Name: Carag, Carl Jervie B.	Date Performed: 10/12/25
Course/Section: CPE31S2	Date Submitted: 10/14/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2025-2026

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

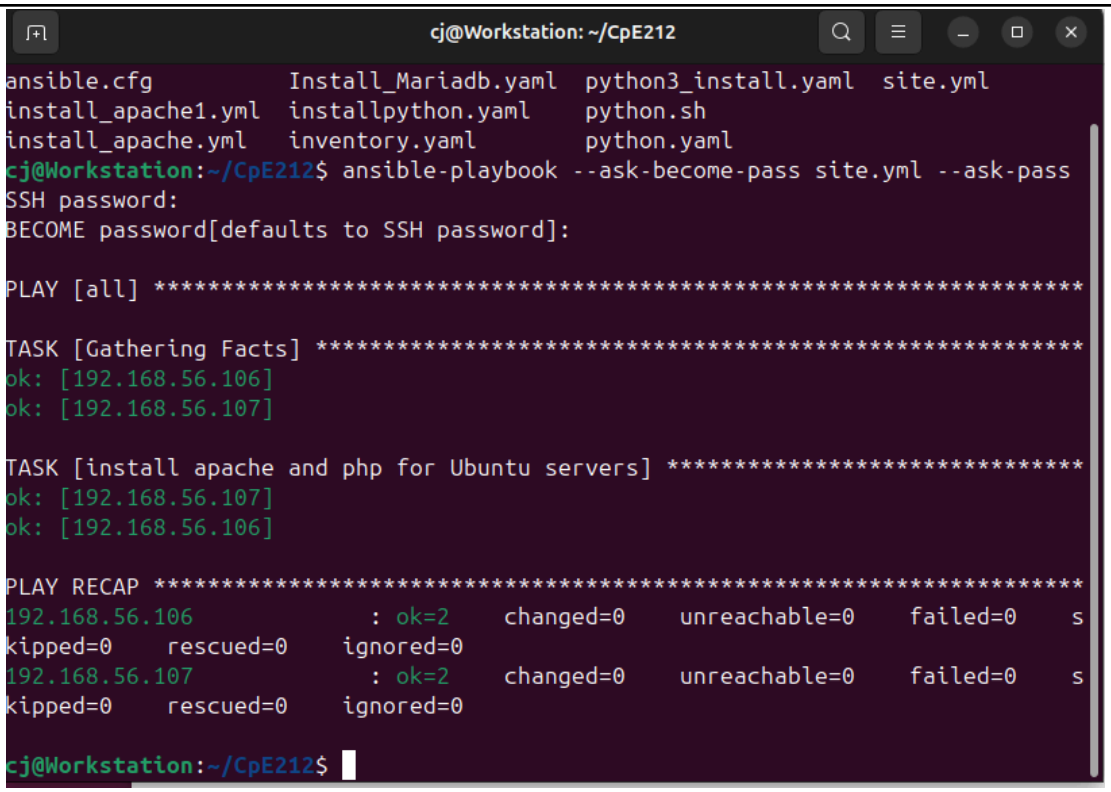
1. Create a new playbook and named it site.yml. Follow the commands as shown in the image below. Make sure to save the file and exit.

```
---

- 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.


cj@Workstation: ~/CpE212
ansible.cfg Install_Mariadb.yaml python3_install.yaml site.yml
install_apache1.yml installpython.yaml python.sh
install_apache.yml inventory.yaml python.yaml
cj@Workstation:~/CpE212$ ansible-playbook --ask-become-pass site.yml --ask-pass
SSH password:
BECOME password[defaults to SSH password]:
PLAY [all] **
TASK [Gathering Facts] **
ok: [192.168.56.106]
ok: [192.168.56.107]
TASK [install apache and php for Ubuntu servers] ******************************
ok: [192.168.56.107]
ok: [192.168.56.106]
PLAY RECAP **
192.168.56.106 : ok=2 changed=0 unreachable=0 failed=0 s
kipped=0 rescued=0 ignored=0
192.168.56.107 : ok=2 changed=0 unreachable=0 failed=0 s
kipped=0 rescued=0 ignored=0
cj@Workstation:~/CpE212$


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"

```
carl@Carl:~/CpEMidterm$ ansible-playbook site.yml -K
BECOME password:

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

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

TASK [install apache and php for Ubuntu servers] *******************************
changed: [192.168.56.101]

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

carl@Carl:~/CpEMidterm$
```

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.

