

<b>Name:</b> John Renzo L. Mendoza	<b>Date Performed:</b> September 22, 2023
<b>Course/Section:</b> CPE31S5	<b>Date Submitted:</b> September 23, 2023
<b>Instructor:</b> Engr. Roman Richard	<b>Semester and SY:</b> 1st Semester (23 - 24)

### 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 installation. 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

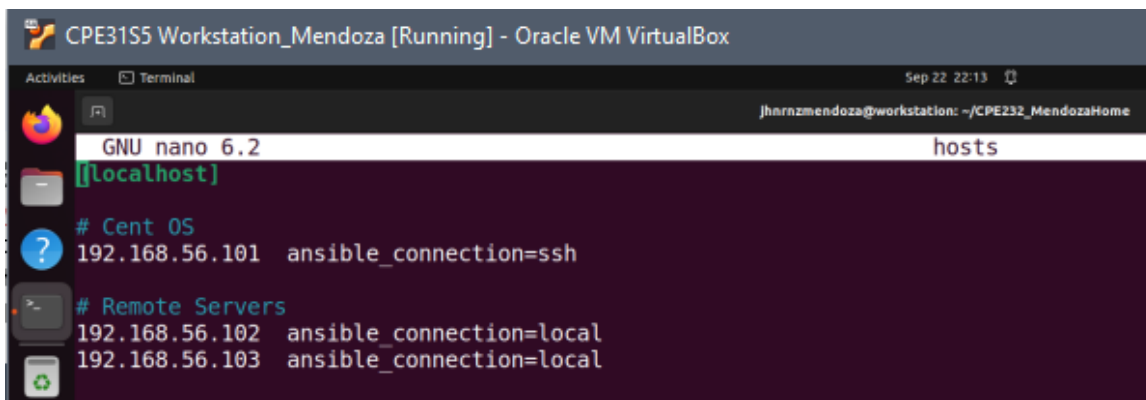
1. In the local machine, make sure you are in the local repository directory (*CPE232\_yourname*). Issue the command `git pull`. When prompted, enter the correct passphrase or password. Describe what happens when you issue this command. Did something happen? Why?

```
jhnrmendoza@workstation:~/CPE232_MendozaHome$ git pull
Already up to date.
```

