Name: Hazel Aillson T. Manuel	**Date Performed:** 08/28/2025
Course/Section: CPE31S4	**Date Submitted:** 09/12/2025
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Sem 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?

```
hazel@LocalMachine:~/CPE232_Manuel$ git pull
Already up to date.
```

Yes, something did happen which notified me that the system is already up to date. Git Pull ensures that any changes in the repository will also be updated to our 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."

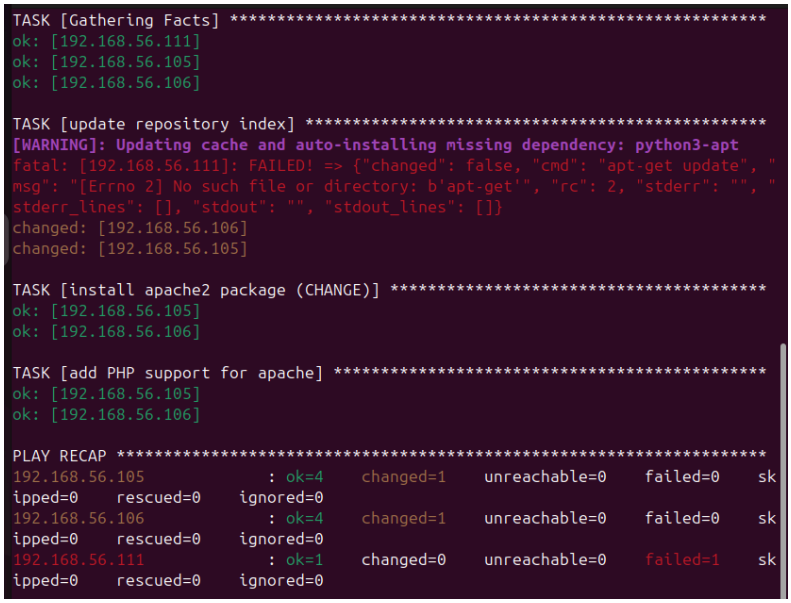


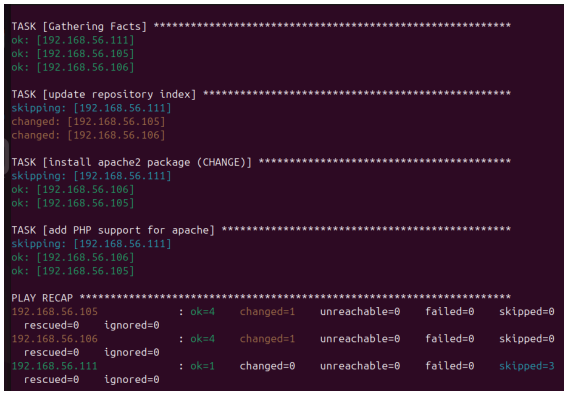
Figure 1: Running the playbook file with CentOS host, failed execution

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.



It shows that it skipped the IP address of the CentOS. This occurred as it does not satisfy the condition of the distribution being in Ubuntu

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.

```
TASK [update repository index] ******************
skipping: [192.168.56.111]
changed: [192.168.56.106]
changed: [192.168.56.105]

TASK [install apache2 package (CHANGE)] *********
skipping: [192.168.56.111]
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [add PHP support for apache] ***************
skipping: [192.168.56.111]
ok: [192.168.56.106]
ok: [192.168.56.105]

PLAY RECAP **************************************
192.168.56.105             : ok=4    changed=1
  rescued=0    ignored=0
192.168.56.106             : ok=4    changed=1
  rescued=0    ignored=0
192.168.56.111             : ok=1    changed=0
  rescued=0    ignored=0
```

It shows that the playbook still skips the CentOS IP address. This happens because our playbook uses the apt package manager. With that, it does not make any changes in the CentOS Machine.

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.

```
TASK [update repository index] *************************************************
skipping: [192.168.56.111]
changed: [192.168.56.105]
changed: [192.168.56.106]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.111]
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.111]
ok: [192.168.56.106]
ok: [192.168.56.105]
```

```
TASK [update repository index] *************************************************
skipping: [192.168.56.105]
skipping: [192.168.56.106]
ok: [192.168.56.111]

TASK [install apache2 package] *************************************************
skipping: [192.168.56.105]
skipping: [192.168.56.106]
ok: [192.168.56.111]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.56.105]
skipping: [192.168.56.106]
changed: [192.168.56.111]
```

