Name:Tracey Dee Bringuela	**Date Performed:02/10/24**
Course/Section:CPE31S2	**Date Submitted:02/10/24**
Instructor: Robin Valenzuela	**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 ru

1.3 Managing Services from remote servers using playbooks

2. **Discussion**:

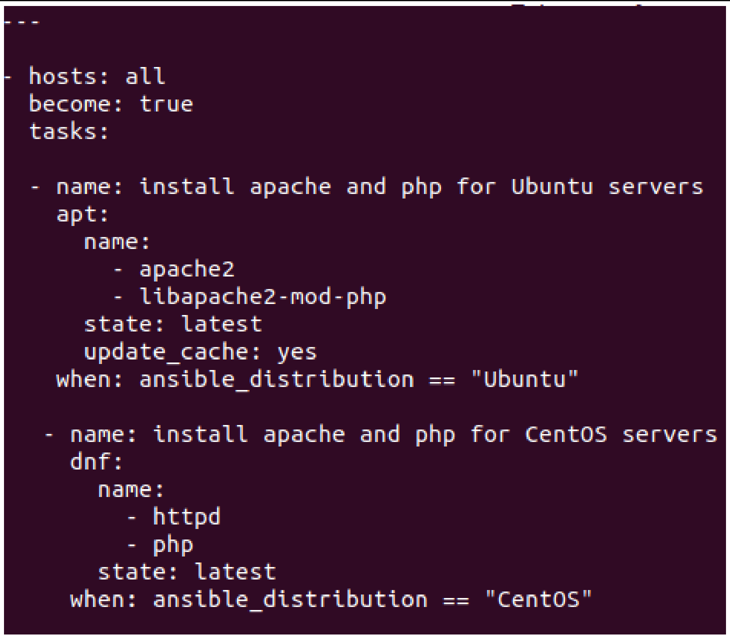
In this activity, we try to individualize hosts. For example, we don't want apache on all our servers, or maybe only one of our servers is a web server, or maybe we have different servers like database or file servers running different things on different categories of servers and that is what we are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

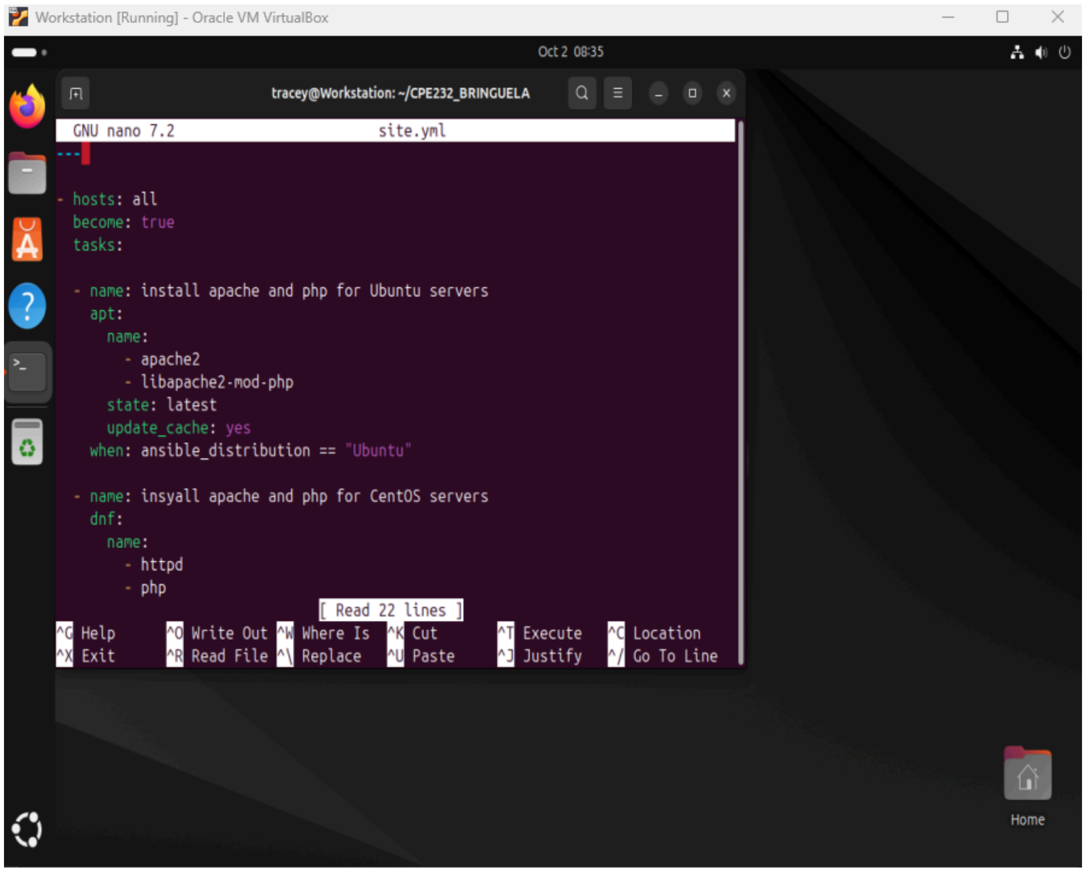
Requirement:
In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command *ssh-copy-id* to copy the public key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

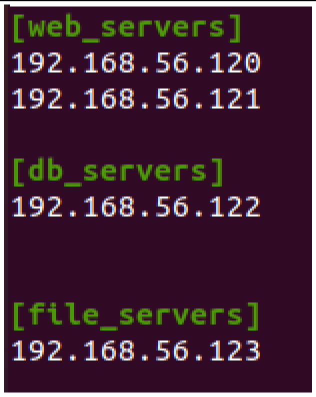


```
---

- 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"
```


Workstation [Running] - Oracle VM VirtualBox
Oct 2 08:35
tracey@Workstation: ~/CPE232_BRINGUELA
Home

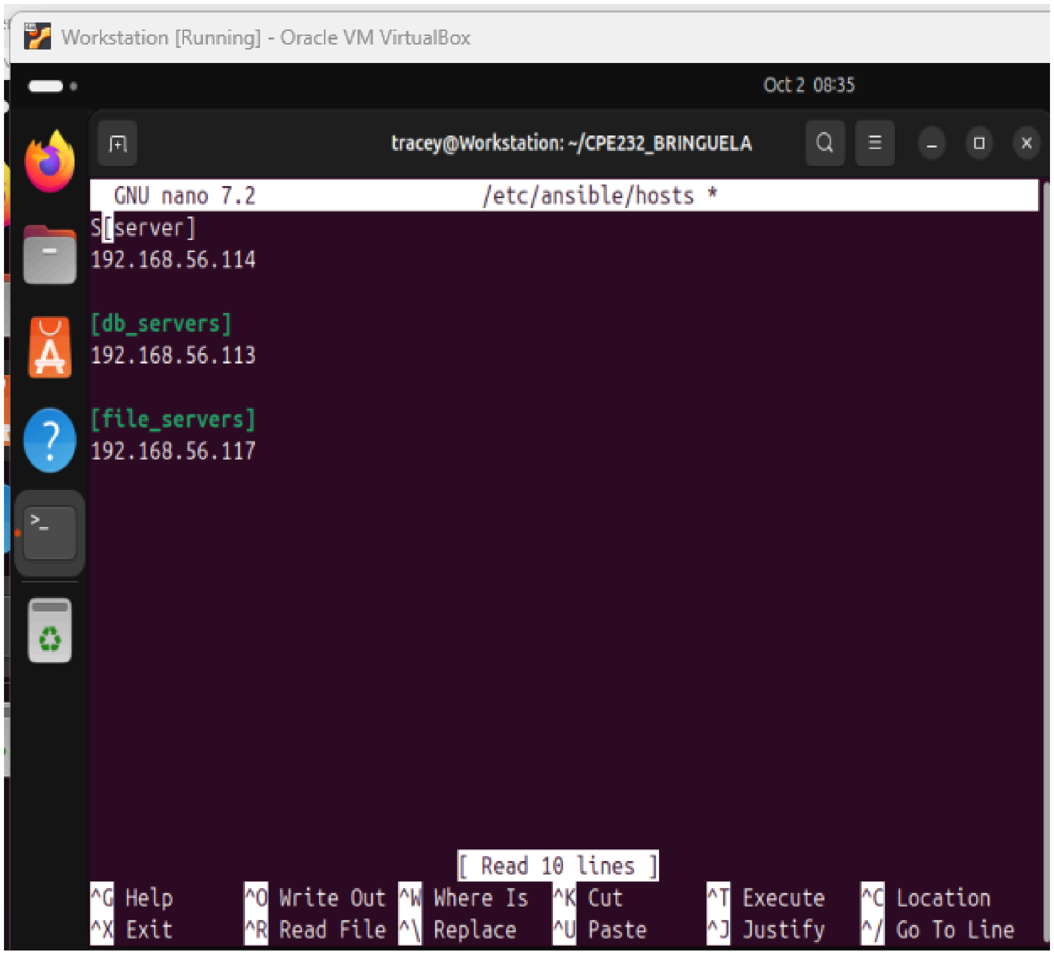
```
  GNU nano 7.2                         site.yml
---

- 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: insyall apache and php for CentOS servers
    dnf:
      name:
        - httpd
        - php
                             [ Read 22 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

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
```

