Name: Robin E. Valenzuela	Date Performed: 10/8/2022
Course/Section: CPE31S24	Date Submitted:
Instructor: Engr. Jonathan Taylar	Semester and SY: 1st Sem, 2022-2023

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"
```

```
---

- 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
```

BEFORE:

```
[Default]
192.168.56.102 apache_package=apache2 php_package=libapache2-mod-
192.168.56.103 apache_package=apache2 php_package=libapache2-mod-
192.168.56.104 apache_package=httpd php_package=php
```

AFTER:

```
[web_servers]
192.168.56.102
192.168.56.104

[db_servers]
192.168.56.102

[file_servers]
192.168.56.104
```

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"
```

Make sure to save the file and exit.

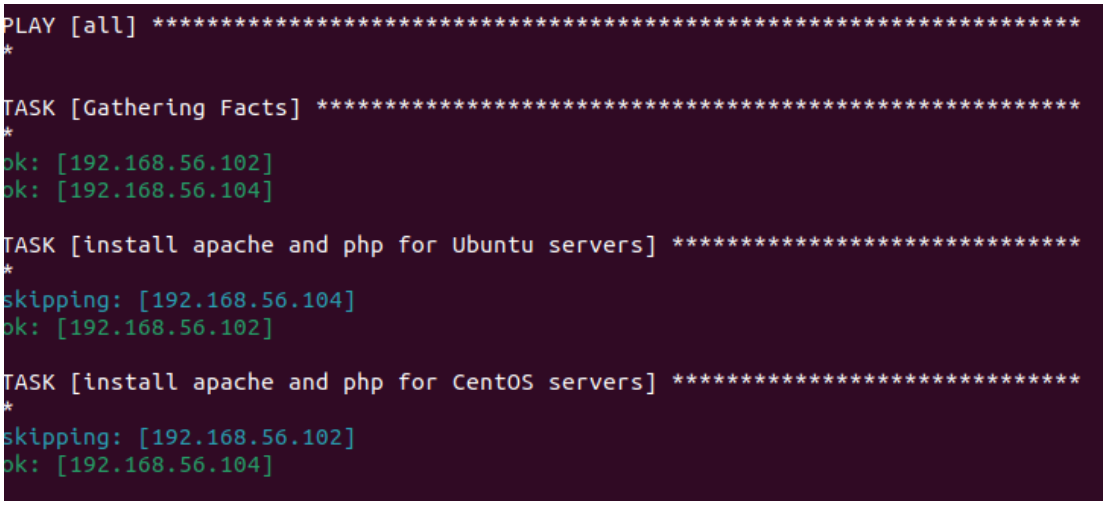
```
- hosts: all
  become: true
  pre_tasks:

  - 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"
```

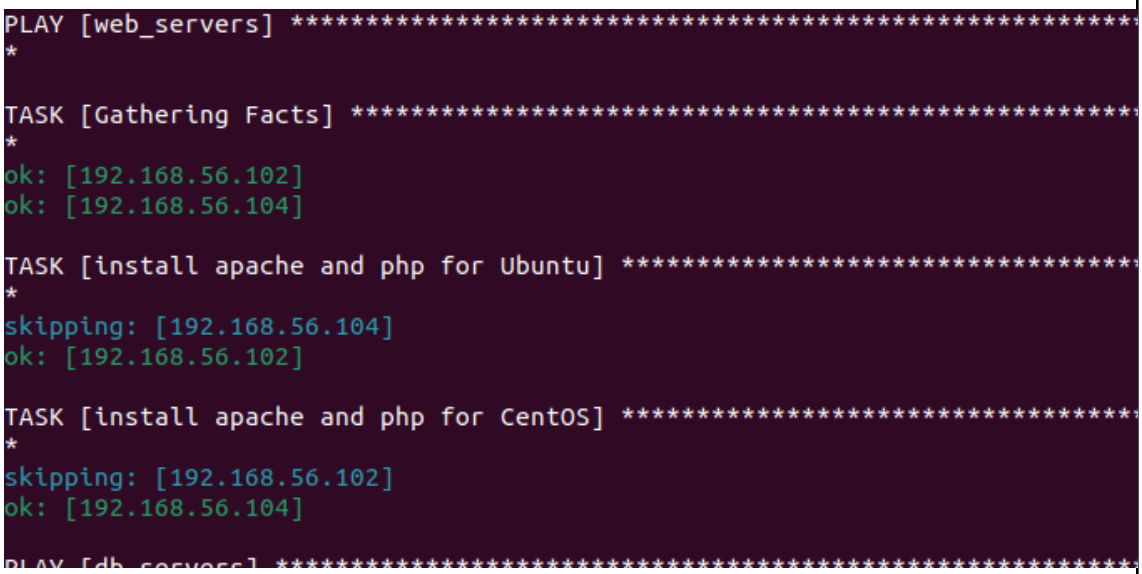
```
- hosts: web_servers
  become: true
  tasks:

  - 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"
```

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.





