Name: Vincent Bryan Bose	**Date Performed: 30/09/2024**
Course/Section: CPE31S2	**Date Submitted: 30/09/2024**
Instructor: Mr. Robin Valenzuela	**Semester and SY: 1st sem 2024-2025**

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.

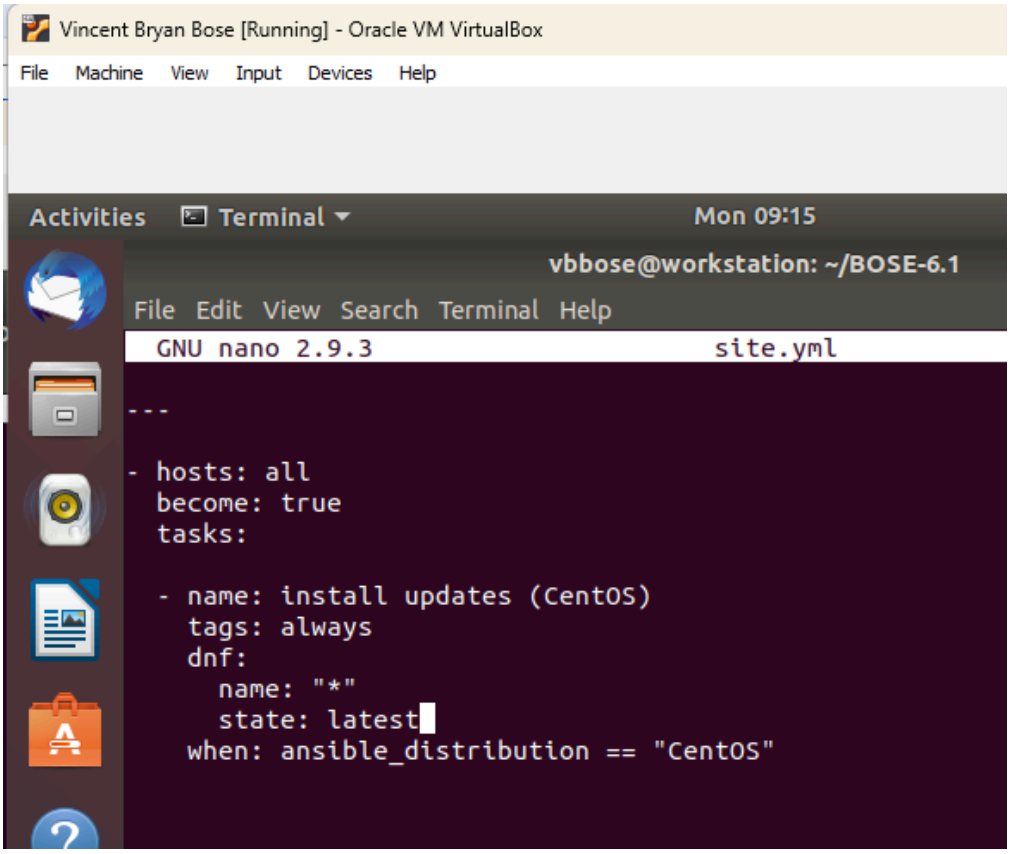
We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.

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

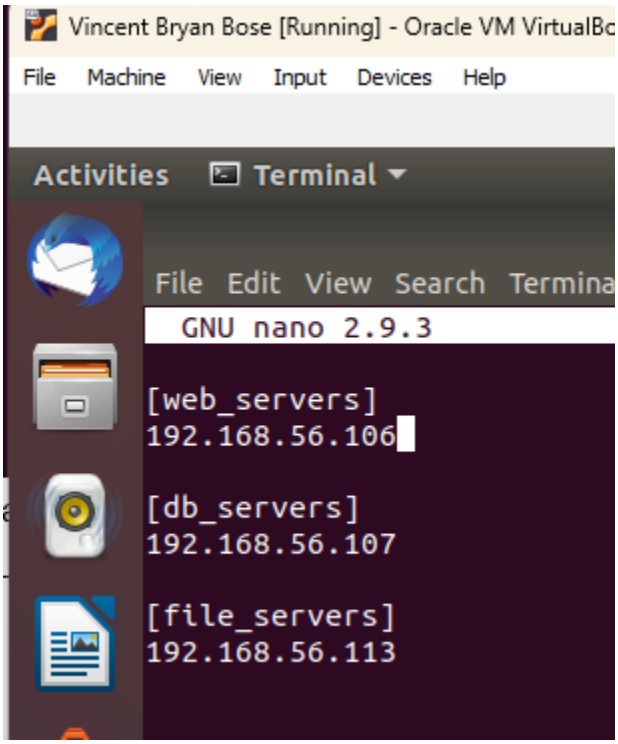
1. Create a new playbook and named it site.yml. Follow the commands as shown in the image below. Make sure to save the file and exit.

```
---

- 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
```