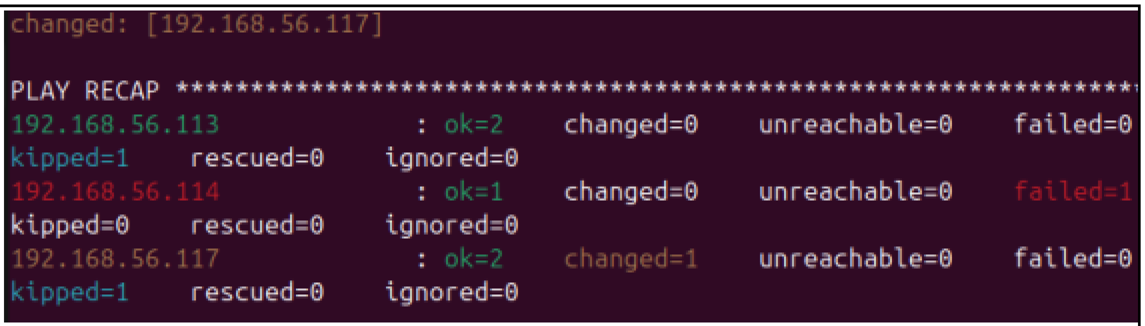
Make sure to save the file and exit.



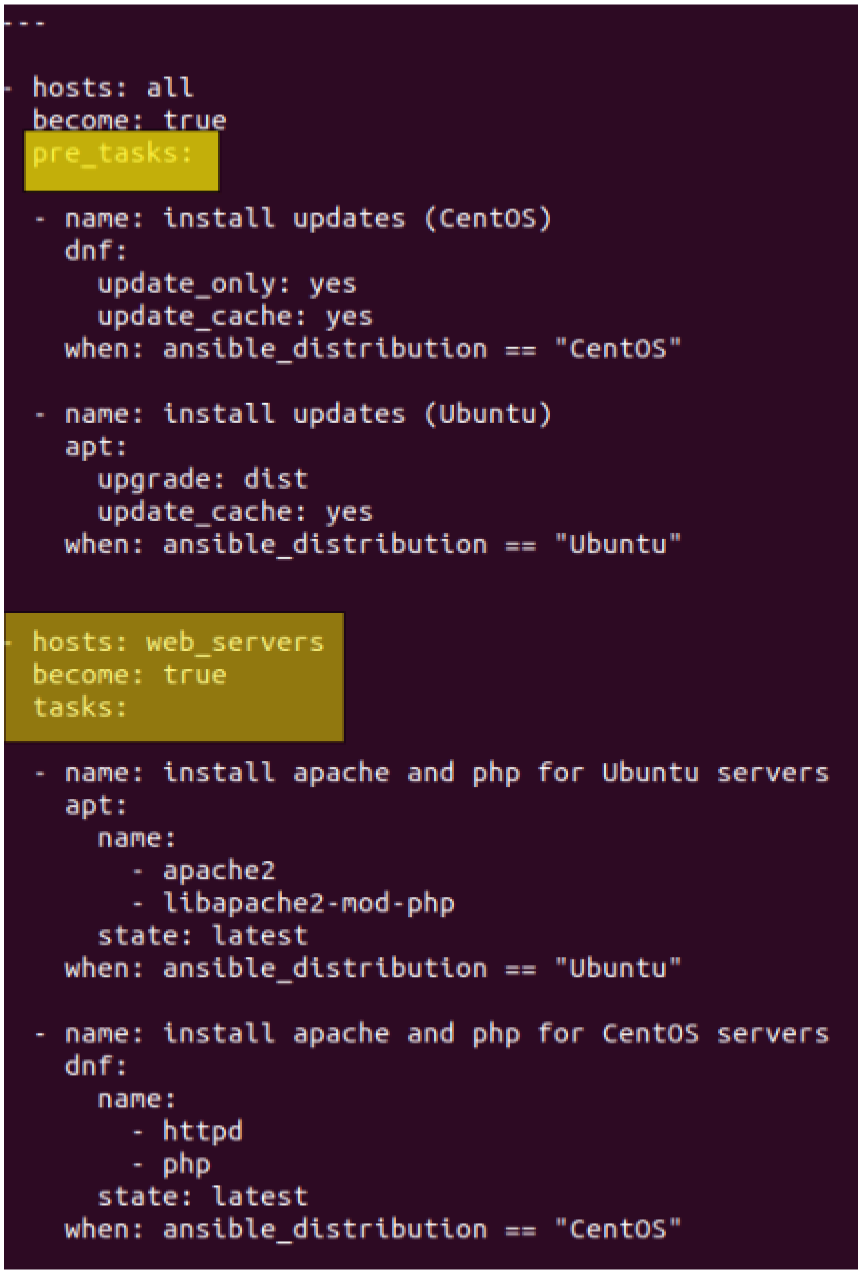
Workstation [Running] - Oracle VM VirtualBox
Oct 2 08:35
tracey@Workstation: ~/CPE232_BRINGUELA

```
  GNU nano 7.2                    /etc/ansible/hosts *
S[server]
192.168.56.114

[db_servers]
192.168.56.113

[file_servers]
192.168.56.117










                                      [ Read 10 lines ]
^G Help        ^O Write Out  ^W Where Is   ^K Cut         ^T Execute    ^C Location
^X Exit        ^R Read File  ^\ Replace    ^U Paste       ^J Justify    ^/ Go To Line
```

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.


```
changed: [192.168.56.117]

PLAY RECAP ********************************************************************
192.168.56.113             : ok=2    changed=0    unreachable=0    failed=0
kipped=1    rescued=0    ignored=0
192.168.56.114             : ok=1    changed=0    unreachable=0    failed=1
kipped=0    rescued=0    ignored=0
192.168.56.117             : ok=2    changed=1    unreachable=0    failed=0
kipped=1    rescued=0    ignored=0
```

