Name: ESCOSIA, JERICO JAMES	**Date Performed: 10-4-24**
Course/Section: CPE212- CPE31S21	**Date Submitted: 10-4-24**
Instructor: Engr. 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 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.

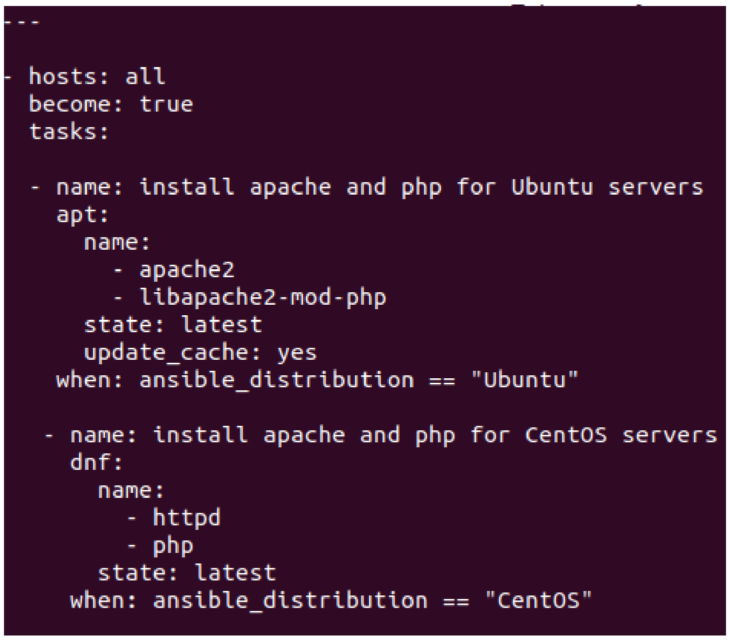
We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.

Requirement:

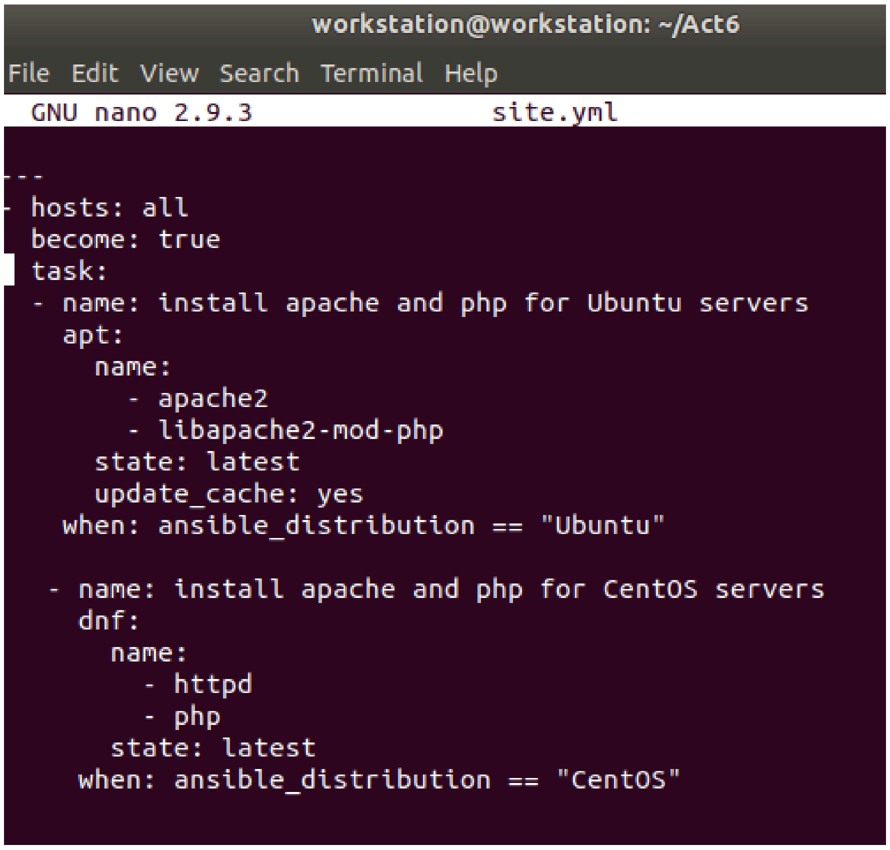
In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command *ssh-copy-id* to copy the public key to Server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

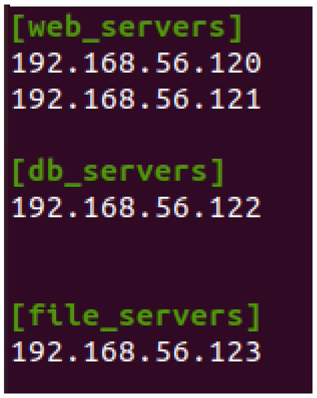


```
---

- 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"
```



