Name: Calderon Ricardo B.	**Date Performed:03/04/2024**
Course/Section: CPE 232-CPE31S1	**Date Submitted:03/05/2024**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st sem/2023-2024**

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.

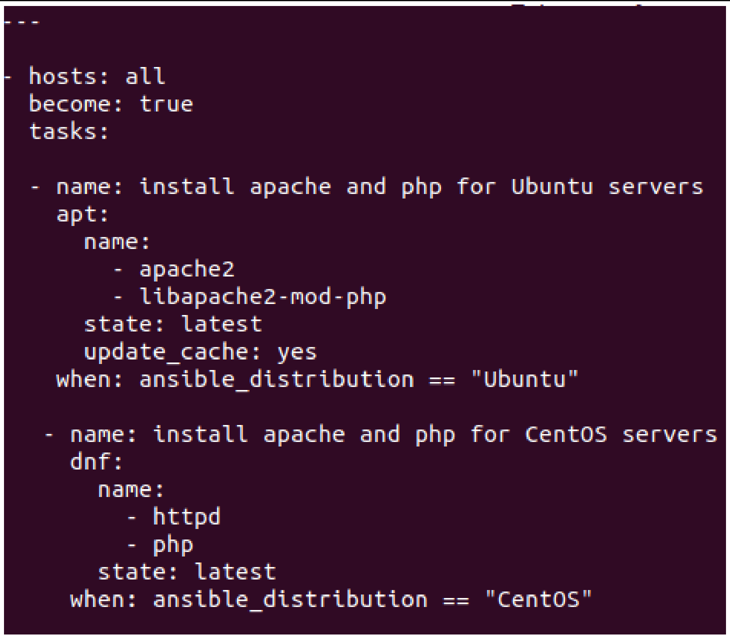
We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.

Requirement:

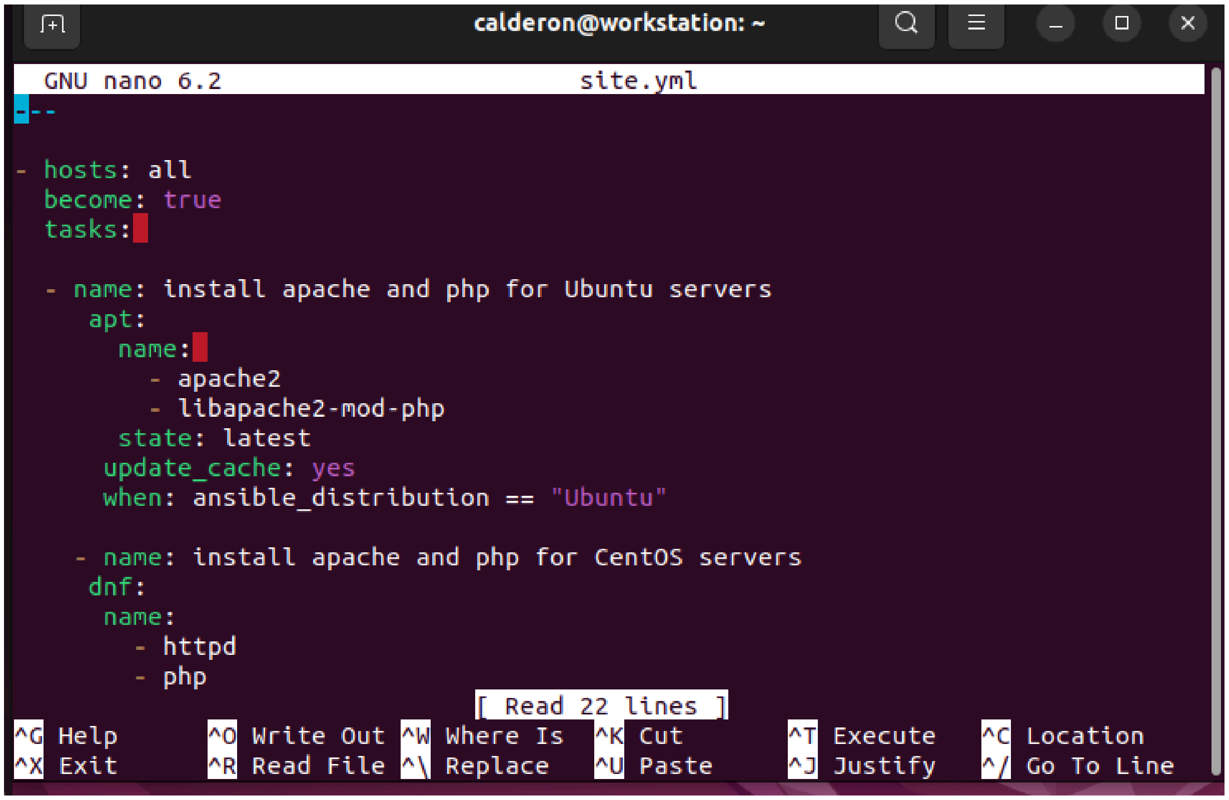
In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command *ssh-copy-id* to copy the public key to Server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

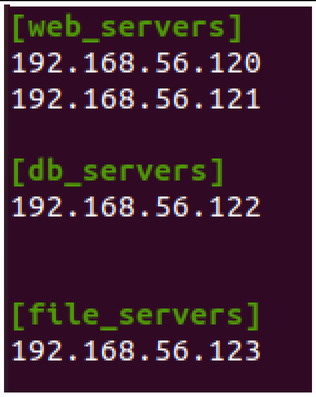


```
---

- 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"
```

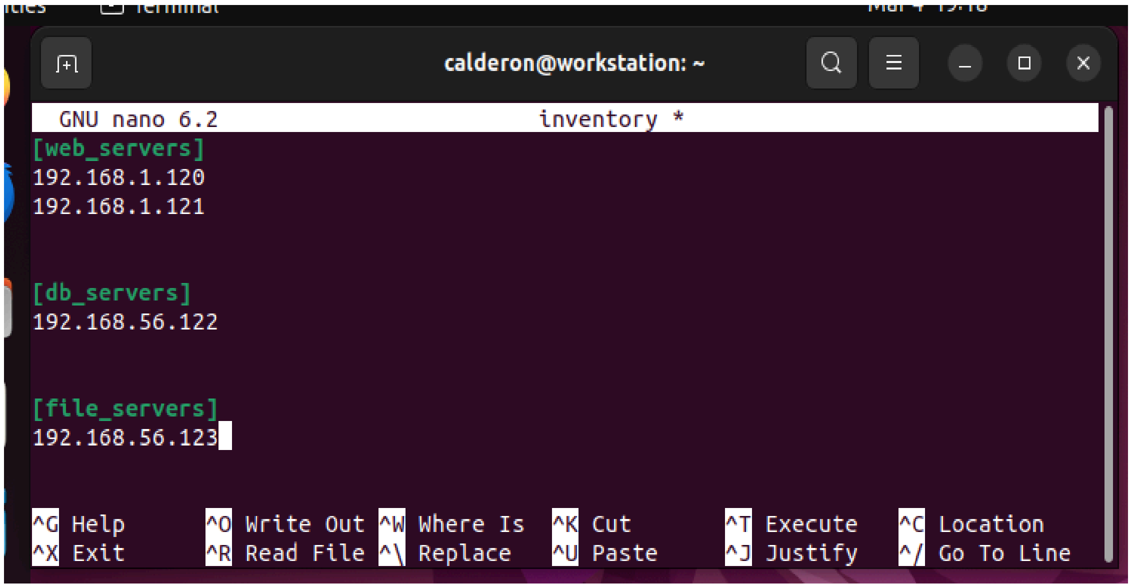


```
GNU nano 6.2                        site.yml
---

- 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: install 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
```

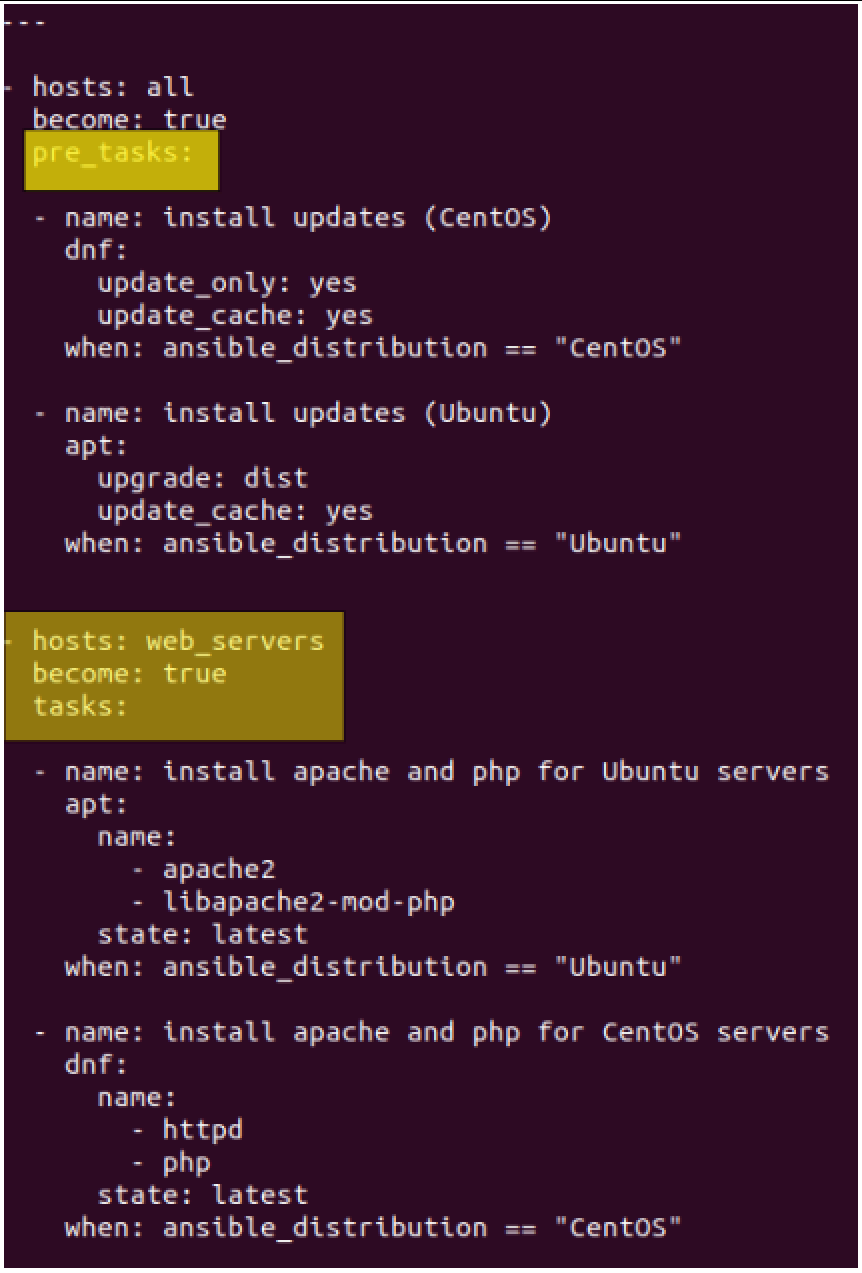


```
  GNU nano 6.2                        inventory *
[web_servers]
192.168.1.120
192.168.1.121


[db_servers]
192.168.56.122


[file_servers]
192.168.56.123

^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
```