```
GNU nano 7.2                                site.yml *

  - name: install updates (Ubuntu)
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

- hosts: db_servers
  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"
```

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

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.

```
carl@Carl:~/CpEMidterm$ ansible-playbook site.yml -K
BECOME password:

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

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

TASK [install updates (Ubuntu)] ************************************************
ok: [192.168.56.101]

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

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

TASK [install apache and php for Ubuntu servers] *******************************
ok: [192.168.56.101]

PLAY RECAP *********************************************************************
192.168.56.101             : ok=4    changed=0    unreachable=0    failed=0
kipped=0    rescued=0    ignored=0
```

Analysis: It executes the first task which is installing updates on my ubuntu manage node before installing apache and php.

```
- 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.

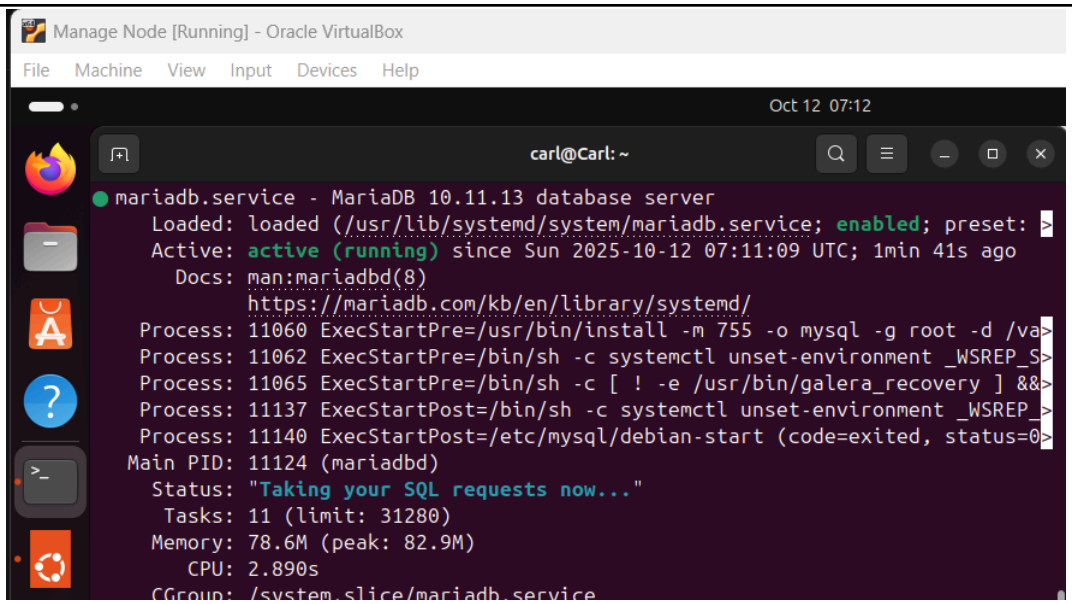
```
inventory.ini   new.yml   site.yml   beforesite.yml
site.yml
  1  ---
  2
  3  - hosts: file_servers
  4    become: true
  5    tasks:
  6
  7    - name: install mariadb package (Ubuntu)
  8      apt:
  9        name: mariadb-server
 10        state: latest
 11      when: ansible_distribution == "Ubuntu"
 12
 13    - name: "Mariadb -Restart/Enabling"
 14      service:
 15        name: mariadb
 16        state: restarted
 17        enabled: true
```

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

```
carl@Carl:~/CpEMidterm$ ansible-playbook site.yml -K
ok: [192.168.56.101]

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

TASK [Mariadb -Restart/Enabling] ***********************************
***************
changed: [192.168.56.101]

PLAY RECAP *********************************************************
***************
192.168.56.101             : ok=3    changed=2    unreachable=0
 failed=0    skipped=0    rescued=0    ignored=0

carl@Carl:~/CpEMidterm$
                                                          Ln 16, Col
```

Output: As we can see from our managenode there are changes that happen which means the file has been installed.

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.