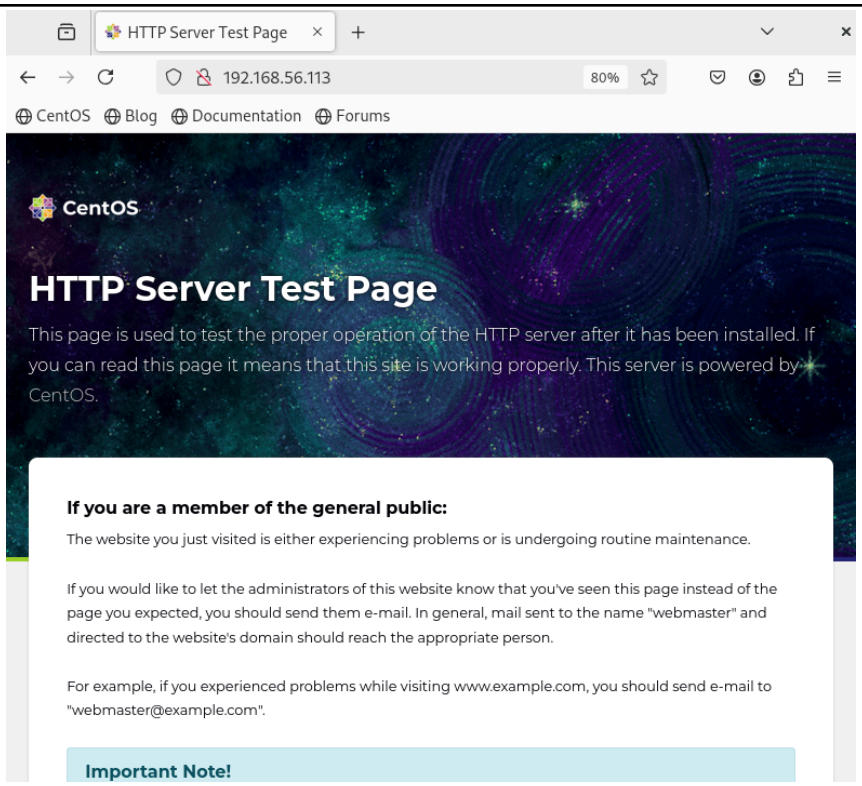
In this part, we are able to make changes with our CentOS machine. As you can see, some of the IP addresses are skipped. This is because it does not satisfy the condition under that task such as the distribution. Satisfying that condition, either result to an ok or a changed status.

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.

   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)

   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)



Running the IP address of the Machine, it shows us the Test page of the HTTP Server. This indicates that the httpd has been installed and started properly.

Task 2: Refactoring playbook

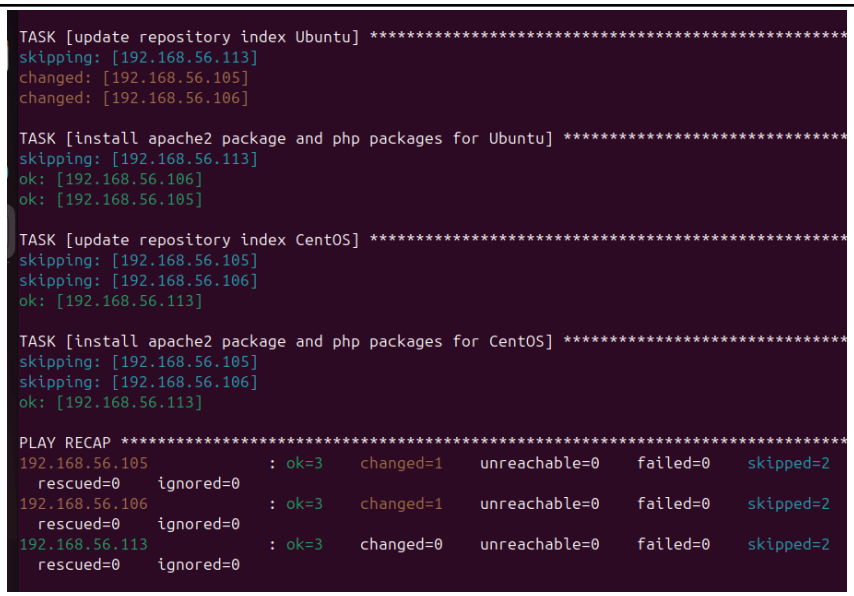
This time, we want to make sure that our playbook is efficient and that the codes are easier to read. This will also makes run ansible more quickly if it has to execute fewer tasks to do the same thing.

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.



This is somewhat the same as the previous one but much shorter and simplified. This lets us see how we can install multiple packages in one task. In broader terms, multiple things to do in one task. This makes our playbook much shorter and easier to maintain.