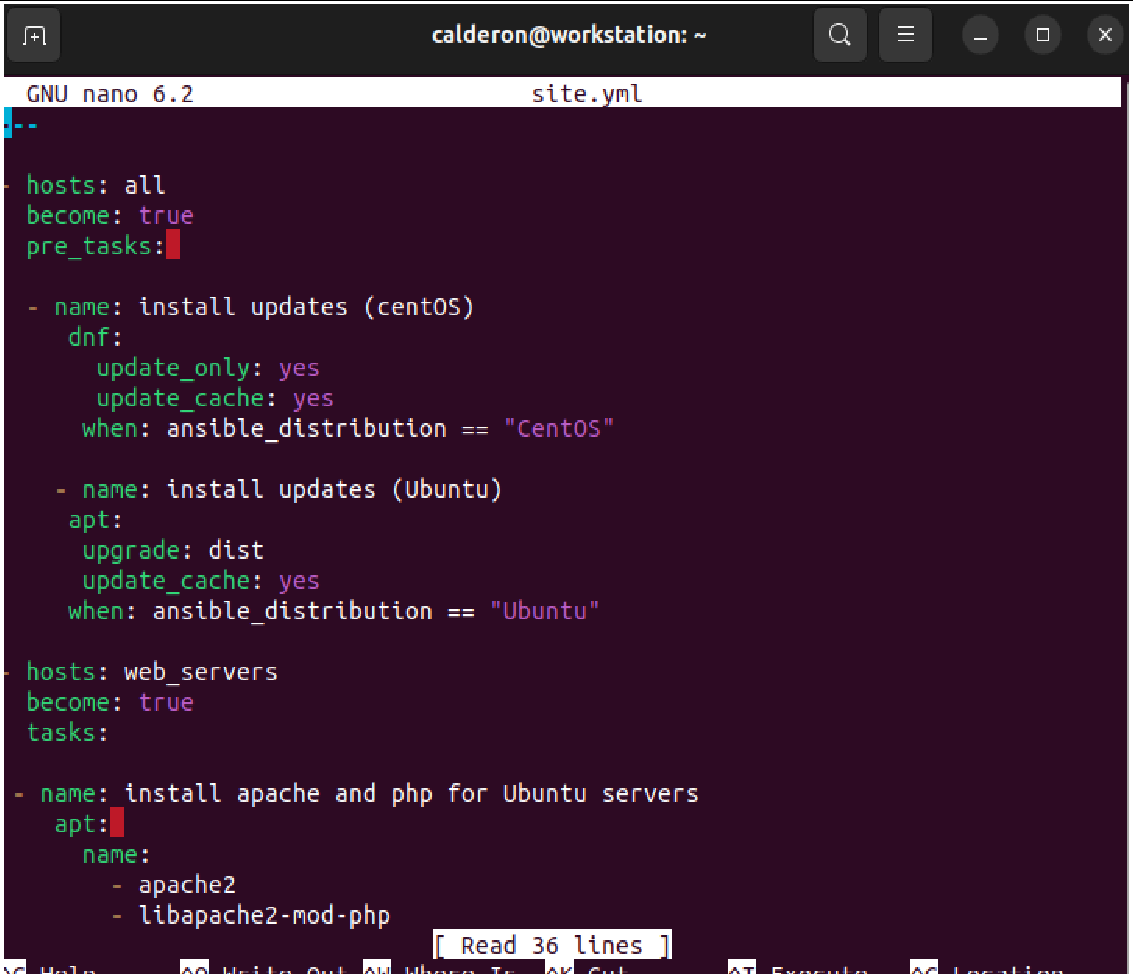
Make sure to save the file and exit.

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:



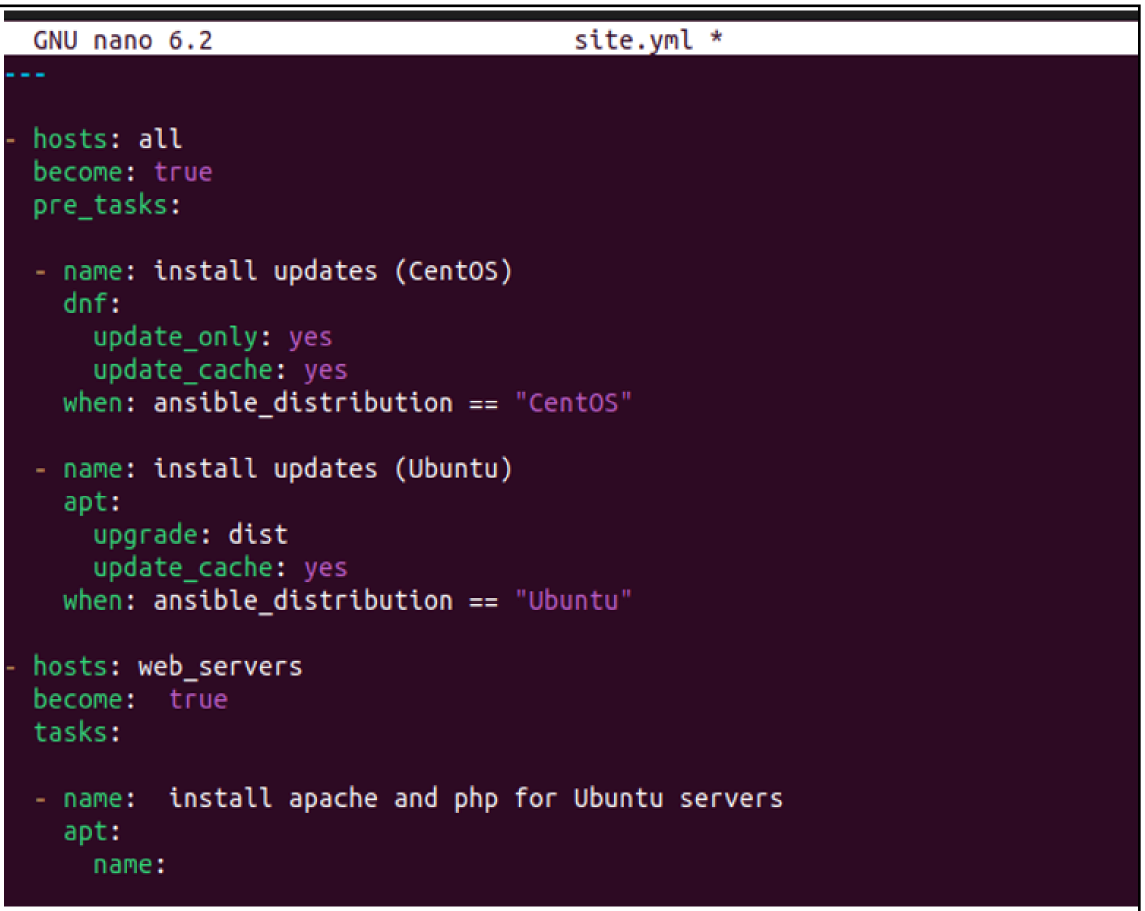
```
---

- 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"
```



calderon@workstation: ~

```
 GNU nano 6.2                        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:
      name:
        - apache2
        - libapache2-mod-php
                              [ Read 36 lines ]
```

