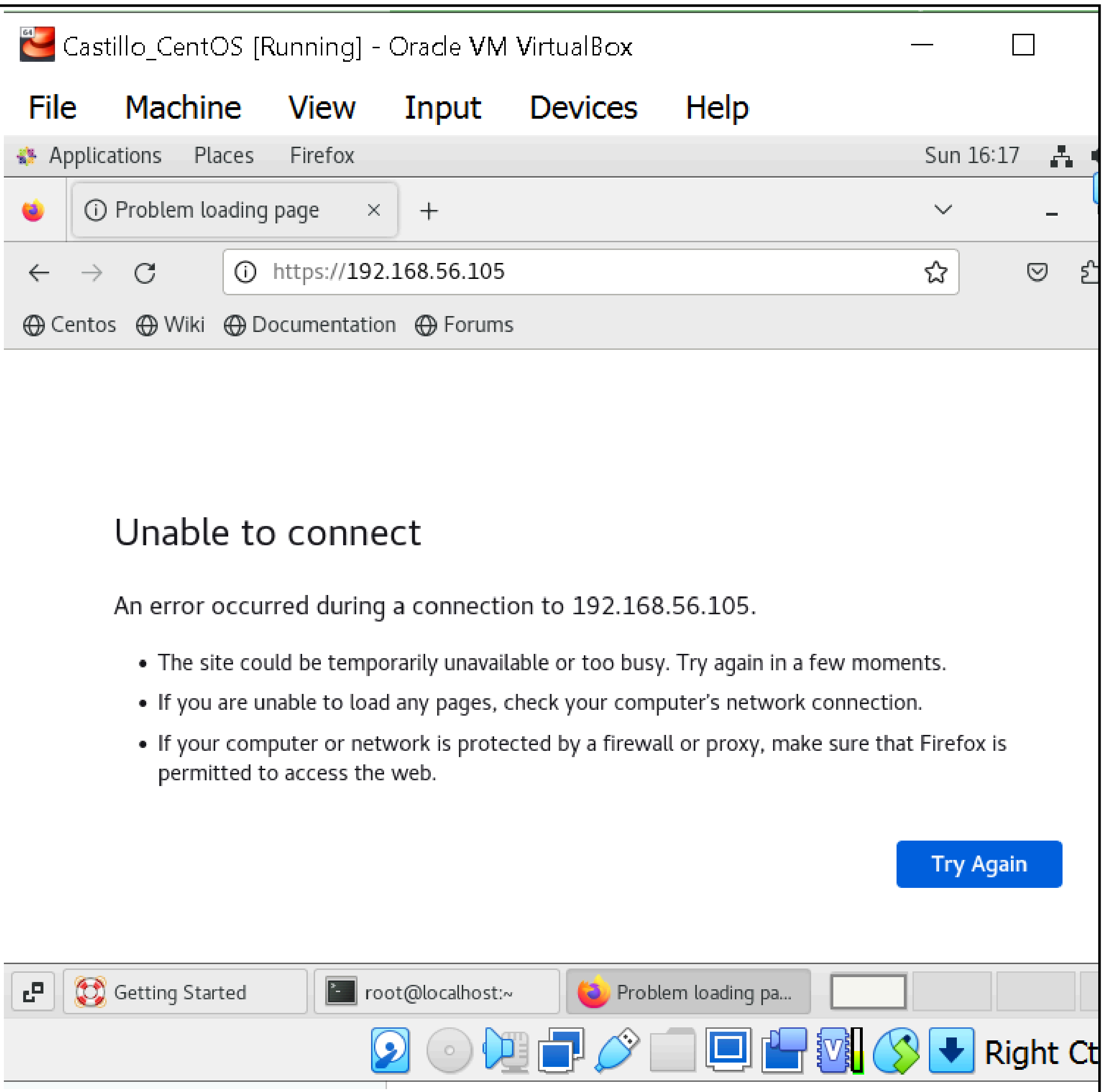


```
- hosts: db_servers
  become: true
  tasks:

  - 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.

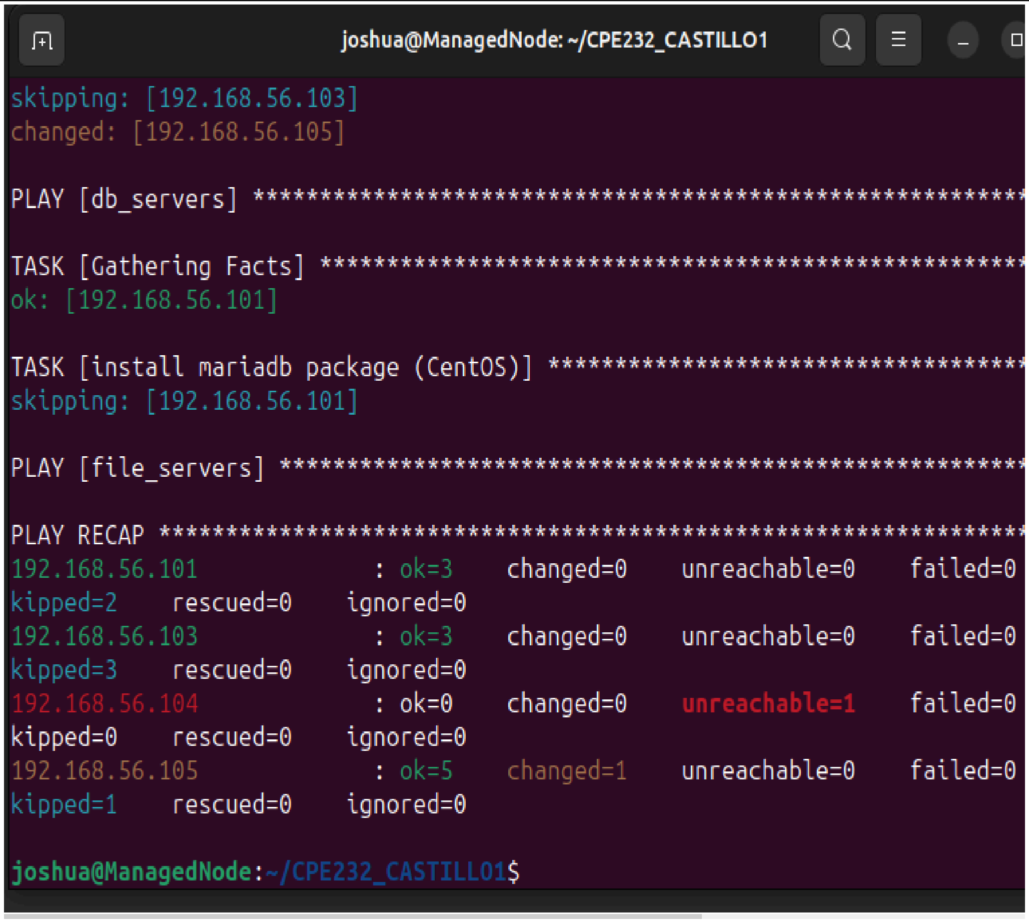

Castillo_CentOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox
Sun 16:17
Problem loading page
https://192.168.56.105
Centos Wiki Documentation Forums
Try Again
Getting Started
root@localhost:~
Problem loading pa...