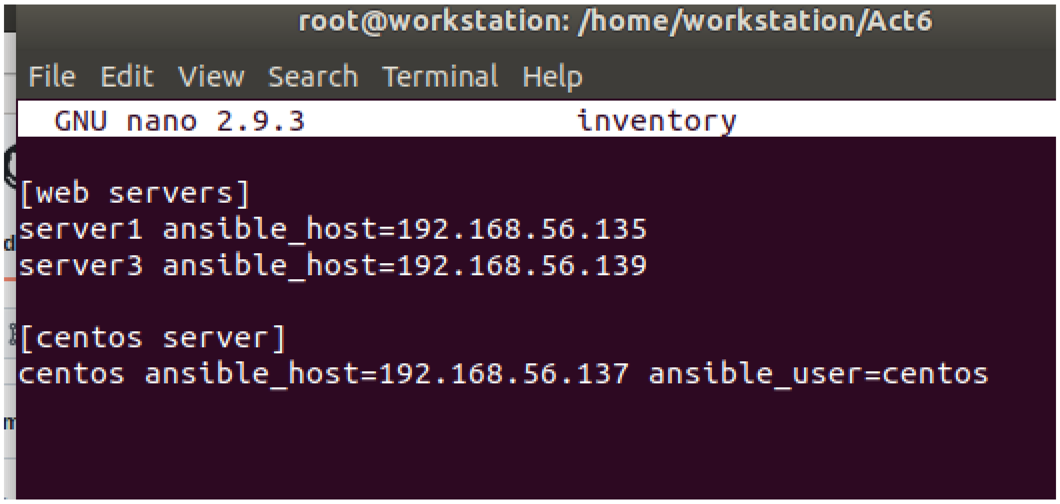
```
workstation@workstation: ~/Act6
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                  site.yml

---
- hosts: all
  become: true
  task:
  - 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
```

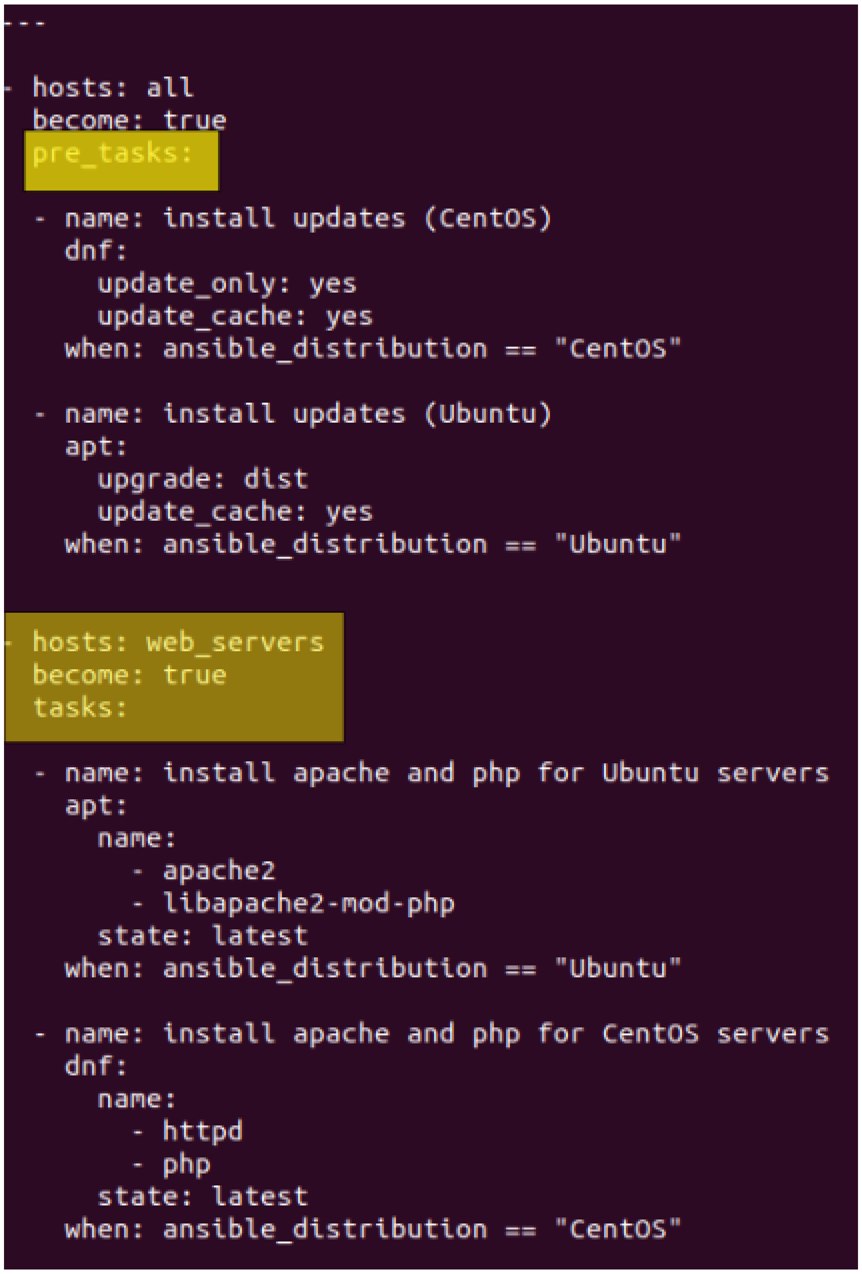
Make sure to save the file and exit.