### Observation:

The command output that the local repository is "Already up to date". I believe this is because the `git pull` command is responsible for getting the progress on the cloud repository and then importing it on the 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."



```
CPE31S5 Workstation_Mendoza [Running] - Oracle VM VirtualBox
Activities Terminal Sep 22 22:13 jhnrmendoza@workstation: ~/CPE232_MendozaHome
GNU nano 6.2 hosts
[[localhost]]
# Cent OS
192.168.56.101 ansible_connection=ssh
# Remote Servers
192.168.56.102 ansible_connection=local
192.168.56.103 ansible_connection=local
```

```

jhrnzendoza@workstation:~/CPE232_MendozaHome$ 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: [192.168.56.101]

TASK [update repository index] *****
[WARNING]: Updating cache and auto-installing missing dependency: python-apt
fatal: [192.168.56.101]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory", "rc":
2, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
changed: [192.168.56.103]
changed: [192.168.56.102]

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

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

PLAY RECAP *****
192.168.56.101      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.56.102      : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

### Observation:

The ip address which is for the CentOS virtual machine does not perform the tasks prompted by the playbook as the package manager for CentOS is yum or dnf while the stated package manager in the playbook is apt which is for Debian distributions.

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"

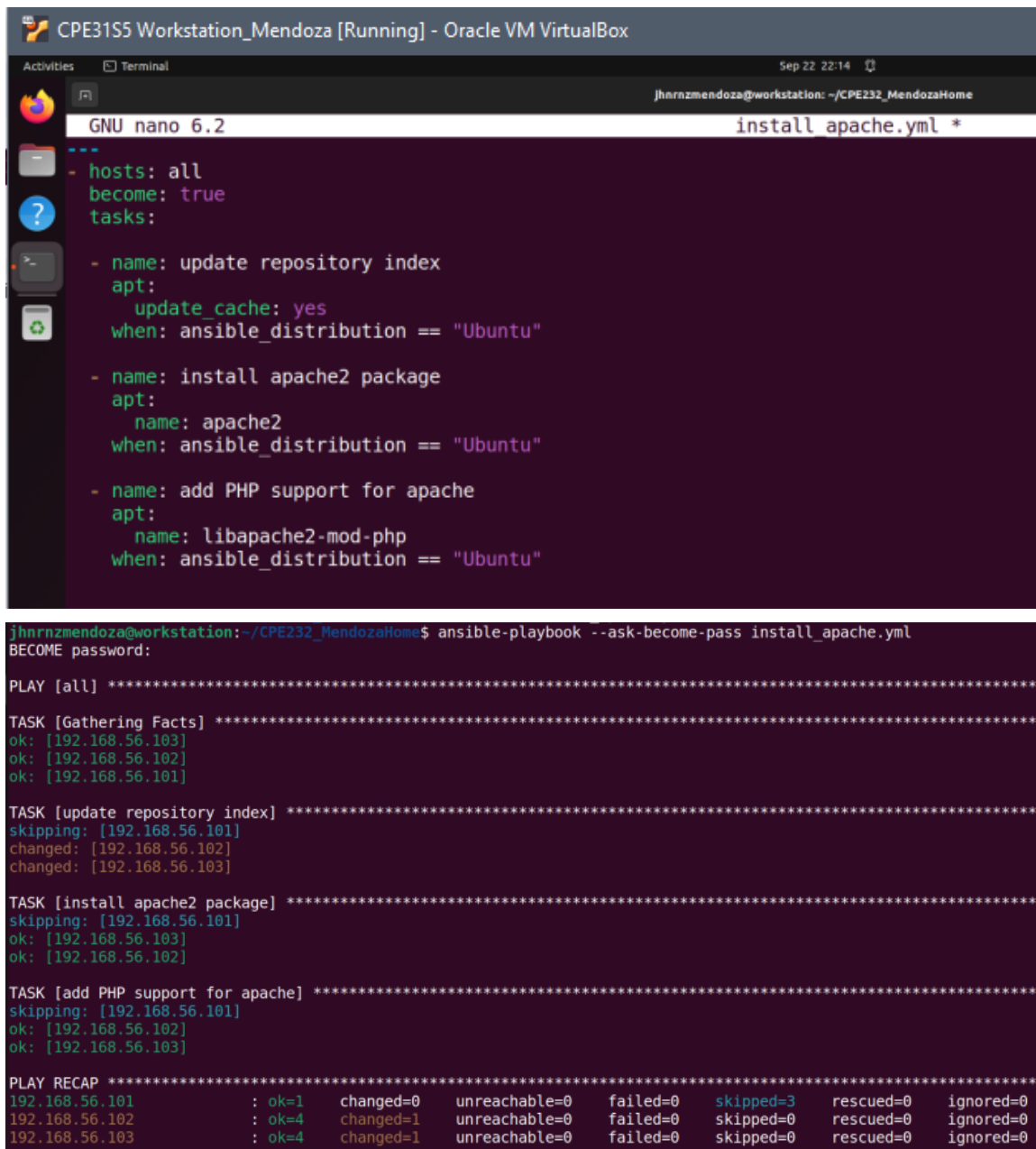
    - name: install apache2 package
      apt:
        name: apache2
        when: ansible_distribution == "Ubuntu"

    - 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.



```
CPE31S5 Workstation_Mendoza [Running] - Oracle VM VirtualBox
Activities Terminal Sep 22, 22:14 jhnrnmendoza@workstation: ~/CPE232_MendozaHome
GNU nano 6.2 install_apache.yml *
---
- 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"

jhnrnmendoza@workstation:~/CPE232_MendozaHome$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.102]
ok: [192.168.56.101]

TASK [update repository index] *****
skipping: [192.168.56.101]
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] *****
skipping: [192.168.56.101]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
skipping: [192.168.56.101]
ok: [192.168.56.102]
ok: [192.168.56.103]

PLAY RECAP *****
192.168.56.101      : ok=1    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.102      : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

### Observation:

The CentOS virtual machine did not result in a failure but instead, it was able to skip the tasks on the playbook since we added constraints wherein the tasks can only be performed by the hosts that have "Ubuntu" as their distribution.

