

Name: LEE IVAN TITONG	Date Performed: 9-12-2025
Course/Section: CPE 212-CPE31S2 - Automating Server Management	Date Submitted: 9-12-2025
Instructor: Engr. Robin Valenzuela	Semester and SY: 2025-2026

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:

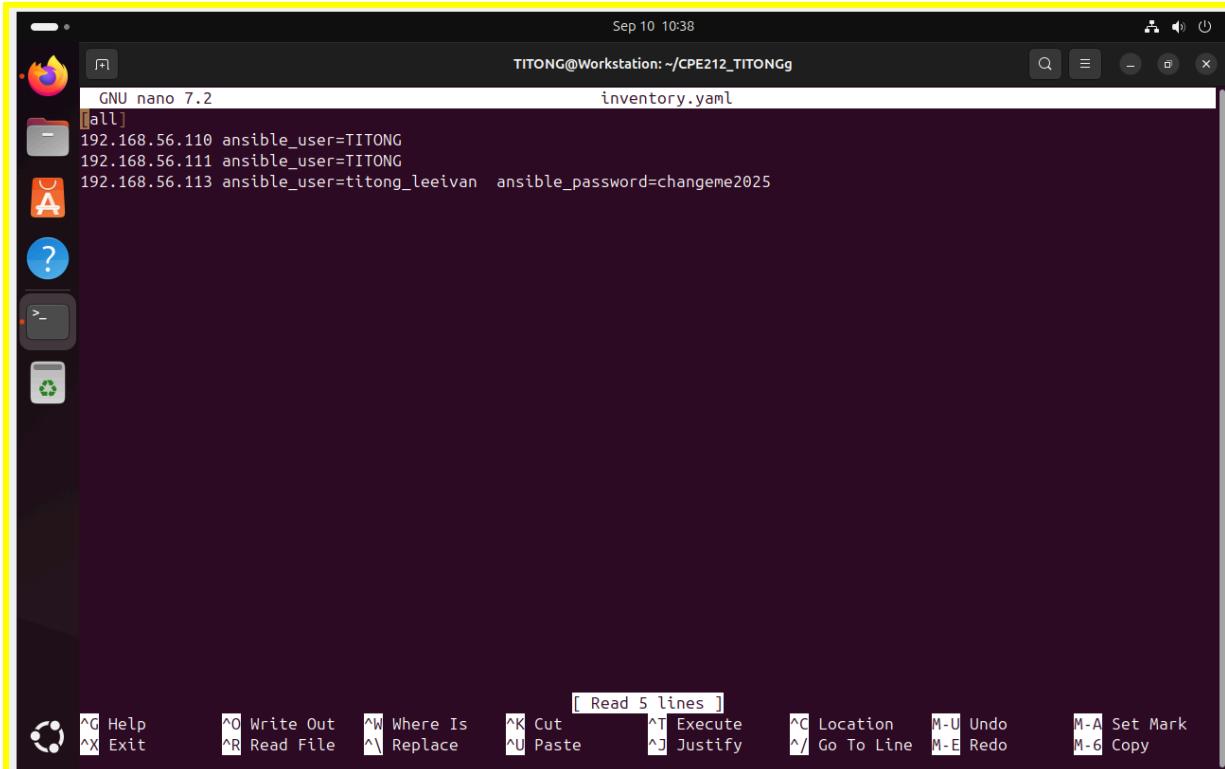
In this activity, you will need to create a CentOS VM. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the CentOS VM. Make sure to use the command **ssh-copy-id** to copy the public key to CentOS. Verify if you can successfully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

```
TITONG@Workstation:~/CPE212_TITONGg$ git pull  
Already up to date.  
TITONG@Workstation:~/CPE212_TITONGg$ S
```

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."