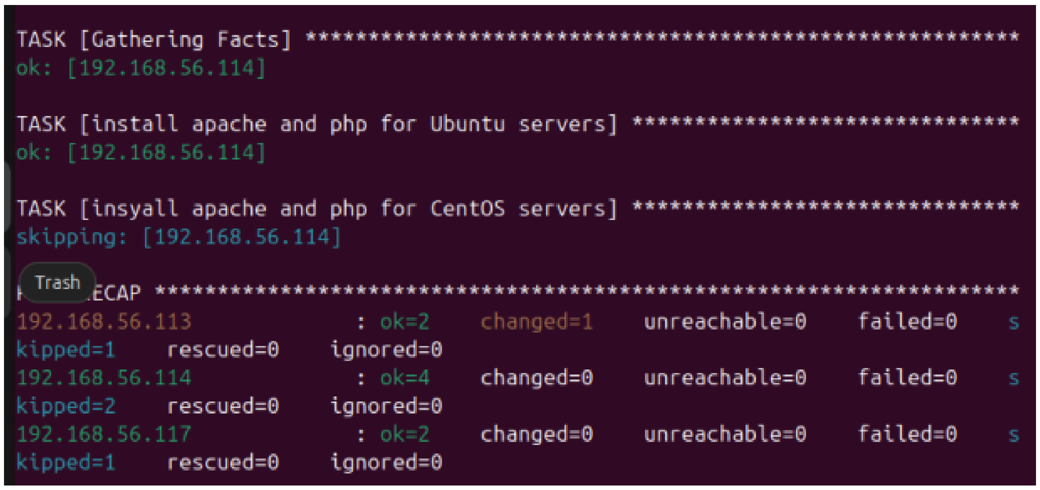
3. Edit the *site.yml* by following the image below:


```
---

- 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.

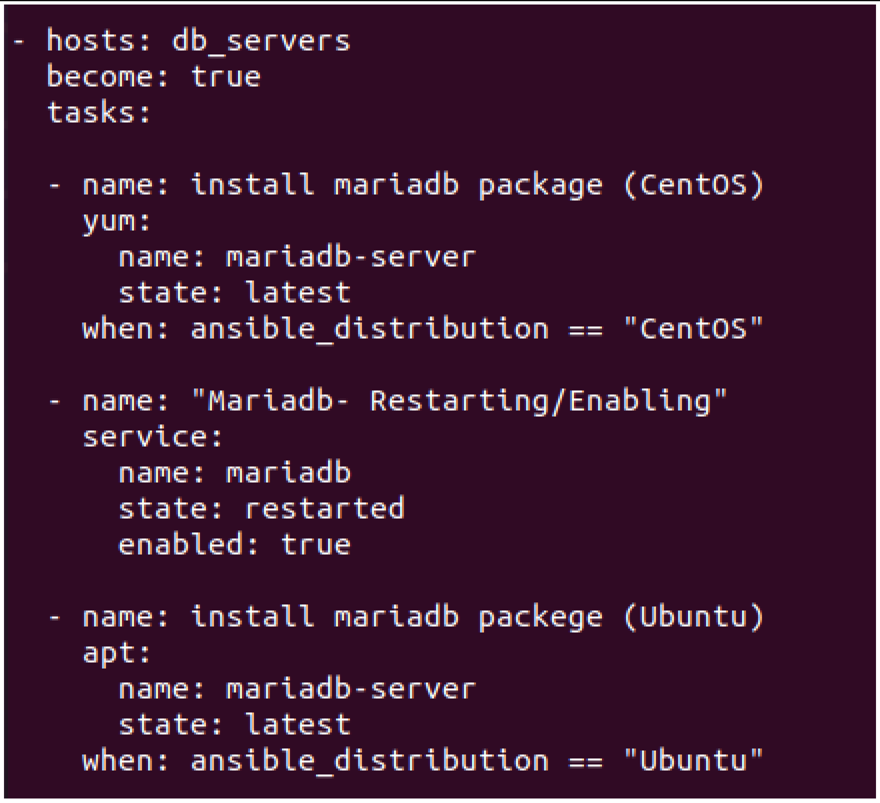
The *pre-tasks* command tells the ansible to run it before any other thing. In the *pre-tasks*, CentOS will install updates while Ubuntu will upgrade its distribution package. This will run before running the second play, which is targeted at *web_servers*. In the second play, apache and php will be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

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

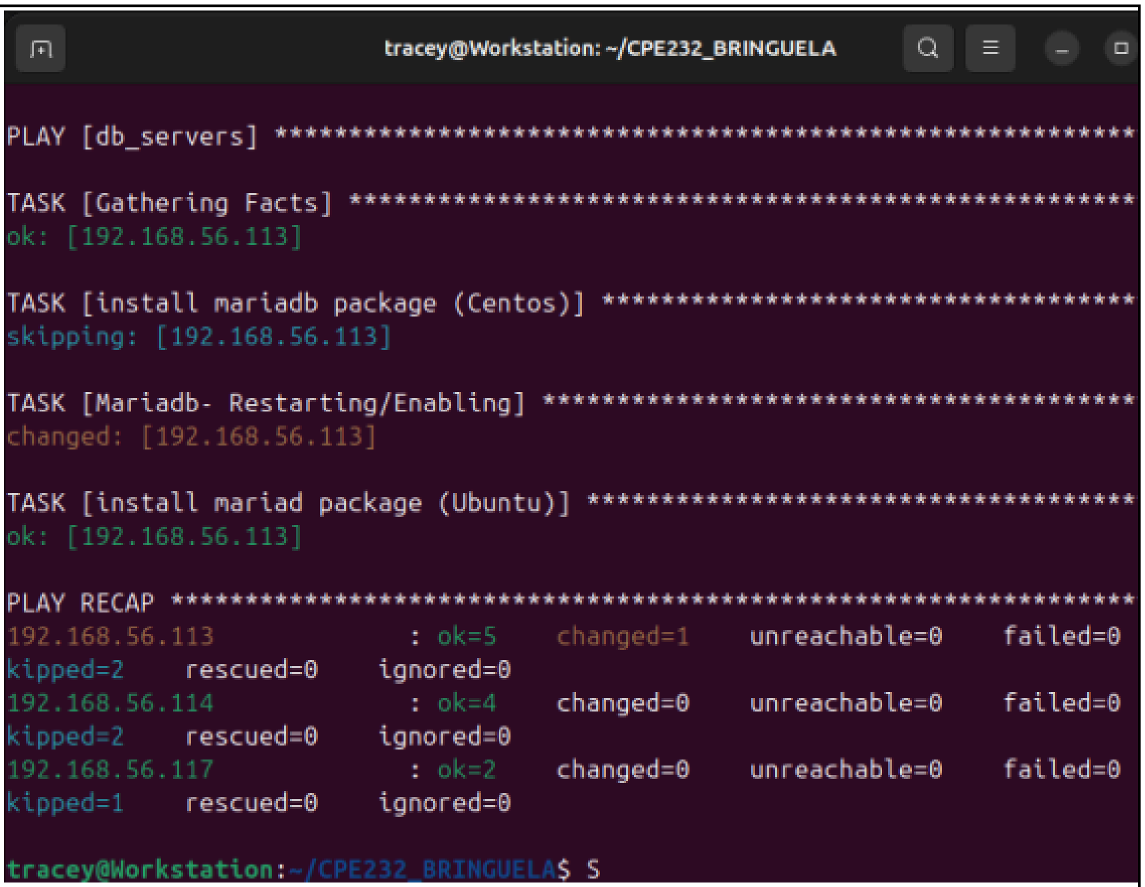


```
TASK [Gathering Facts] *********************************************************
ok: [192.168.56.114]

TASK [install apache and php for Ubuntu servers] ********************************
ok: [192.168.56.114]

TASK [insyall apache and php for CentOS servers] ********************************
skipping: [192.168.56.114]

Trash ECAP *********************************************************************
192.168.56.113             : ok=2    changed=1    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

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.

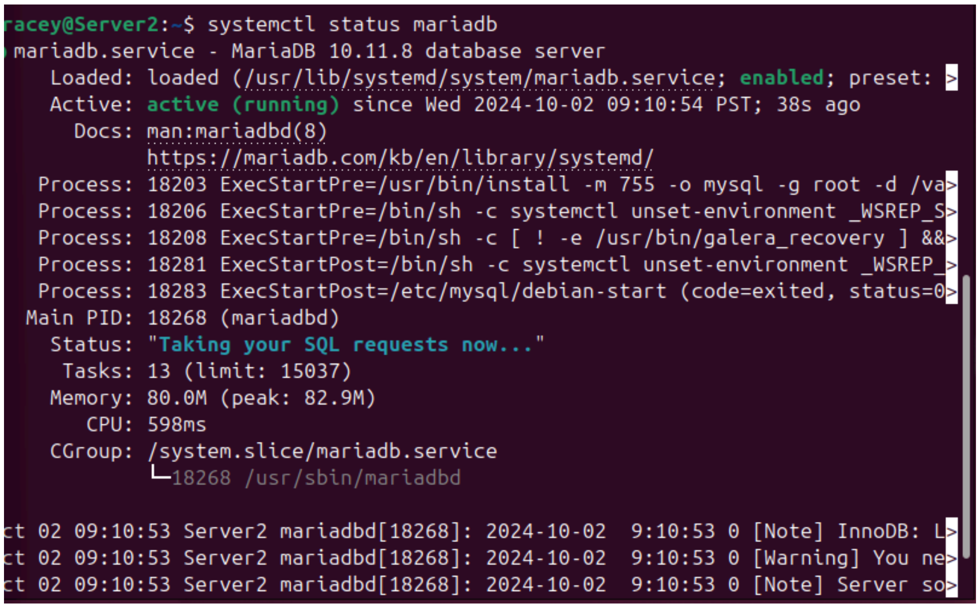
Run the *site.yml* file and describe the result.



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

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

TASK [install mariadb package (Centos)] **************************************
skipping: [192.168.56.113]

TASK [Mariadb- Restarting/Enabling] ******************************************
changed: [192.168.56.113]

TASK [install mariad package (Ubuntu)] ***************************************
ok: [192.168.56.113]

PLAY RECAP *******************************************************************
192.168.56.113             : ok=5    changed=1    unreachable=0    failed=0
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=2    changed=0    unreachable=0    failed=0
kipped=1    rescued=0    ignored=0

tracey@Workstation:~/CPE232_BRINGUELA$ S
```

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.



```
racey@Server2:~$ systemctl status mariadb
 mariadb.service - MariaDB 10.11.8 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: >
     Active: active (running) since Wed 2024-10-02 09:10:54 PST; 38s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 18203 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /va>
    Process: 18206 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_S>
    Process: 18208 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] &&>
    Process: 18281 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_>
    Process: 18283 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0>
   Main PID: 18268 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 13 (limit: 15037)
     Memory: 80.0M (peak: 82.9M)
        CPU: 598ms
     CGroup: /system.slice/mariadb.service
             └─18268 /usr/sbin/mariadbd

