Name: Buenvenida, Ken Benedict	Date Performed: 09/18/2023
Course/Section: CpE232 - CpE31S4	Date Submitted: 09/18/2023
Instructor: Engr. Jonathan Taylar	Semester and SY: 1st Semester 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.

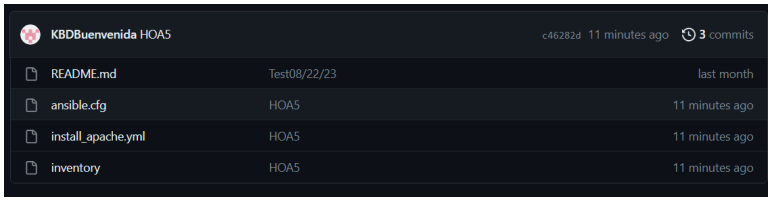
It is a best practice in ansible when you are working in a collaborative environment to use the command git pull. git pull is a Git command used to update the local version of a repository from a remote. By default, git pull does two things. Updates the current local working branch (currently checked 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

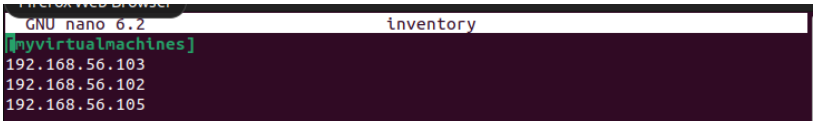
1. In the local machine, make sure you are in the local repository directory (*CPE232_yourname*). Issue the command git pull. When prompted, enter the correct passphrase or password. Describe what happened when you issue this command. Did something happen? Why?

INPUT	
OUTPUT	

Observation:

The moment we used the git pull command it prompted me that it is already up to date. If we changed or created a new file that uses the same repository, I can easily transfer the changes in the repository to my computer. The git pull command copies or fetches the changes from the repository to my local machine.

2. 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."

INPUT	
PROCESS	

OUTPUT

```
ken@manageNode:~/ansible$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update repository index] *************************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] *************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

PLAY RECAP *********************************************************************
192.168.56.102             : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.103             : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

ken@manageNode:~/ansible$
```