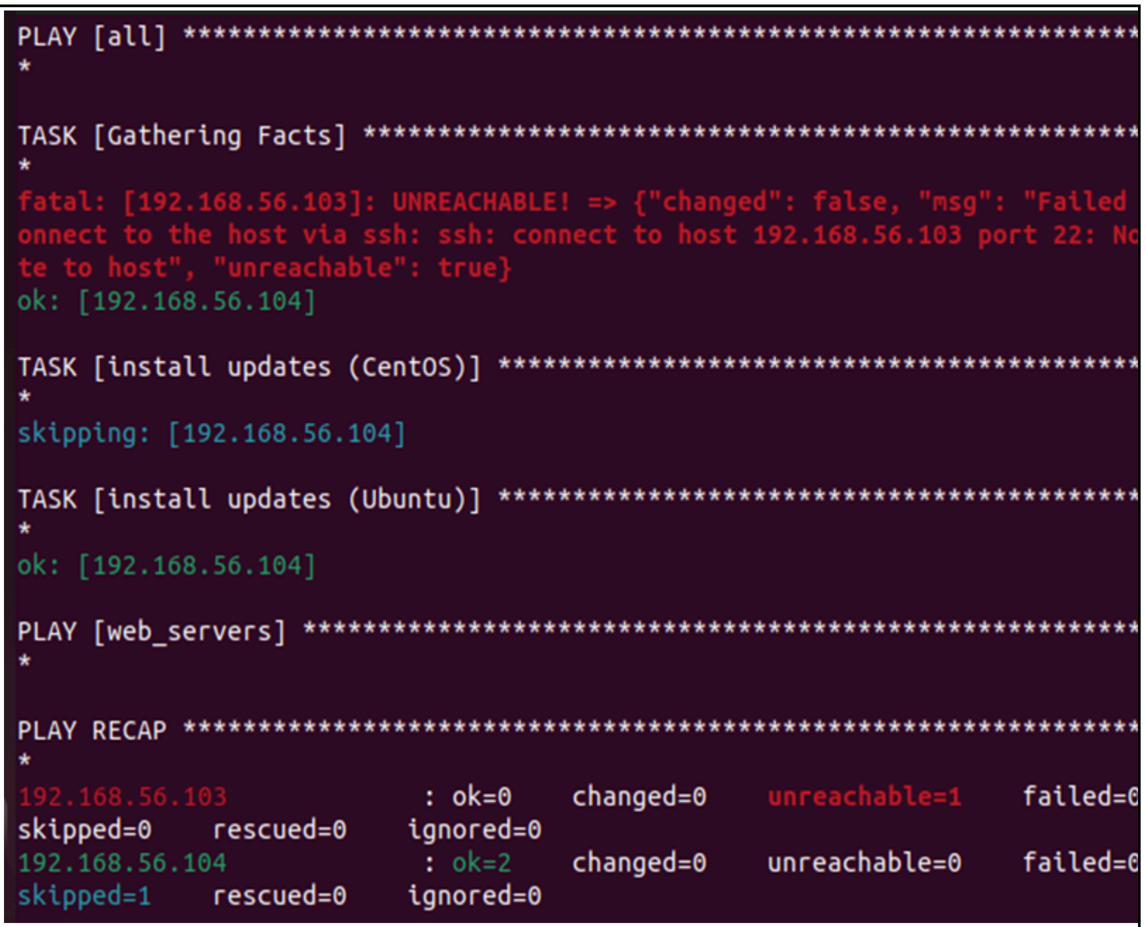
Make sure to save the file and exit.



```
GNU nano 6.2                                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:
      name:
```

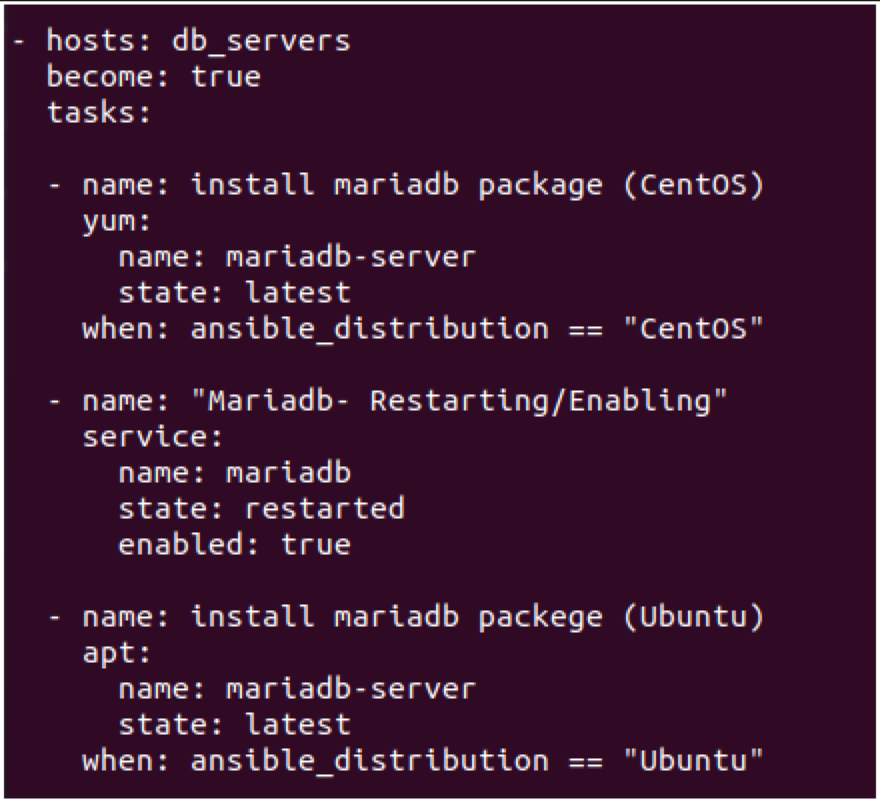
The *pre-tasks* command tells the ansible to run it before any other thing. In the *pre-tasks*, CentOS will install updates while Ubuntu will upgrade its distribution package. This will run before running the second play, which is targeted at *web_servers*. In the second play, apache and php will be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

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



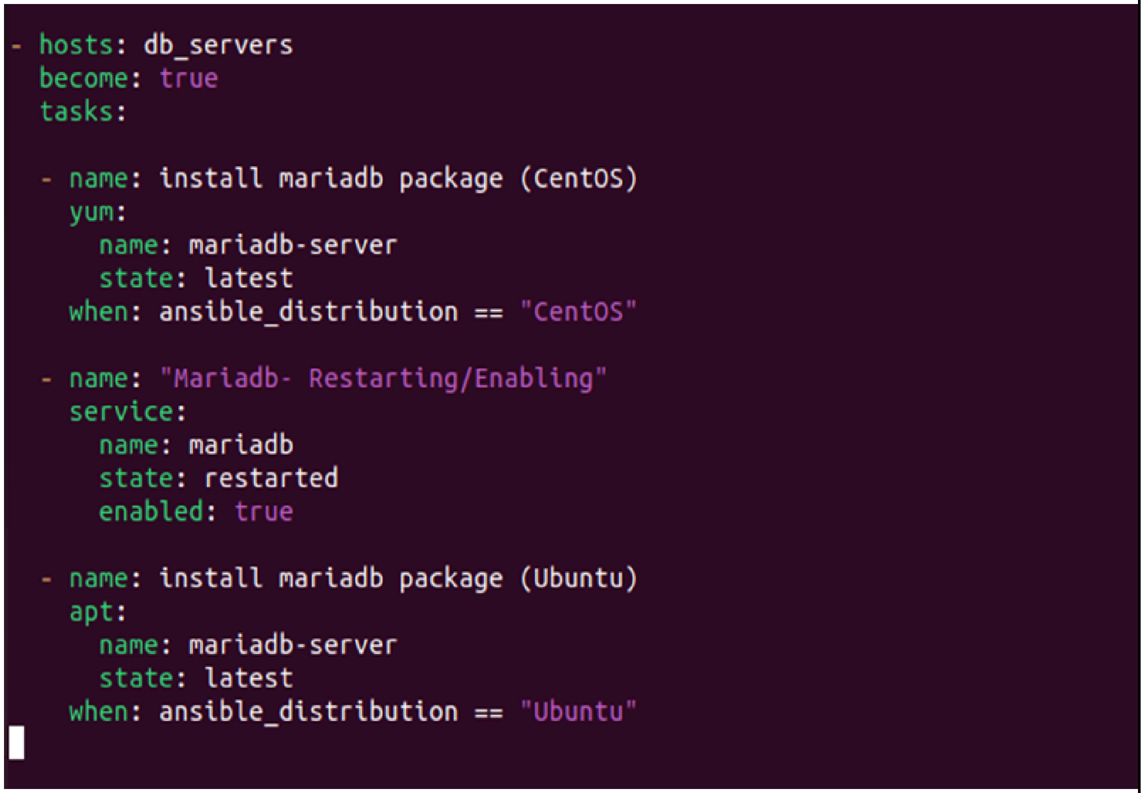
```
PLAY [all] *****************************************************************
*

TASK [Gathering Facts] *****************************************************
*
fatal: [192.168.56.103]: UNREACHABLE! => {"changed": false, "msg": "Failed
onnect to the host via ssh: ssh: connect to host 192.168.56.103 port 22: No
te to host", "unreachable": true}
ok: [192.168.56.104]

TASK [install updates (CentOS)] ********************************************
*
skipping: [192.168.56.104]

TASK [install updates (Ubuntu)] ********************************************
*
ok: [192.168.56.104]

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

PLAY RECAP *****************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

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.



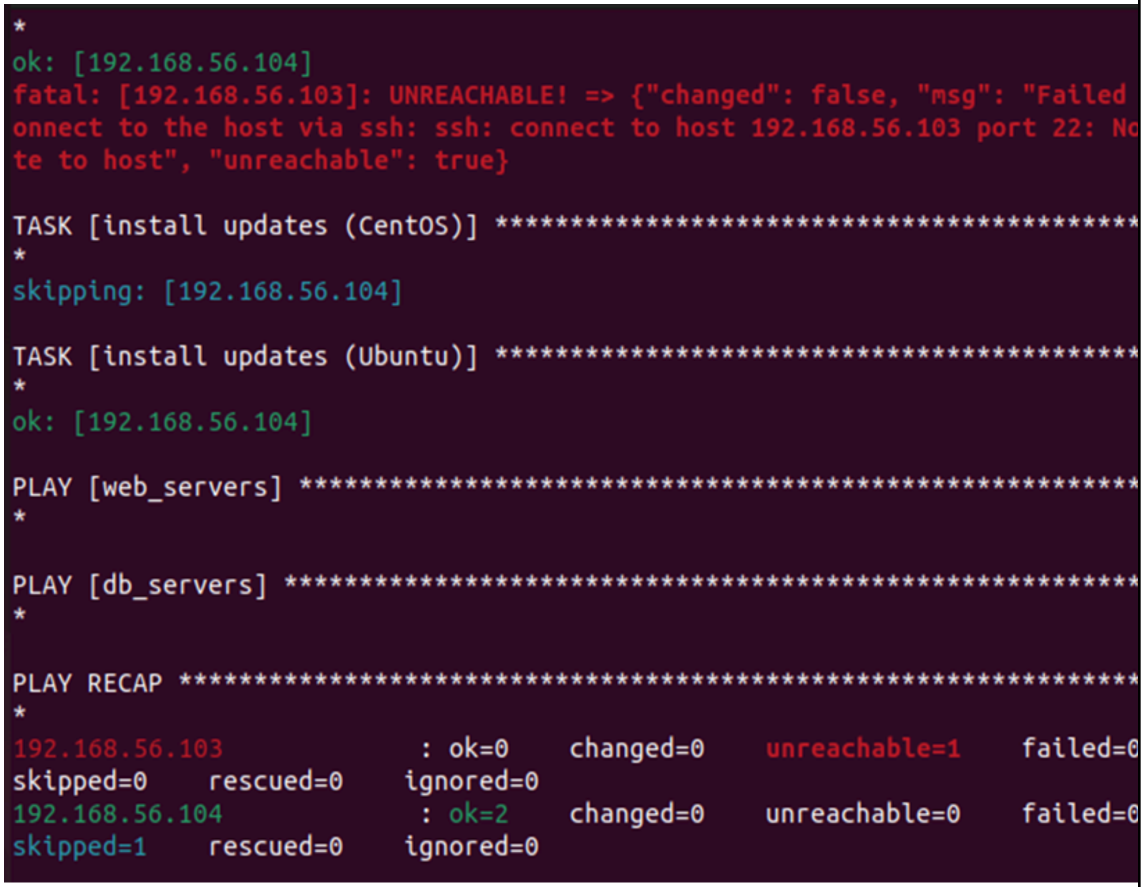
```
- hosts: db_servers
  become: true
  tasks:

  - 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.



```
- 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 package (Ubuntu)
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "Ubuntu"
```

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

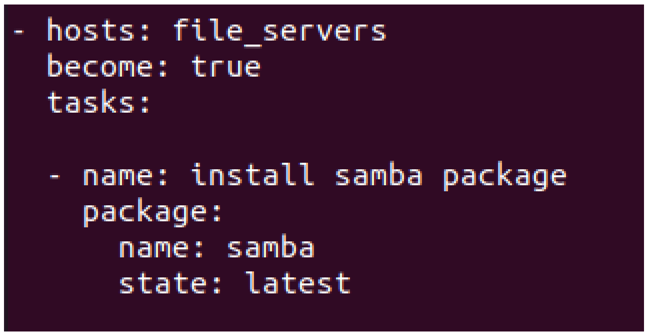


```
*
ok: [192.168.56.104]
fatal: [192.168.56.103]: UNREACHABLE! => {"changed": false, "msg": "Failed 
onnect to the host via ssh: ssh: connect to host 192.168.56.103 port 22: No
te to host", "unreachable": true}

TASK [install updates (CentOS)] ******************************************
*
skipping: [192.168.56.104]

TASK [install updates (Ubuntu)] ******************************************
*
ok: [192.168.56.104]

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

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

PLAY RECAP ***************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

4. 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.

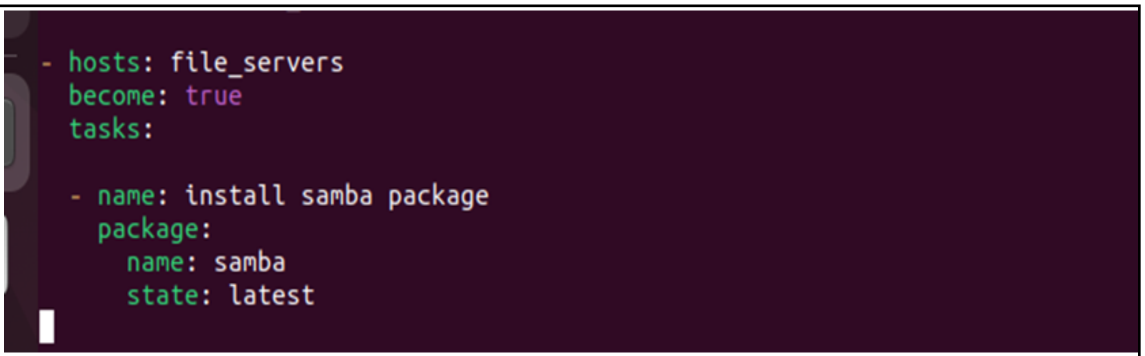
   Describe the output.

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



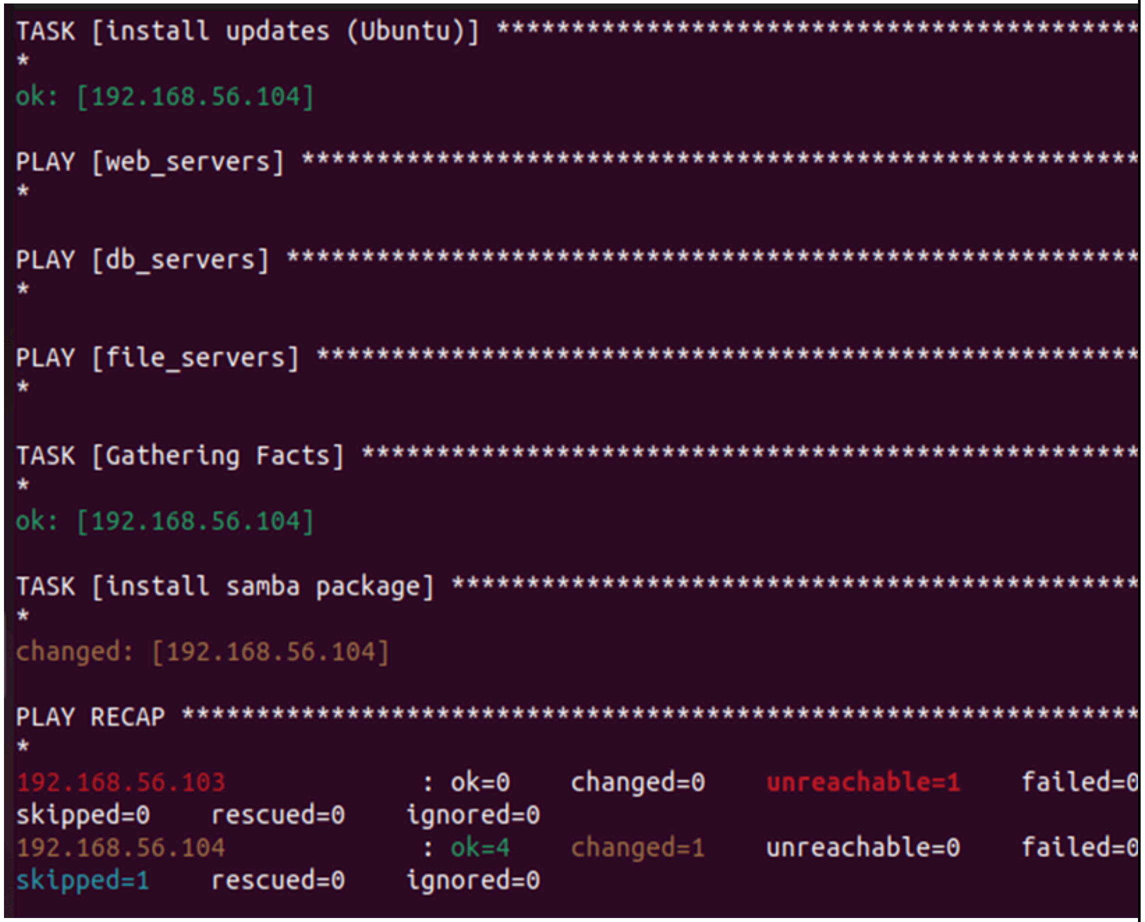
```
- hosts: file_servers
  become: true
  tasks:

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

Make sure to save the file and exit.