root@workstation: /home/workstation/Act6
File Edit View Search Terminal Help

```
  GNU nano 2.9.3                    inventory

[web servers]
server1 ansible_host=192.168.56.135
server3 ansible_host=192.168.56.139

[centos server]
centos ansible_host=192.168.56.137 ansible_user=centos
```

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:

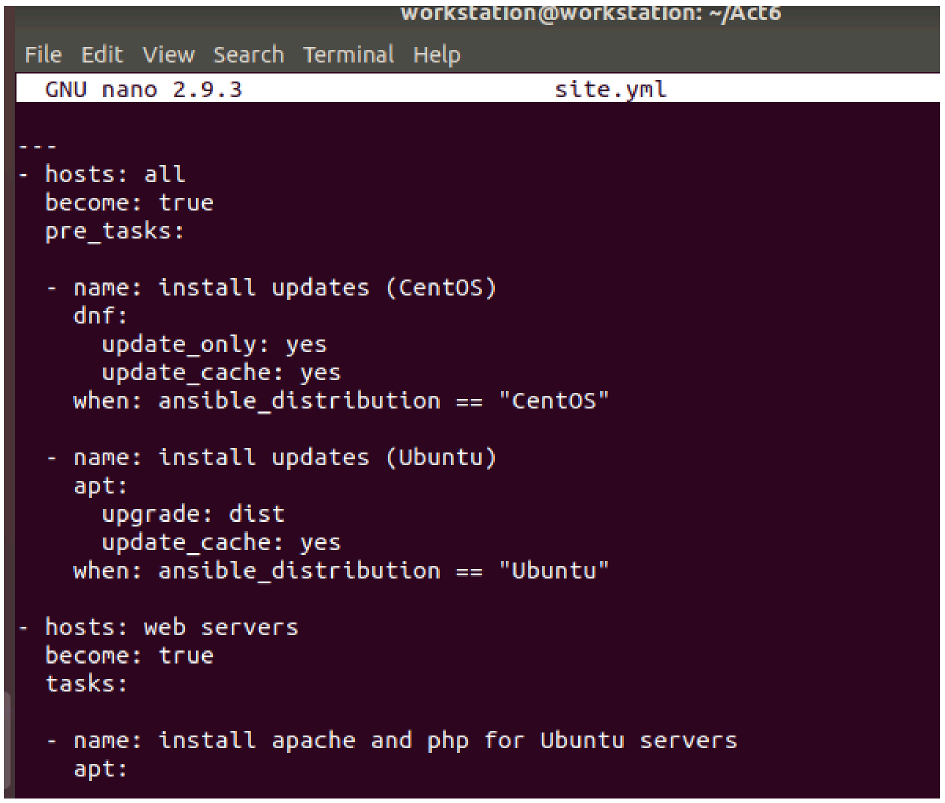


```
---

- 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.

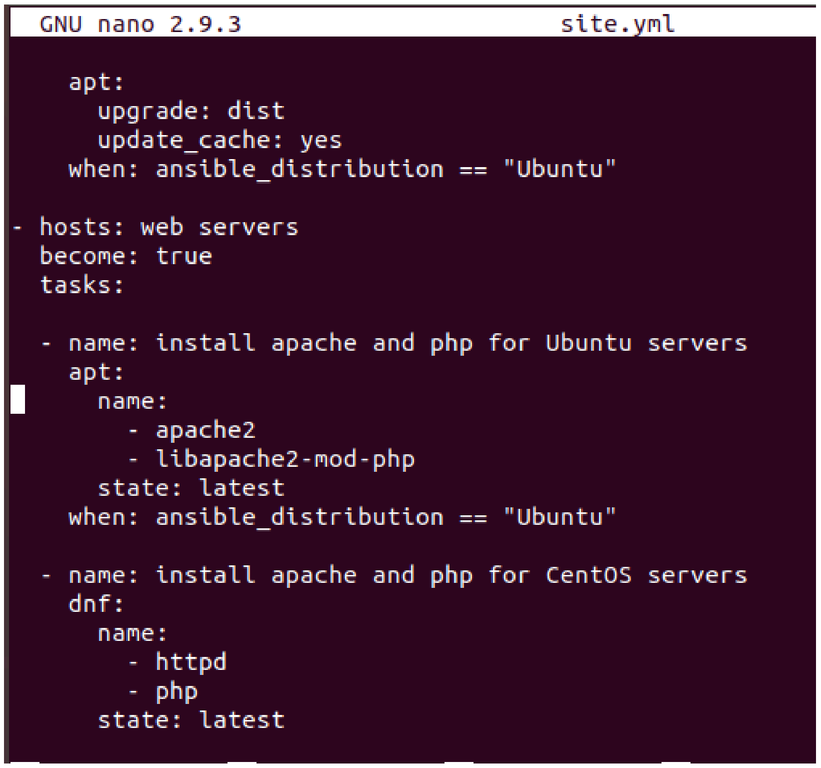
Run the *site.yml* file and describe the result.



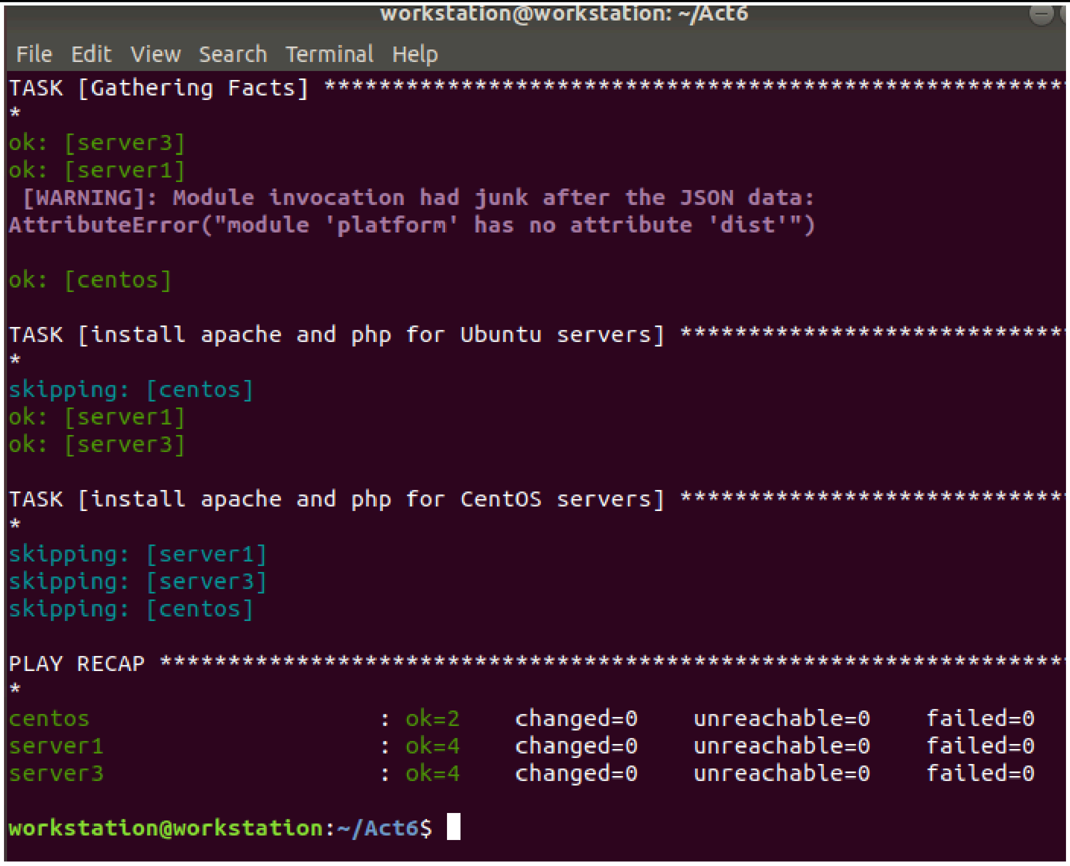
workstation@workstation: ~/Act6

File Edit View Search Terminal Help