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

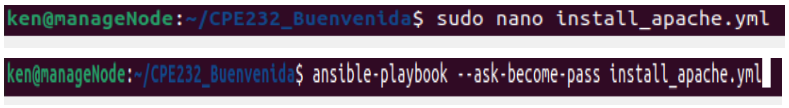
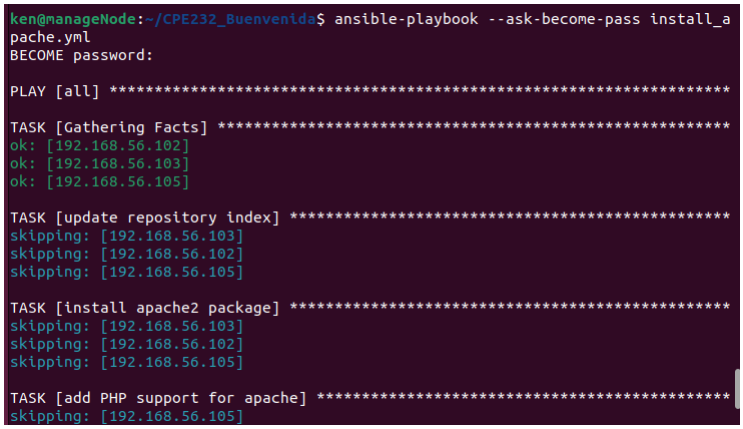
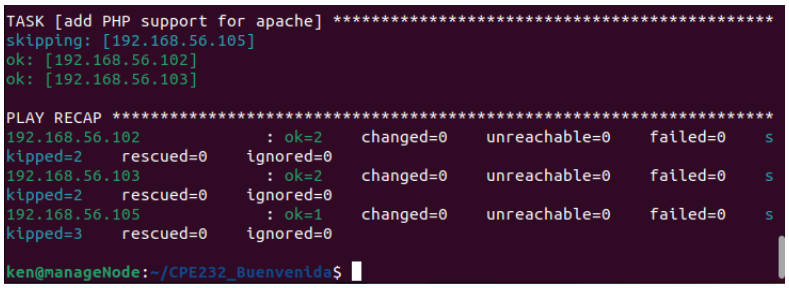
```
---
- hosts: all
  become: true
  tasks:

  - 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.

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	 

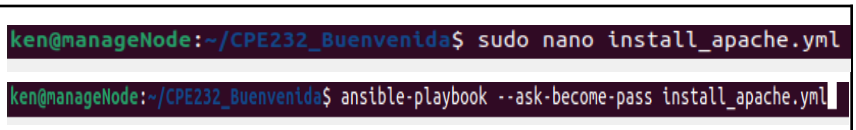
Note: This will work also if you try. Notice the changes are highlighted.

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
      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	
PROCESS	


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

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == ["Debian","Ubuntu"]

  - name: install apache2 package
    apt:
      name: apache2
    when: ansible_distribution == ["Debian","Ubuntu"]

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
    when: ansible_distribution in ["Debian","Ubuntu"]

  - name: update respository index
    yum:
      update_cache: yes
    when: ansible_distribution == "CentOS"

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

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

OUTPUT

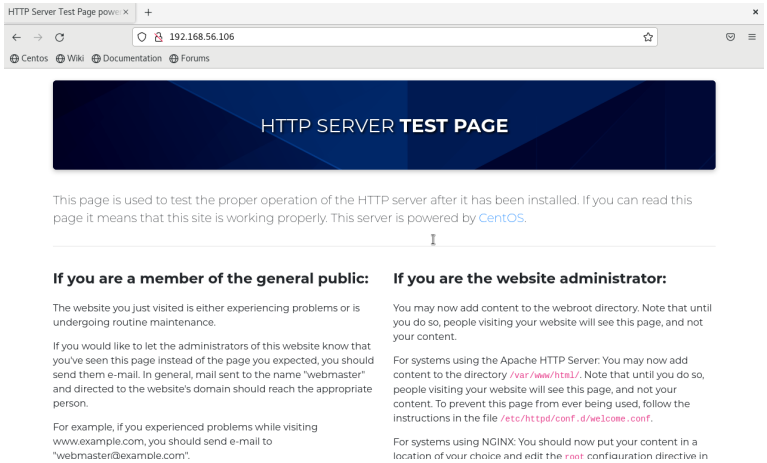
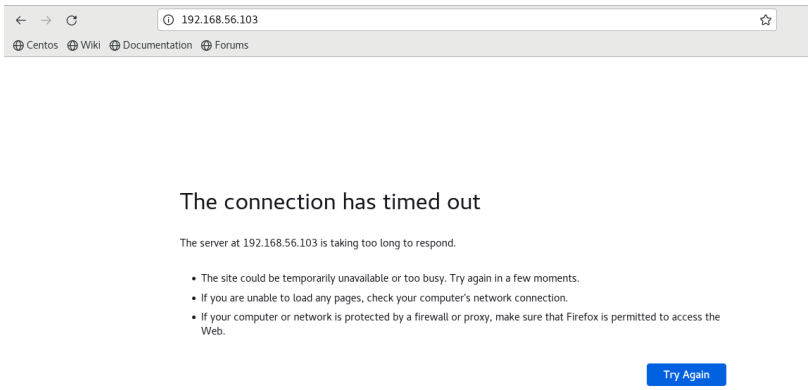
```
ken@manageNode:~/CPE232_Buenvenida$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]
ok: [192.168.56.106]

TASK [update repository index] *************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
skipping: [192.168.56.106]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
skipping: [192.168.56.106]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.106]
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update respository index] ************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
ok: [192.168.56.106]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
changed: [192.168.56.106]
```

```
TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
changed: [192.168.56.106]

PLAY RECAP *********************************************************************
192.168.56.102             : ok=2    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
192.168.56.103             : ok=2    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
192.168.56.106             : ok=4    changed=2    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```

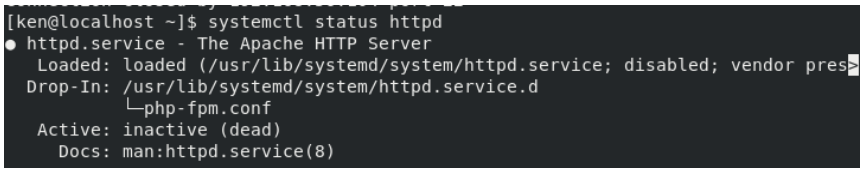
Observation:

I was able to update the repository for all the servers, I was also able to install the apache2 package for all the servers and add PHP support for all the servers. The reason why Ubuntu is skipping the server of CentOS means that the playbook is running smoothly.