ct 02 09:10:53 Server2 mariadbd[18268]: 2024-10-02  9:10:53 0 [Note] InnoDB: L>
ct 02 09:10:53 Server2 mariadbd[18268]: 2024-10-02  9:10:53 0 [Warning] You ne>
ct 02 09:10:53 Server2 mariadbd[18268]: 2024-10-02  9:10:53 0 [Note] Server so>
```

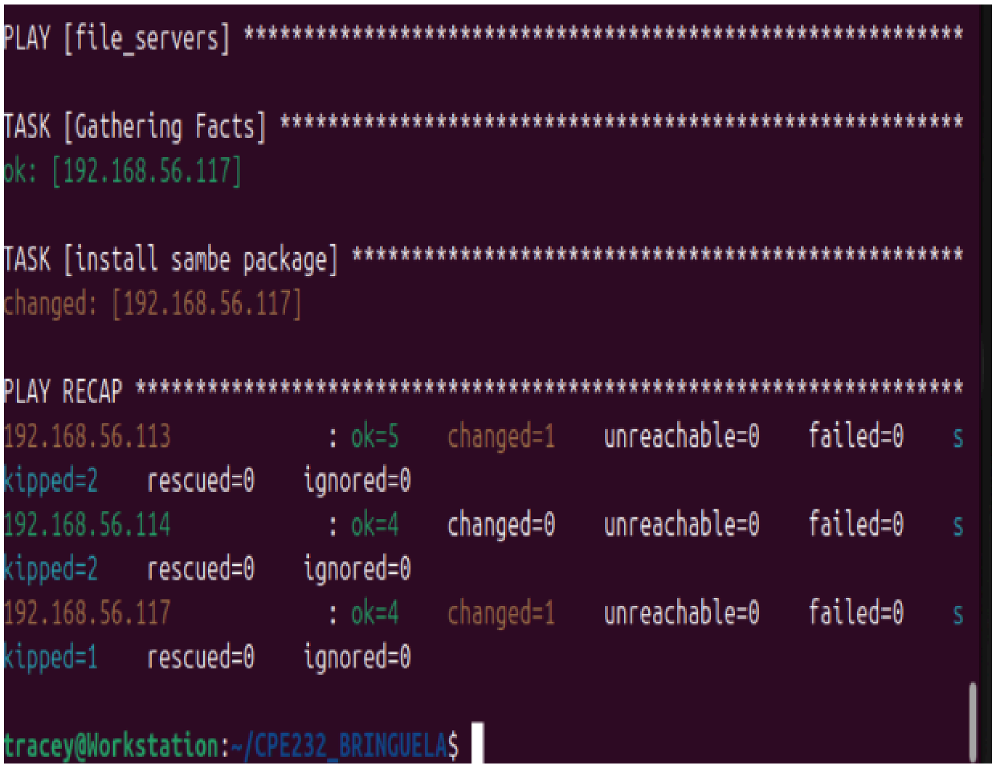
it is enabled now

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.



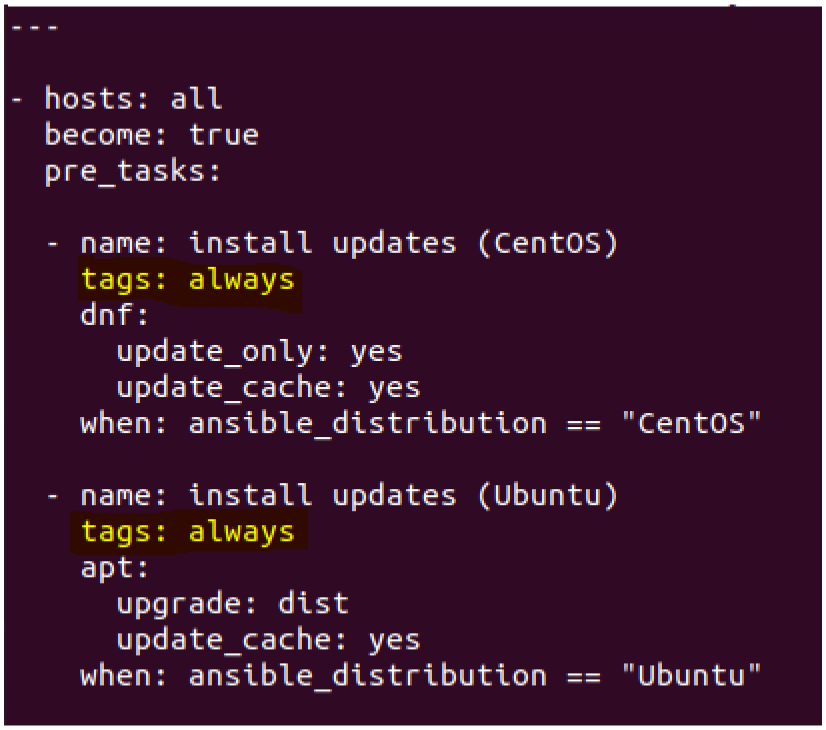
```
PLAY [file_servers] ************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.117]

TASK [install sambe package] ***************************************************
changed: [192.168.56.117]

PLAY RECAP *********************************************************************
192.168.56.113             : ok=5    changed=1    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=4    changed=1    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

tracey@Workstation:~/CPE232_BRINGUELA$
```

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.

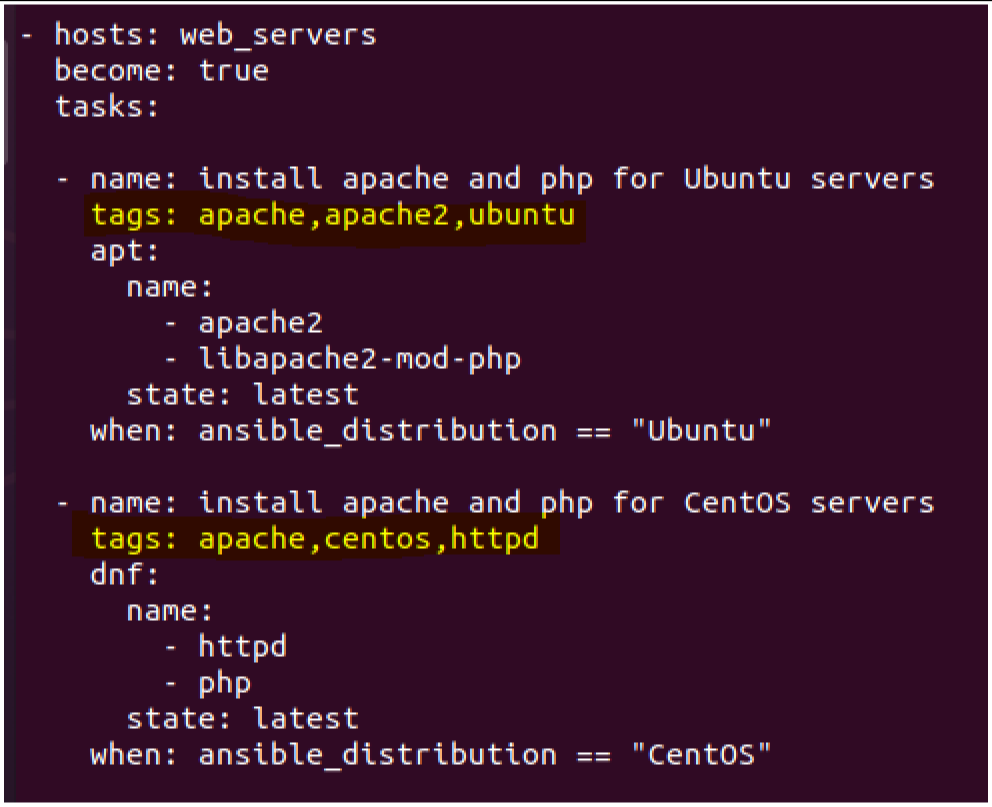
Task 2: Using Tags in running playbooks

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.

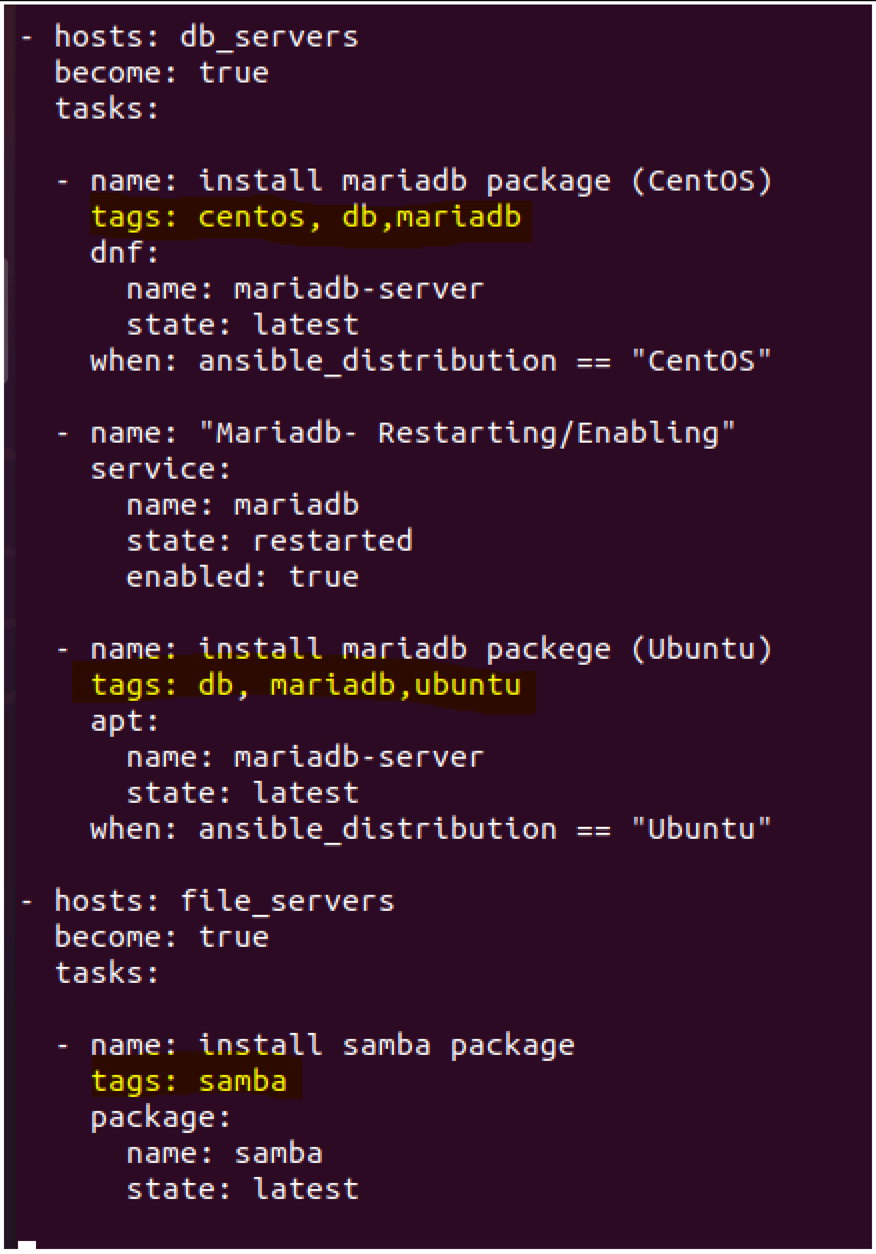
1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.