```
  GNU nano 2.9.3                                  site.yml

---
- 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:
```



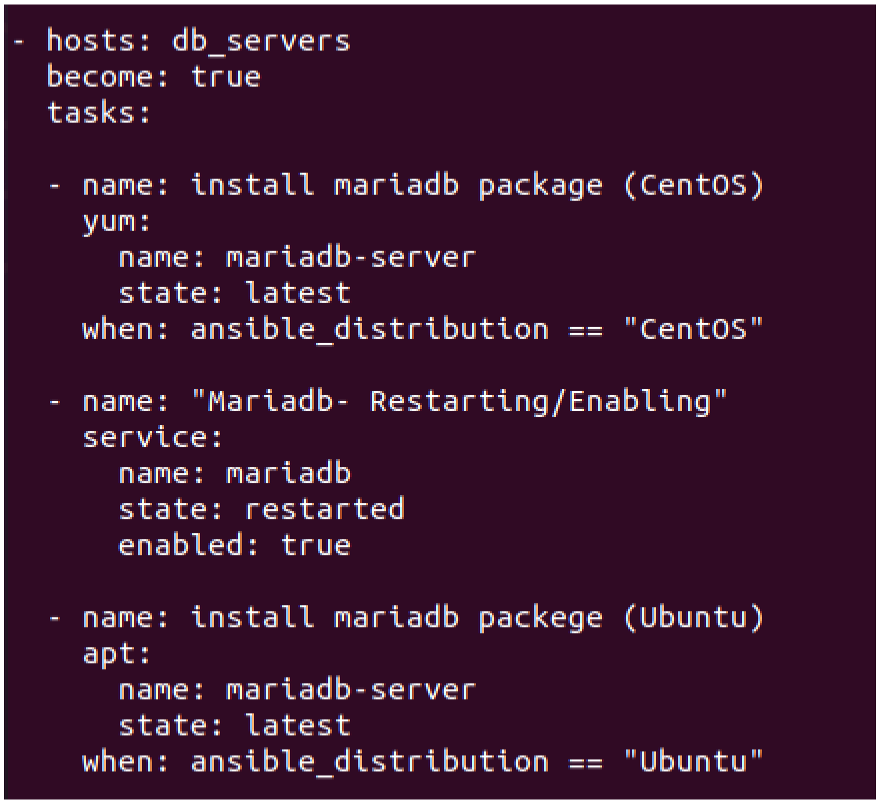
```
  GNU nano 2.9.3                                  site.yml

