

Name: San Jose, Kier Justin F.	Date Performed: 10/4/2024
Course/Section: CPE 212 - CPE31S21	Date Submitted: 10/4/2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 2024-2025
Activity 6: Targeting Specific Nodes and Managing Services	
<p>1. Objectives:</p> <ul style="list-style-type: none"> 1.1 Individualize hosts 1.2 Apply tags in selecting plays to run 1.3 Managing Services from remote servers using playbooks 	
<p>2. Discussion:</p> <p>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.</p> <p>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.</p> <p>Requirement:</p> <p>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 <i>ssh-copy-id</i> to copy the public key to Server 3. Verify if you can successfully SSH to Server 3.</p>	
Task 1: Targeting Specific Nodes	
<ul style="list-style-type: none"> 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.

```

GNU nano 2.9.3                                inventory

[servers]
server1 ansible_host=192.168.56.151 ansible_user=kier
test3 ansible_host=192.168.56.155 ansible_user=kier

[server_centos]
centos ansible_host=192.168.56.154 ansible_user=kiersanjose

```

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:

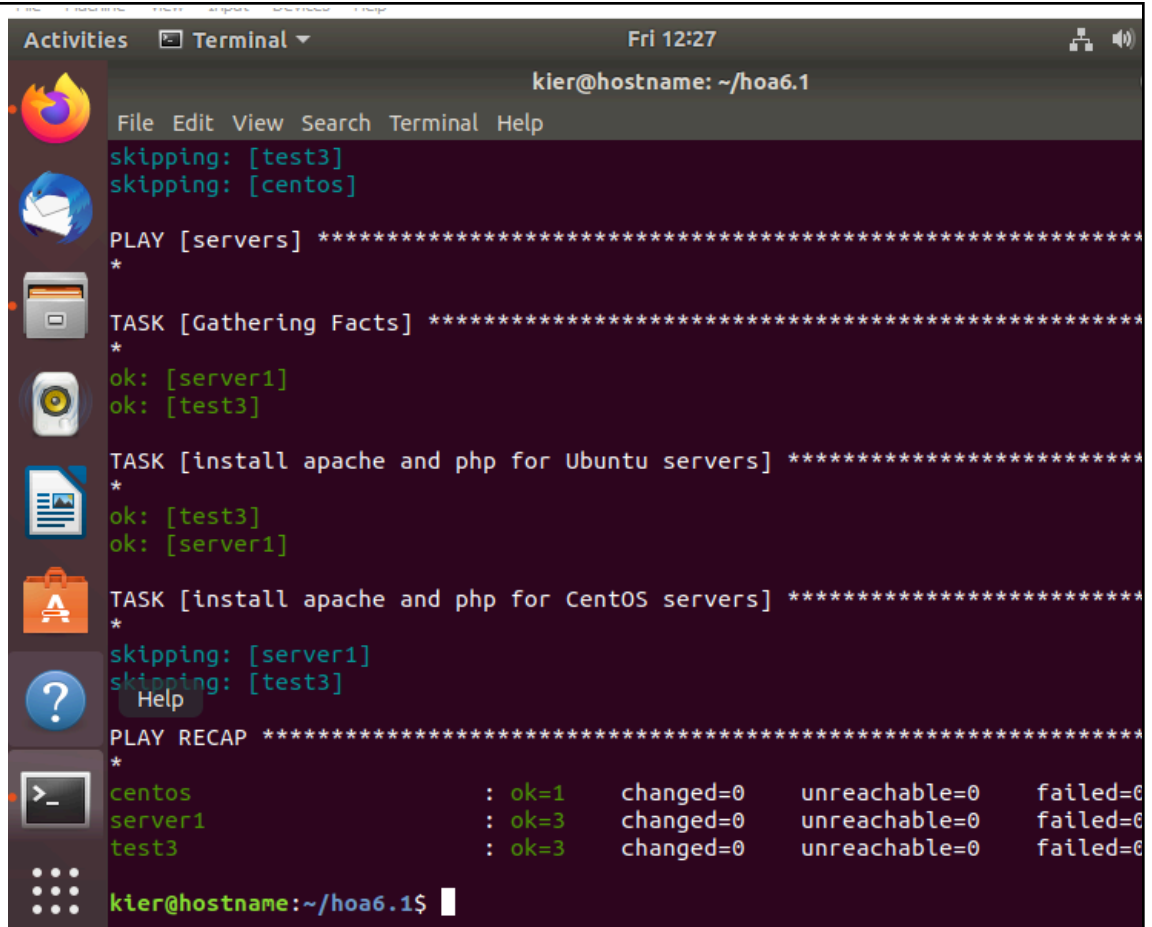
```

- - -
- 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.

A terminal window titled 'Terminal' showing the output of an Ansible playbook. The user is 'kier' on a host named 'hostname' in the directory '~/hoa6.1'. The output shows several plays and tasks. The first play is '[servers]' which skips '[test3]' and '[centos]'. The second play is '[Gathering Facts]' which is successful for '[server1]' and '[test3]'. The third play is '[install apache and php for Ubuntu servers]' which is successful for '[test3]' and '[server1]'. The fourth play is '[install apache and php for CentOS servers]' which skips '[server1]' and '[test3]'. Finally, a 'PLAY RECAP' section shows the summary for '[centos]', '[server1]', and '[test3]'.