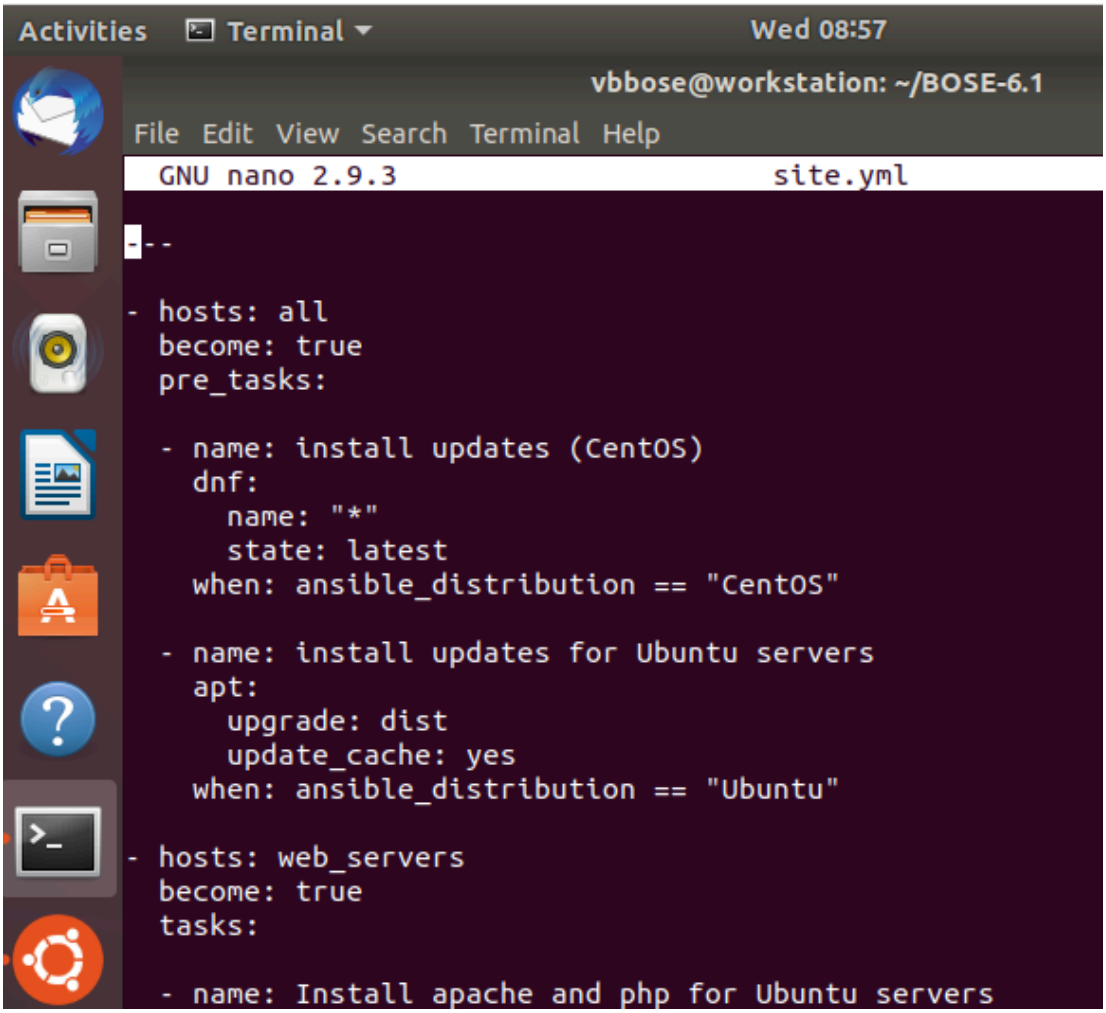


Make sure to save the file and exit.