    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
```


workstation@workstation: ~/Act6
File Edit View Search Terminal Help

```
TASK [Gathering Facts] *******************************************************
*
ok: [server3]
ok: [server1]
 [WARNING]: Module invocation had junk after the JSON data:
AttributeError("module 'platform' has no attribute 'dist'")

ok: [centos]

TASK [install apache and php for Ubuntu servers] *****************************
*
skipping: [centos]
ok: [server1]
ok: [server3]

TASK [install apache and php for CentOS servers] *****************************
*
skipping: [server1]
skipping: [server3]
skipping: [centos]

PLAY RECAP *******************************************************************
*
centos                     : ok=2    changed=0    unreachable=0    failed=0
server1                    : ok=4    changed=0    unreachable=0    failed=0
server3                    : ok=4    changed=0    unreachable=0    failed=0

workstation@workstation:~/Act6$ 
```

In this command I was able to install php and apache on CentOS and Ubuntu machines at the same time.

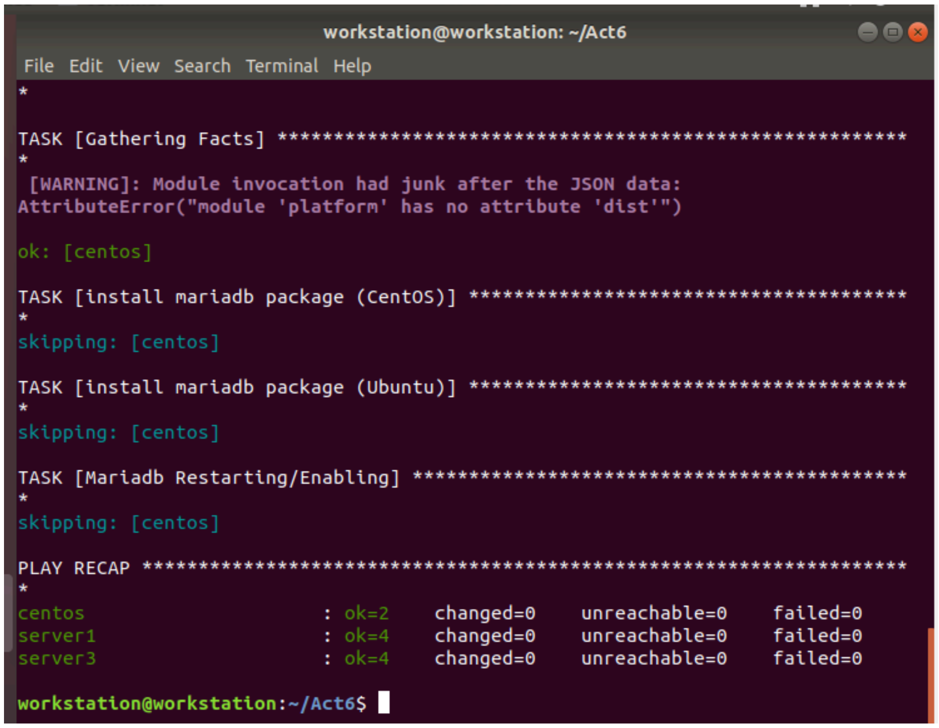
4. Let’s try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db_servers* by adding it on the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3.



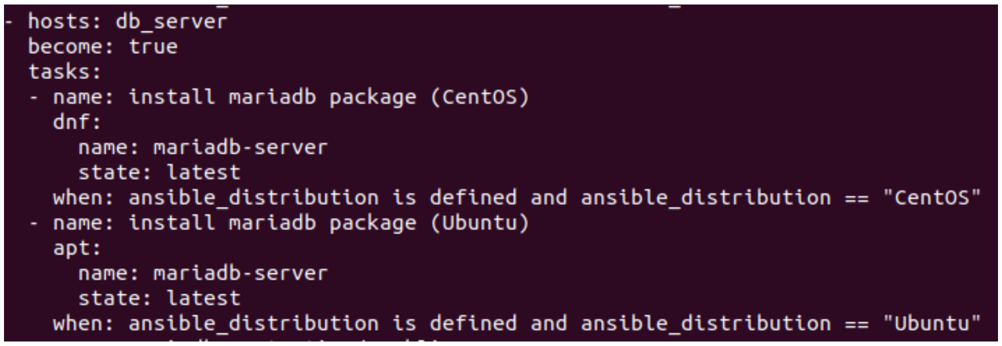
```
- hosts: db_servers
  become: true
  tasks:

  - 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.



```
workstation@workstation: ~/Act6
File  Edit  View  Search  Terminal  Help
*

TASK [Gathering Facts] *******************************************************
*
 [WARNING]: Module invocation had junk after the JSON data:
AttributeError("module 'platform' has no attribute 'dist'")

ok: [centos]

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

TASK [install mariadb package (Ubuntu)] ***************************************
*
skipping: [centos]

TASK [Mariadb Restarting/Enabling] ********************************************
*
skipping: [centos]

PLAY RECAP *******************************************************************
*
centos                     : ok=2    changed=0    unreachable=0    failed=0
server1                    : ok=4    changed=0    unreachable=0    failed=0
server3                    : ok=4    changed=0    unreachable=0    failed=0

workstation@workstation:~/Act6$ 
```