```
Activities Terminal Fri 12:27
kier@hostname: ~/hoa6.1
File Edit View Search Terminal Help
skipping: [test3]
skipping: [centos]
PLAY [servers] *****
*
TASK [Gathering Facts] *****
*
ok: [server1]
ok: [test3]
TASK [install apache and php for Ubuntu servers] *****
*
ok: [test3]
ok: [server1]
TASK [install apache and php for CentOS servers] *****
*
skipping: [server1]
skipping: [test3]
PLAY RECAP *****
*
centos : ok=1 changed=0 unreachable=0 failed=0
server1 : ok=3 changed=0 unreachable=0 failed=0
test3 : ok=3 changed=0 unreachable=0 failed=0
kier@hostname:~/hoa6.1$
```

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.

- 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 package (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.

```

TASK [Gathering Facts] *****
*
ok: [server1]
ok: [test3]

TASK [Install MariaDB package (CentOS)] *****
*
skipping: [server1]
skipping: [test3]

TASK [Install MariaDB package (Ubuntu)] *****
*
ok: [server1]
ok: [test3]

TASK [Restart and enable MariaDB] *****
*
skipping: [server1]
skipping: [test3]

PLAY RECAP *****
*
centos                : ok=1    changed=0    unreachable=0    failed=0
server1               : ok=6    changed=0    unreachable=0    failed=0
test3                 : ok=6    changed=0    unreachable=0    failed=0

kier@hostname:~/hoa6.1$

```

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.

- It doesn't detect the mariadb service.

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.

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 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.

```

File Edit View Search Terminal Help
TASK [Install MariaDB package (CentOS)] *****
*
skipping: [server1]
skipping: [test3]

TASK [Install MariaDB package (Ubuntu)] *****
*
ok: [server1]
ok: [test3]

TASK [Restart and enable MariaDB] *****
*
skipping: [server1]
skipping: [test3]
[WARNING]: Could not match supplied host pattern, ignoring: file_servers

PLAY [file_servers] *****
*
skipping: no hosts matched

PLAY RECAP *****
*
centos                : ok=1    changed=0    unreachable=0    failed=0
server1              : ok=6    changed=0    unreachable=0    failed=0
test3                 : ok=6    changed=0    unreachable=0    failed=0

kier@hostname:~/hoa6.1$

```

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

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

```

kier@hostname:~/hoa6.1$ ansible-playbook --list-tags site.yml
[WARNING]: Could not match supplied host pattern, ignoring:

playbook: site.yml

  play #1 (all): all    TAGS: []
    TASK TAGS: []

  play #2 (servers): servers    TAGS: []
    TASK TAGS: []

  play #3 (servers): servers    TAGS: []
    TASK TAGS: []

  play #4 (file_servers): file_servers    TAGS: []
    TASK TAGS: []
kier@hostname:~/hoa6.1$

```

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