```
---

- 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"
```

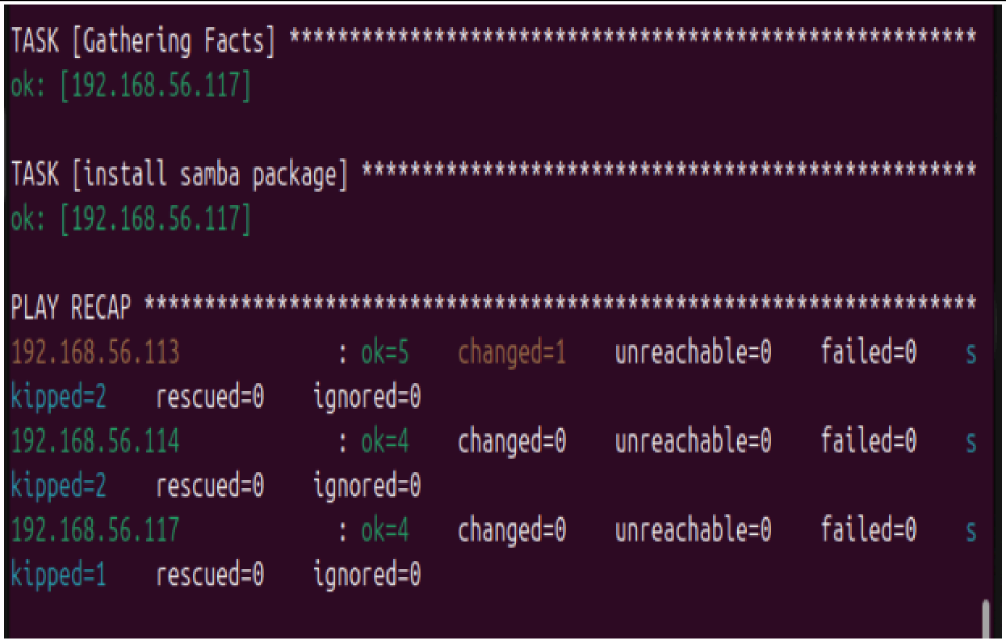


```
- hosts: web_servers
  become: true
  tasks:

  - 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"
```



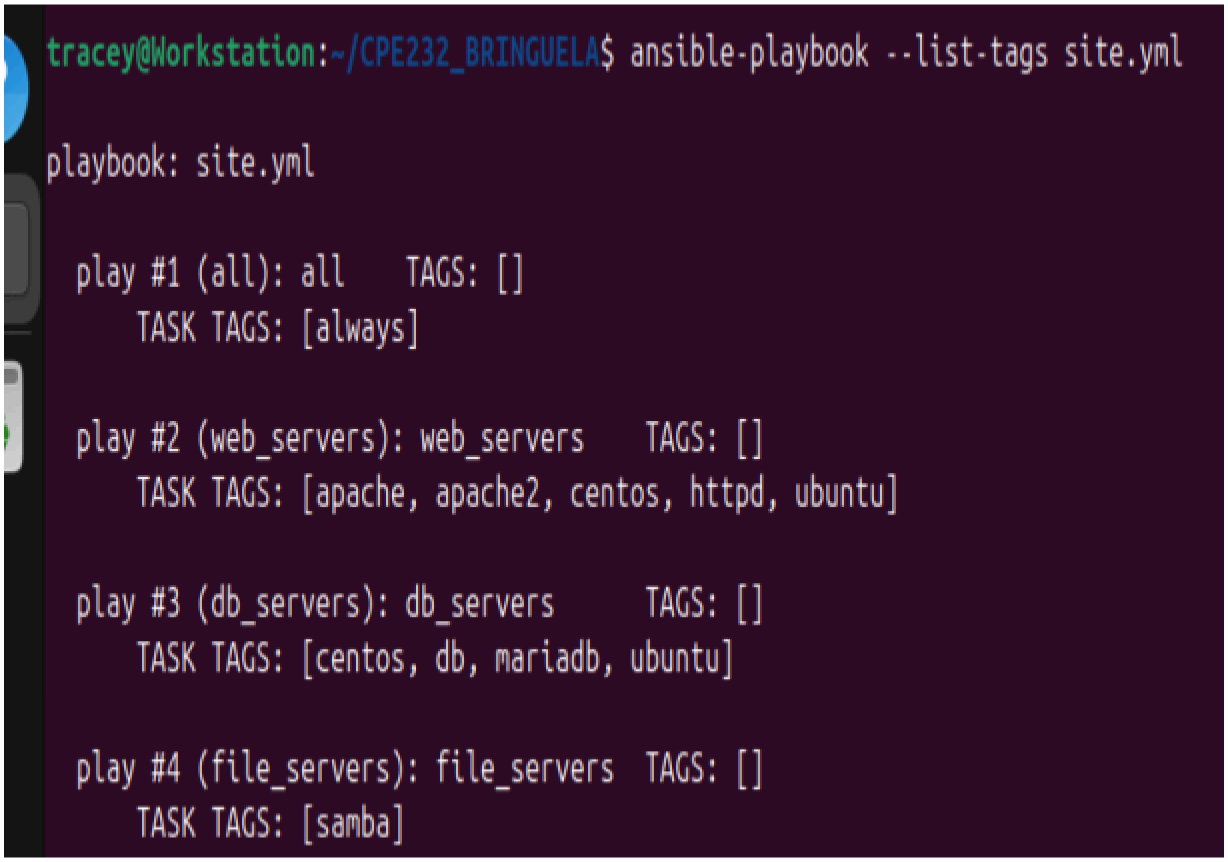
```
- hosts: db_servers
  become: true
  tasks:

  - 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.



