Name: Jozette Fermin	**Date Performed:** Sept 18, 2023
Course/Section: CPE232	**Date Submitted:** Sept 18, 2023
Instructor: Engr. Jonathan Taylar	**Semester and SY:** 1st Sem 2023-2024

Activity 5: Consolidating Playbook plays

1. **Objectives:**

1.1 Use **when** command in playbook for different OS distributions

1.2 Apply refactoring techniques in cleaning up the playbook codes

2. **Discussion**:

We are going to look at a way that we can differentiate a playbook by a host in terms of which distribution the host is running. It's very common in most Linux shops to run multiple distributions, for example, Ubuntu shop or Debian shop and you need a different distribution for a one off-case or perhaps you want to run plays only on certain distributions.

It is a best practice in ansible when you are working in a collaborative environment to use the command git pull. git pull is a Git command used to update the local version of a repository from a remote. By default, git pull does two things. Updates the current local working branch (currently checked out branch) and updates the remote-tracking branches for all other branches. git pull essentially pulls down any changes that may have happened since the last time you worked on the repository.

Requirement:

In this activity, you will need to create a CentOS VM. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the CentOS VM. Make sure to use the command *ssh-copy-id* to copy the public key to CentOS. Verify if you can successfully SSH to CentOS VM.

Task 1: Use when command for different distributions

1. In the local machine, make sure you are in the local repository directory (*CPE232_yourname*). Issue the command git pull. When prompted, enter the correct passphrase or password. Describe what happened when you issue this command. Did something happen? Why?

2.
```
jozette@ManageNode:~/HOA5$ git pull
Username for 'https://github.com': qjpfermin@tip.edu.ph
Password for 'https://qjpfermin@tip.edu.ph@github.com':
Already up to date.
```

3. Edit the inventory file and add the IP address of the Centos VM. Issue the command we used to execute the playbook (the one we used in the last activity): *ansible-playbook --ask-become-pass install_apache.yml*. After executing this command, you may notice that it did not become successful in the Centos VM. You can see that the Centos VM has failed=1. Only the two remote servers have been changed. The reason is that Centos VM does not support "apt" as the package manager. The default package manager for Centos is "yum."

```
jozette@ManageNode:~/HOA5$ ansible-playbook --ask-become-pass install_apache.yaml
SUDO password:

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

TASK [Gathering Facts] *********************************************************************************
ok: [192.168.56.108]
fatal: [192.168.56.110]: FAILED! => {"msg": "Incorrect sudo password"}

TASK [update repository index] *************************************************************************
changed: [192.168.56.108]

TASK [install apache2 package] *************************************************************************
ok: [192.168.56.108]

TASK [add PHP support for apache] **********************************************************************
ok: [192.168.56.108]
        to retry, use: --limit @/home/jozette/HOA5/install_apache.retry

PLAY RECAP *********************************************************************************************
192.168.56.108             : ok=4    changed=1    unreachable=0    failed=0
192.168.56.110             : ok=0    changed=0    unreachable=0    failed=1
```

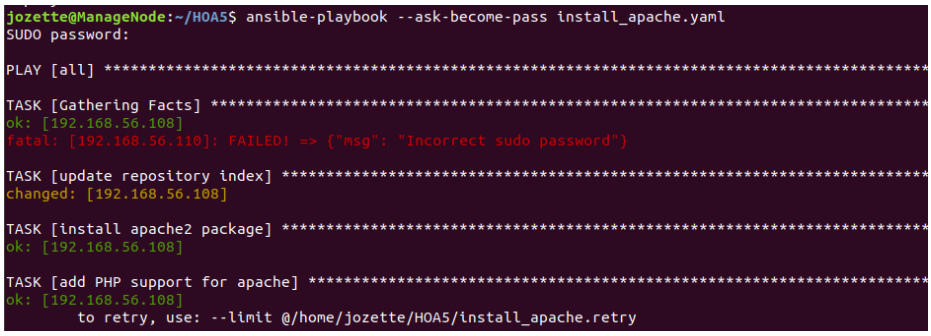
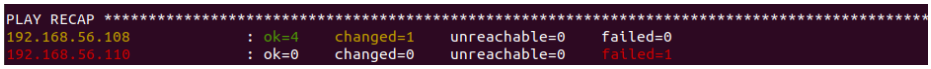
4. Edit the *install_apache.yml* file and insert the lines shown below.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 package
    apt:
      name: apache2
    when: ansible_distribution == "Ubuntu"