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

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

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



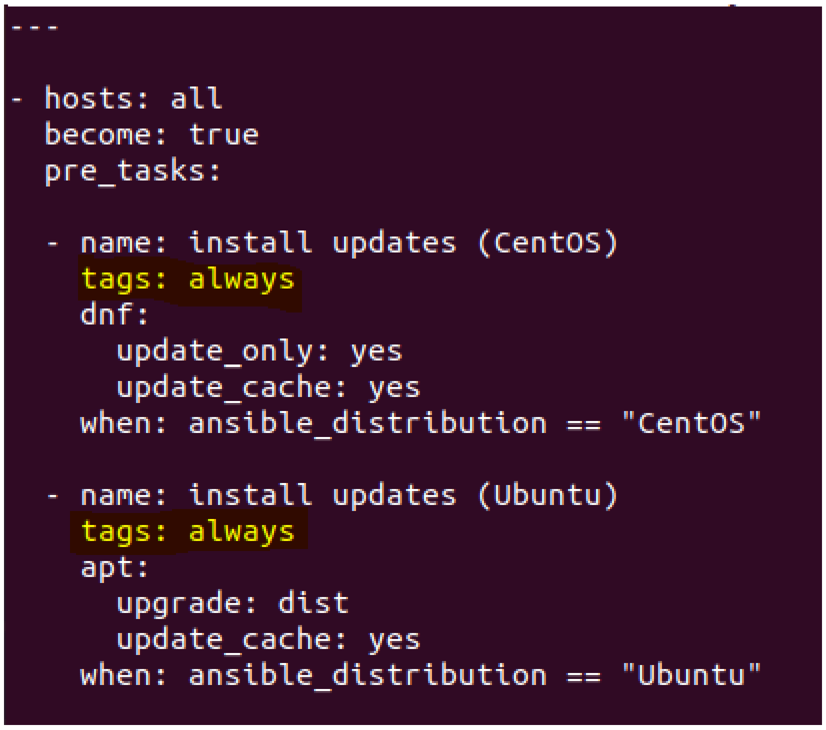
```
TASK [install updates (Ubuntu)] *******************************************
*
ok: [192.168.56.104]

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

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 ****************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=4    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

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.

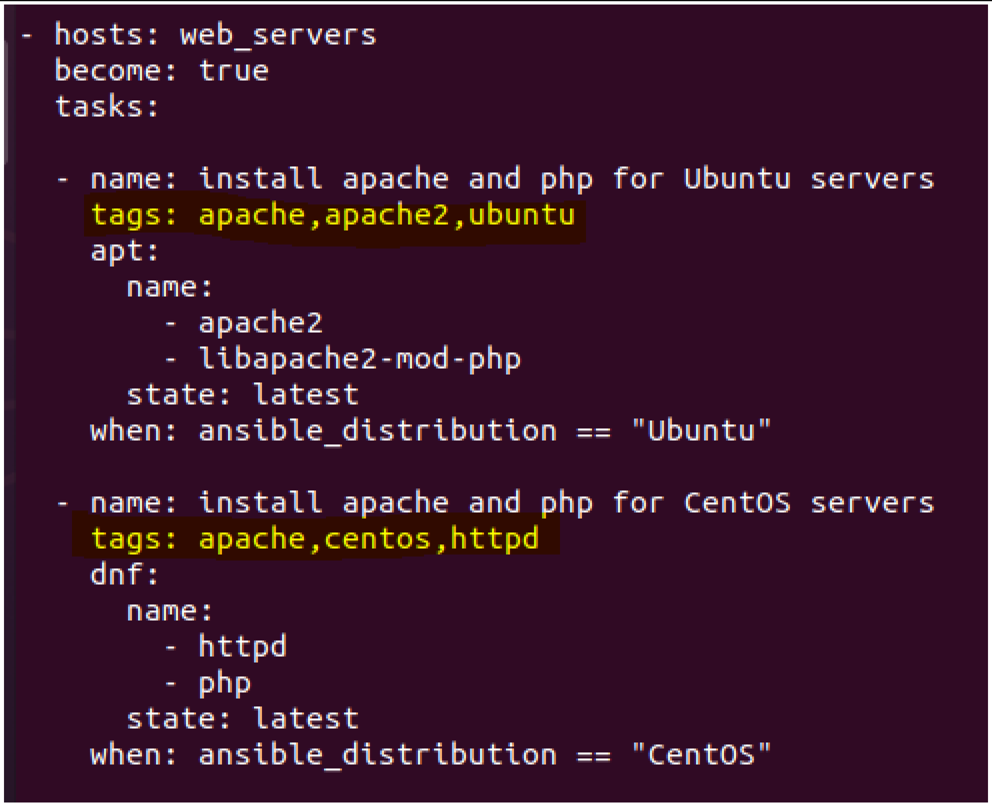
Task 2: Using Tags in running playbooks

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.

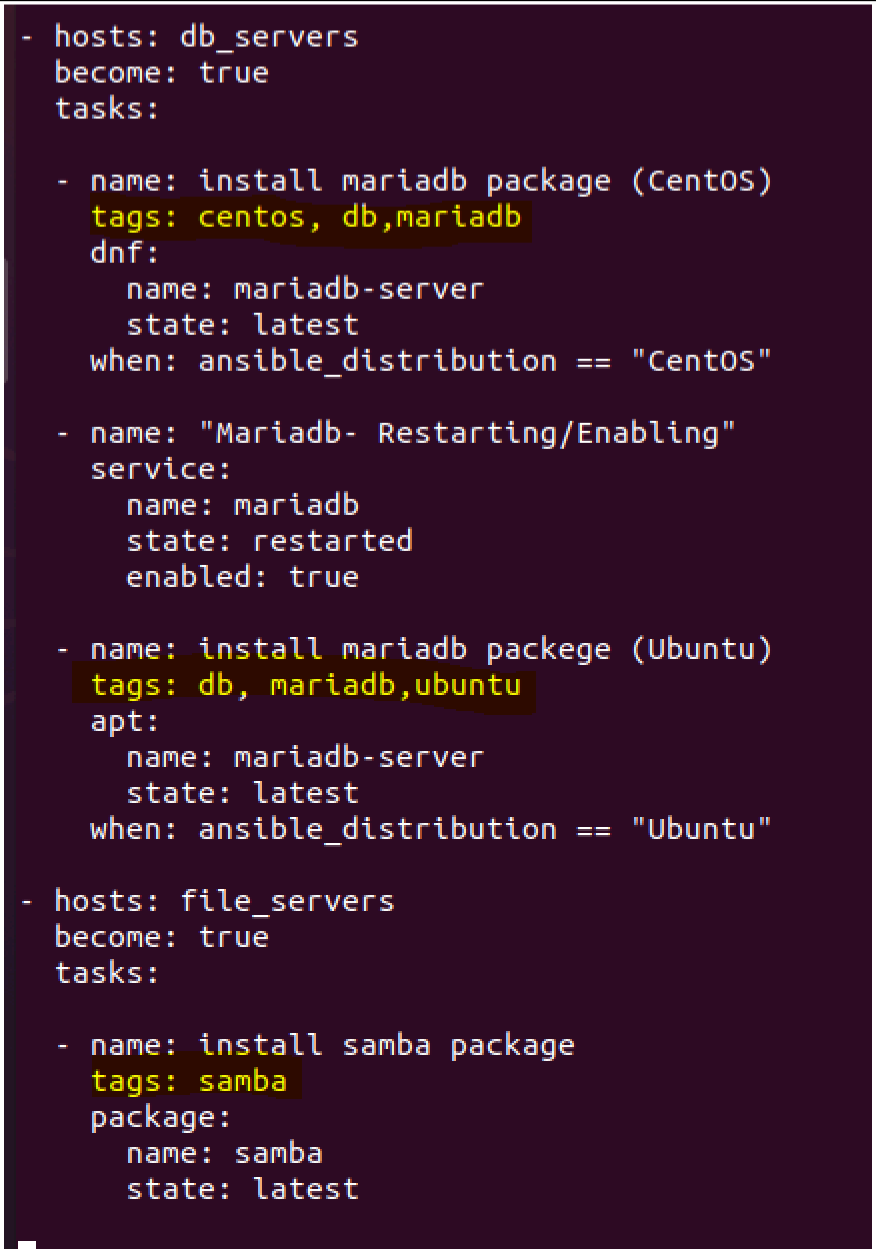
1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name_of_tag*. This is an arbitrary command, which means you can use any name for a tag.



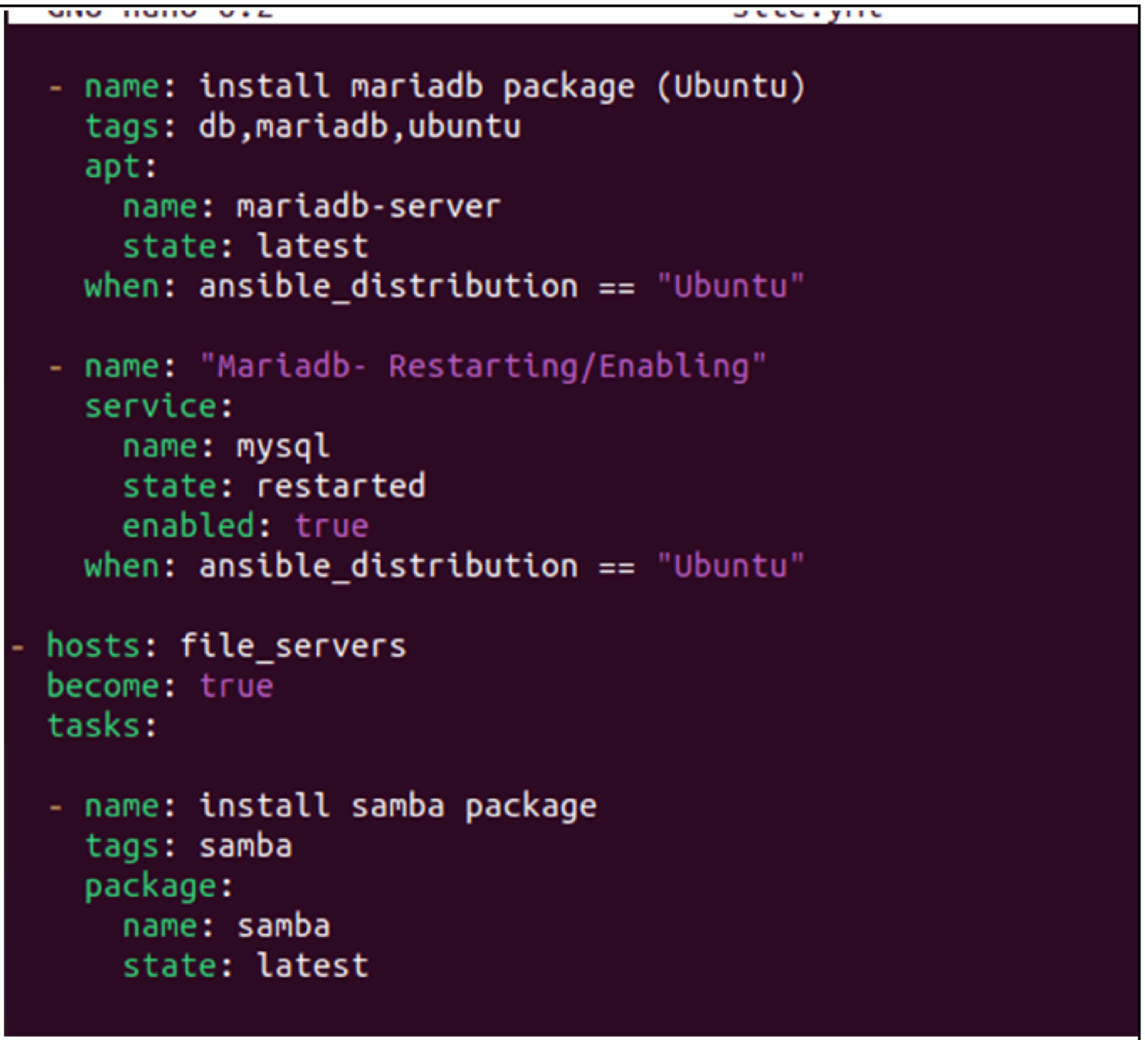
```
---