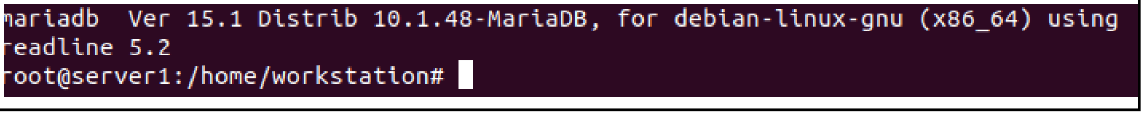


```
- hosts: db_server
  become: true
  tasks:
  - name: install mariadb package (CentOS)
    dnf:
      name: mariadb-server
      state: latest
    when: ansible_distribution is defined and ansible_distribution == "CentOS"
  - name: install mariadb package (Ubuntu)
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution is defined and ansible_distribution == "Ubuntu"
```

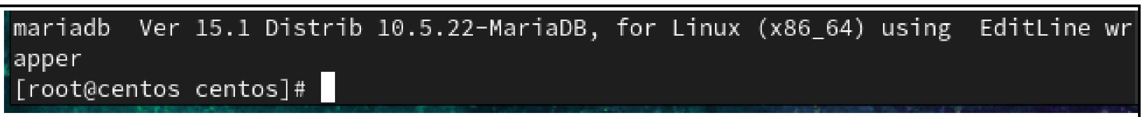
In this part we installed the mariadb package in both centos and ubuntu using ansible, the tasks is specified depending on the operation system hence the the skipping of task in the terminal.

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.



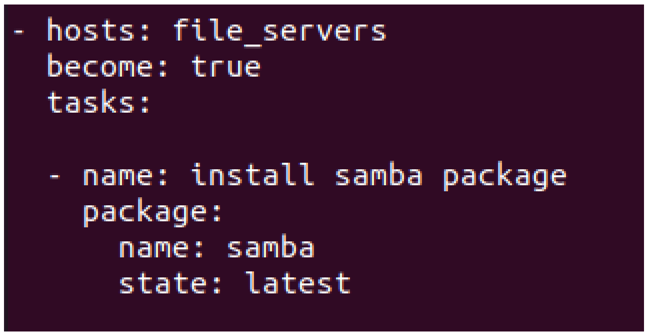
```
mariadb  Ver 15.1 Distrib 10.1.48-MariaDB, for debian-linux-gnu (x86_64) using
readline 5.2
root@server1:/home/workstation# 
```



```
mariadb  Ver 15.1 Distrib 10.5.22-MariaDB, for Linux (x86_64) using  EditLine wr
apper
[root@centos centos]# 
```

In the manage nodes you can see the latest version of mariadb upon installing it using ansible.