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
    when: ansible_distribution == "Ubuntu"
```

Make sure to save the file and exit.

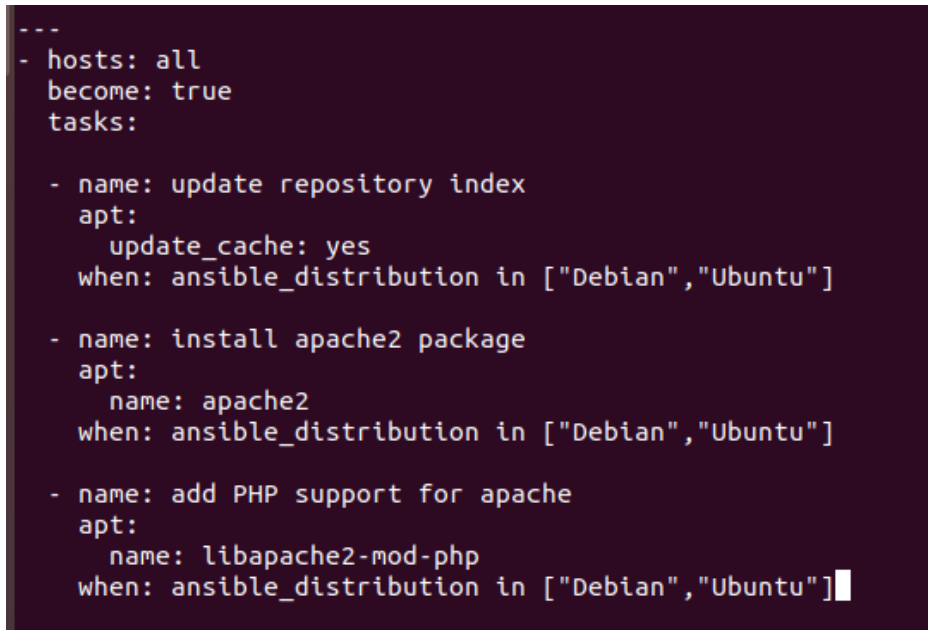
Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

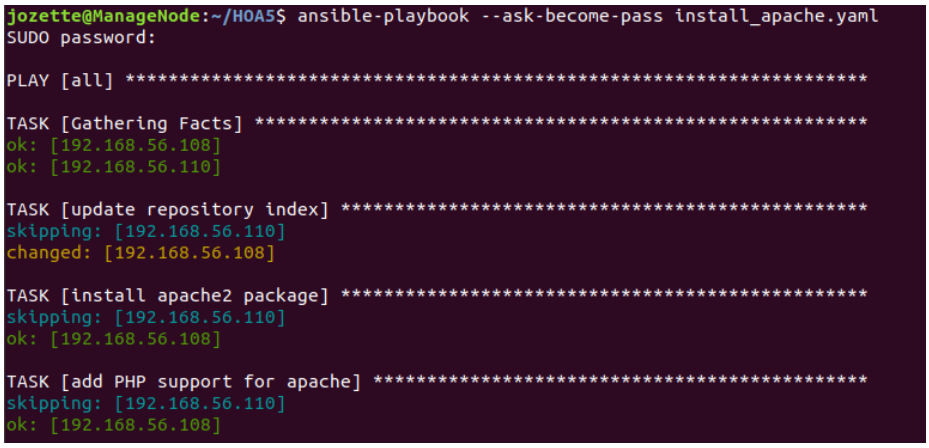
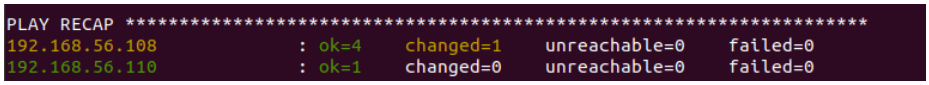
INPUT	--- - hosts: all become: true tasks: - name: update repository index apt: update_cache: yes when: ansible_distribution == "Ubuntu" - name: install apache2 package apt: name: apache2 when: ansible_distribution == "Ubuntu" - name: add PHP support for apache apt: name: libapache2-mod-php when: ansible_distribution == "Ubuntu"

PROCESS	
OUTPUT	

If you have a mix of Debian and Ubuntu servers, you can change the configuration of your playbook like this.

- name: update repository index
  apt:
    update_cache: yes
  when: ansible_distribution in [“Debian”, “Ubuntu]

INPUT	

PROCESS	
OUTPUT	

Note: This will work also if you try. Notice the changes are highlighted.

5. Edit the *install_apache.yml* file and insert the lines shown below.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 package
    apt:
      name: apache2
      stae: latest
    when: ansible_distribution == "Ubuntu"

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: update repository index
    dnf:
      update_cache: yes
    when: ansible_distribution == "CentOS"

  - name: install apache2 package
    dnf:
      name: httpd
      state: latest
    when: ansible_distribution == "CentOS"

  - name: add PHP support for apache
    dnf:
      name: php
      state: latest
    when: ansible_distribution == "CentOS"
```