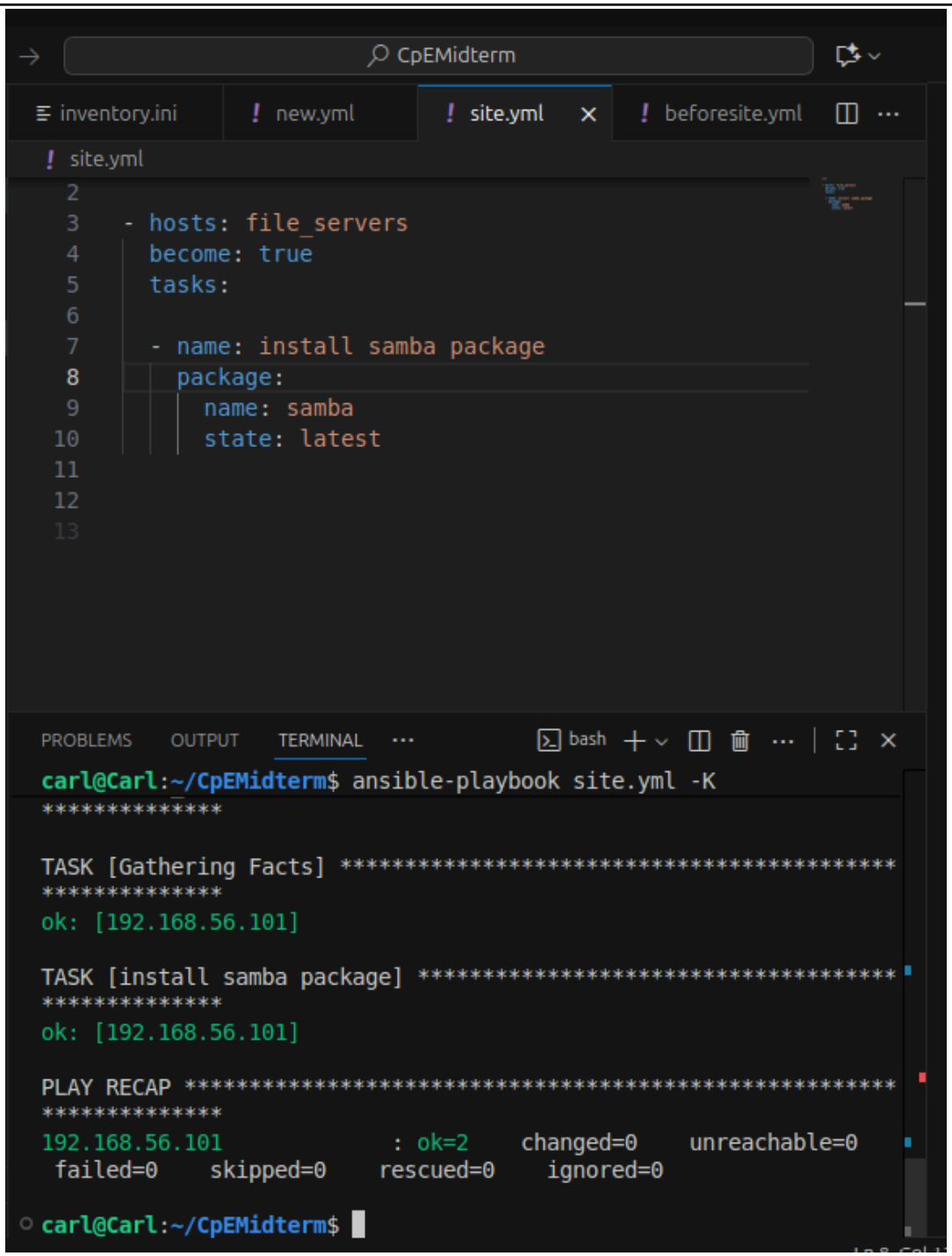


Describe the output.
Analysis: The mariadb is running and is enabled in my ubuntu managenode.

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.

Output: It successfully installed the samba package, but there are no changes indicated in the status of the playbook I run.

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.