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"]

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  
        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 == "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.

```
jhnrmendoza@workstation:~/CPE232_MendozaHome$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.102]
ok: [192.168.56.101]

TASK [update repository index] *****
skipping: [192.168.56.101]
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] *****
skipping: [192.168.56.101]
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [add PHP support for apache] *****
skipping: [192.168.56.101]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [update repository index] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [192.168.56.101]

TASK [install apache2 package] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
changed: [192.168.56.101]

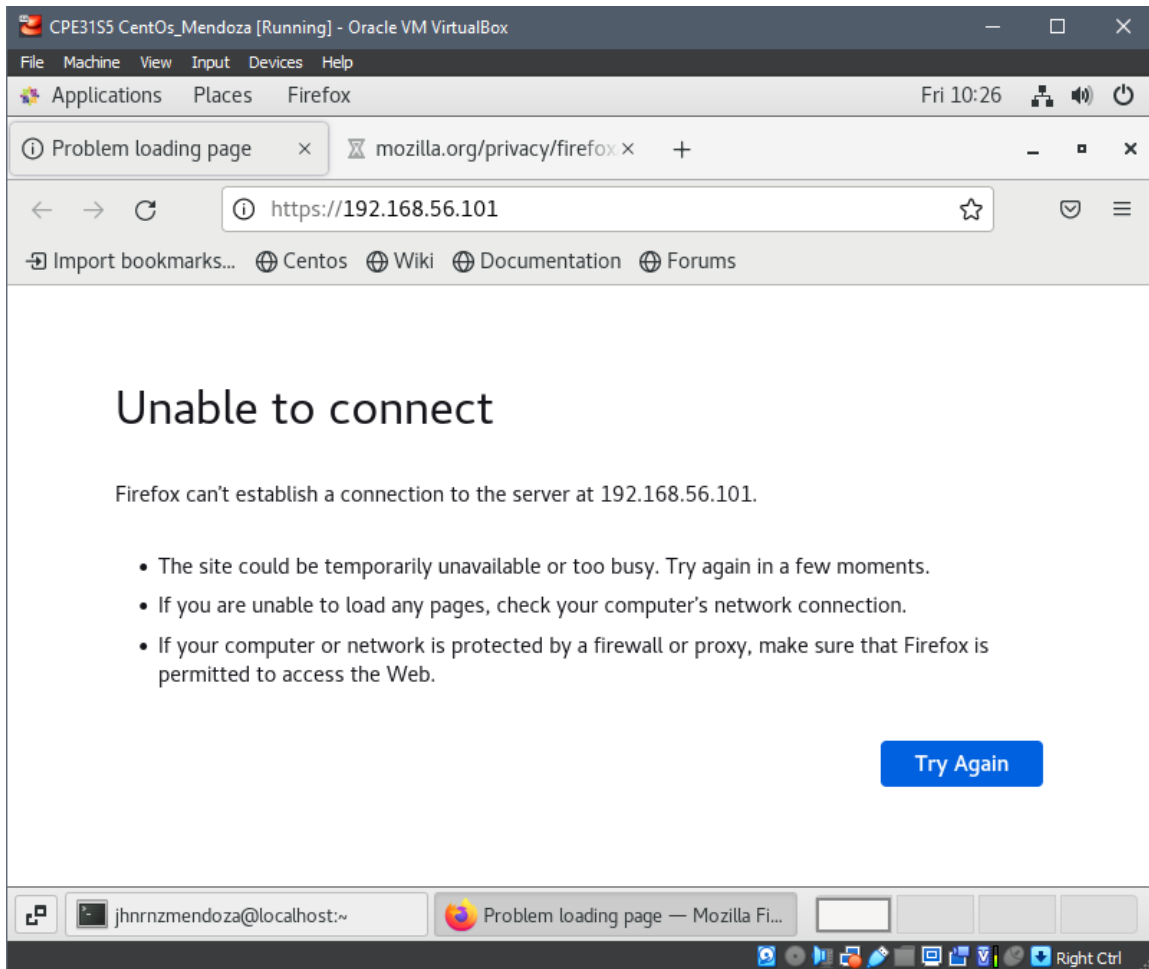
TASK [add PHP support for apache] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
changed: [192.168.56.101]

PLAY RECAP *****
192.168.56.101      : ok=4    changed=2    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.102      : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.103      : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```

### Observation:

Instead of using “dnf” as the package manager, I have used the “yum” since there is an error with dnf to my installed CentOS image file. Since “yum” package manager is still a CentOS based package manager, It still makes the distinction between Debian and RedHat virtual machines.

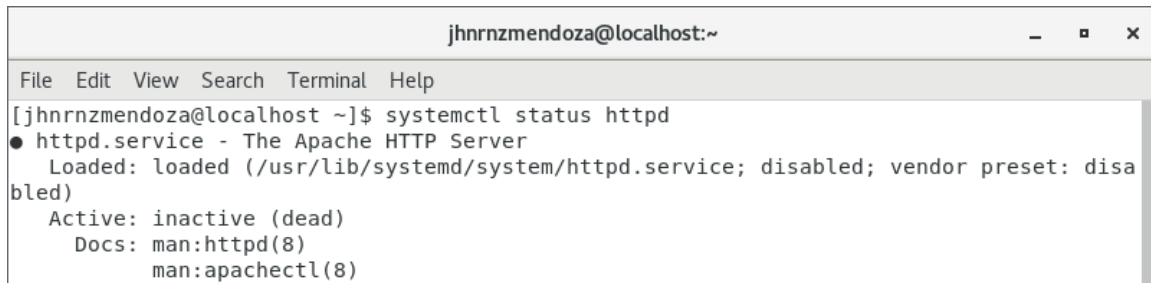
5. To verify the installations, go to CentOS VM and type its IP address on the browser. Was it successful? The answer is no. It's because the httpd service or the Apache HTTP server in CentOS is not yet active. Thus, you need to activate it first.



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.

A terminal window titled 'jhnrnmendoza@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[jhnrnmendoza@localhost ~]\$ systemctl status httpd' has been executed. The output shows '● httpd.service - The Apache HTTP Server', 'Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)', 'Active: inactive (dead)', and 'Docs: man:httpd(8), man:apachectl(8)'.