```
*  
  
TASK [Gathering Facts] *****  
*  
ok: [test3]  
ok: [server1]  
  
PLAY [servers] *****  
*  
  
TASK [Gathering Facts] *****  
*  
ok: [server1]  
ok: [test3]  
[WARNING]: Could not match supplied host pattern, ignoring:  
  
PLAY [file_servers] *****  
*  
skipping: no hosts matched  
  
PLAY RECAP *****  
*  
centos           : ok=1    changed=0    unreachable  
server1          : ok=3    changed=0    unreachable  
test3            : ok=3    changed=0    unreachable
```

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

```
ok: [server1]  
ok: [test3]  
[WARNING]: Could not match supplied host pattern, ignoring:  
  
PLAY [file_servers] *****  
*  
skipping: no hosts matched  
  
PLAY RECAP *****  
*  
centos           : ok=1    changed=0    unreachable  
server1          : ok=3    changed=0    unreachable  
test3            : ok=3    changed=0    unreachable  
  
kier@hostname:~/hoa6.1$
```

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

```
File Edit View Search Terminal Help
*
TASK [Gathering Facts] *****
*
ok: [server1]
ok: [test3]

PLAY [servers] *****
*

TASK [Gathering Facts] *****
*
ok: [server1]
ok: [test3]
[WARNING]: Could not match supplied host pattern, ignoring:

PLAY [file_servers] *****
*
skipping: no hosts matched

PLAY RECAP *****
*
centos                : ok=1    changed=0    unreachable
server1               : ok=3    changed=0    unreachable
test3                 : ok=3    changed=0    unreachable
```

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

```

*
TASK [Gathering Facts] *****
*
ok: [test3]
ok: [server1]

PLAY [servers] *****
*

TASK [Gathering Facts] *****
*
ok: [server1]
ok: [test3]
[WARNING]: Could not match supplied host pattern, ignoring:

PLAY [file_servers] *****
*
skipping: no hosts matched

PLAY RECAP *****
*
centos           : ok=1    changed=0    unreachable
server1          : ok=3    changed=0    unreachable
test3            : ok=3    changed=0    unreachable

kier@hostname:~/hoa6.1$

```

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?
 - Grouping remote servers together is important for several reasons. Firstly, it allows system administrators to manage and monitor the corporate network remotely, making it easier to perform tasks on multiple servers at once. Secondly, it enables faster troubleshooting, as technicians can more quickly identify and troubleshoot problems that may occur. Thirdly, grouping servers together can improve security by applying security policies to groups of servers, ensuring that all servers in the group have the same level of security. Lastly, remote server access can increase productivity by allowing system administrators to respond to issues more quickly, resulting in faster and more responsive services for customers.
2. What is the importance of tags in playbooks?
 - Tags are metadata that can be attached to tasks in an Ansible playbook, allowing for selective targeting of certain tasks at runtime. Adding tags to a play or statically imported tasks and roles adds those tags to all of the contained tasks, referred to as tag inheritance. Tags can be added to a single task or include, or to multiple tasks by defining them at the level of a block, play, role, or import. Tags can be used to select or skip tasks when running a playbook, making it possible to run specific tasks within a playbook on demand. Understanding how tags are passed within the hierarchy of imports is important because tags become tricky when using includes or imports. Tags can also be used to filter roles in a playbook, allowing for the selective running of one or two roles out of many. Overall, tags are important in playbooks because they allow for more efficient and selective execution of tasks.
3. Why do think some services need to be managed automatically in playbooks?
 - Managing services automatically in playbooks can provide several benefits, including increased efficiency, faster response times, improved customer

satisfaction, operational excellence, and scalability, as automated playbooks can detect new threats faster, reduce the need for manual intervention, and provide self-service options for users