- 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"
```

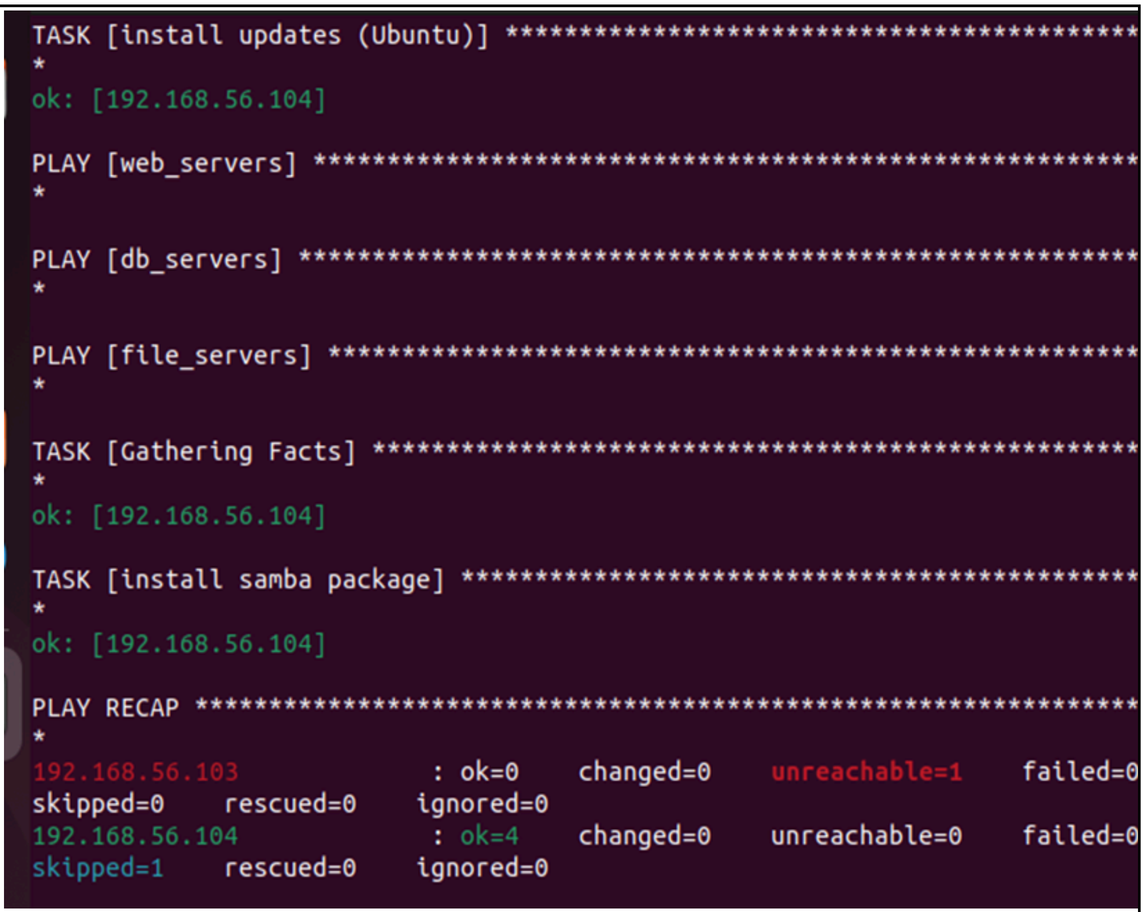

```
- hosts: db_servers
  become: true
  tasks:

  - 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.



```
  - name: install mariadb package (Ubuntu)
    tags: db,mariadb,ubuntu
    apt:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mysql
      state: restarted
      enabled: true
    when: ansible_distribution == "Ubuntu"

- hosts: file_servers
  become: true
  tasks:

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

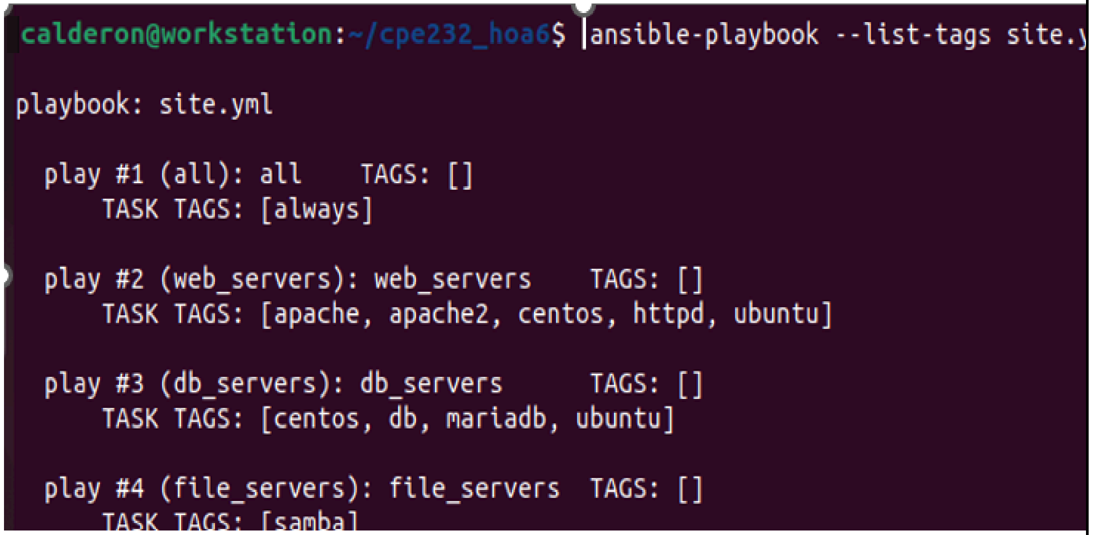
Run the *site.yml* file and describe the result.



```
TASK [install updates (Ubuntu)] ************************************************
*
ok: [192.168.56.104]

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

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

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

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

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

PLAY RECAP *********************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=4    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

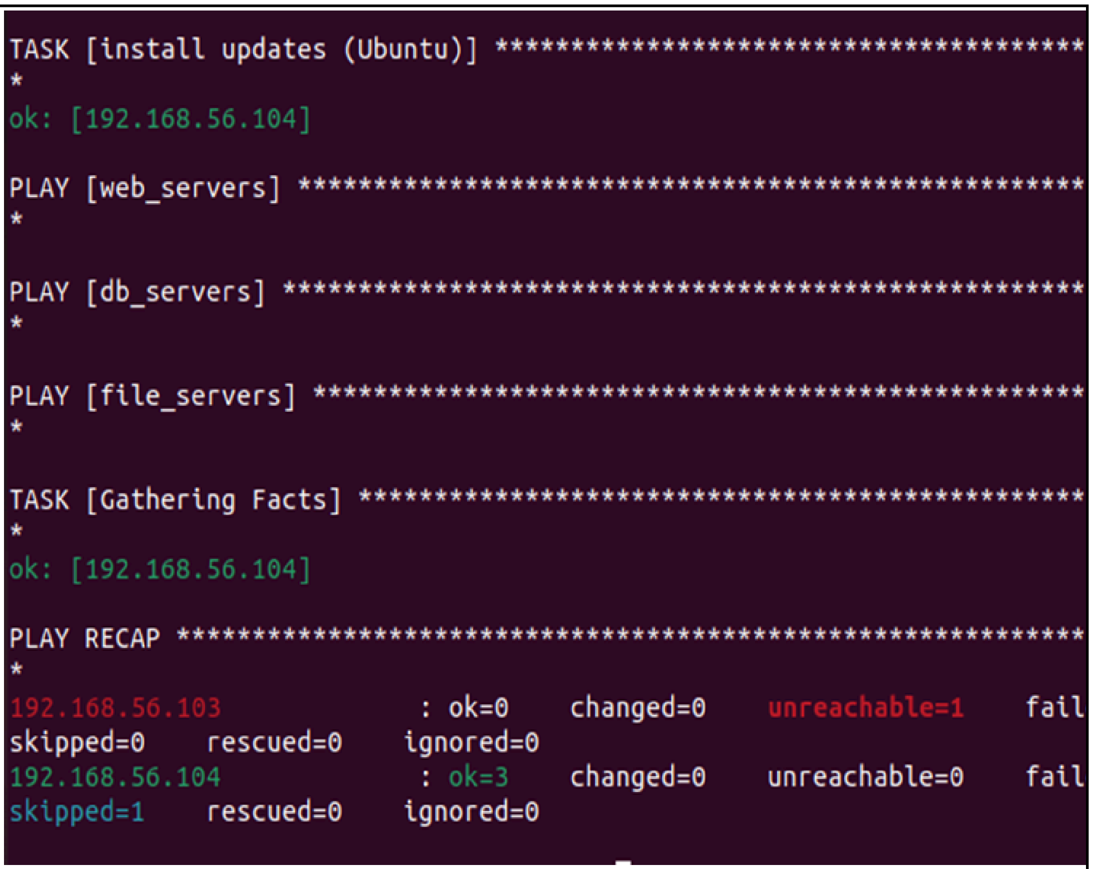
2. On the local machine, try to issue the following commands and describe each result:

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



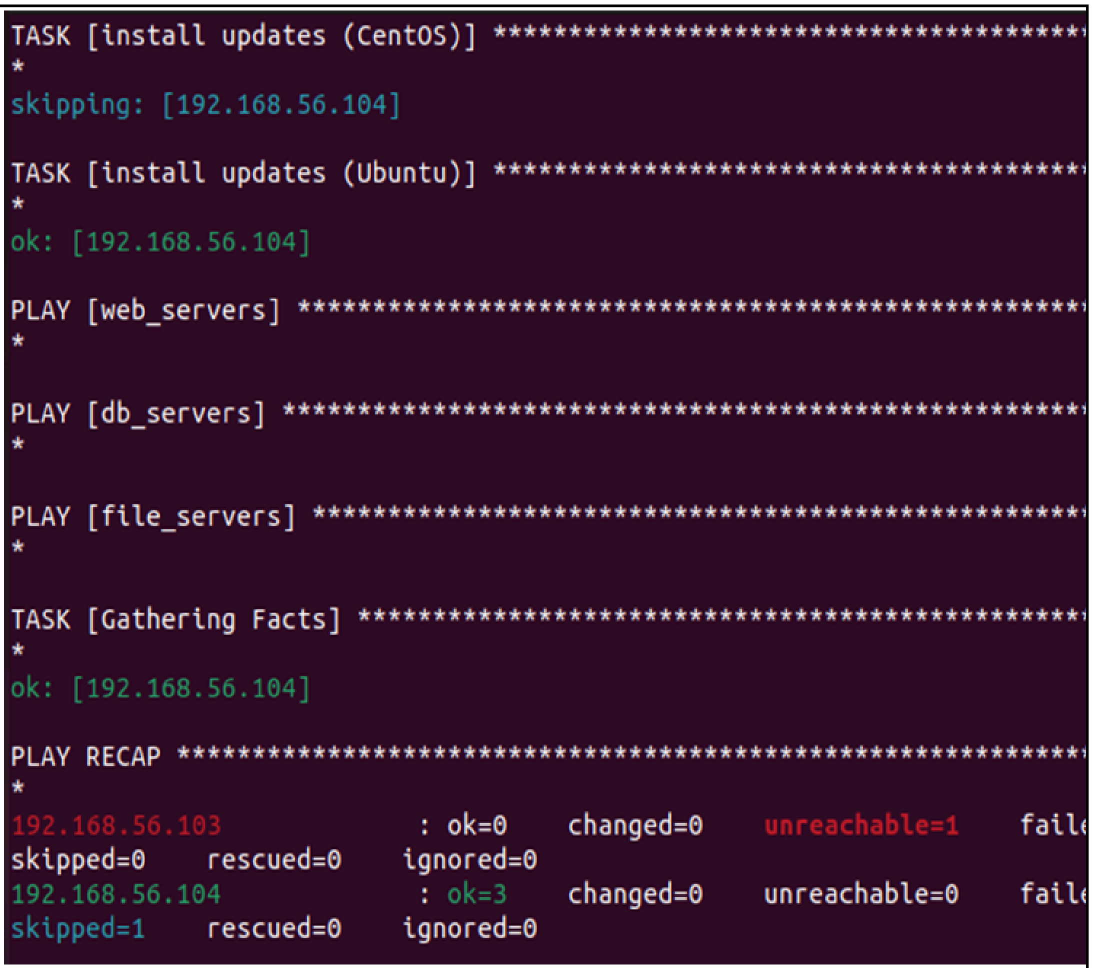
```
calderon@workstation:~/cpe232_hoa6$ ansible-playbook --list-tags site.y

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*



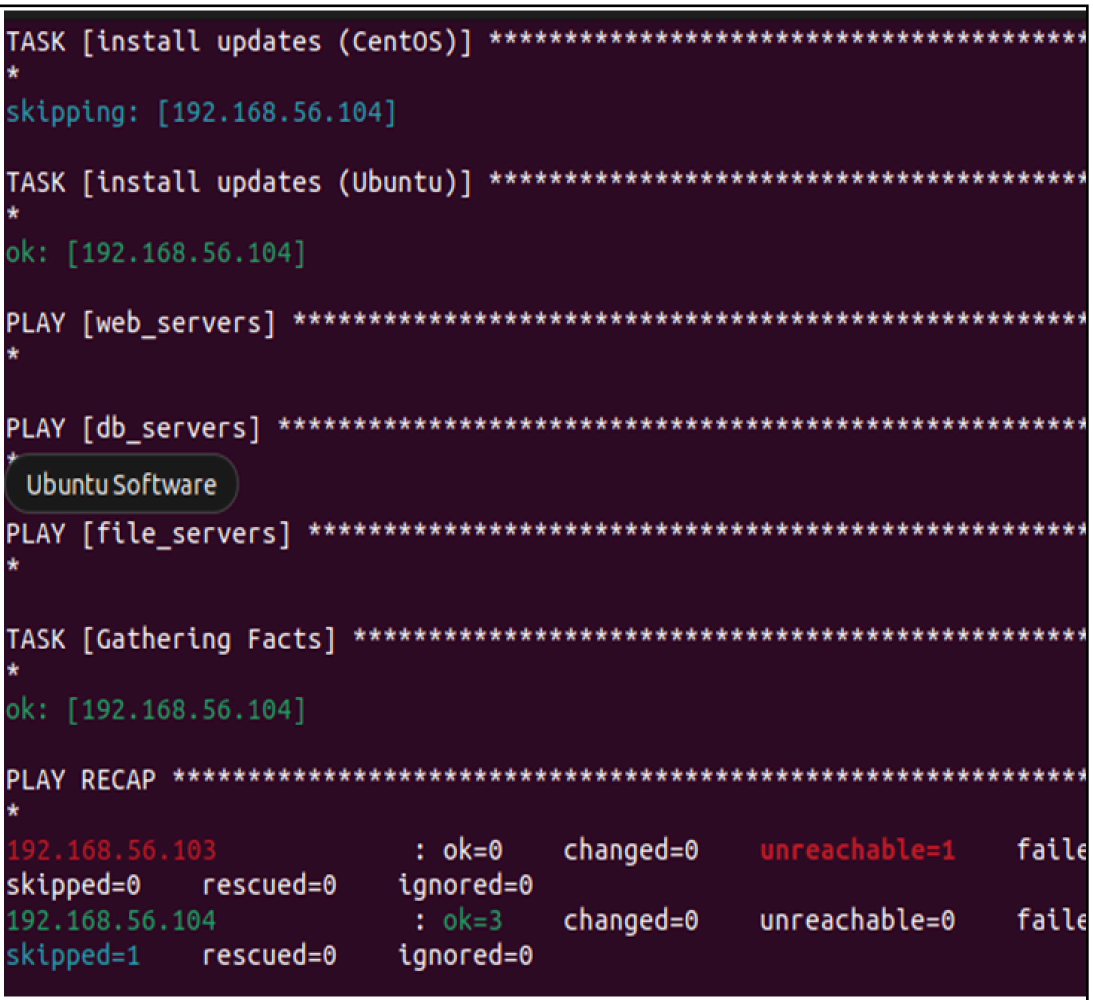
```
TASK [install updates (Ubuntu)] ****************************************
*
ok: [192.168.56.104]

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

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

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

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

PLAY RECAP *************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    fail
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=3    changed=0    unreachable=0    fail
skipped=1    rescued=0    ignored=0
```

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



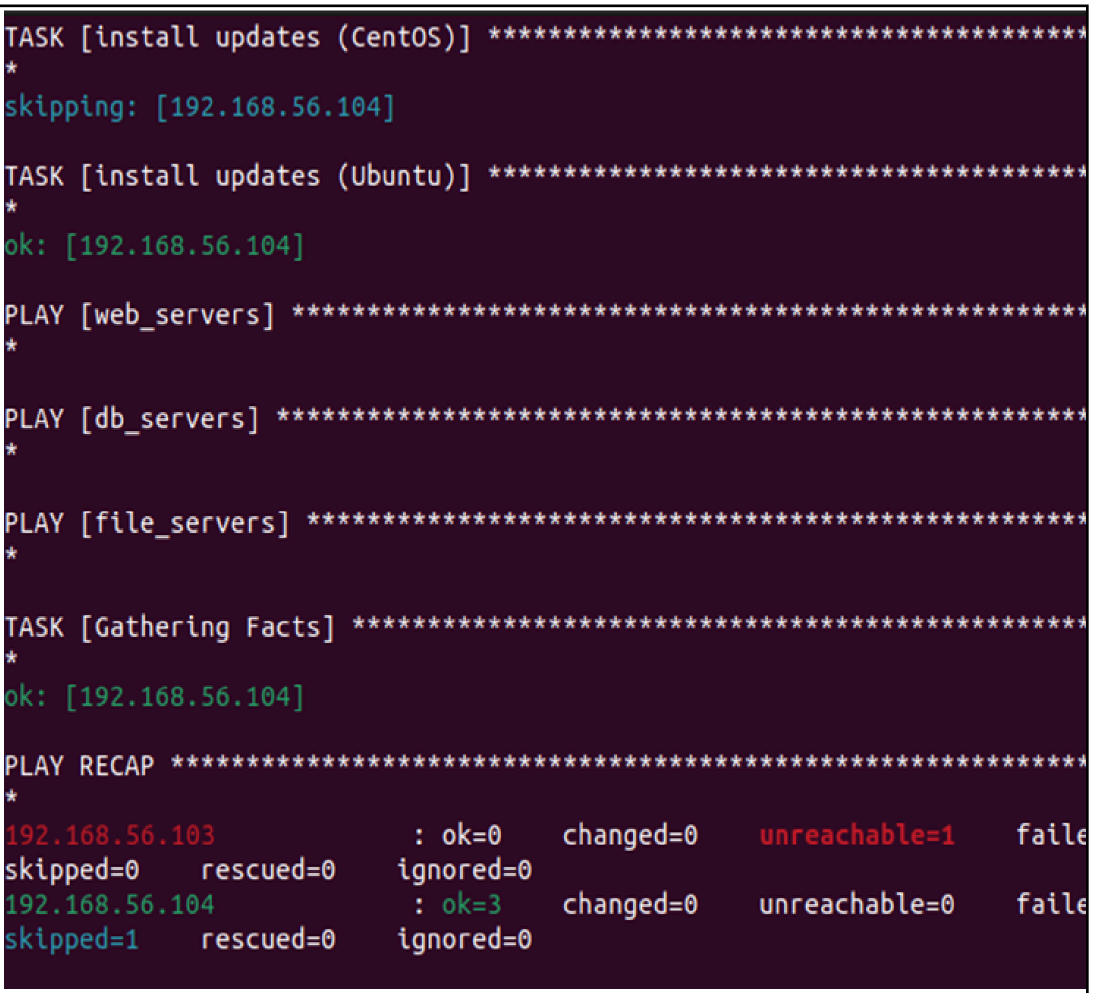
```
TASK [install updates (CentOS)] ***************************************
*
skipping: [192.168.56.104]

TASK [install updates (Ubuntu)] ***************************************
*
ok: [192.168.56.104]