5. 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.

INPUT	
OUTPUT	

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.

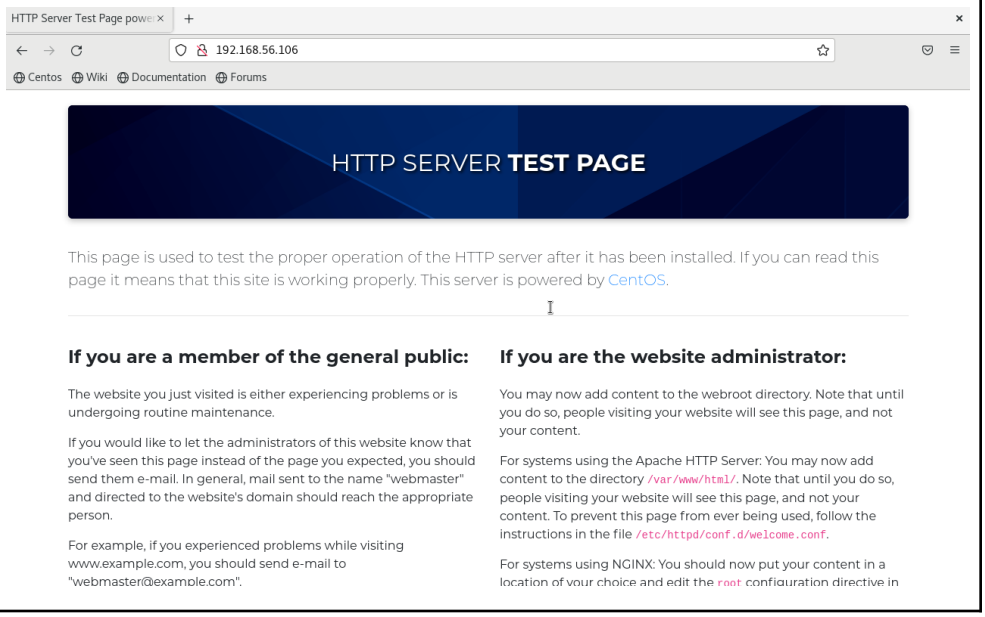
INPUT & OUTPUT	

5.2 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)

INPUT & OUTPUT	`[ken@localhost ~]$ sudo systemctl start httpd`
	`[sudo] password for ken:`
	`[ken@localhost ~]$ sudo firewall-cmd --add-port=80/tcp`
	`success`

5.3 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)

INPUT & OUTPUT



Observation:

It was successful. The moment we activated the httpd, We were able to see the server test page which means that the http service is active.

Task 2: Refactoring playbook

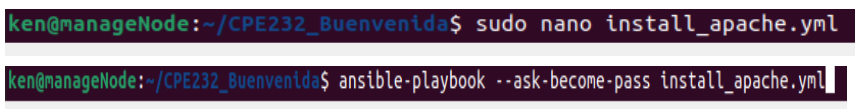
This time, we want to make sure that our playbook is efficient and that the codes are easier to read. This will also makes run ansible more quickly if it has to execute fewer tasks to do the same thing.

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.