```
jhnrnmendoza@localhost:~
File Edit View Search Terminal Help
[jhnrnmendoza@localhost ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)
```

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)

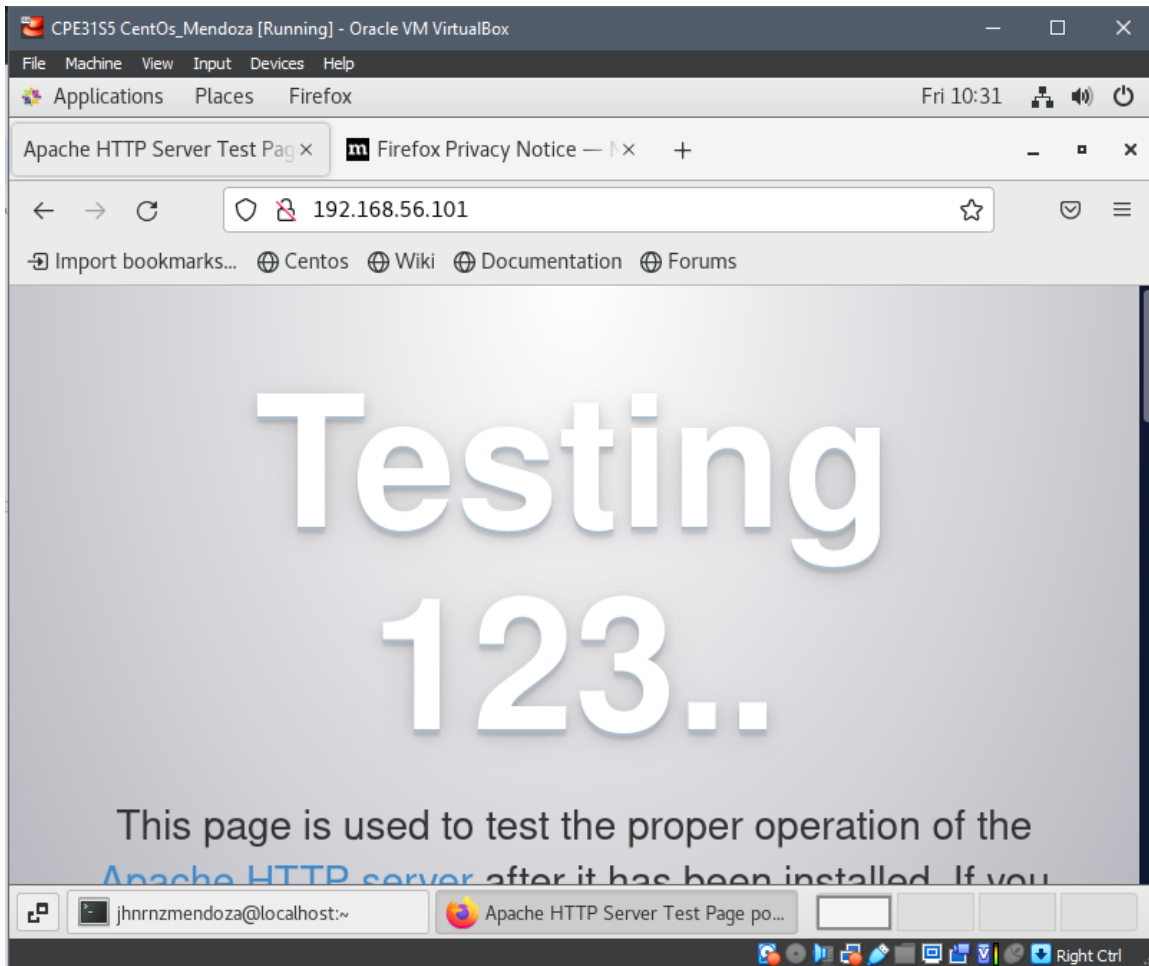
```
[jhnrnmendoza@localhost ~]$ sudo systemctl start httpd
[jhnrnmendoza@localhost ~]$ sudo firewall-cmd --add-port=80/tcp
success
```



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)