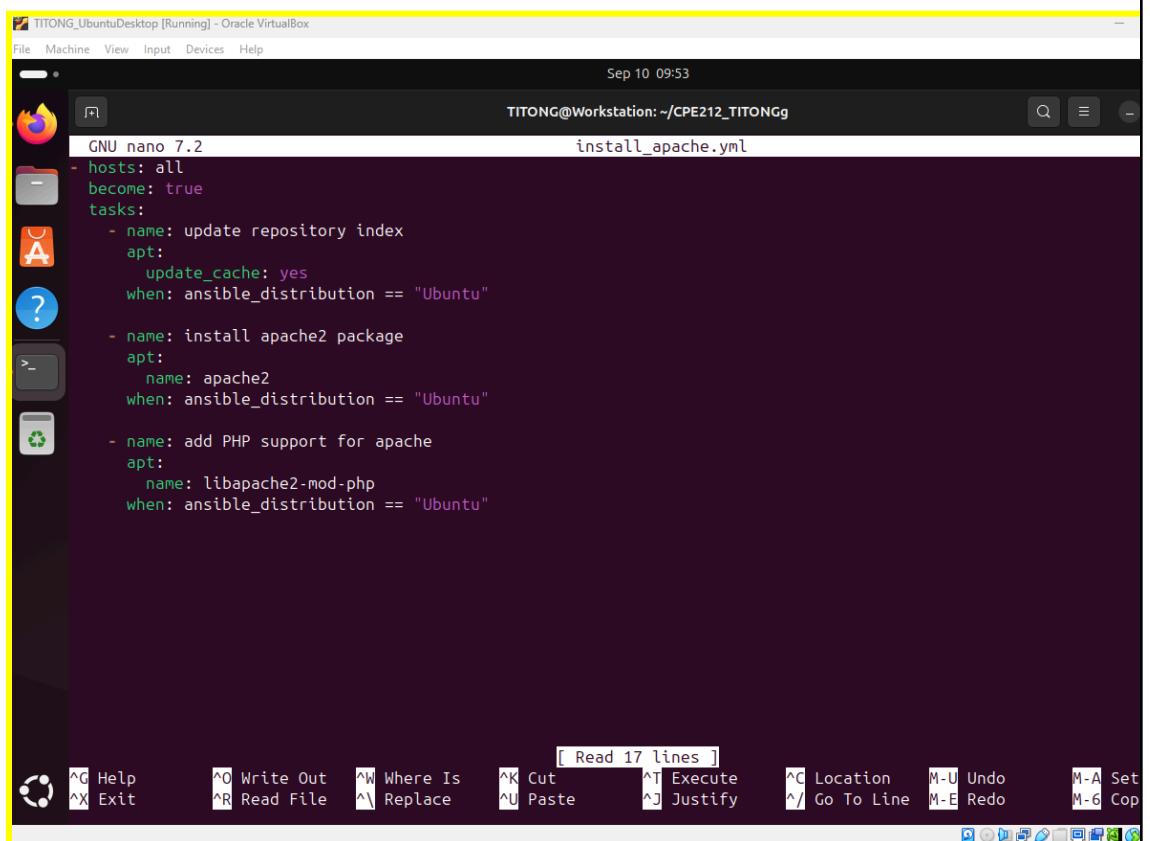
The screenshot shows a Linux desktop environment with a terminal window titled "inventory.yaml". The terminal displays the following content:

```
[all]
192.168.56.110 ansible_user=TITONG
192.168.56.111 ansible_user=TITONG
192.168.56.113 ansible_user=titong_leeeivan ansible_password=changeme2025
```

The terminal window is running the "nano" editor. The bottom status bar shows various keyboard shortcuts for the editor, such as "Read 5 lines", "Help", "Exit", "Write Out", "Where Is", "Cut", "Execute", "Location", "Undo", "Set Mark", etc.