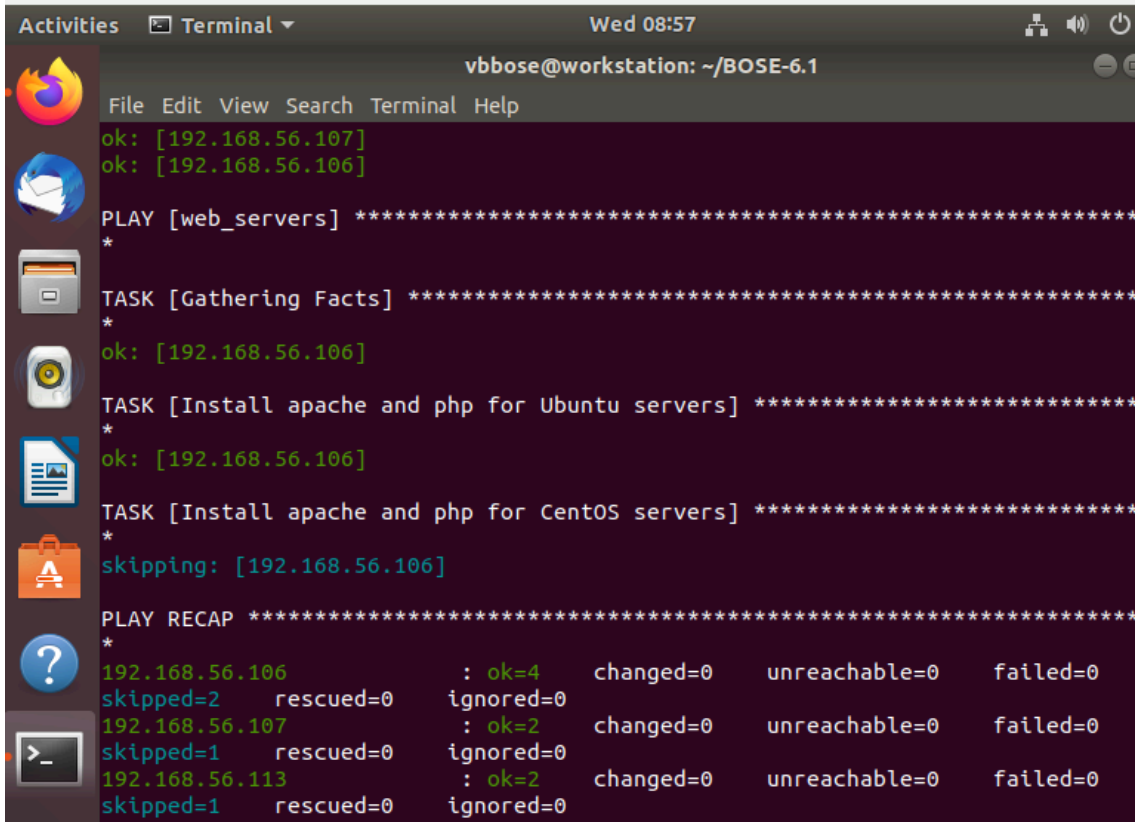
Right now, we have created groups in our inventory file and put each server in its own group. In other cases, you can have a server be a member of multiple groups, for example you have a test server that is also a web server.

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

```yaml
---

- 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"
```


Activities
Terminal
Wed 08:57
vbbose@workstation: ~/BOSE-6.1
File Edit View Search Terminal Help
GNU nano 2.9.3
site.yml

- hosts: all
  become: true
  pre_tasks:
  - name: install updates (CentOS)
    dnf:
      name: "*"
      state: latest
    when: ansible_distribution == "CentOS"
  - name: install updates for Ubuntu servers
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"
- hosts: web_servers
  become: true
  tasks:
  - name: Install apache and php for Ubuntu servers



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.

- All the necessary installs are installed.

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"
```

Activities Terminal Wed 09:02

vbbose@workstation: ~/BOSE-6.1

File Edit View Search Terminal Help

GNU nano 2.9.3 site.yml

```
    when: ansible_distribution == "CentOS"

- 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"
```

```
Activities   Terminal   Wed 09:02
vbbose@workstation: ~/BOSE-6.1
File  Edit  View  Search  Terminal  Help
*

TASK [Gathering Facts] ***********************************************
*
ok: [192.168.56.107]

TASK [install mariadb package (CentOS)] ******************************
*
skipping: [192.168.56.107]

TASK [Mariadb- Restarting/Enabling] **********************************
*
changed: [192.168.56.107]

TASK [install mariadb package (Ubuntu)] ******************************
*
ok: [192.168.56.107]

PLAY RECAP ***********************************************************
*
192.168.56.106             : ok=4    changed=0    unreachable=0    failed=
skipped=2    rescued=0    ignored=0
192.168.56.107             : ok=5    changed=1    unreachable=0    failed=
skipped=2    rescued=0    ignored=0
192.168.56.113             : ok=2    changed=0    unreachable=0    failed=
skipped=1    rescued=0    ignored=0
```

Make sure to save the file and exit.

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

- The mariadb package has installed successfully.

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.