```
[jhnrmendoza@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 Fri 2023-09-22 10:29:23 EDT; 1min 18s ago
     Docs: man:httpd(8)
           man:apachectl(8)
    Main PID: 8487 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:  0 B/sec"
     Tasks: 6
    CGroup: /system.slice/httpd.service
            └─8487 /usr/sbin/httpd -DFOREGROUND
              └─8491 /usr/sbin/httpd -DFOREGROUND
                └─8492 /usr/sbin/httpd -DFOREGROUND
                  └─8493 /usr/sbin/httpd -DFOREGROUND
                    └─8494 /usr/sbin/httpd -DFOREGROUND
                      └─8495 /usr/sbin/httpd -DFOREGROUND

Sep 22 10:29:23 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
Sep 22 10:29:23 localhost.localdomain httpd[8487]: AH00558: httpd: Could not reliab...e
Sep 22 10:29:23 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
```



## 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 make 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.

CPE31S5 Workstation\_Mendoza [Running] - Oracle VM VirtualBox

Activities Terminal Sep 22 22:34 jhernzmendoza@workstation: ~/CPE232\_Mendoza

GNU nano 6.2 install apache.yml

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

    # For Ubuntu and Debian

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

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

    # For CentOS
    - name: update repository index
      yum:
        update_cache: yes
        when: ansible_distribution == "CentOS"

    - name: install apache2 package and php packages for CentOS
      yum:
        name:
          - httpd
          - php
        state: latest
        when: ansible_distribution == "CentOS"
```

Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.

```
jhnrmendoza@workstation:~/CPE232_MendozaHome$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.101]
ok: [192.168.56.102]

TASK [update repository index] *****
skipping: [192.168.56.101]
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package and php packaged for Ubuntu] *****
skipping: [192.168.56.101]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [update repository index] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [192.168.56.101]

TASK [install apache2 package and php packages for CentOS] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [192.168.56.101]

PLAY RECAP *****
192.168.56.101      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.102      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.103      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

### Observation:

Upon changing the playbook, I have observed that the lines of code after executing the ansible playbook was shorter than the previous version. I believe this is because we have combined the installation of apache and adding php packages as one task. Therefore, instead of having 6 tasks, we only had 4 tasks on the playbook.

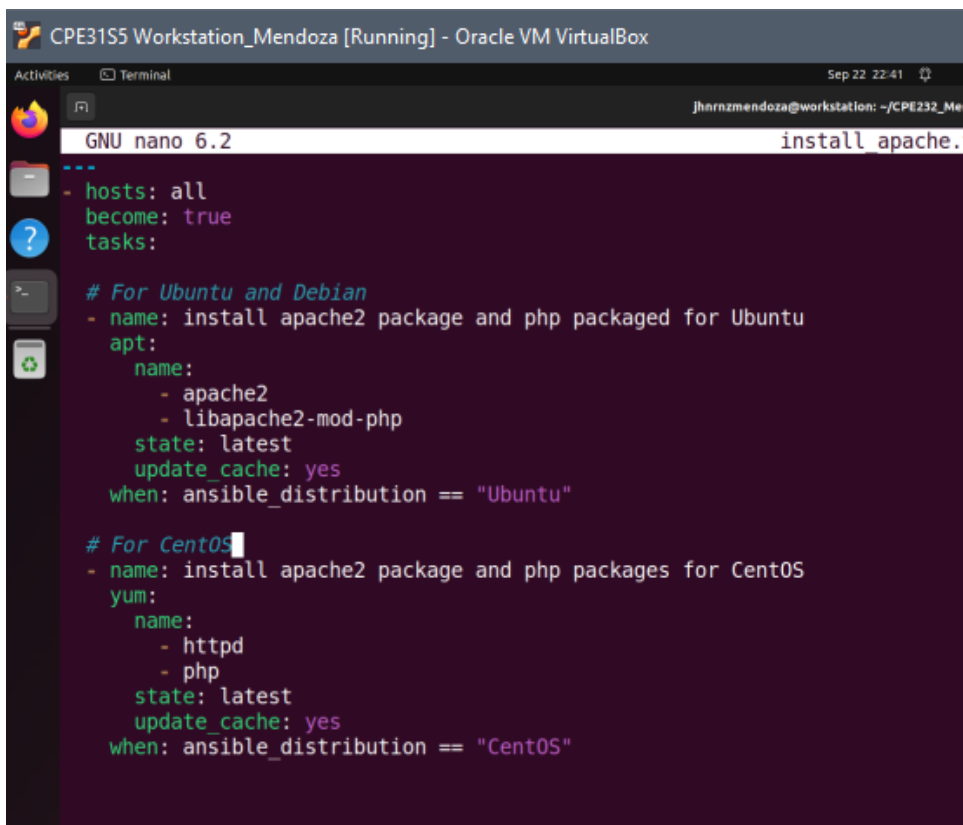
2. Edit the playbook `install_apache.yml` again. In task 2.1, we consolidated the plays into one play. This time we can actually consolidate everything in just 2 plays. This can be done by removing the update repository play and putting the command `update_cache: yes` below the command `state: latest`.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