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"
```



The screenshot shows a terminal window titled 'TITONG_UbuntuDesktop [Running] - Oracle VirtualBox'. The window title bar includes 'File Machine View Input Devices Help' and the date 'Sep 10 09:53'. The main area of the terminal displays the YAML configuration file 'install_apache.yml' using the 'GNU nano 7.2' editor. The file content is identical to the one shown above. The terminal window has a dark background and light-colored text. At the bottom, there is a toolbar with various icons and keyboard shortcuts for navigating the nano editor.

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"]

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"

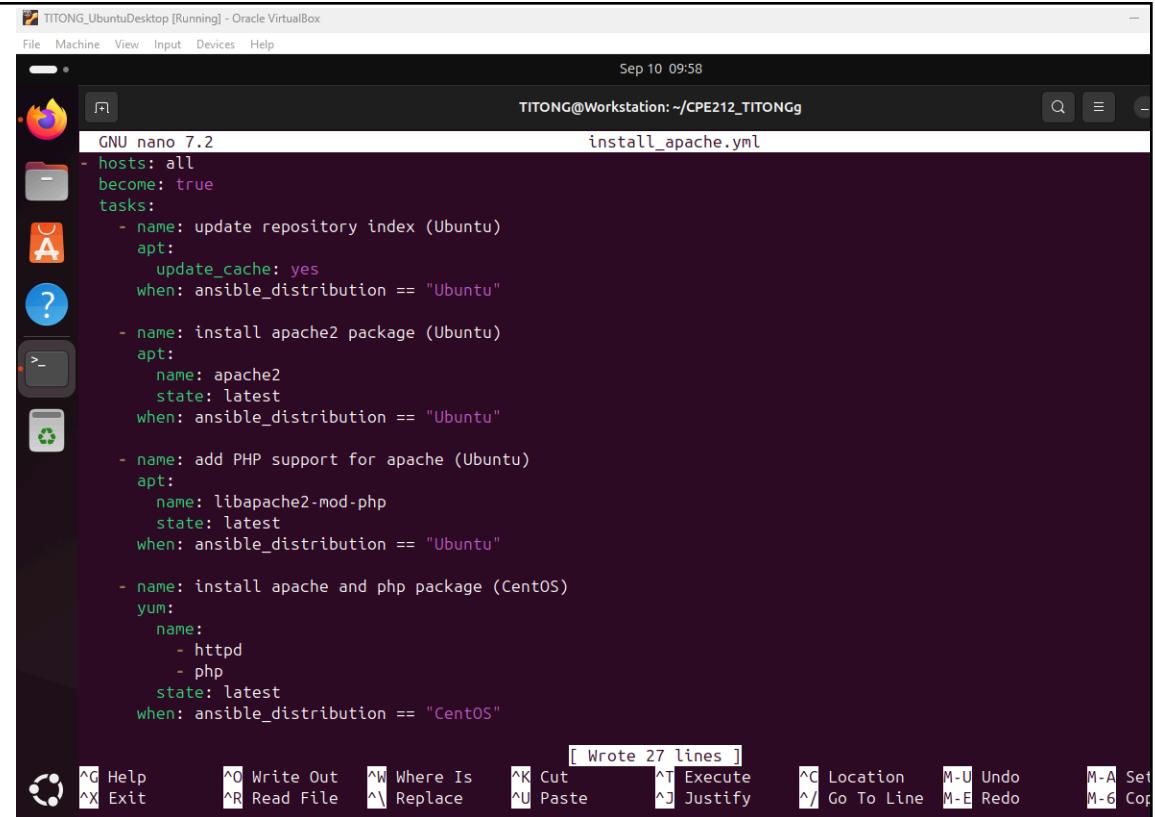
    - name: install apache2 package
      apt:
        name: apache2
        state: latest
      when: ansible_distribution == "Ubuntu"

    - 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"
```



The screenshot shows a terminal window titled "TITONG_UbuntuDesktop [Running] - Oracle VirtualBox". The terminal is running on a Ubuntu system, indicated by the icon in the top left and the "Ubuntu" distribution in the Ansible code. The user is editing an Ansible playbook named "install_apache.yml" using the "nano" text editor. The code in the editor is as follows:

```
GNU nano 7.2
hosts: all
become: true
tasks:
- name: update repository index (Ubuntu)
  apt:
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

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

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

- name: install apache and php package (CentOS)
  yum:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"
```

The terminal status bar indicates "Wrote 27 lines". Below the terminal are various keyboard shortcuts for the nano editor.

Make sure to save and exit.