Right Ct

Unable to connect

An error occurred during a connection to 192.168.56.105.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

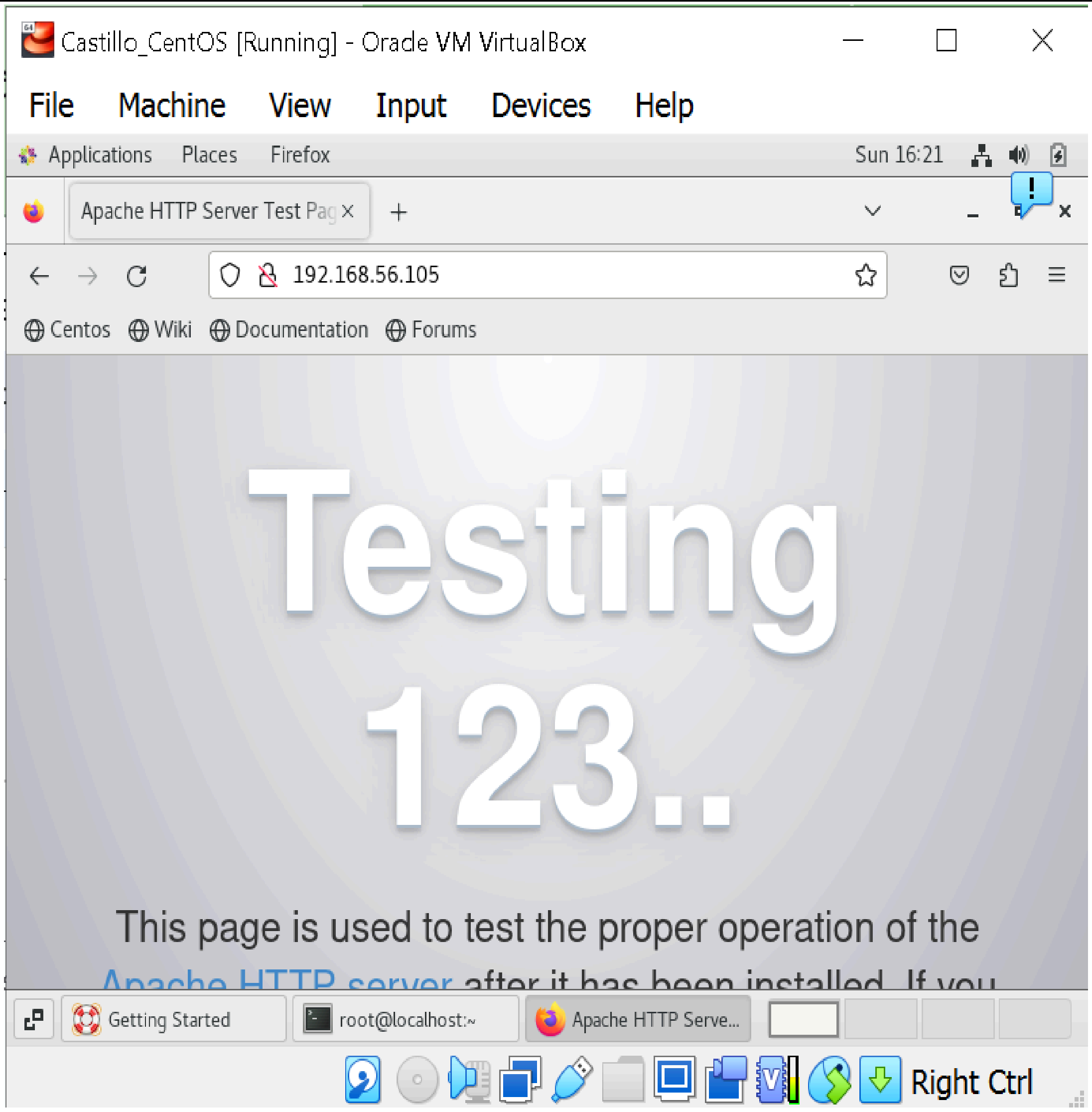
3. Go to the local machine and this time, run the *site.yml* file. Then after running the file, go again to the CentOS server and enter its IP address on the browser. Describe the result.



joshua@ManagedNode: ~/CPE232_CASTILLO1