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
```

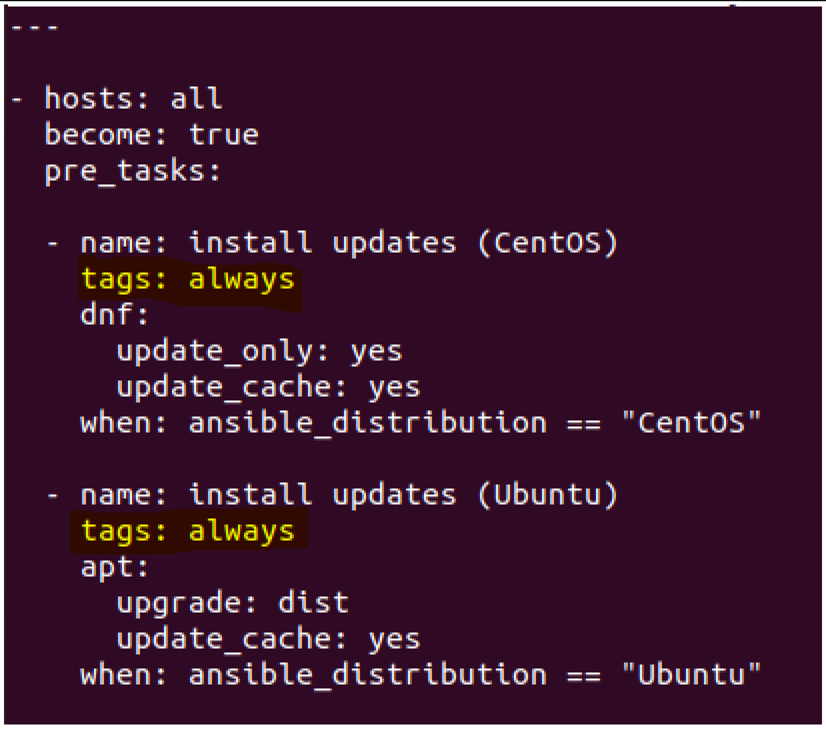
Make sure to save the file and exit.

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

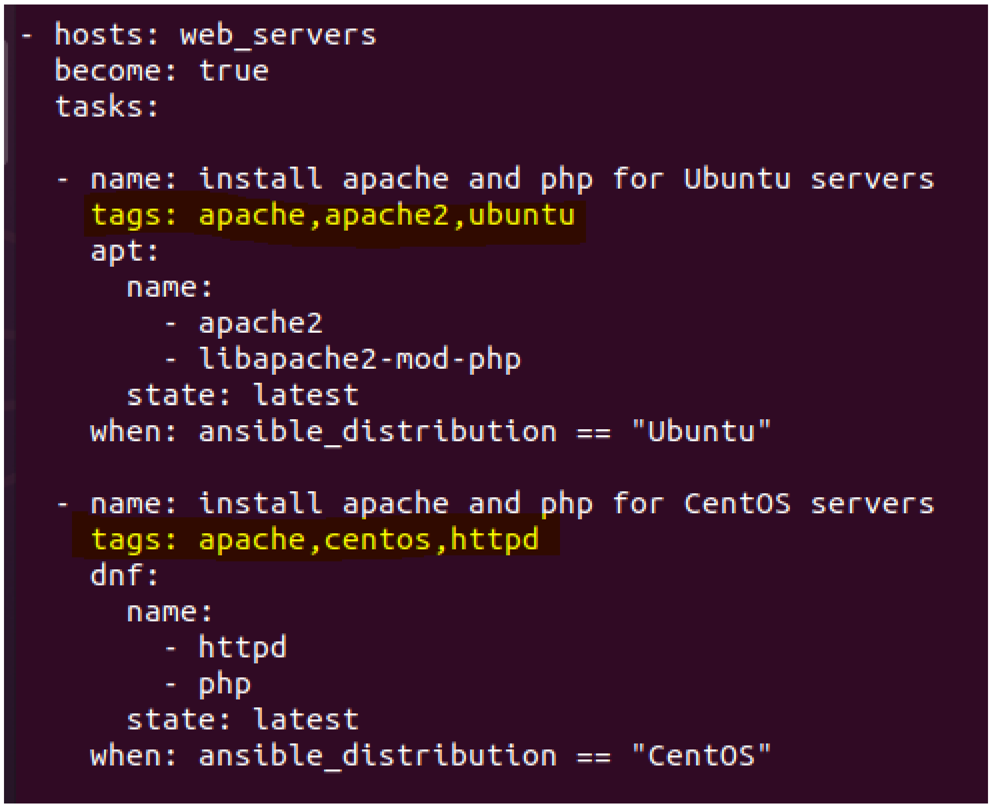
The testing of the *file_servers* is beyond the scope of this activity, and as well as our topics and objectives. However, in this activity we were able to show that we can target hosts or servers using grouping in ansible playbooks.

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.

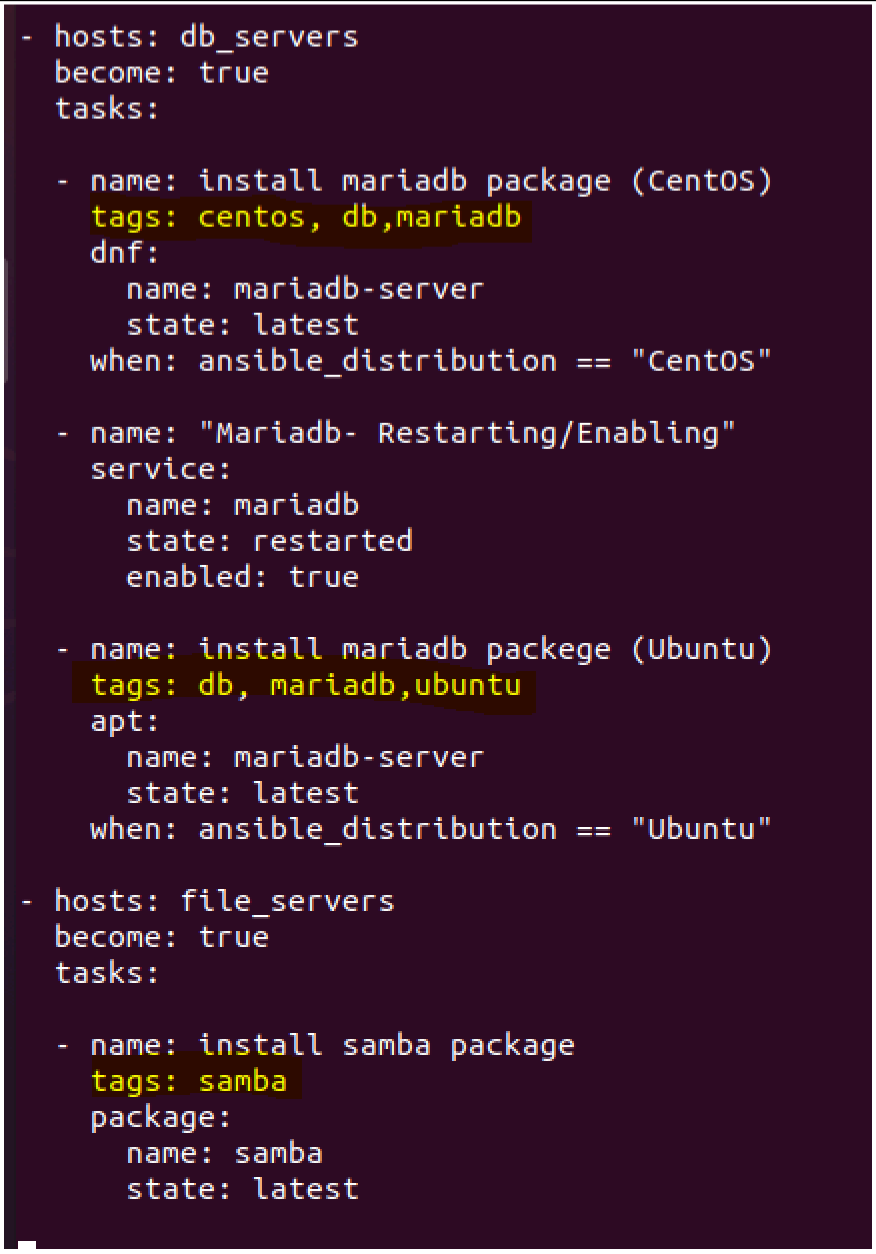
1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.