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

```
Sep 10 10:43
TITONG@Workstation:~/CPE212_TITONGg

TASK [Gathering Facts]
ok: [192.168.56.110]
ok: [192.168.56.113]
ok: [192.168.56.111]

TASK [update repository index (Ubuntu)]
skipping: [192.168.56.113]
changed: [192.168.56.111]
changed: [192.168.56.110]

TASK [install apache2 package (Ubuntu)]
skipping: [192.168.56.113]
ok: [192.168.56.110]
ok: [192.168.56.111]

TASK [add PHP support for apache (Ubuntu)]
skipping: [192.168.56.113]
ok: [192.168.56.111]
ok: [192.168.56.110]

TASK [install apache and php package (CentOS)]
skipping: [192.168.56.110]
skipping: [192.168.56.111]
changed: [192.168.56.113]

PLAY RECAP
192.168.56.110      : ok=4    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.111      : ok=4    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.113      : ok=2    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0

TITONG@Workstation:~/CPE212_TITONGg$
```

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.

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.

```
titong_leeivan@vbox:~ — systemctl status httpd
```

```
[titong_leeivan@vbox ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: enabled)
   Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Fri 2025-09-12 16:38:44 PST; 18min ago
       Docs: man:httpd.service(8)
   Main PID: 937 (httpd)
      Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes sent/sec: 0"
      Tasks: 177 (limit: 23004)
     Memory: 44.7M
        CPU: 2.429s
       CGrou...  
Sep 12 16:38:44 localhost.localdomain systemd[1]: Starting The Apache HTTP Se...  
Sep 12 16:38:44 localhost.localdomain httpd[937]: AH00558: httpd: Could not r...  
Sep 12 16:38:44 localhost.localdomain httpd[937]: Server configured, listening...  
Sep 12 16:38:44 localhost.localdomain systemd[1]: Started The Apache HTTP Ser...  
lines 1-22/22 (END)
```

5.2 Issue the following command to start the service:

sudo systemctl start httpd

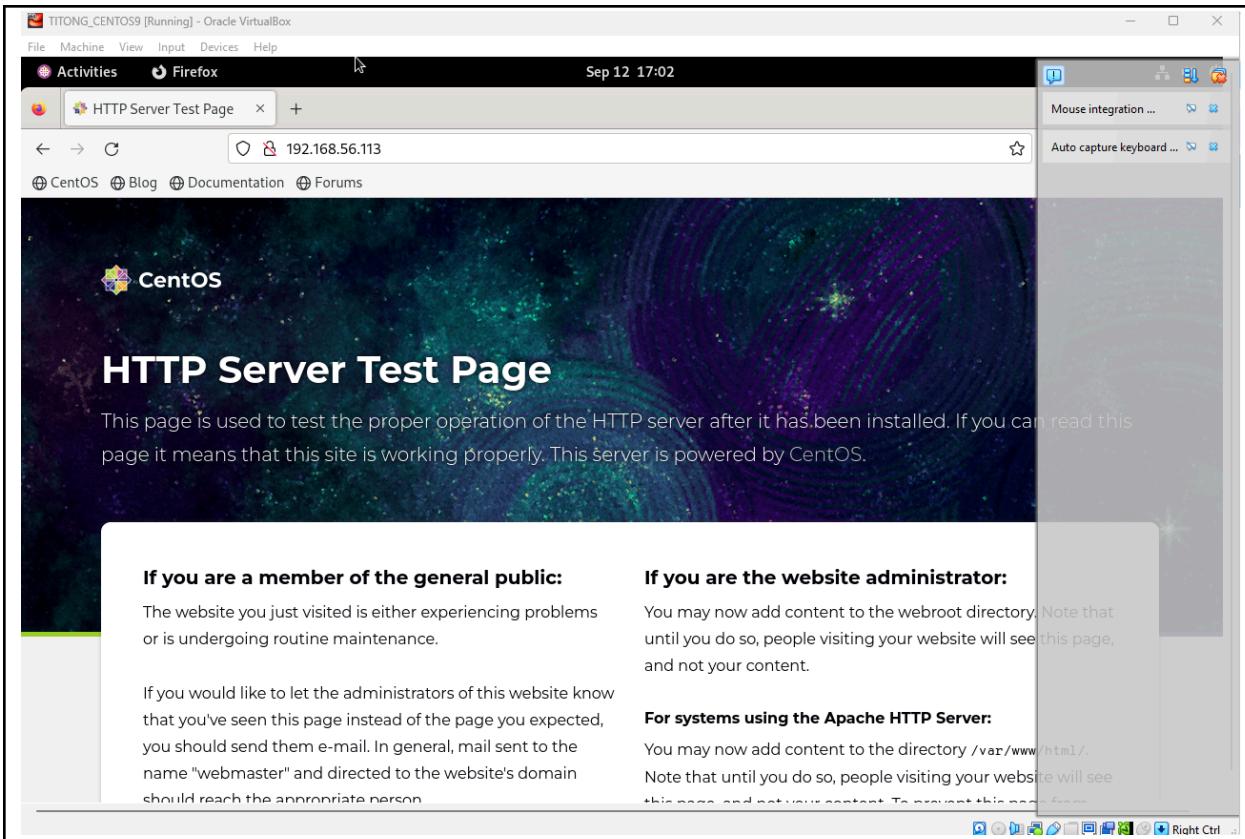
(When prompted, enter the sudo password)