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

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

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

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

PLAY RECAP ************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    faile
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=3    changed=0    unreachable=0    faile
skipped=1    rescued=0    ignored=0
```

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



```
TASK [install updates (CentOS)] ****************************************
*
skipping: [192.168.56.104]

TASK [install updates (Ubuntu)] ****************************************
*
ok: [192.168.56.104]

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

PLAY [db_servers] ******************************************************
*
Ubuntu Software
PLAY [file_servers] ****************************************************
*

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

PLAY RECAP *************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    faile
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=3    changed=0    unreachable=0    faile
skipped=1    rescued=0    ignored=0
```

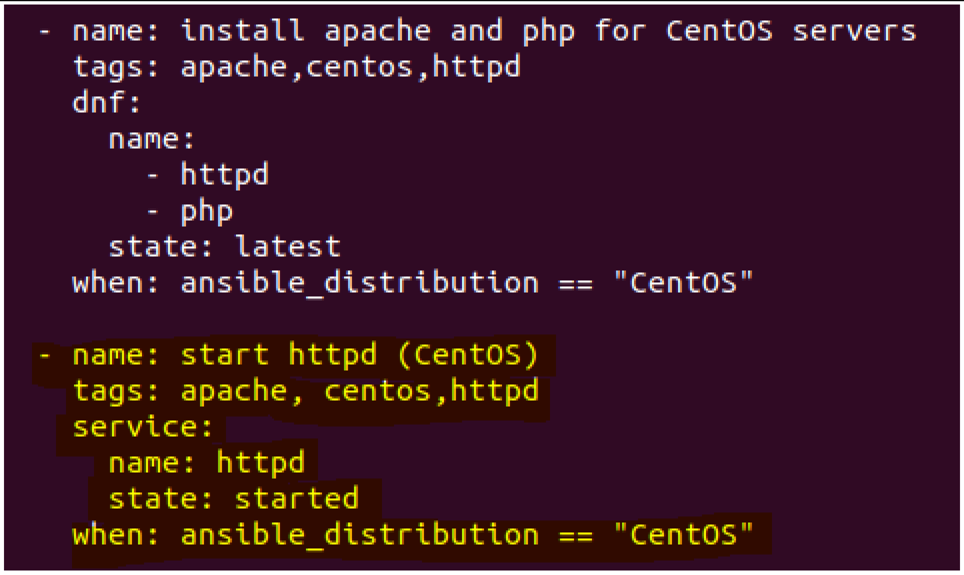
2.5 *ansible-playbook --tags “apache,db” --ask-become-pass site.yml*



```
TASK [install updates (CentOS)] *****************************************
*
skipping: [192.168.56.104]

TASK [install updates (Ubuntu)] *****************************************
*
ok: [192.168.56.104]

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

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

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

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

PLAY RECAP **************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    faile
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=3    changed=0    unreachable=0    faile
skipped=1    rescued=0    ignored=0
```

Task 3: Managing Services

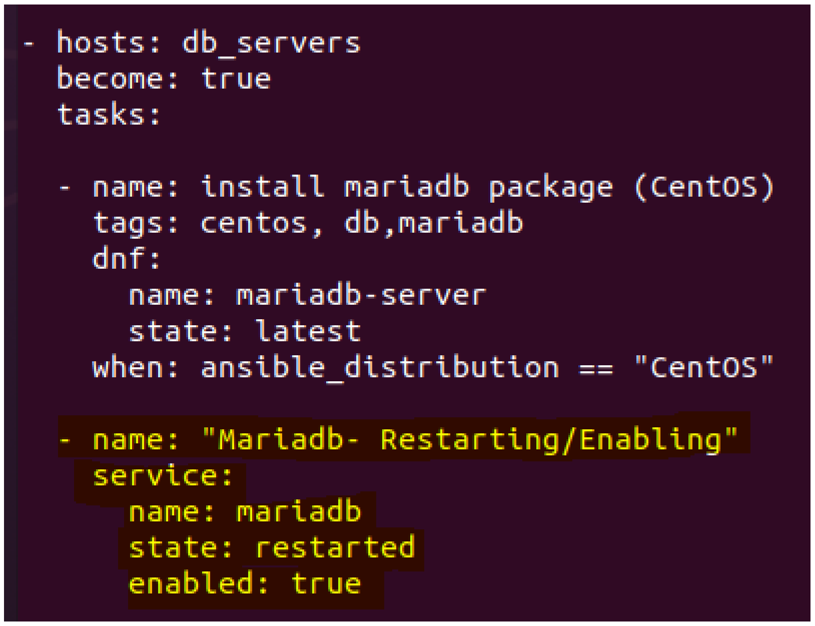
1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.



```
- 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.



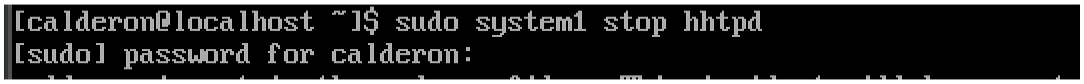
```
- hosts: db_servers
  become: true
  tasks:

  - 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.

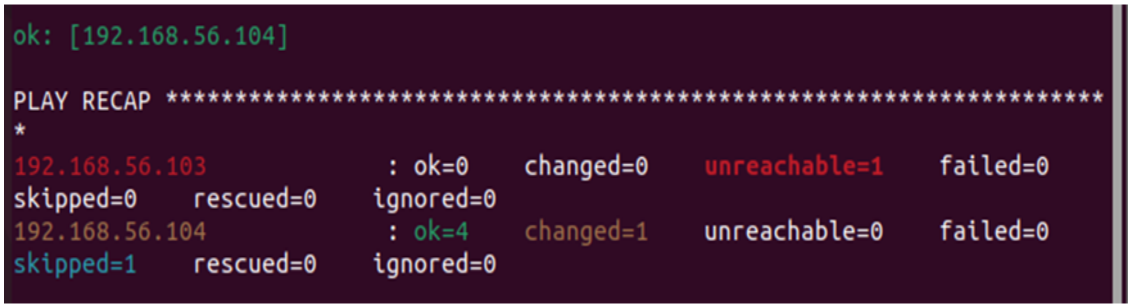
2. To test it, before you run the saved playbook, go to the CentOS server and stop the currently running httpd using the command *sudo systemctl stop httpd.*

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.

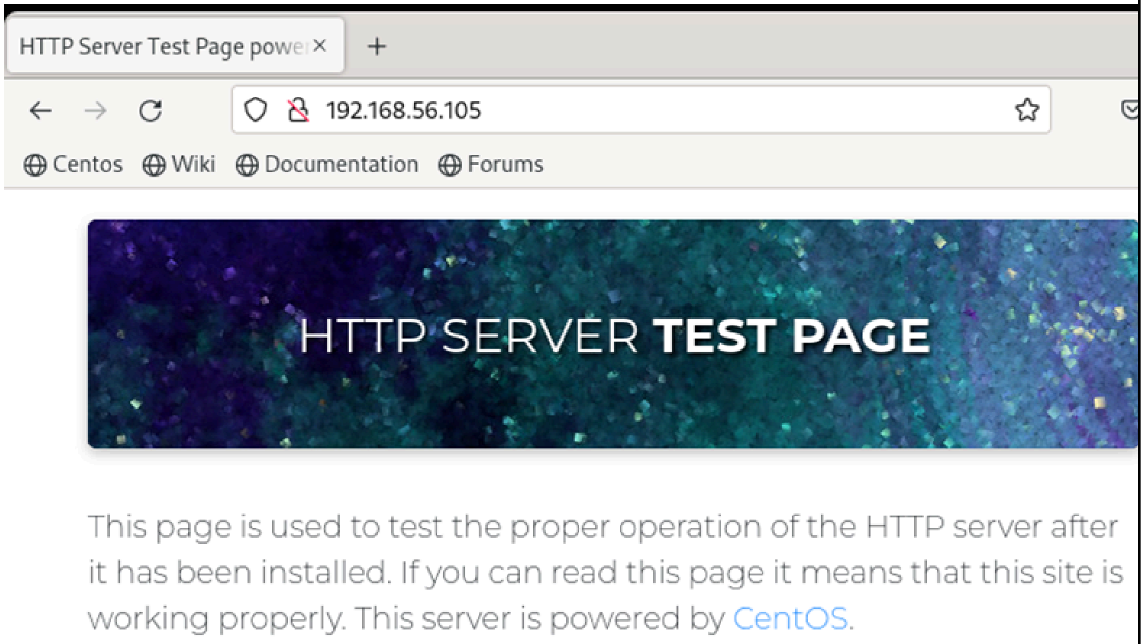


```
[calderon@localhost ~]$ sudo systeml stop hhtpd
[sudo] password for calderon:
```

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.



```
ok: [192.168.56.104]

PLAY RECAP *********************************************************************
*
192.168.56.103             : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.104             : ok=4    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```



HTTP SERVER **TEST PAGE**

This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

Reflections:

Answer the following:

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

Remote server access enables administrators to manage and monitor computer systems and networks remotely via a network connection. This capability eliminates the need for physical presence for repairs and maintenance tasks. It empowers system administrators to efficiently oversee and control corporate networks from any location with an internet connection.

2. What is the importance of tags in playbooks?

   Tags play a crucial role in managing and orchestrating ansible playbooks by providing selective execution capabilities, enhancing playbook organization and readability, and enabling efficient testing, debugging, and deployment of infrastructure configurations.

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

   Automating service management tasks in playbooks offers numerous benefits, including improved efficiency, consistency, speed, scalability, reliability, security, and resource optimization. By leveraging automation, organizations can enhance operational efficiency, agility, and security while reducing the burden on IT staff.

Conclusion

This activity has underscored the importance of optimizing the execution of large Ansible playbooks by leveraging tags. By strategically applying tags to tasks, includes, blocks, plays, roles, and imports within playbooks, administrators gain the flexibility to selectively execute or skip specific parts of the playbook based on their requirements. This approach enhances playbook management and execution efficiency, particularly in complex environments where precise control over task execution is necessary.