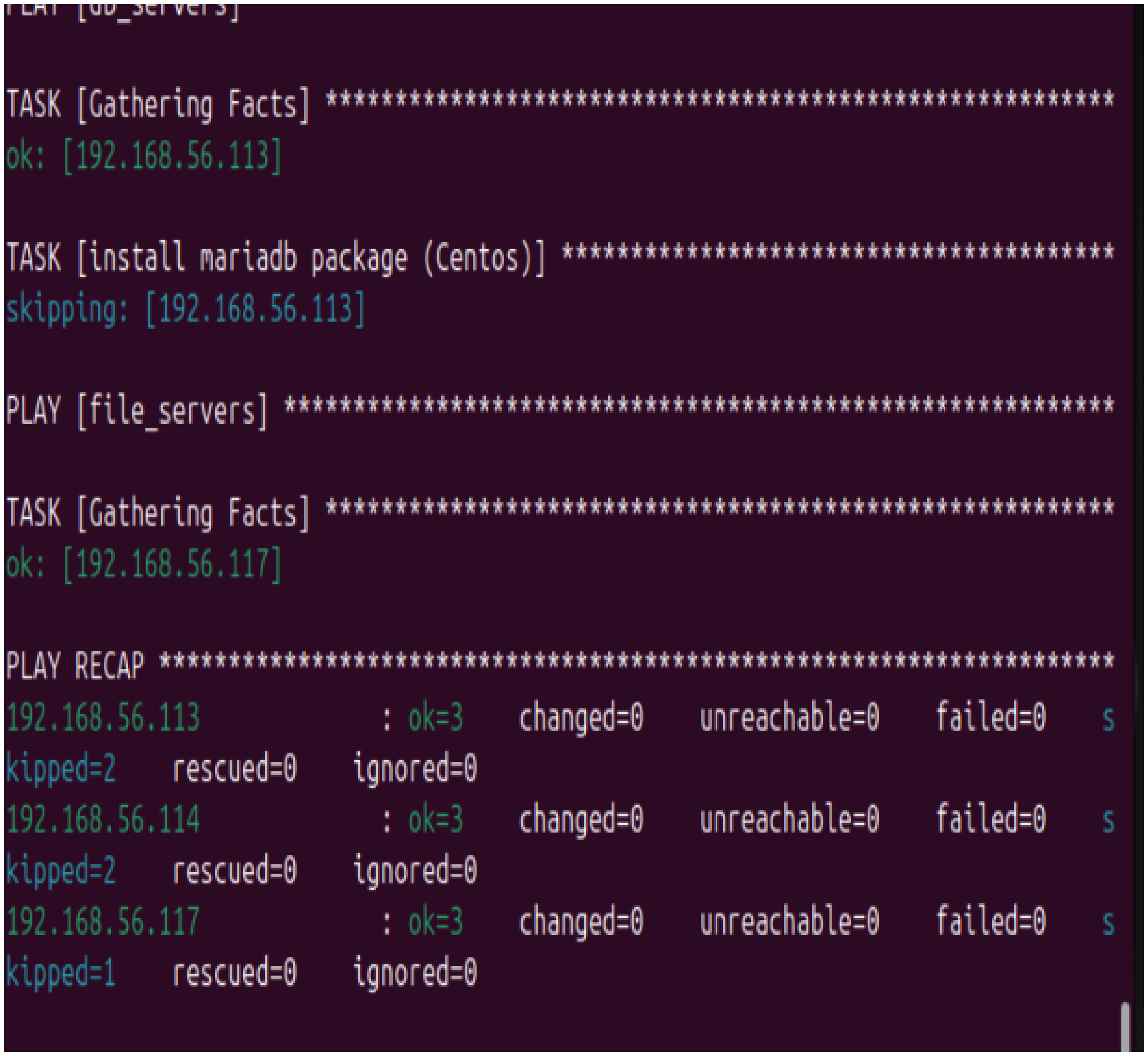
```
TASK [Gathering Facts] *************************************************************
ok: [192.168.56.117]

TASK [install samba package] ******************************************************
ok: [192.168.56.117]

PLAY RECAP ***************************************************************************
192.168.56.113             : ok=5    changed=1    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

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



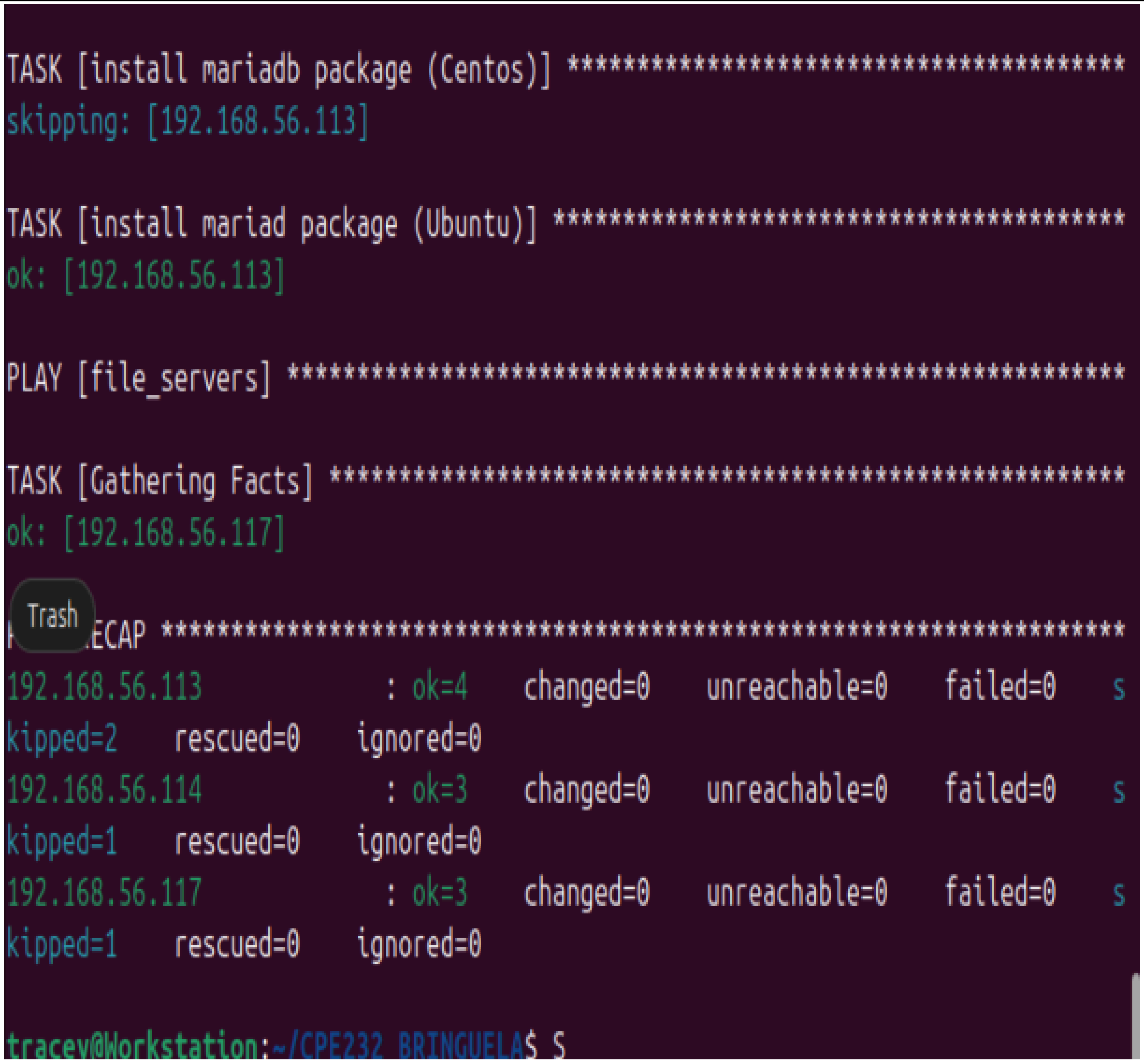
```
tracey@Workstation:~/CPE232_BRINGUELA$ ansible-playbook --list-tags site.yml

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]
```

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



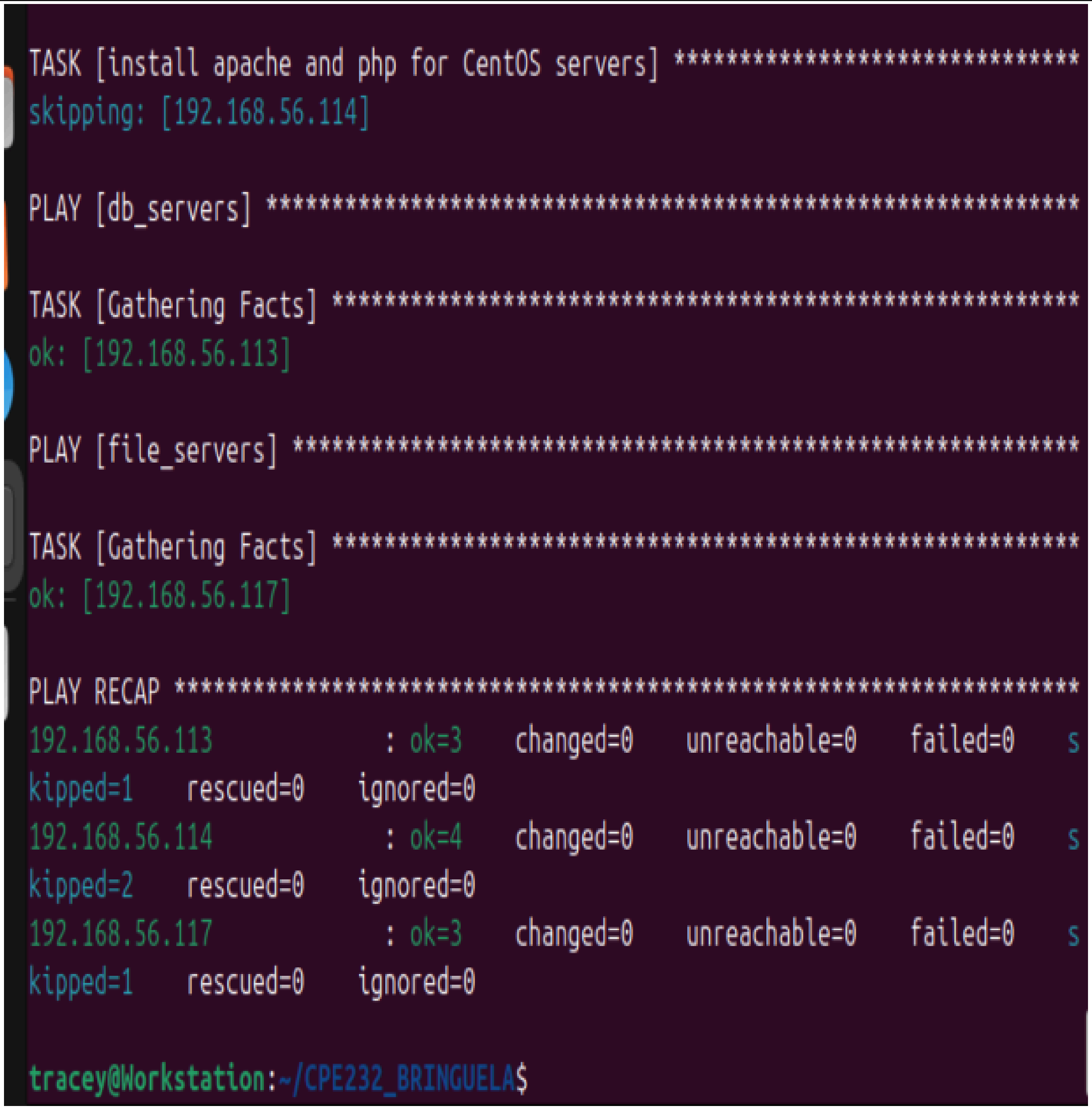
```
PLAY [db_servers]