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

(The result should be a success)

```
[titong_leeivan@vbox ~]$ sudo systemctl start httpd
[titong_leeivan@vbox ~]$ 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)



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.

```

TITONG@Workstation:~/CPE212_TITONG$ ansible-playbook --ask-become-pass install_apache_lab5.yml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.110]
ok: [192.168.56.113]
ok: [192.168.56.111]

TASK [update repository index Ubuntu] ****
skipping: [192.168.56.113]
changed: [192.168.56.110]
changed: [192.168.56.111]

TASK [install apache2 and php packages for Ubuntu] ****
skipping: [192.168.56.113]
ok: [192.168.56.111]
ok: [192.168.56.110]

TASK [update repository index for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TASK [install apache and php packages for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TITONG@Workstation:~/CPE212_TITONG$ ****
TASK [update repository index Ubuntu] ****
skipping: [192.168.56.113]
changed: [192.168.56.110]
changed: [192.168.56.111]

TASK [install apache2 and php packages for Ubuntu] ****
skipping: [192.168.56.113]
ok: [192.168.56.111]
ok: [192.168.56.110]

TASK [update repository index for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TASK [install apache and php packages for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

PLAY RECAP ****
192.168.56.110      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0
=0
192.168.56.111      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0
=0
192.168.56.113      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
=0

TITONG@Workstation:~/CPE212_TITONG$ 

```

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.

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

```

TITONG@Workstation:~/CPE212_TITONGg$ ansible-playbook --ask-become-pass install_5_task2.yml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.111]
ok: [192.168.56.110]
ok: [192.168.56.113]

TASK [update repository index Ubuntu] ****
skipping: [192.168.56.113]
changed: [192.168.56.111]
changed: [192.168.56.110]

TASK [install apache2 and php packages for Ubuntu] ****
skipping: [192.168.56.113]
ok: [192.168.56.110]
ok: [192.168.56.111]

TASK [update repository index for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TASK [install apache and php packages for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TITONG@Workstation:~/CPE212_TITONGg$ 

TASK [update repository index Ubuntu] ****
skipping: [192.168.56.113]
changed: [192.168.56.111]
changed: [192.168.56.110]

TASK [install apache2 and php packages for Ubuntu] ****
skipping: [192.168.56.113]
ok: [192.168.56.110]
ok: [192.168.56.111]

TASK [update repository index for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

TASK [install apache and php packages for CentOS] ****
skipping: [192.168.56.110]
skipping: [192.168.56.111]
ok: [192.168.56.113]

PLAY RECAP ****
192.168.56.110      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0
=0
192.168.56.111      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0
=0
192.168.56.113      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
=0