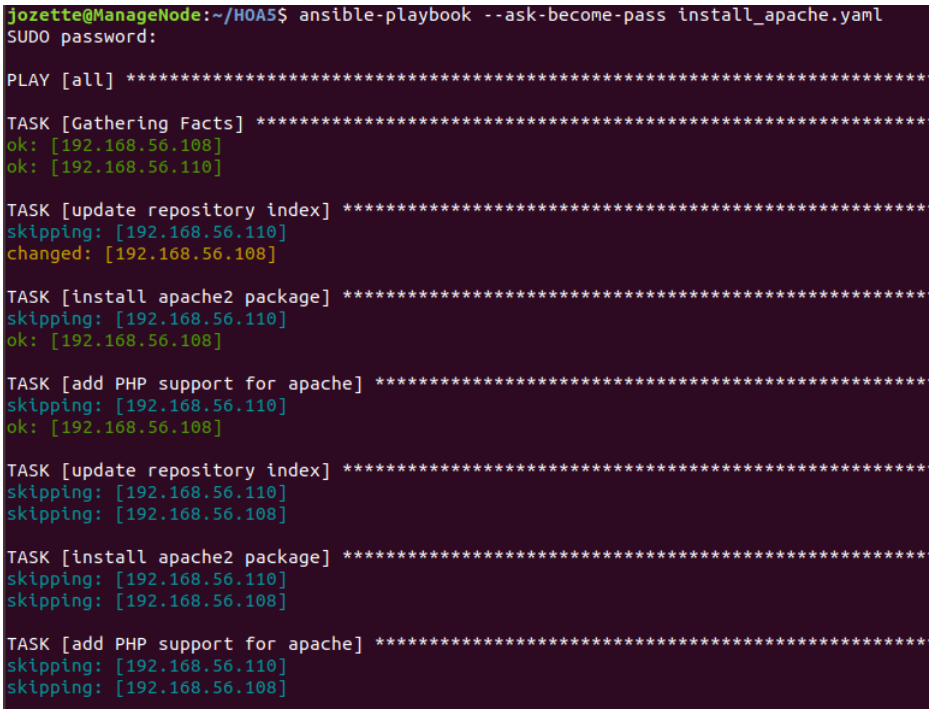
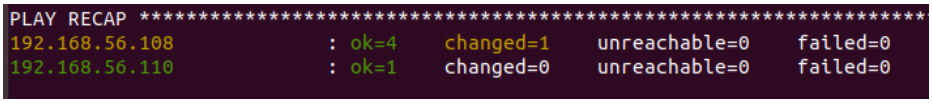
Make sure to save and exit.

Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 package
    apt:
      name: apache2
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"
```

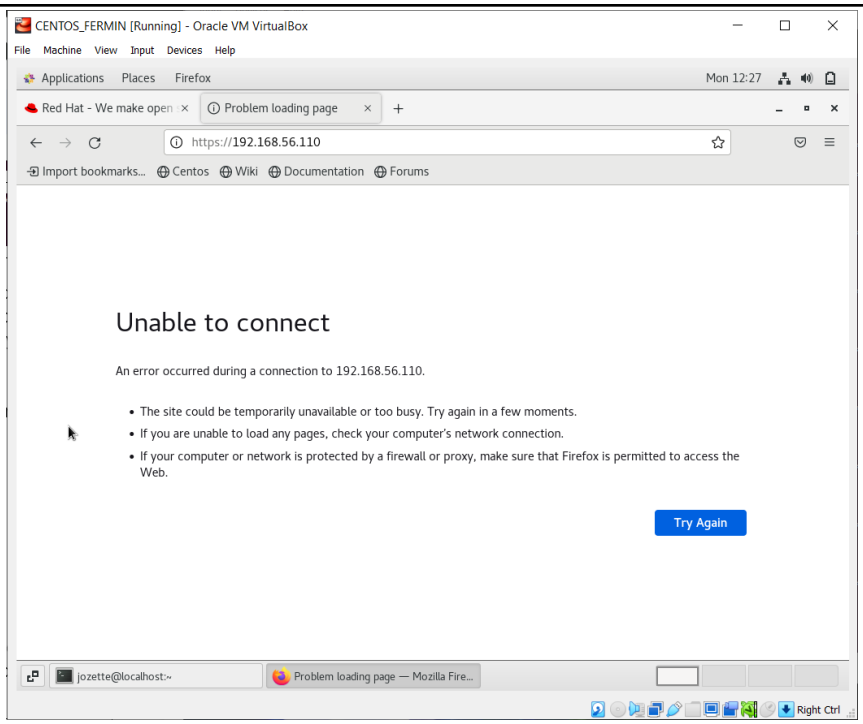
```
  - name: update repository index
    dnf:
      update_cache: yes
    when: ansible_distribution == "Debian"

  - name: install apache2 package
    dnf:
      name: httpd
      state: latest
    when: ansible_distribution == "Debian"

  - name: add PHP support for apache
    dnf:
      name: php
      status: latest
    when: ansible_distribution == "Debian"
```

PROCESS	
OUTPUT	

```
jozette@ManageNode:~/HOA5$ ansible-playbook --ask-become-pass install_apache.yaml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.108]
ok: [192.168.56.110]

TASK [update repository index] *************************************************
skipping: [192.168.56.110]
changed: [192.168.56.108]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.110]
ok: [192.168.56.108]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.110]
ok: [192.168.56.108]

TASK [update repository index] *************************************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]
```

```
PLAY RECAP *********************************************************************
192.168.56.108             : ok=4    changed=1    unreachable=0    failed=0
192.168.56.110             : ok=1    changed=0    unreachable=0    failed=0
```

6. To verify the installations, go to CentOS VM and type its IP address on the browser. Was it successful? The answer is no. It's because the httpd service or the Apache HTTP server in the CentOS is not yet active. Thus, you need to activate it first.



5.1 To activate, go to the CentOS VM terminal and enter the following:

systemctl status httpd

The result of this command tells you that the service is inactive.