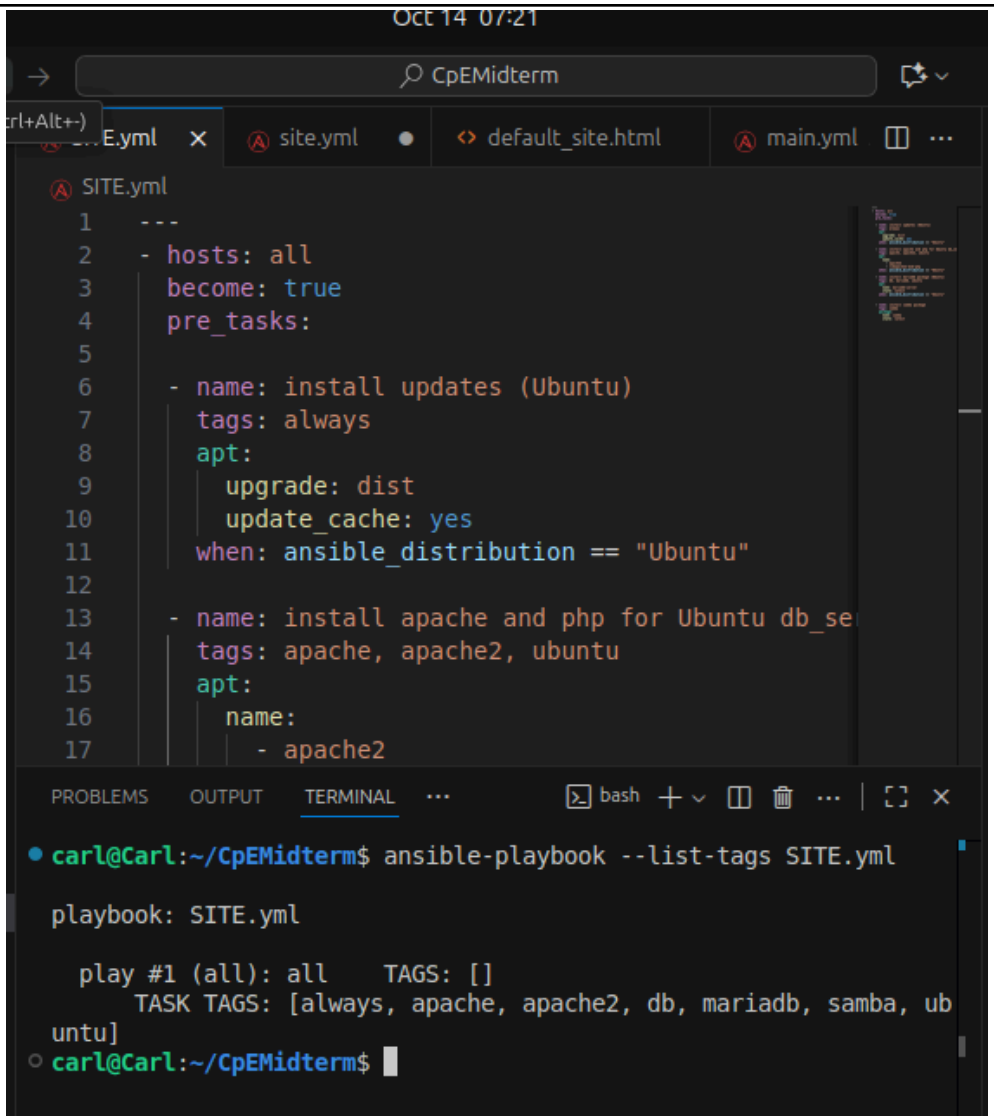
```
---

- 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*



Observation: When we typed "--list-tags" on our ansible playbook command we are listing all of the tags we used each specific play on our playbook.

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

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

```
carl@Carl:~/CpEMidterm$ ansible-playbook --tags db --ask-become-p
ass SITE.yml
BECOME password:

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

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

TASK [install updates (Ubuntu)] ************************************
***************
ok: [192.168.56.101]