```
vbbose@server2:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.1.48 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
   Active: active (running) since Wed 2024-10-02 09:02:46 +08; 2min 57s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 15078 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_ST
  Process: 15075 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/S
  Process: 14974 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && V
  Process: 14972 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_STA
  Process: 14971 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/
 Main PID: 15048 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 27 (limit: 4915)
   CGroup: /system.slice/mariadb.service
           └─15048 /usr/sbin/mysqld
```

```
vbbose@server1:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.1.48 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
   Active: active (running) since Wed 2024-10-02 07:48:32 +08; 1h 17min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 1318 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_STA
  Process: 1314 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SU
  Process: 913 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR
  Process: 911 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START
  Process: 905 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/ru
 Main PID: 1040 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 27 (limit: 4915)
   CGroup: /system.slice/mariadb.service
           └─1040 /usr/sbin/mysqld
```

Describe the output.

- The mariadb server is in the server1

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
```

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

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

```
PLAY RECAP *********************************************************************
*
192.168.56.106             : ok=4    changed=0    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.56.107             : ok=5    changed=1    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

Make sure to save the file and exit.

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

- The samba package has been successfully installed.

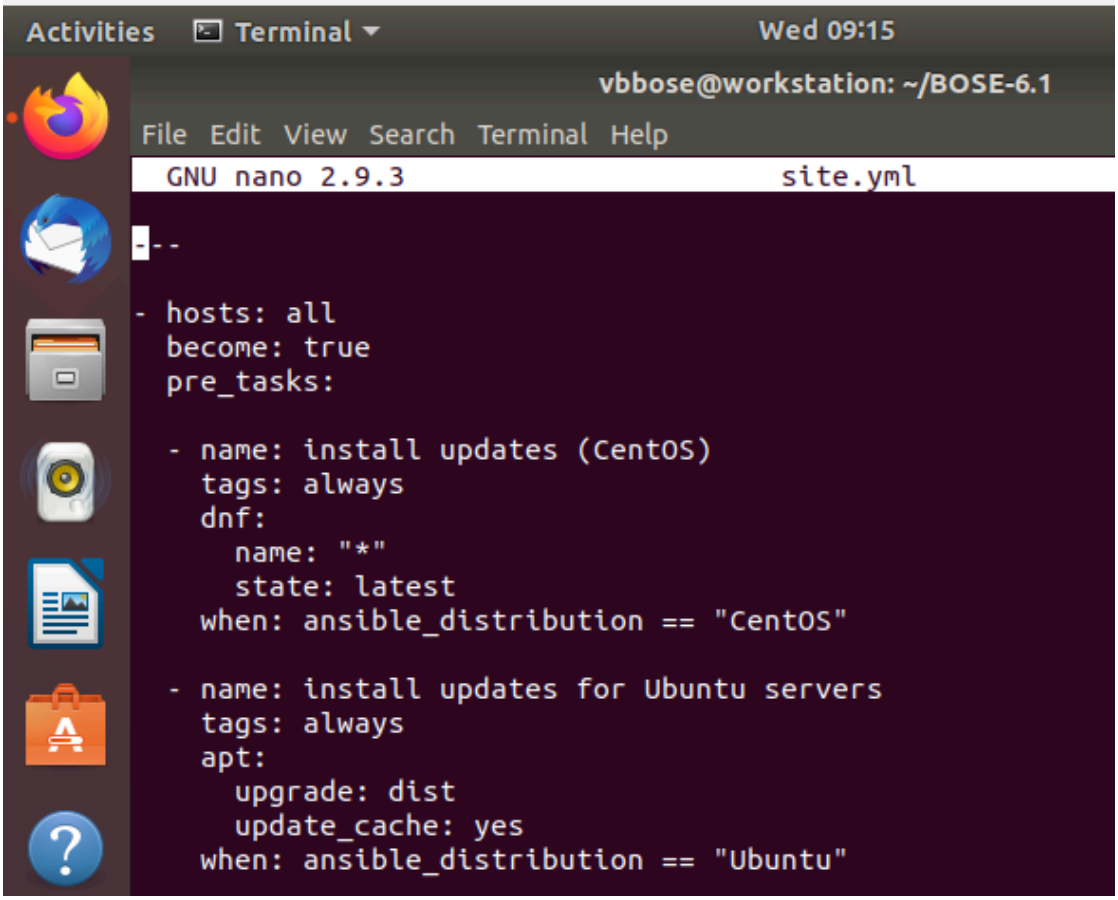
The testing of the *file_servers* is beyond the scope of this activity, and as well as our topics and objectives. However, in this activity we were able to show that we can target hosts or servers using grouping in ansible playbooks.

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"
```