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

TASK [install mariadb package (Centos)] *****************************************
skipping: [192.168.56.113]

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

TASK [Gathering Facts] **********************************************************
ok: [192.168.56.117]

PLAY RECAP **********************************************************************
192.168.56.113             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

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



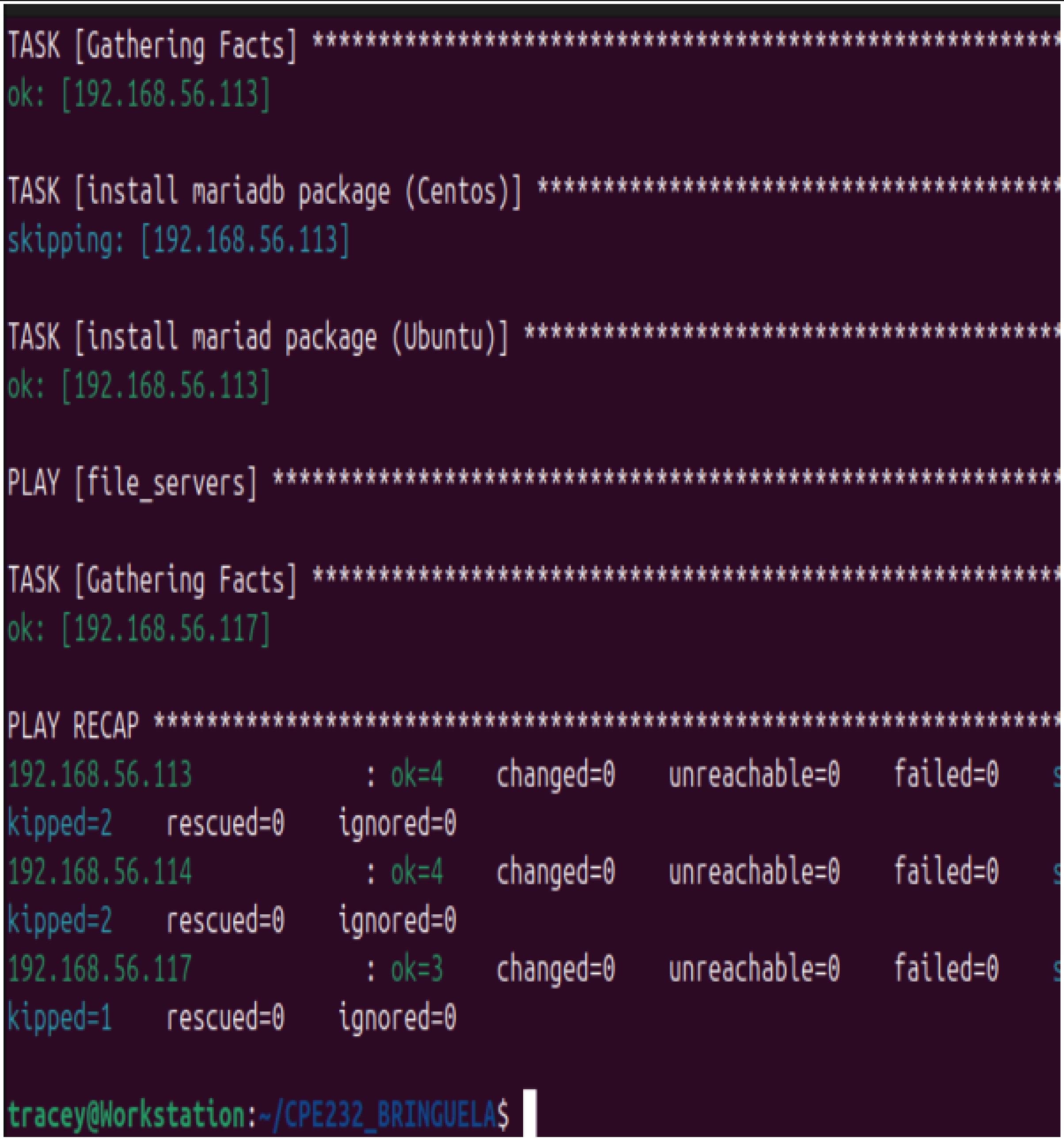
```
TASK [install mariadb package (Centos)] ****************************************
skipping: [192.168.56.113]

TASK [install mariad package (Ubuntu)] *****************************************
ok: [192.168.56.113]

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.117]

Trash ECAP *********************************************************************
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.117             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

tracev@Workstation:~/CPE232_BRINGUELA$ $
```

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



```
TASK [install apache and php for CentOS servers] ********************************
skipping: [192.168.56.114]

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

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

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

TASK [Gathering Facts] **********************************************************
ok: [192.168.56.117]

PLAY RECAP **********************************************************************
192.168.56.113             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

tracey@Workstation:~/CPE232_BRINGUELA$
```

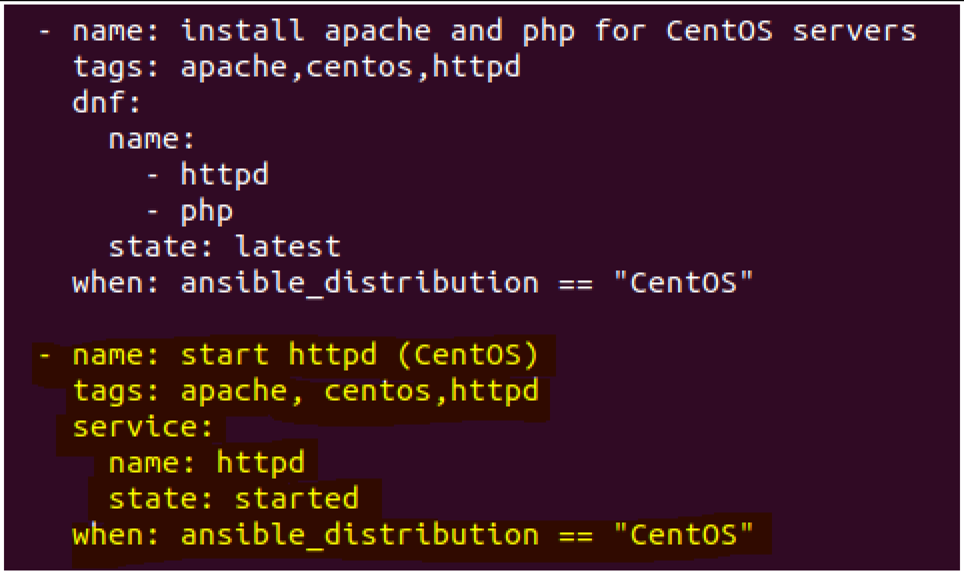
2.5 *ansible-playbook --tags “apache,db” --ask-become-pass site.yml*



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

TASK [install mariadb package (Centos)] *************************************
skipping: [192.168.56.113]

TASK [install mariad package (Ubuntu)] **************************************
ok: [192.168.56.113]

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

TASK [Gathering Facts] ******************************************************
ok: [192.168.56.117]

PLAY RECAP ******************************************************************
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.114             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.117             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0

tracey@Workstation:~/CPE232_BRINGUELA$
```

Task 3: Managing Services

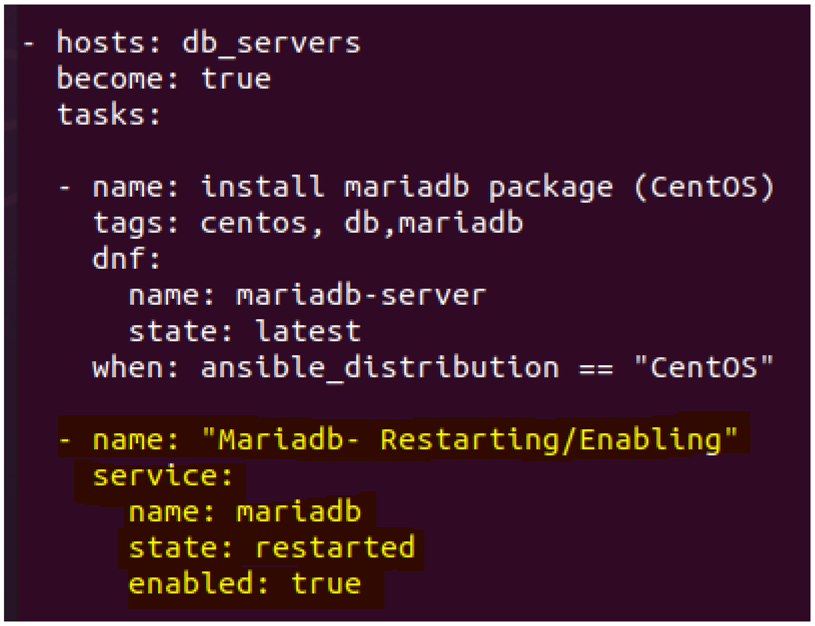
1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.