TITONG@Workstation:~/CPE212_TITONGg$ 

```

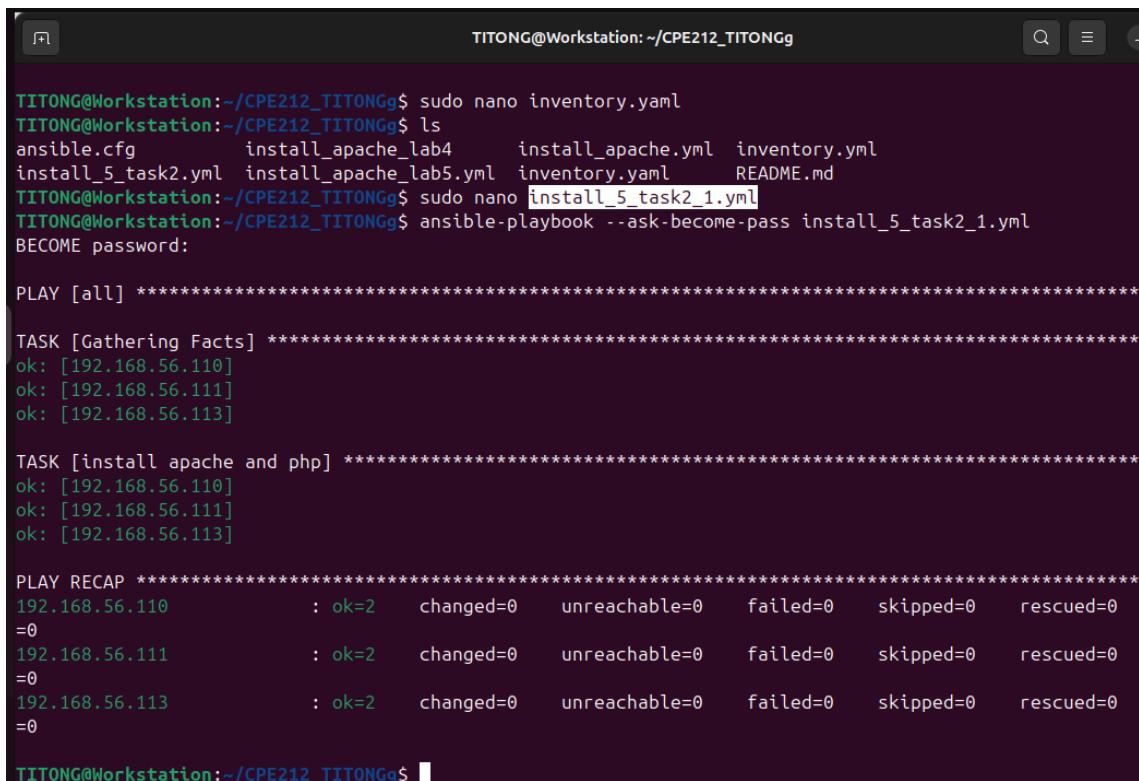
3. Finally, we can consolidate these 2 plays in just 1 play. This can be done by declaring variables that will represent the packages that we want to install. Basically, the `apache_package` and `php_package` are variables. The names

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.



The screenshot shows a terminal window titled "Thunderbird Mail" with the command line interface. The user is running an Ansible playbook named "install_5_task2_1.yml". The terminal output shows the playbook's tasks being executed on three hosts (192.168.56.110, 192.168.56.111, and 192.168.56.113). The "TASK [Gathering Facts]" step shows the hosts are up and running. The "TASK [install apache and php]" step shows the packages were installed successfully. The "PLAY RECAP" summary indicates all hosts completed the task with 2 ok status and 0 errors.

```
TITONG@Workstation:~/CPE212_TITONGg$ sudo nano inventory.yaml
TITONG@Workstation:~/CPE212_TITONGg$ ls
ansible.cfg      install_apache_lab4      install_apache.yml  inventory.yaml
install_5_task2.yml  install_apache_lab5.yml  inventory.yaml  README.md
TITONG@Workstation:~/CPE212_TITONGg$ sudo nano install_5_task2_1.yml
TITONG@Workstation:~/CPE212_TITONGg$ ansible-playbook --ask-become-pass install_5_task2_1.yml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.110]
ok: [192.168.56.111]
ok: [192.168.56.113]

TASK [install apache and php] ****
ok: [192.168.56.110]
ok: [192.168.56.111]
ok: [192.168.56.113]

PLAY RECAP ****
192.168.56.110      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
=0
192.168.56.111      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
=0
192.168.56.113      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
=0

TITONG@Workstation:~/CPE212_TITONGg$
```

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.