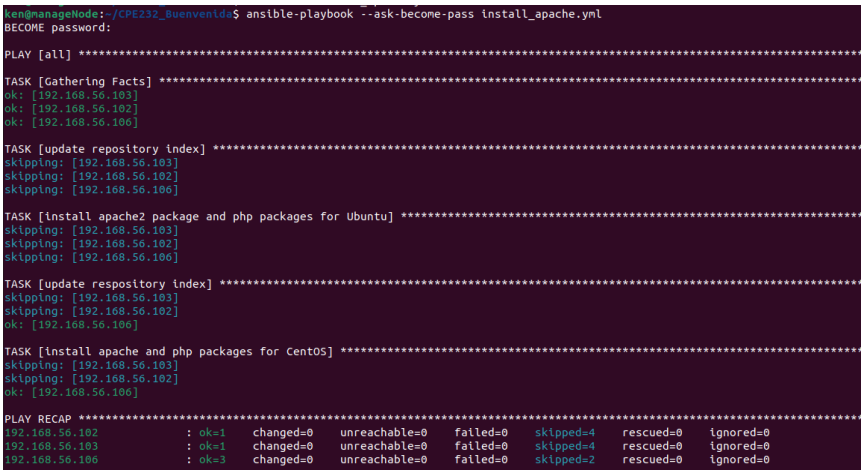
Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

INPUT	

PROCESS

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

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == ["Debian","Ubuntu"]

  - name: install apache2 package and php packages for Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
    when: ansible_distribution == ["Debian","Ubuntu"]

  - name: update respository index
    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"
```

OUTPUT



Observation:

It still worked even though the codes are much shorter and simpler. There were no issues that the playbook ran into even though we modified it a lot by merging the apache and php package.

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

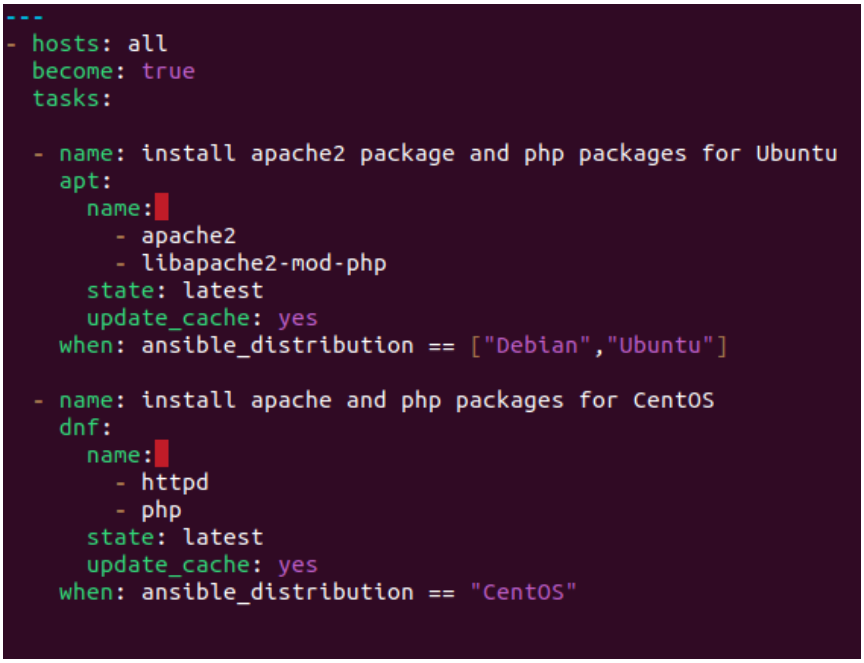
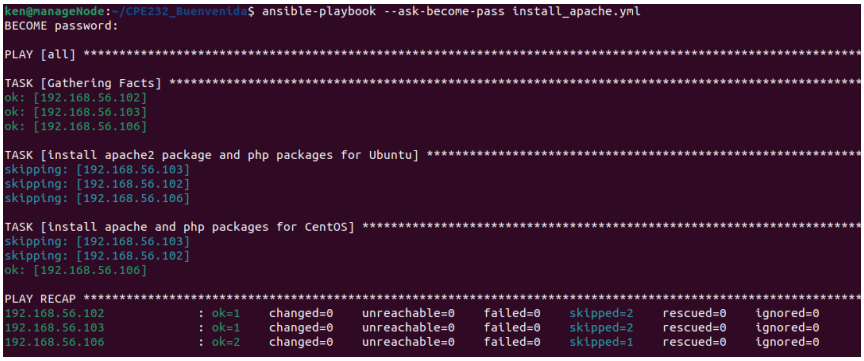
command *update_cache: yes* below the command *state: latest*. See below for reference:

```
---
- 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	`ken@manageNode:~/CPE232_Buenvenida$ sudo nano install_apache.yml` `ken@manageNode:~/CPE232_Buenvenida$ ansible-playbook --ask-become-pass install_apache.yml`

PROCESS	
OUTPUT	

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

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

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

```
ken@manageNode:~/CPE232_Buenvenida$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]
ok: [192.168.56.106]

TASK [install apache2 package and php packages for Ubuntu] *********************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
skipping: [192.168.56.106]

TASK [install apache and php packages for CentOS] ******************************************************************
skipping: [192.168.56.103]
skipping: [192.168.56.102]
ok: [192.168.56.106]

PLAY RECAP *********************************************************************************************************
192.168.56.102             : ok=1    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.103             : ok=1    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

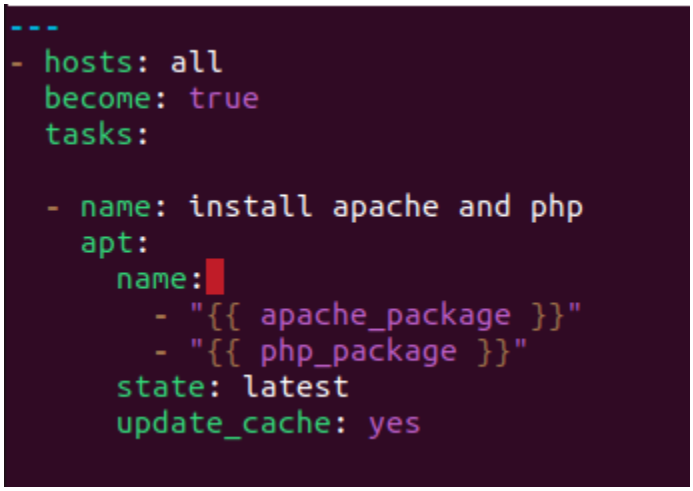
Observation:

The play still worked even though I changed and configured the playbook only into two parts where I separated the packages for Ubuntu and CentOS.

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	`ken@manageNode:~/CPE232_Buenvenida$ sudo nano install_apache.yml` `ken@manageNode:~/CPE232_Buenvenida$ ansible-playbook --ask-become-pass install_apache.yml`
PROCESS	

OUTPUT

```
ken@manageNode:~/CPE232_Buenvenida$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]
ok: [192.168.56.106]

TASK [install apache and php] **************************************************************************************************************************
fatal: [192.168.56.103]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package'
 is undefined\n\nThe error appears to be in '/home/ken/CPE232_Buenvenida/install_apache.yml': line 6, column 5, but may\nbe elsewher
e in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n  - name: install apache and php\n
 ^ here\n"}
fatal: [192.168.56.102]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package'
 is undefined\n\nThe error appears to be in '/home/ken/CPE232_Buenvenida/install_apache.yml': line 6, column 5, but may\nbe elsewher
e in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n  - name: install apache and php\n
 ^ here\n"}
fatal: [192.168.56.106]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package'
 is undefined\n\nThe error appears to be in '/home/ken/CPE232_Buenvenida/install_apache.yml': line 6, column 5, but may\nbe elsewher
e in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n  - name: install apache and php\n
 ^ here\n"}

PLAY RECAP *********************************************************************************************************************************************
192.168.56.102             : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.56.103             : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.56.106             : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Observation:

Based on the output, The play did not work because there an unidentified element in the playbook.yaml where the playbook can't identify what type of package is needed.