```
- 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.

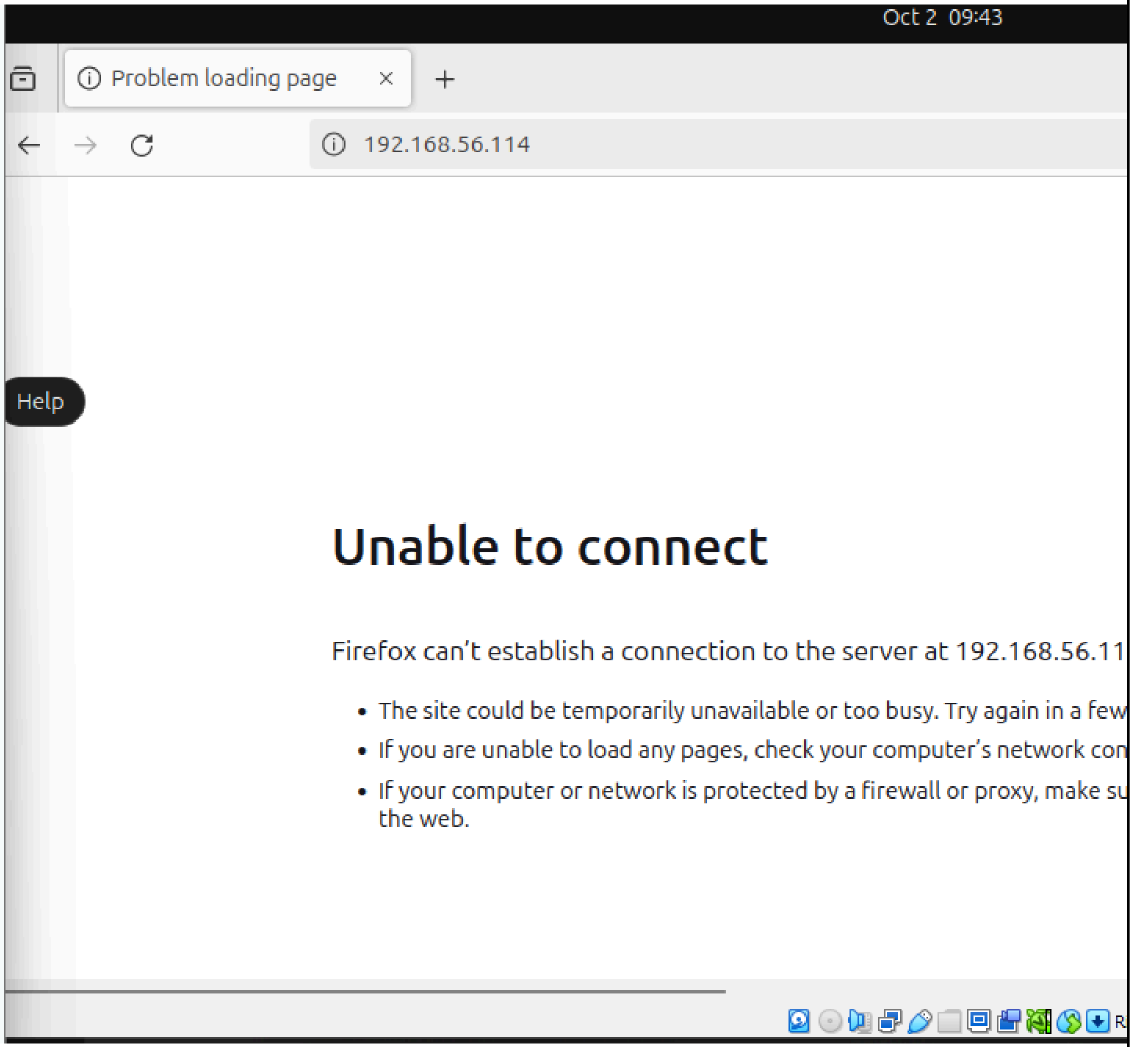


```
- hosts: db_servers
  become: true
  tasks:

  - 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.

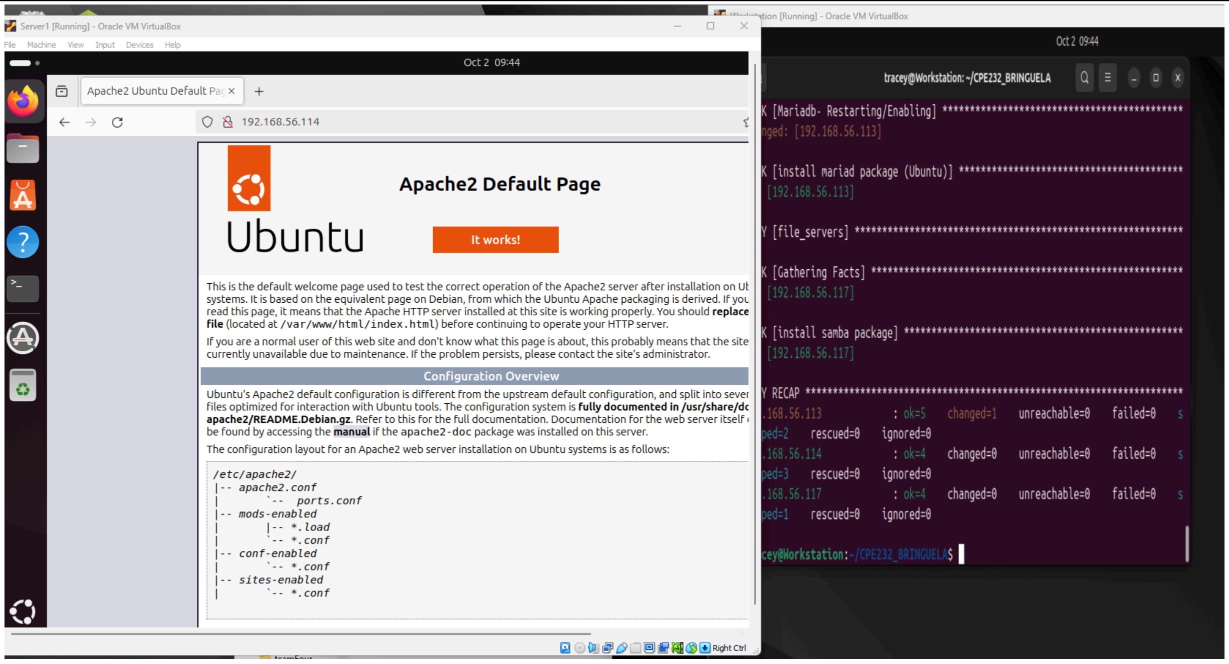


Unable to connect

Firefox can't establish a connection to the server at 192.168.56.11

- The site could be temporarily unavailable or too busy. Try again in a few
- If you are unable to load any pages, check your computer's network con
- If your computer or network is protected by a firewall or proxy, make su 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.
   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.


Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ub systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into sever files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/dc apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

```
K [Mariadb- Restarting/Enabling] *****************************************
nged: [192.168.56.113]

K [install mariad package (Ubuntu)] **************************************
 [192.168.56.113]

Y [file_servers] *********************************************************

K [Gathering Facts] ******************************************************
 [192.168.56.117]

K [install samba package] ************************************************
 [192.168.56.117]

Y RECAP ******************************************************************
.168.56.113          : ok=5    changed=1    unreachable=0    failed=0    s
ped=2    rescued=0    ignored=0
.168.56.114          : ok=4    changed=0    unreachable=0    failed=0    s
ped=3    rescued=0    ignored=0
.168.56.117          : ok=4    changed=0    unreachable=0    failed=0    s
ped=1    rescued=0    ignored=0

cey@Workstation:~/CPE232_BRINGUELA$
```

Reflections:

Answer the following:

1. What is the importance of putting our remote servers into groups?
   Grouping remote servers allows you to efficiently manage and organize servers with similar roles or configurations. This makes it easier to apply specific tasks or configurations to a targeted subset of servers, improving scalability. It also simplifies maintenance and updates by allowing for consistent changes across grouped systems.

2. What is the importance of tags in playbooks?
   Tags allow you to selectively execute specific tasks within a playbook without running the entire set of tasks. This saves time and resources, especially in large playbooks, by focusing on only the tasks you need. Tags also make debugging and testing more efficient, as you can isolate sections of the playbook.

3. Why do think some services need to be managed automatically in playbooks?
   Automating the management of services ensures consistent and reliable operations, reduces the risk of human error, and enables faster recovery or scaling during changes or failures.