```
CPE3155 Workstation_Mendoza [Running] - Oracle VM VirtualBox
Sep 22 22:41
jhnrmendoza@workstation: ~/CPE232_Men
GNU nano 6.2 install_apache.
---
- hosts: all
  become: true
  tasks:

    # For Ubuntu and Debian
    - name: install apache2 package and php packaged for Ubuntu
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        update_cache: yes
      when: ansible_distribution == "Ubuntu"

    # For CentOS
    - name: install apache2 package and php packages for CentOS
      yum:
        name:
          - httpd
          - php
        state: latest
        update_cache: yes
      when: ansible_distribution == "CentOS"
```

Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.

```
jhnrmendoza@workstation:~/CPE232_MendozaHome$ sudo nano install_apache.yml
jhnrmendoza@workstation:~/CPE232_MendozaHome$ 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: [192.168.56.101]

TASK [install apache2 package and php packaged for Ubuntu] *****
skipping: [192.168.56.101]
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [install apache2 package and php packages for CentOS] *****
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [192.168.56.101]

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

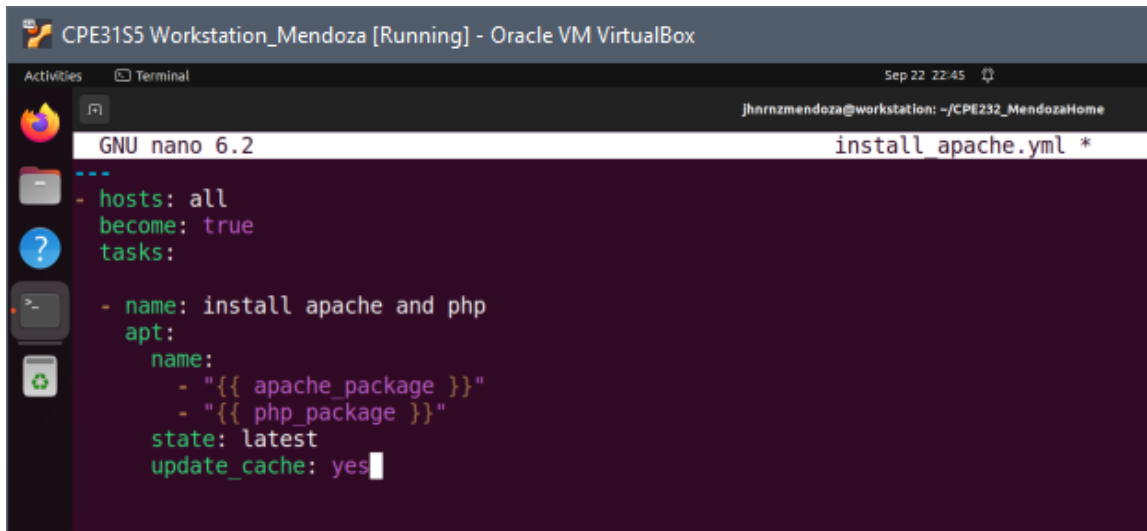
### Observation:

Similarly, the output on this version of the playbook was even shorter and simplified as all of the tasks are combined as one. Therefore, the installation of apache, adding php packages, and updating the cache was a single task. Instead of having 4 tasks (from the previous version) we only had 2 tasks on the playbook.