```
TITONG@Workstation:~/CPE212_TITONGg$ cat inventory.yaml  
[ubuntu]  
192.168.56.110 ansible_user=TITONG apache_package=apache2 php_package=libapache2-mod-php  
192.168.56.111 ansible_user=TITONG apache_package=apache2 php_package=libapache2-mod-php  
  
[centos]  
192.168.56.113 ansible_user=titong_leeivan ansible_password=changeme2025 apache_package=httpd php_package/php  
  
[all:vars]  
ansible_become=true  
TITONG@Workstation:~/CPE212_TITONGg$
```

```
TITONG@Workstation:~/CPE212_TITONGg$ sudo nano inventory.yaml  
TITONG@Workstation:~/CPE212_TITONGg$ ls  
ansible.cfg      install_apache_lab4    install_apache.yml  inventory.yml  
install_5_task2.yml  install_apache_lab5.yml  inventory.yaml      README.md  
TITONG@Workstation:~/CPE212_TITONGg$ sudo nano install_5_task2_1.yml  
TITONG@Workstation:~/CPE212_TITONGg$ ansible-playbook --ask-become-pass install_5_task2_1.yml  
BECOME password:  
  
PLAY [all] ****  
  
TASK [Gathering Facts] ****  
ok: [192.168.56.110]  
ok: [192.168.56.111]  
ok: [192.168.56.113]  
  
TASK [install apache and php] ****  
ok: [192.168.56.110]  
ok: [192.168.56.111]  
ok: [192.168.56.113]  
  
PLAY RECAP ****  
192.168.56.110      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0  
=0  
192.168.56.111      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0  
=0  
192.168.56.113      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0  
=0  
TITONG@Workstation:~/CPE212_TITONGg$
```

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.

Supplementary Activity:

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

```
TITONG@Workstation:/CPE212_TITONG$ ansible-playbook --ask-become-pass install_5_supplementary.yml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.110]
ok: [192.168.56.113]
ok: [192.168.56.111]

TASK [install apache and php on Red Hat] ****
ok: [192.168.56.113]
ok: [192.168.56.110]
ok: [192.168.56.111]

PLAY RECAP ****
192.168.56.110      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored
=0
192.168.56.111      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored
=0
192.168.56.113      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored
=0

TITONG@Workstation:/CPE212_TITONG$
```

The screenshot shows a terminal window titled "TITONG_UbuntuDesktop [Running] - Oracle VirtualBox". The window has a dark theme. The title bar includes "File Machine View Input Devices Help" and the date "Sep 12 09:28". The main area displays a file named "install_5_supplementary.yml" with the following content:

```
GNU nano 7.2                               install_5_supplementary.yml
hosts: all
become: true
tasks:
  - name: install apache and php on Red Hat
    package:
      name:
        - "{{ apache_package }}"
        - "{{ php_package }}"
    state: latest
    update_cache: yes
```

The bottom of the terminal shows a menu of keyboard shortcuts: [Read 11 lines], ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^/ Go To Line, M-U Undo, M-E Redo.

Reflections:

Answer the following:

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

Refactoring makes the playbook cleaner, shorter, and easier to manage

Refactoring playbook codes is important because:

- It makes the playbook clearer and easier to read – tasks become simple and understandable.
- It is easier to maintain and update – changes can be done in one place instead of many.
- It removes duplication – you don't repeat the same tasks for different operating systems.
- It becomes more reusable and scalable – you can use the same playbook for more hosts or OS without rewriting everything.
- It makes execution and troubleshooting faster – errors are easier to find and fix

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

We use “when” to control which tasks run depending on conditions such as OS, variables, or system state.

We use the when command when we want a task to run only under certain conditions.

Examples:

- When a task should run only on Ubuntu (using apt) or only on CentOS (using dnf).
- When we want a task to run only if a variable has a specific value.
- When we need to check the OS version, host name, or the presence of a file before running a task.