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"
```

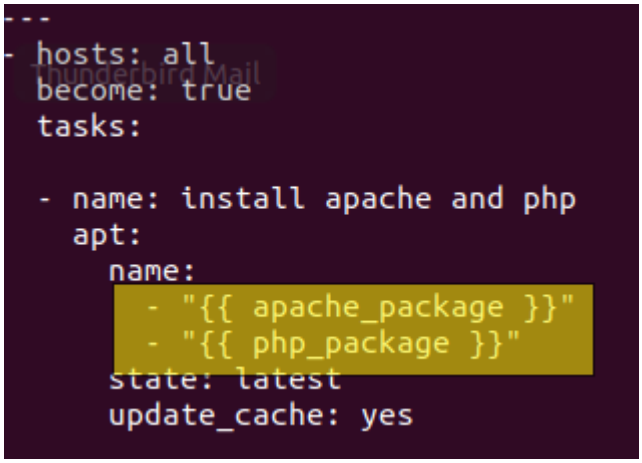
Make sure to save the file and exit.

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

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.106]
ok: [192.168.56.105]
ok: [192.168.56.113]

TASK [install apache2 package and php packages for Ubuntu] *********************
skipping: [192.168.56.113]
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [install apache2 package and php packages for CentOS] *********************
skipping: [192.168.56.105]
skipping: [192.168.56.106]
ok: [192.168.56.113]

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

In this playbook, we are able to implement the update_cache. The purpose of this is that the specific packages are the only one that updates. This ensures that all of the packages that we are installing are up to date.

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.



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

Running this playbook results in a failed one where the changes that we did are undefined. Due to this being undefined, an Error/Failure of execution occurs.

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
```

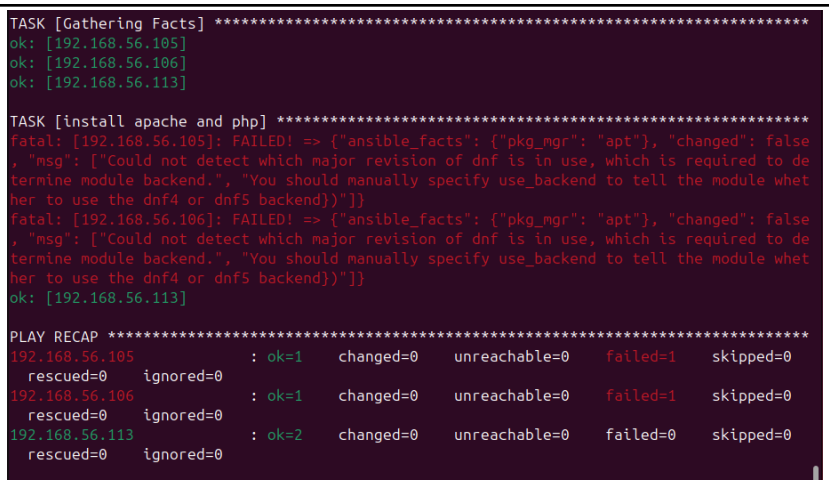
Make sure to save the *inventory* file and exit.

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.

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.105]
ok: [192.168.56.106]
ok: [192.168.56.113]

TASK [install apache and php] **************************************************
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
fatal: [192.168.56.113]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[E
rrno 2] No such file or directory: b'apt-get'", "rc": 2, "stderr": "", "stderr_lines": [],
 "stdout": "", "stdout_lines": []}
ok: [192.168.56.105]
ok: [192.168.56.106]

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

Initially running the playbook, it is able to run for both Ubuntu Machines. However, CentOS Machine resulted in a failure because of the apt package manager. To resolve this issue, we can change the apt to dnf then the CentOS Machine will run perfectly, however the two Ubuntu will result in a failure. Refer to the image below for the CentOS one.



Supplementary Activity:

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

Note: Can't run as unable to install centos

```
None
---
- name: Previous Tasks but on Red Hat
  hosts: redhat
  become: true
  tasks:

       - name: Updating all packages
         dnf:
           name: "*"
           state: latest
           update_cache: yes
         when: ansible_distribution == "RefHat"

       - name: Instal Apache and PHP on RedHat
         dnf:
           name:
             - httpd
             - php
           state: present
         when: ansible_distribution == "RedHat"
```

In this supplementary activity, it is somewhat the same as the Ubuntu and CentOS one. The difference is that the host is for redhat group only and the distribution for when is RedHat. Of course, if we made the host for all, it will skip all the machines that are not Red Hat distribution.

Reflections:

Answer the following:

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

   Refractory the playbook code is important as it makes the code readable, maintainable and much efficient. This basically allows us to consolidate tasks through variables and cleanly organize the logic. This reduces the complexity of our playbook and allows other users or our fellow teammates to easily understand how our playbook works.

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

   We utilized the when command in the playbook when we want to apply a conditional logic where it will only execute the task when it satisfies the criteria. This is useful when having remote servers that have different operating systems. Depending on the ansible distribution that we have specified, it will only run the tasks on that specific machine.