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

PLAY RECAP *********************************************************
***************
192.168.56.101             : ok=3    changed=0    unreachable=0
  failed=0    skipped=0    rescued=0    ignored=0

carl@Carl:~/CpEMidterm$
```

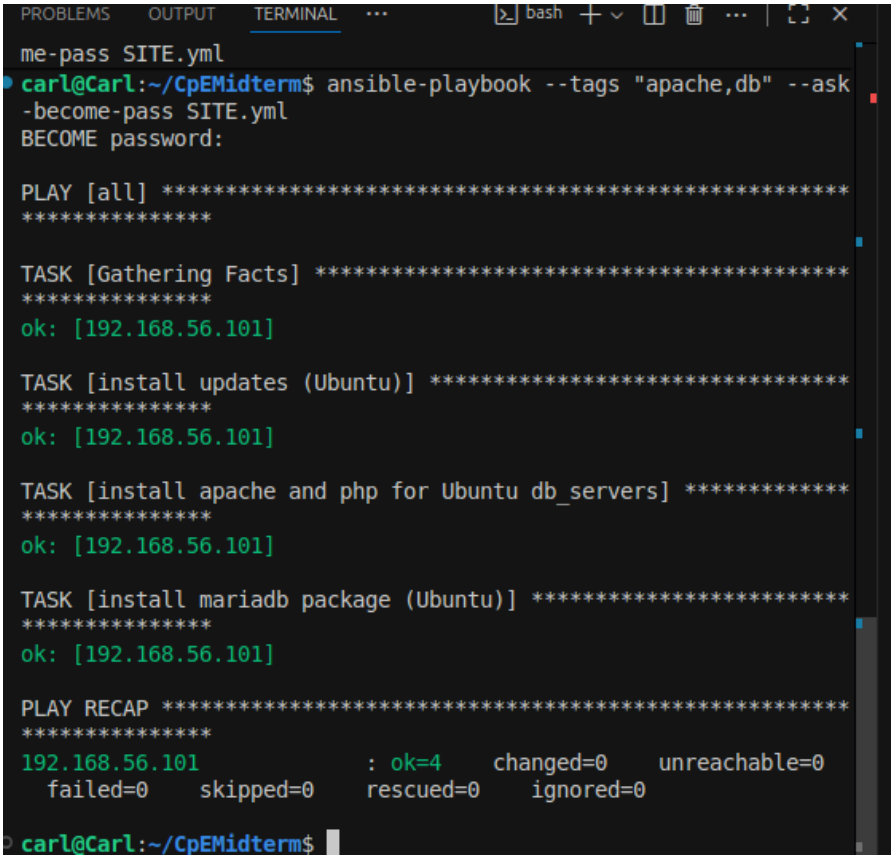
Observation: As we can see here our playbook does the task for installing mariadb on our managenode because we have chosen one of the tags we placed on the playbook for installing mariadb.

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

```
carl@Carl:~/CpEMidterm$ ansible-playbook --tags apache --ask-beco
me-pass SITE.yml
BECOME password:

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

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

TASK [install updates (Ubuntu)] ************************************
***************
ok: [192.168.56.101]

TASK [install apache and php for Ubuntu db_servers] ****************
***************
ok: [192.168.56.101]

PLAY RECAP *********************************************************
***************
192.168.56.101             : ok=3    changed=0    unreachable=0
  failed=0    skipped=0    rescued=0    ignored=0

carl@Carl:~/CpEMidterm$
```

Observation: Using the tags “apache” enables our playbook to do only a specific task which is installing apache for ubuntu servers.

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



Observation: As we can see the tags we placed for the playbook of installing mariadb and apache were called here “apache,db” making the playbook do the tasks only for installing mariadb and apache.

Task 3: Managing Services

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

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

- name: start httpd (CentOS)
  tags: apache, centos,httpd
  service:
    name: httpd
    state: started
  when: ansible_distribution == "CentOS"
```

Figure 3.1.1
Make sure to save the file and exit.

You would also notice from our previous activity that we already created a module that runs a service.

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

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

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

Figure 3.1.2

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

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

When prompted, enter the sudo password. After that, open the browser and enter the CentOS server's IP address. You should not be getting a display because we stopped the httpd service already.

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?
   - It makes the tasks of installing applications or files be specified to the number or remote servers present in the group.
2. What is the importance of tags in playbooks?
   - It reduces the possibility of installing redundant files than the particular file we are only to install to our remote servers using ansible playbook.
3. Why do think some services need to be managed automatically in playbooks?
   - It allows the practice of automation reducing manually installing of files per device. It allows the programmer to know which servers a specific tasks is assigned to and the outcomes or result of those tasks are predicted.

Note: Sir I was only able to do the ansible playbook on my ubuntu manage node and not on the CentOS but I do appreciate your considerations.