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
```



The screenshot shows a terminal window titled "CPE3155 Workstation\_Mendoza [Running] - Oracle VM VirtualBox". The terminal is running GNU nano 6.2, editing the file `install_apache.yml`. The content of the file is as follows:

```
--
- 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.

```
jhrnzmendoza@workstation:~/CPE232_MendozaHome$ sudo nano install_apache.yml
jhrnzmendoza@workstation:~/CPE232_MendozaHome$ 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: [192.168.56.101]

TASK [install apache and php] *****
fatal: [192.168.56.101]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined. 'apache_package' is undefined\n\nThe error appears to be in '/home/jhrnzmendoza/CPE232_MendozaHome/install_apache.yml': line 6, column 5, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\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. 'apache_package' is undefined\n\nThe error appears to be in '/home/jhrnzmendoza/CPE232_MendozaHome/install_apache.yml': line 6, column 5, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n- name: install apache and php\n  ^ here\n"}
fatal: [192.168.56.103]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined. 'apache_package' is undefined\n\nThe error appears to be in '/home/jhrnzmendoza/CPE232_MendozaHome/install_apache.yml': line 6, column 5, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n- name: install apache and php\n  ^ here\n"}

PLAY RECAP *****
192.168.56.101      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
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
```

### Observation:

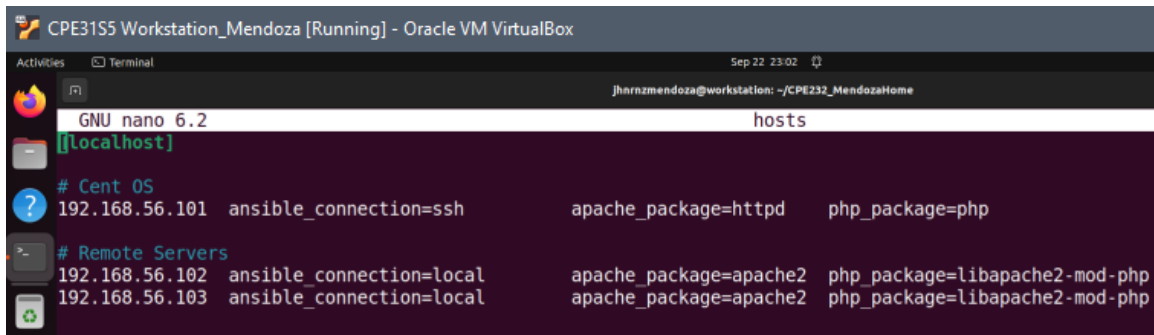
The playbook was not successful since we added arbitrary variables on the task within the playbook but these variables `apache_package` and `php_package` were not yet initialized. Therefore, the playbook would not know what to execute, hence why the process has failed. To fix this problem, we may declare the variables with the appropriate packages first.