```
skipping: [192.168.56.103]
changed: [192.168.56.105]

PLAY [db_servers] **********************************************************

TASK [Gathering Facts] *****************************************************
ok: [192.168.56.101]

TASK [install mariadb package (CentOS)] ************************************
skipping: [192.168.56.101]

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

PLAY RECAP *****************************************************************
192.168.56.101             : ok=3    changed=0    unreachable=0    failed=0
kipped=2    rescued=0    ignored=0
192.168.56.103             : ok=3    changed=0    unreachable=0    failed=0
kipped=3    rescued=0    ignored=0
192.168.56.104             : ok=0    changed=0    unreachable=1    failed=0
kipped=0    rescued=0    ignored=0
192.168.56.105             : ok=5    changed=1    unreachable=0    failed=0
kipped=1    rescued=0    ignored=0

joshua@ManagedNode:~/CPE232_CASTILLO1$
```

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.



Testing 123..

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you

Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?

   The importance of putting remote servers into groups is to facilitate easier management and organization of the servers. By grouping servers based on their roles or characteristics, such as web servers, database servers, or application servers, it becomes easier to apply configuration changes, perform maintenance tasks, and monitor the servers collectively. Grouping servers also allows for more efficient automation and orchestration of tasks across multiple servers simultaneously.

2. What is the importance of tags in playbooks?
   Tags in playbooks provide a way to selectively apply tasks or configurations to specific groups of servers. They allow for fine-grained control over which servers receive certain actions, enabling more flexibility and customization in playbook execution. Tags can be assigned to servers based on different criteria, such as their roles, environments, or specific attributes. By using tags, administrators can target specific subsets of servers for specific tasks, making playbook execution more efficient and reducing the risk of unintended changes.

3. Why do I think some services need to be managed automatically in playbooks?
   Some services need to be managed automatically in playbooks to ensure consistent and reliable deployment and maintenance processes. Manual management of services can be time-consuming, error-prone, and inconsistent across different environments. By automating service management in playbooks, tasks like service installation, configuration, starting, stopping, and monitoring can be executed consistently and reliably. Automation also allows for centralized control and the ability to scale operations effectively, especially when dealing with a large number of servers or complex infrastructures. Additionally, automation provides the advantage of repeatability and auditability, as the entire service management process can be documented and version-controlled within the playbook.