```
[jozette@localhost ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor prese
t: disabled)
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)
```

6.2 Issue the following command to start the service:

sudo systemctl start httpd

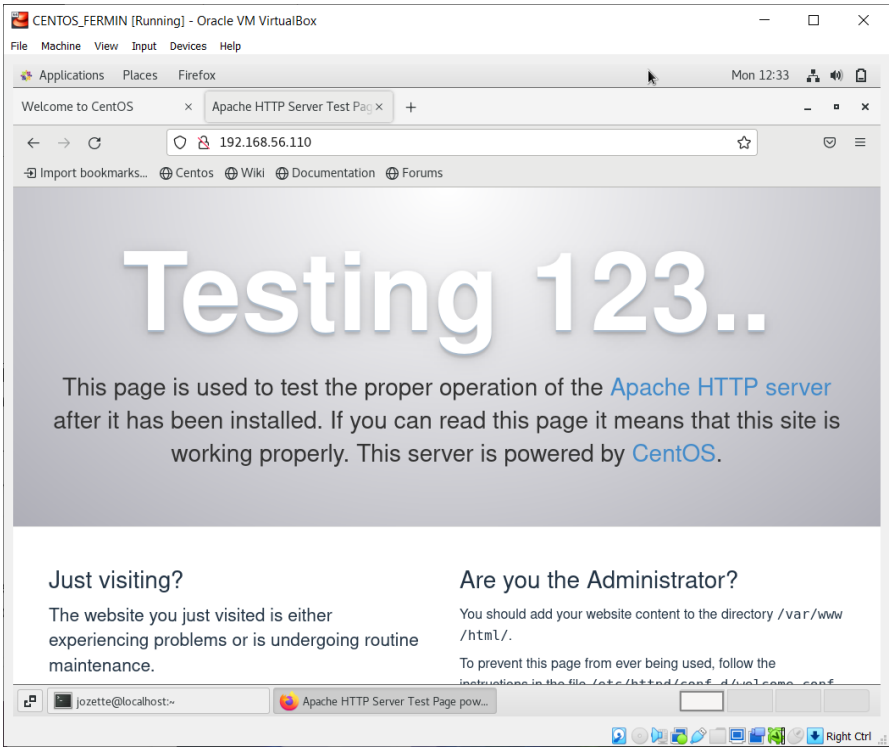
(When prompted, enter the sudo password)

sudo firewall-cmd --add-port=80/tcp

(The result should be a success)

```
[jozette@localhost ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled
   Active: active (running) since Mon 2023-09-18 12:30:47 PST; 50s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 12275 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
    Tasks: 6
   CGroup: /system.slice/httpd.service
           ├─12275 /usr/sbin/httpd -DFOREGROUND
           ├─12280 /usr/sbin/httpd -DFOREGROUND
           ├─12281 /usr/sbin/httpd -DFOREGROUND
           ├─12282 /usr/sbin/httpd -DFOREGROUND
           ├─12283 /usr/sbin/httpd -DFOREGROUND
           └─12284 /usr/sbin/httpd -DFOREGROUND

Sep 18 12:30:47 localhost.localdomain systemd[1]: Starting The Apache HTTP Se...
Sep 18 12:30:47 localhost.localdomain httpd[12275]: AH00558: httpd: Could not...
Sep 18 12:30:47 localhost.localdomain systemd[1]: Started The Apache HTTP Ser...
Hint: Some lines were ellipsized, use -l to show in full.
```

6.3 To verify the service is already running, go to CentOS VM and type its IP address on the browser. Was it successful? (Screenshot the browser)



Task 2: Refactoring playbook

This time, we want to make sure that our playbook is efficient and that the codes are easier to read. This will also makes run ansible more quickly if it has to execute fewer tasks to do the same thing.

1. Edit the playbook *install_apache.yml*. Currently, we have three tasks targeting our Ubuntu machines and 3 tasks targeting our CentOS machine. Right now, we try to consolidate some tasks that are typically the same. For example, we can consolidate two plays that install packages. We can do that by creating a list of installation packages as shown below:

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index Ubuntu
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 and php packages for Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: update repository index for CentOS
    dnf:
      update_cache: yes
    when: ansible_distribution == "CentOS"

  - name: install apache and php packages for CentOS
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "CentOS"
```

Make sure to save the file and exit.

Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 and php packages for Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"
```

```
  - name: update repository index
    dnf:
      update_cache: yes
    when: ansible_distribution == "Debian"

  - name: install apache and php packages for CentOS
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "Debian"
```

PROCESS

```
jozette@ManageNode:~/HOA5$ ansible-playbook --ask-become-pass install_apache.yaml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.108]
ok: [192.168.56.110]

TASK [update repository index] *************************************************
skipping: [192.168.56.110]
changed: [192.168.56.108]

TASK [install apache2 and php packages for Ubuntu] *****************************
skipping: [192.168.56.110]
ok: [192.168.56.108]

TASK [update repository index] *************************************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]

