Name: Potestades, North Nygel G.	**Date Performed:** 9/19/25
Course/Section: CPE31S2	**Date Submitted:** 9/19/25
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Semester 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.

```
[web_servers]
192.168.56.112
192.168.56.113
[db_servers]
192.168.56.107
[file_servers]
192.168.56.119
```

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.

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.112             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=1    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

For two of the servers, only 2 tasks were performed and 1 skipped, as they only performed the pre-tasks. The targeted group, being web servers had 2 more tasks and 1 more skipped task, as they were both Ubuntu distributions.

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.

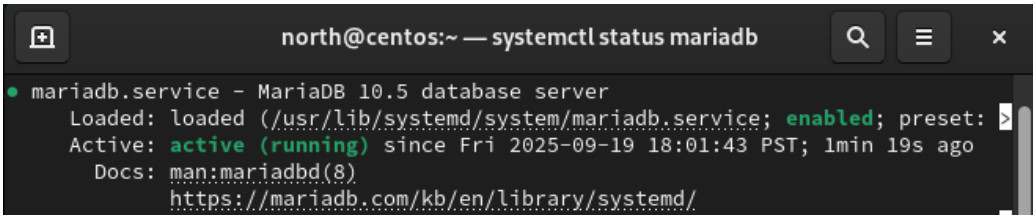
Run the *site.yml* file and describe the result.

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=5    changed=2    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.112             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

The result is that in addition to the previous tasks, 107, which is my CentOS distribution, has 5 more tasks performed and 1 more skipped, as the last task is for Ubuntu distributions only.

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.



There is only one CentOS server assigned to db_servers, so only the CentOS output can be seen. As seen in the screenshot, the CentOS MariaDB is enabled, while if there was an Ubuntu distribution added in the same group, it would be installed but not enabled.

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 RECAP *********************************************************************
192.168.56.107             : ok=5    changed=1    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.112             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=4    changed=1    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

The result is that 119, the only server in file_servers, has 3 more tasks completed, as it is the only server specified to install the latest version of samba.

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

```
north@workstation:~/CPE212_Potestades$ ansible-playbook --list-tags site.yaml

playbook: site.yaml

  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 displays the tags which are associated with the different tasks and the different groups of tasks.

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

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.112             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

This command plays only those tasks with the tags centos, resulting in only the tasks which are labeled with centos to play.

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

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.112             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.113             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.119             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

The same as the previous, though this time only playing those tasks which are labeled with db.

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

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
192.168.56.112             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

This command plays only tasks which are tagged with apache, resulting in more tasks being completed by those in the web servers group.

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

```
PLAY RECAP *********************************************************************
192.168.56.107             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.112             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.113             : ok=4    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
192.168.56.119             : ok=3    changed=0    unreachable=0    failed=0    s
kipped=1    rescued=0    ignored=0
```

This command plays only tasks which are tagged with the tags apache or db, resulting in a higher number of tasks performed for all 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"
```

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.

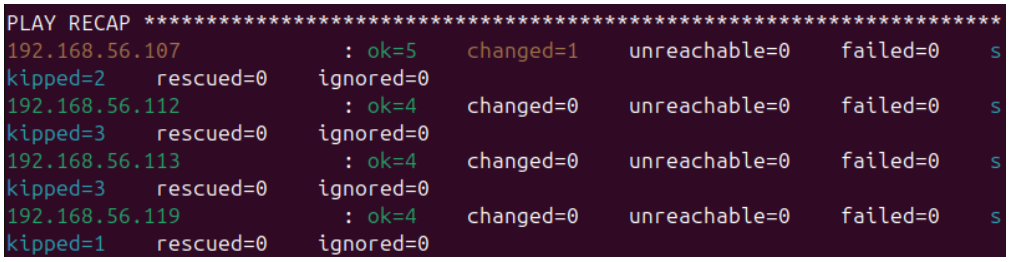
```
- hosts: db_servers
  become: true
  tasks:

  - 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.



It still does not work, as my CentOS server is not under the web_servers group, though if it was then it would work now, as it would be activated.

GitHub Link: Git

Reflections:

Answer the following:

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

Grouping remote servers allows you to target specific groups whenever needed, and adding a new group is as easy as just adding a new group to the inventory file. It also helps with readability, as having just IP addresses can be confusing if you do not know which systems they are assigned to.

2. What is the importance of tags in playbooks?

Tags are important as they let you run tasks with only specific tags in playbooks, letting you shorten the execution time and letting you test certain tasks before running the full playbook. It can also help with updating only specific packages on remote servers, so that you do not need to create new playbooks for each package you want to upgrade.

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

They need to be managed automatically because most services do not run immediately after being installed, meaning you need to run them automatically if you want to be able to use them as soon as the playbook is done running.