It will first run the pre-task then the webservers

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"
```

```
- 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: mariabdb-server
     state: latest
   when: ansible_distribution == "Ubuntu"
```

y

Make sure to save the file and exit.

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

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

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

TASK [install mariadb package (CentOS)] ****
*
changed: [192.168.56.104]

TASK [Mariadb - Restarting/Enabling] *******
*
changed: [192.168.56.104]

TASK [install mariadb package (Ubuntu)] ****
*
skipping: [192.168.56.104]

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

since linux works with Structural programming, (meaning up to down) it does not read Ubuntu since it is not at the top, so I used my CentOS ip instead so I could install MariaDB.

This is the ubuntu, the figure above is CentOS:

```
LAY [db_servers] ****************************************************


ASK [Gathering Facts] ***********************************************

k: [192.168.56.102]

ASK [install mariadb package (CentOS)] ******************************

kipping: [192.168.56.102]

ASK [Mariadb - Restarting/Enabling] *********************************

atal: [192.168.56.102]: FAILED! => {"changed": false, "msg": "Could not f
e requested service mariadb: host"}

LAY RECAP ***********************************************************
```

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.

```
File  Edit  View  Search  Terminal  Help
[valenzuela@workstationCent ~]$ systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: dis
abled)
   Active: active (running) since Sat 2022-10-08 10:13:12 PST; 13min ago
  Process: 6032 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited, st
atus=0/SUCCESS)
  Process: 5945 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n (code=exited, statu
s=0/SUCCESS)
 Main PID: 6031 (mysqld_safe)
    Tasks: 20
   CGroup: /system.slice/mariadb.service
           ├─6031 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           └─6196 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plug...
```

```
es   Terminal                    Oct 8  10:32
                                 valenzuela@server1: ~
valenzuela@server1:~$ systemctl status mariadb
Unit mariadb.service could not be found.
valenzuela@server1:~$ S
```

   The output is different for each since it will not install on the Ubuntu one since the IP that I put in inventory is for CentOS.

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: [192.168.56.102]

TASK [install samba package] ***************************************
*
changed: [192.168.56.102]

PLAY RECAP *********************************************************
*
192.168.56.102             : ok=6    changed=1    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.56.104             : ok=7    changed=1    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0
```

Since I have set the samba package to the IP of Ubuntu, it is 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.

```yaml
---

- 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"
```

```yaml
- 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
```

```yaml
- 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"


  - name: install apache and php for Ubuntu servers
    apt:
       name:
          - apache2
```


- 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"
- hosts: web_servers
become: true
tasks:
- name: install apache and php for Ubuntu
tags: apache,apache2,ubuntu
apt:
name:

```
    state: latest
  when: ansible_distribution == "Ubuntu"

- name: install apache and php for CentOS
  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
  yum:
    name: mariadb-server
    state: latest
  when: ansible_distribution == "CentOS"

- name: "Mariadb - Restarting/Enabling"
  service:
```

```
  when: ansible_distribution == "CentOS"

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

- name: install mariadb package (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

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

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible-playbook --list-tags site.y
ml

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]
```

- *This command lists down the tags that were used in the .yml*

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

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible-playbook --tags ce
k-become-pass site.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************
*
ok: [192.168.56.104]
ok: [192.168.56.102]

TASK [install updates (CentOS)] ************************************
*
skipping: [192.168.56.102]
ok: [192.168.56.104]

TASK [install updates (Ubuntu)] ************************************
*
skipping: [192.168.56.104]
ok: [192.168.56.102]

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

- *The command ran with tags 'centos'*

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

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

TASK [Gathering Facts] *********************************************
*
ok: [192.168.56.102]
ok: [192.168.56.104]

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

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

TASK [install mariadb package (CentOS)] ****************************
*
ok: [192.168.56.104]

TASK [install mariadb package (Ubuntu)] ****************************
*
skipping: [192.168.56.104]

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

- *this command ran with tags 'db'*

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

-this command ran with tags ' apache'

```
TASK [install apache and php for CentOS] ***********************************
*
skipping: [192.168.56.102]
ok: [192.168.56.104]

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

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

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

TASK [Gathering Facts] *****************************************************
*
ok: [192.168.56.102]

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

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

- *this command ran with tags 'apache and db'*

```
TASK [install apache and php for Ubuntu] ***********************************
*
skipping: [192.168.56.104]
ok: [192.168.56.102]

TASK [install apache and php for CentOS] ***********************************
*
skipping: [192.168.56.102]
ok: [192.168.56.104]

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

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

TASK [install mariadb package (CentOS)] ************************************
*
ok: [192.168.56.104]

TASK [install mariadb package (Ubuntu)] ************************************
*
skipping: [192.168.56.104]

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

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: start httdp (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.