TASK [install apache and php packages for CentOS] ******************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]
```

OUTPUT	PLAY RECAP ********** 192.168.56.108 : ok=3 changed=1 unreachable=0 failed=0 192.168.56.110 : ok=1 changed=0 unreachable=0 failed=0

2. Edit the playbook *install_apache.yml* again. In task 2.1, we consolidated the plays into one play. This time we can actually consolidated everything in just 2 plays. This can be done by removing the update repository play and putting the command *update_cache: yes* below the command *state: latest*. See below for reference:

```
---
- hosts: all
  become: true
  tasks:

   - name: install apache2 and php packages for Ubuntu
     apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
     when: ansible_distribution == "Ubuntu"

   - name: install apache and php packages for CentOS
     dnf:
       name:
         - httpd
         - php
       state: latest
       update_cache: yes
     when: ansible_distribution == "CentOS"
```

Make sure to save the file and exit.

Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT

```
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 and php packages for Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php packages for CentOS
    dnf:
      name:
        - httpd
        - php
      state: latest
      update_cache: yes
    when: ansible_distribution == "Debian"
```

PROCESS

```
jozette@ManageNode:~/HOA5$ ansible-playbook --ask-become-pass install_apache.yaml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.108]
ok: [192.168.56.110]

TASK [install apache2 and php packages for Ubuntu] *****************************
skipping: [192.168.56.110]
ok: [192.168.56.108]

TASK [install apache and php packages for CentOS] ******************************
skipping: [192.168.56.110]
skipping: [192.168.56.108]
```

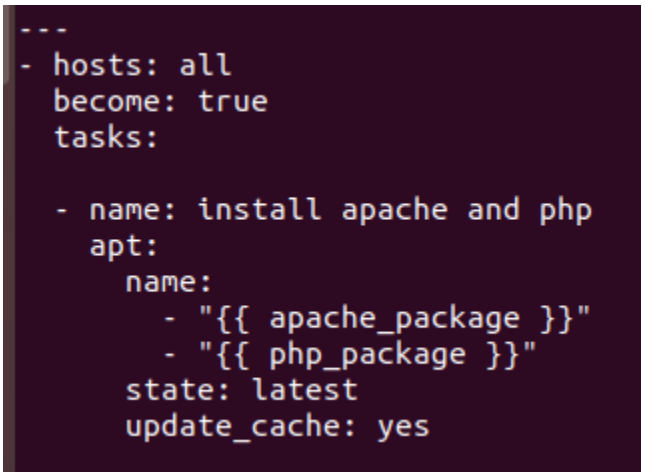
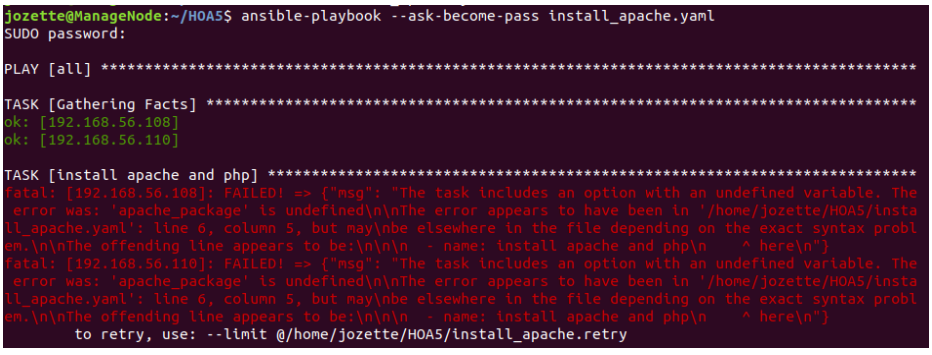
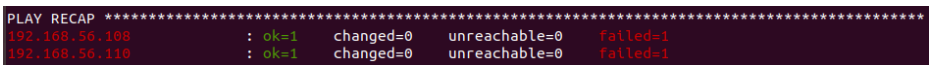
OUTPUT

```
PLAY RECAP *********************************************************************
192.168.56.108             : ok=2    changed=0    unreachable=0    failed=0
192.168.56.110             : ok=1    changed=0    unreachable=0    failed=0
```

3. Finally, we can consolidate these 2 plays in just 1 play. This can be done by declaring variables that will represent the packages that we want to install. Basically, the apache_package and php_package are variables. The names are arbitrary, which means we can choose different names. We also take out the line when: ansible_distribution. Edit the playbook *install_apache.yml* again and make sure to follow the below image. Make sure to save the file and exit.