Activities
Terminal
Wed 09:15
vbbose@workstation: ~/BOSE-6.1
File Edit View Search Terminal Help
GNU nano 2.9.3
site.yml

- hosts: all
  become: true
  pre_tasks:

  - name: install updates (CentOS)
    tags: always
    dnf:
      name: "*"
      state: latest
    when: ansible_distribution == "CentOS"

  - name: install updates for Ubuntu servers
    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: 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
```

```
- 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
```

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

  - name: install mariadb package (CentOS)
    tags: centos,db,mariadb
    yum:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true
```

```
PLAY RECAP *********************************************************************
*
192.168.56.106             : ok=4    changed=0    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.56.107             : ok=5    changed=1    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

Make sure to save the file and exit.

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

- We have added tags to execute more codes in the playbook

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

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

```
vbbose@workstation:~/BOSE-6.1$ 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]
```

- *Lists all tags.*

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

```
TASK [install updates (CentOS)] ***********************************************
*
skipping: [192.168.56.106]
skipping: [192.168.56.107]
ok: [192.168.56.113]
```

- *successfully installed centos*

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

```
TASK [install updates for Ubuntu servers] *************************************
*
skipping: [192.168.56.113]
ok: [192.168.56.107]
ok: [192.168.56.106]
```

- *successfully installed db*

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

```
TASK [Install apache and php for Ubuntu servers] ******************************
*
ok: [192.168.56.106]

TASK [Install apache and php for CentOS servers] ******************************
*
skipping: [192.168.56.106]
```

- *successfully installed apache*

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

```
TASK [install mariadb package (CentOS)] ***************************************
*
skipping: [192.168.56.107]

TASK [install mariadb package (Ubuntu)] ***************************************
*
ok: [192.168.56.107]
```

- *successfully installed apache, db*

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"
```

```
- 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
```

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
```

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

  - name: install mariadb package (CentOS)
    tags: centos,db,mariadb
    yum:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true
```

```
TASK [Gathering Facts] ******************************************************
*
ok: [192.168.56.107]

TASK [install mariadb package (CentOS)] *************************************
*
skipping: [192.168.56.107]

TASK [Mariadb- Restarting/Enabling] *****************************************
*
changed: [192.168.56.107]

TASK [install mariadb package (Ubuntu)] *************************************
*
ok: [192.168.56.107]

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

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

TASK [install samba package] ************************************************
*
ok: [192.168.56.113]

PLAY RECAP ******************************************************************
*
```