```
---

- 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*

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

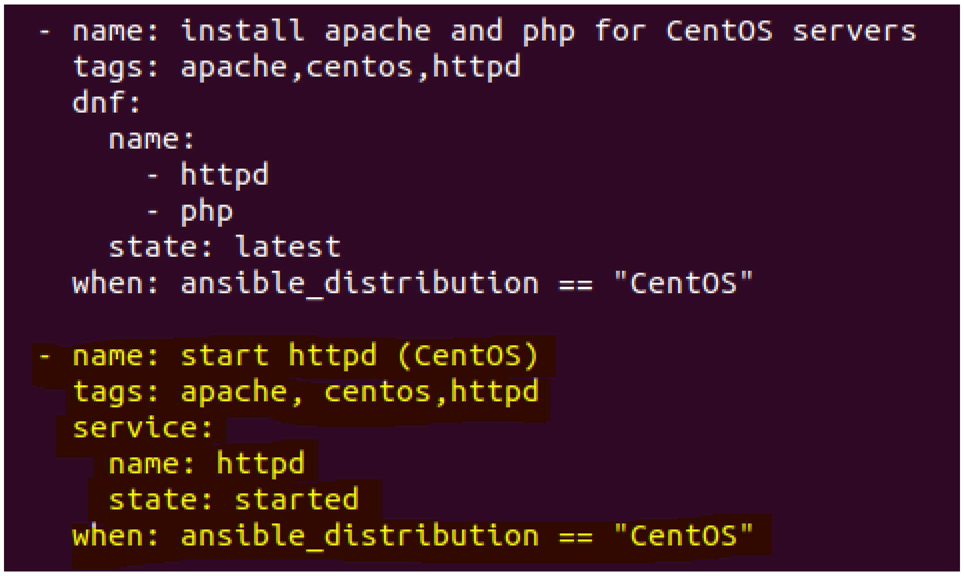
   2.3 *ansible-playbook --tags db --ask-become-pass site.yml*

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

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

Task 3: Managing Services

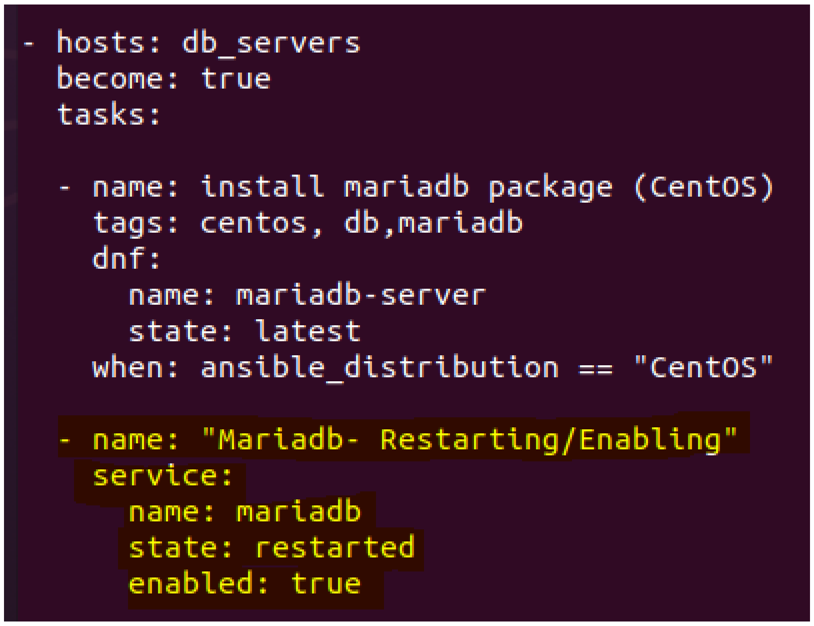
1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.



```
- 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.
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.

Reflections:

Answer the following:

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

   -By putting the remote servers into groups it makes it easier to identify what servers do, it also makes it easier to target that group of servers and do different tasks specifically for the server.
2. What is the importance of tags in playbooks?
   - Tags in playbooks are crucial as they enable selective execution of specific tasks or groups of tasks within a playbook.
3. Why do think some services need to be managed automatically in playbooks?
   - In a larger network management, remotely controlling all the devices in the network is a big help for system admins because they can configure the devices even if they are not on the device physically.