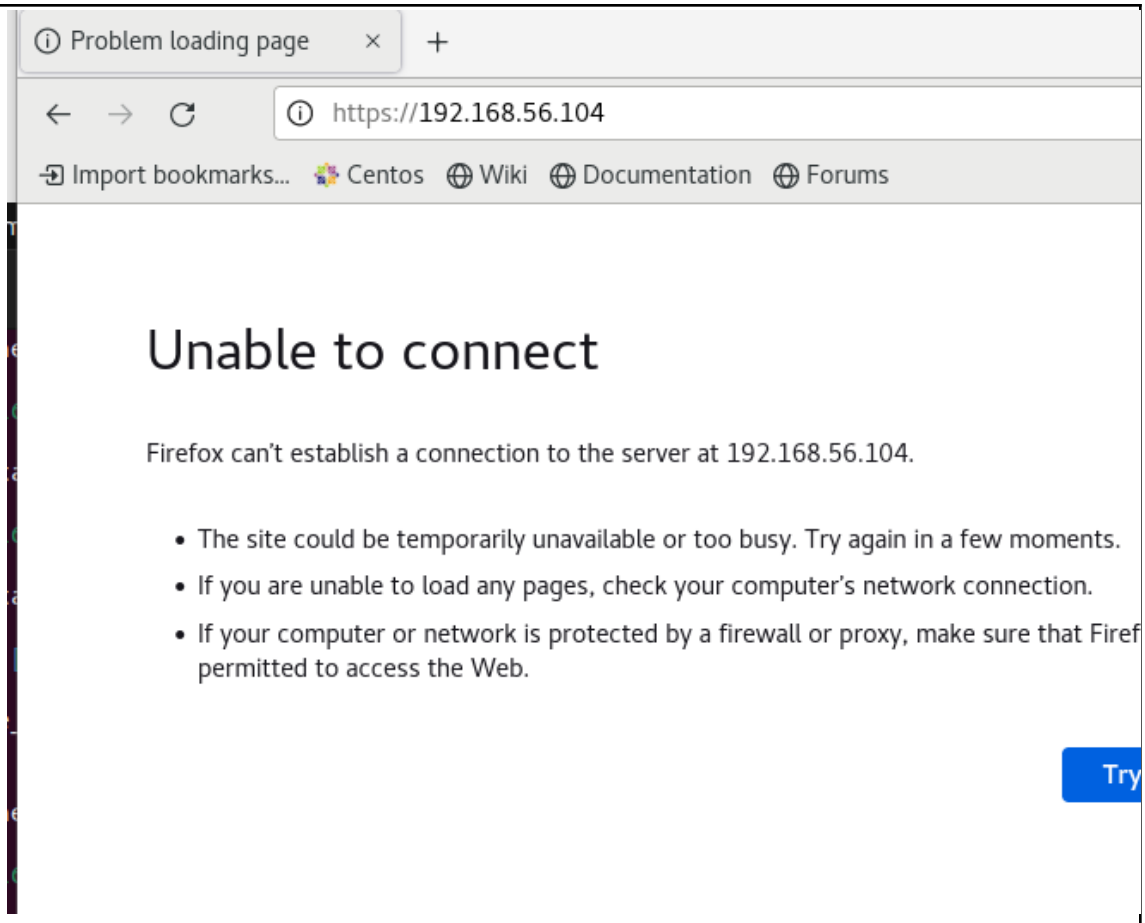
You would also notice from our previous activity that we already created a module that runs a service.

```
- 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.

```
TASK [start httdp (CentOS)] *****
*
skipping: [192.168.56.102]
changed: [192.168.56.104]
```

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?
   - The most important thing that I've learned is that it keeps it organized. Sometimes, we don't use every IP address in the inventory, and we just want a specific server to run specific tasks.ans

2. What is the importance of tags in playbooks?
    - The importance of tags is that you get to call the tags instead of playing every task in the play book. You have the choice to target a specific task.
3. Why do think some services need to be managed automatically in playbooks?
    - I think that you need some services to be automatic so that you have an ease of service. So that you don't need to turn them on manually, like the httpd in CentOS, it is much easier to run them in a playbook rather than typing out the exact command every time you login.