Figure 3.1.2

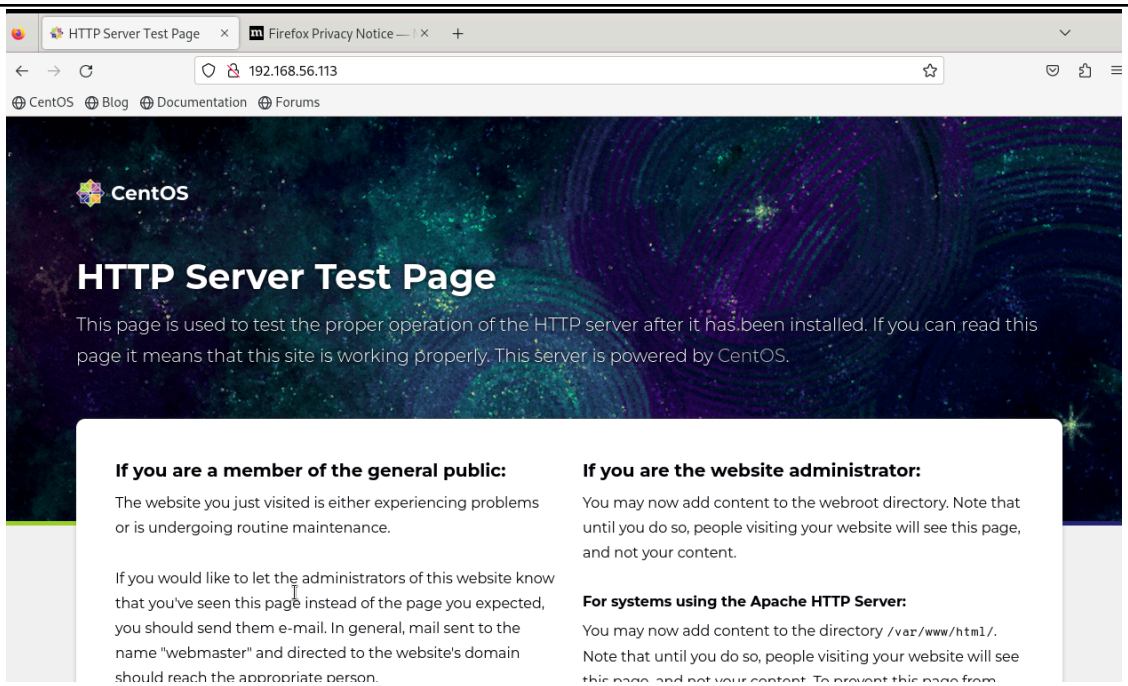
This is because in CentOS, installed packages' services are not run automatically. Thus, we need to create the module to run it automatically.

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.

```
vbbose@BOSECENTOS:~ — systemctl status httpd
[vbbose@BOSECENTOS ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: d>
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Wed 2024-10-02 09:41:05 PST; 45s ago
       Docs: man:httpd.service(8)
   Main PID: 56037 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes>
      Tasks: 177 (limit: 10964)
     Memory: 21.9M
        CPU: 57ms
     CGroup: /system.slice/httpd.service
             ├─56037 /usr/sbin/httpd -DFOREGROUND
             ├─56044 /usr/sbin/httpd -DFOREGROUND
             ├─56045 /usr/sbin/httpd -DFOREGROUND
             ├─56046 /usr/sbin/httpd -DFOREGROUND
             └─56047 /usr/sbin/httpd -DFOREGROUND

Oct 02 09:41:05 BOSECENTOS systemd[1]: Starting The Apache HTTP Server...
Oct 02 09:41:05 BOSECENTOS httpd[56037]: AH00558: httpd: Could not reliably det>
Oct 02 09:41:05 BOSECENTOS systemd[1]: Started The Apache HTTP Server.
Oct 02 09:41:05 BOSECENTOS httpd[56037]: Server configured, listening on: port >
```

```
lines 1-22/22 (END)
[vbbose@BOSECENTOS ~]$ sudo systemctl stop httpd
[sudo] password for vbbose:
[vbbose@BOSECENTOS ~]$
```

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?
   - Grouping remote servers in Ubuntu is important because it allows you to manage multiple servers with similar configurations, security settings, or applications more easily. This can simplify tasks such as applying security updates, configuring software, or monitoring performance. By grouping servers, you can also improve security by applying the same security settings to all servers, reduce the risk of downtime by distributing traffic across multiple servers, and optimize resource utilization by sharing resources between servers. Additionally, grouping servers enables you to monitor and report on multiple servers more easily, making it a valuable tool for managing large-scale infrastructure.
2. What is the importance of tags in playbooks?
   - Tags in Ansible playbooks are a powerful feature that enables you to selectively execute specific tasks or roles, conditionally execute tasks based on groups or conditions, and organize and reuse playbook content. By using tags, you can control the execution of your playbook with precision, allowing you to run specific tasks or roles in isolation, test individual components, or deploy specific configurations to specific hosts or groups. This flexibility and control

make tags an essential tool for managing and maintaining your Ansible automation workflows.

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

   - Some services need to be managed automatically in playbooks because it enables consistency, speed, efficiency, scalability, and repeatability. Manual management can lead to inconsistencies, which can cause issues and downtime. Automated management ensures that services are consistently configured and maintained, reducing the risk of human error. It also allows for faster deployment, scaling, and maintenance of services, which is critical in today's fast-paced digital landscape. Additionally, automation saves time and resources, allowing teams to focus on more strategic tasks and reducing the administrative burden. By managing services automatically through playbooks, you can ensure that your infrastructure is always up-to-date, secure, and running smoothly.