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
```

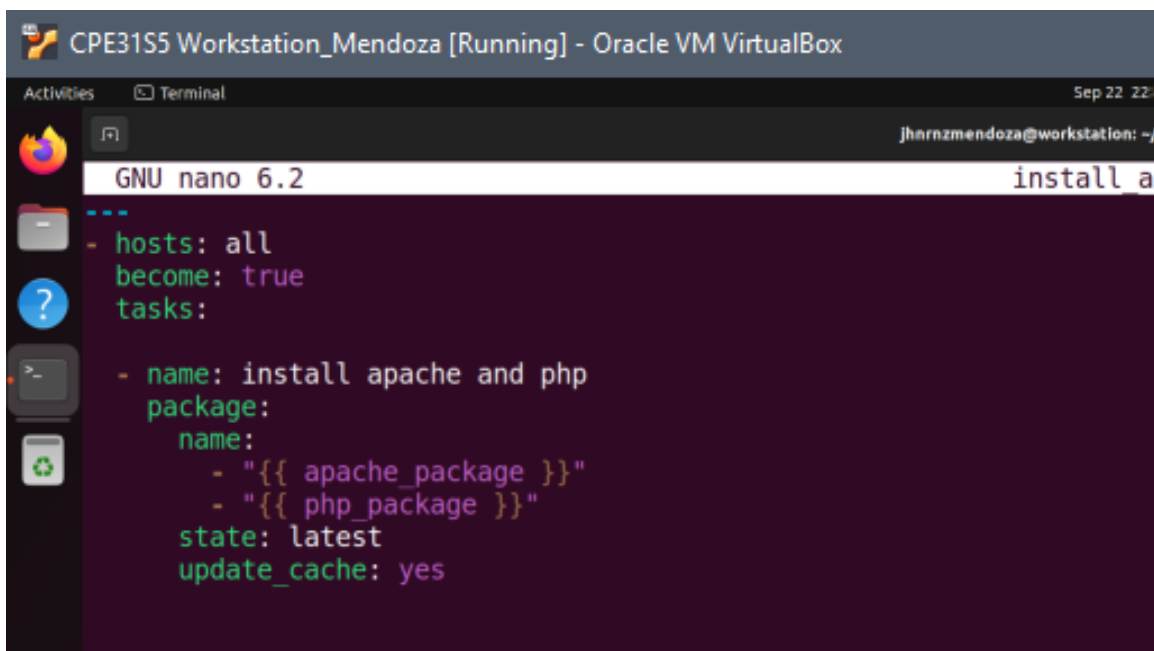
Make sure to save the *inventory* file and exit.



The screenshot shows a terminal window titled "CPE31S5 Workstation\_Mendoza [Running] - Oracle VM VirtualBox". The terminal is running GNU nano 6.2 and editing the /etc/ansible/hosts file. The content of the file is as follows:

```
hosts
# Cent OS
192.168.56.101 ansible_connection=ssh      apache_package=httpd      php_package=php
# Remote Servers
192.168.56.102 ansible_connection=local    apache_package=apache2    php_package=libapache2-mod-php
192.168.56.103 ansible_connection=local    apache_package=apache2    php_package=libapache2-mod-php
```

**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 assigned as *apt*, which will not run in CentOS. Replace the *apt* with the *package* module. *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](https://docs.ansible.com/ansible/latest/modules/package_module.html)



The screenshot shows a terminal window titled "CPE31S5 Workstation\_Mendoza [Running] - Oracle VM VirtualBox". The terminal is running GNU nano 6.2 and editing the /etc/ansible/playbooks/install\_apache.yml file. The content of the file is as follows:

```
---
- hosts: all
  become: true
  tasks:
    - name: install apache and php
      package:
        name:
          - "{{ apache_package }}"
          - "{{ php_package }}"
        state: latest
        update_cache: yes
```

Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.

```
jhnrmendoza@workstation:~/CPE232_MendozaHome$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.102]
ok: [192.168.56.101]

TASK [install apache and php] *****
ok: [192.168.56.103]
ok: [192.168.56.102]
ok: [192.168.56.101]

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

### Observation:

Since we had made the playbook generic and the needed variables were initialized, the playbook was able to execute the tasks. We can also observe that the output was simplified once again into only one play.

**Supplementary Activity:**

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

Step 1: Installing Virtual Machine with Red Hat OS.

Step 2: Connecting the Red Hat OS Virtual Machine to the Local Machine through SSH.

Step 3: Creating a Playbook similar to the Experiment's procedure in a new file.

Step 4: Execute the Playbook.

**Reflections:**

Answer the following:

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

The system administrator should refactor the playbook codes so that the playbook would be much more efficient and simplified in a way that it won't be redundant in terms of the commands. Refactoring can also help in simplifying the output of the playbook. Instead of looking at many plays of the playbook, we could only stick to a few plays that are concise and detailed in terms of the task it achieved, similar to what we have done in this activity.

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

The command when is a condition statement, in which the playbook will test if the virtual machine or a remote host does have the description declared by the when command. In this activity, we have observed this when we are configuring the remote CentOS and remote Ubuntu-based Server. Since CentOS and Ubuntu differ in terms of their package management tool, it is necessary to use a when statement so that the task would only perform itself to the virtual machines who satisfy the conditions. This is useful when we have multiple hosts to configure and we have grouped them accordingly in terms of their distribution, or other characteristics.