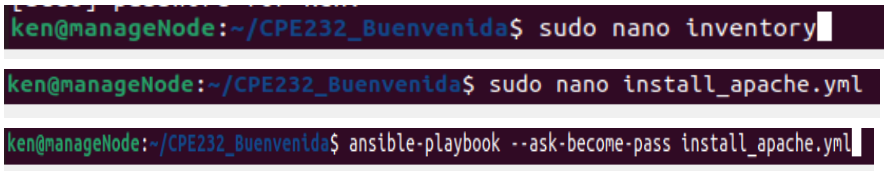
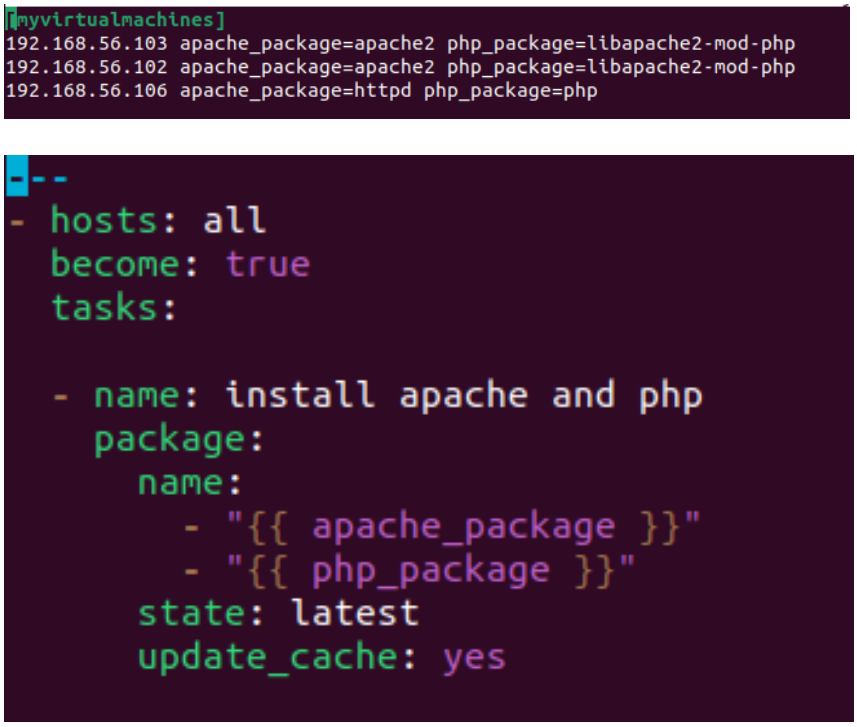
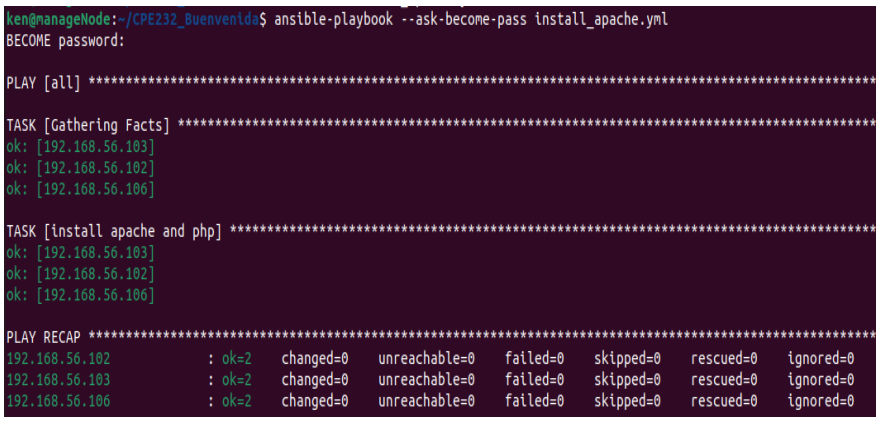
4. Unfortunately, task 2.3 was not successful. It's because we need to change something in the inventory file so that the variables we declared will be in place. Edit the *inventory* file and follow the below configuration:

```
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	
PROCESS	
OUTPUT	

Observation:

We can see that the playbook worked properly due to the changes that we did. We replaced the ‘apt:’ with ‘package:’. This worked due to the fact that package

is an ansible module where it'll automatically adjust for the proper module such as dnf or apt for the respective OS depending if we're using Ubuntu or CentOS.

Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?
    - **It is very important as it shortens and improves the codes by reducing the complexity of the playbook, it makes them more modular and reusable. It can make the code easier to understand and debug as it is shorter. It also improves the performance of the playbook by eliminating the repetition in the codes which can make the playbook faster.**

2. When do we use the "when" command in playbook?
    - **We use the 'when' command when we are needed to conditionally execute multiple tasks, but the task will only be executed if the conditions are met in the 'when' statement. The 'when' command is so dynamic and flexible that it can make the playbook powerful when scripting.**