```
---
- hosts: all
  become: true
  tasks:

  - name: install apache and php
    apt:
      name:
        - "{{ apache_package }}"
        - "{{ php_package }}"
      state: latest
      update_cache: yes
```

Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT	
PROCESS	
OUTPUT	

4. Unfortunately, task 2.3 was not successful. It's because we need to change something in the inventory file so that the variables we declared will be in place. Edit the *inventory* file and follow the below configuration:

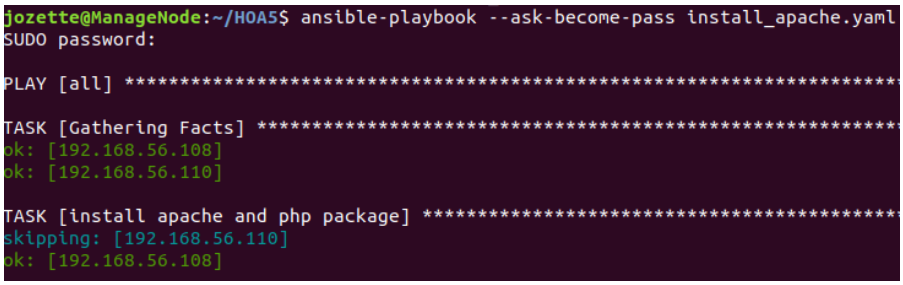
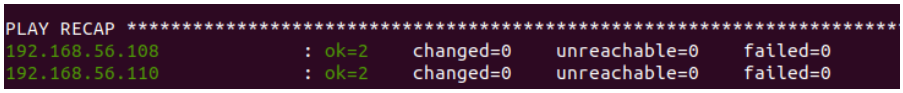
```
192.168.56.120 apache_package=apache2 php_package=libapache2-mod-php
192.168.56.121 apache_package=apache2 php_package=libapache2-mod-php
192.168.56.122 apache_package=httpd php_package=php
```

Make sure to save the *inventory* file and exit.

Finally, we still have one more thing to change in our *install_apache.yml* file. In task 2.3, you may notice that the package is assign as apt, which will not run in CentOS. Replace the *apt* with *package*. Package is a module in ansible that is generic, which is going to use whatever package manager the underlying host or the target server uses. For Ubuntu it will automatically use *apt*, and for CentOS it will automatically use *dnf*. Make sure to save the file and exit. For more details about the ansible package, you may refer to this documentation: ansible.builtin.package – Generic OS package manager — Ansible Documentation

Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT	```
	--- 	
	- hosts: all
	  become: true
	  tasks:
	
	  - name: install apache and php package
	  package:
	  name:
	  - "{{ apache_package }}"
	  - "{{ php_package }}"
	  state: latest
	  update_cache: yes
	  when: ansible_distribution in ["Ubuntu","Debian"]
	``

PROCESS	
OUTPUT	

Supplementary Activity:

1. Create a playbook that could do the previous tasks in Red Hat OS.

Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?

   The refactoring of playbook codes is important because it makes the code more readable and easier to understand. It also helps in improving the script and removing redundant, unused codes. Furthermore, by refactoring the script without changing the external behavior the debugging are much easier.

2. When do we use the “when” command in playbook?

   The when command in playbook acts as an if-else statement. wherein it is used to execute on a specific conditions. This can be useful when you want to execute a task only on specific hosts or when you want to execute a task only if a previous task has failed

Conclusions:

In this hands-on activity, we learned how to use and when do we use the when command in playbook for different OS distributions and how to apply refactoring techniques in cleaning up the playbook codes. I also learned about the importance of refactoring of playbook codes. I learned that by doing refactoring in a script it makes the code more readable and easier to understand and also improves the script and removing redundant, unused codes. Overall, doing this activity helps me a lot in understanding the consolidating of playbooks.