Name: Gayao, Froilan M.	Date Performed: 9/27/24
Course/Section: CPE-31S4	Date Submitted: 9/27/24
Instructor: Enr. Robin Valenzuela	Semester and SY: 1st 24-25

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.

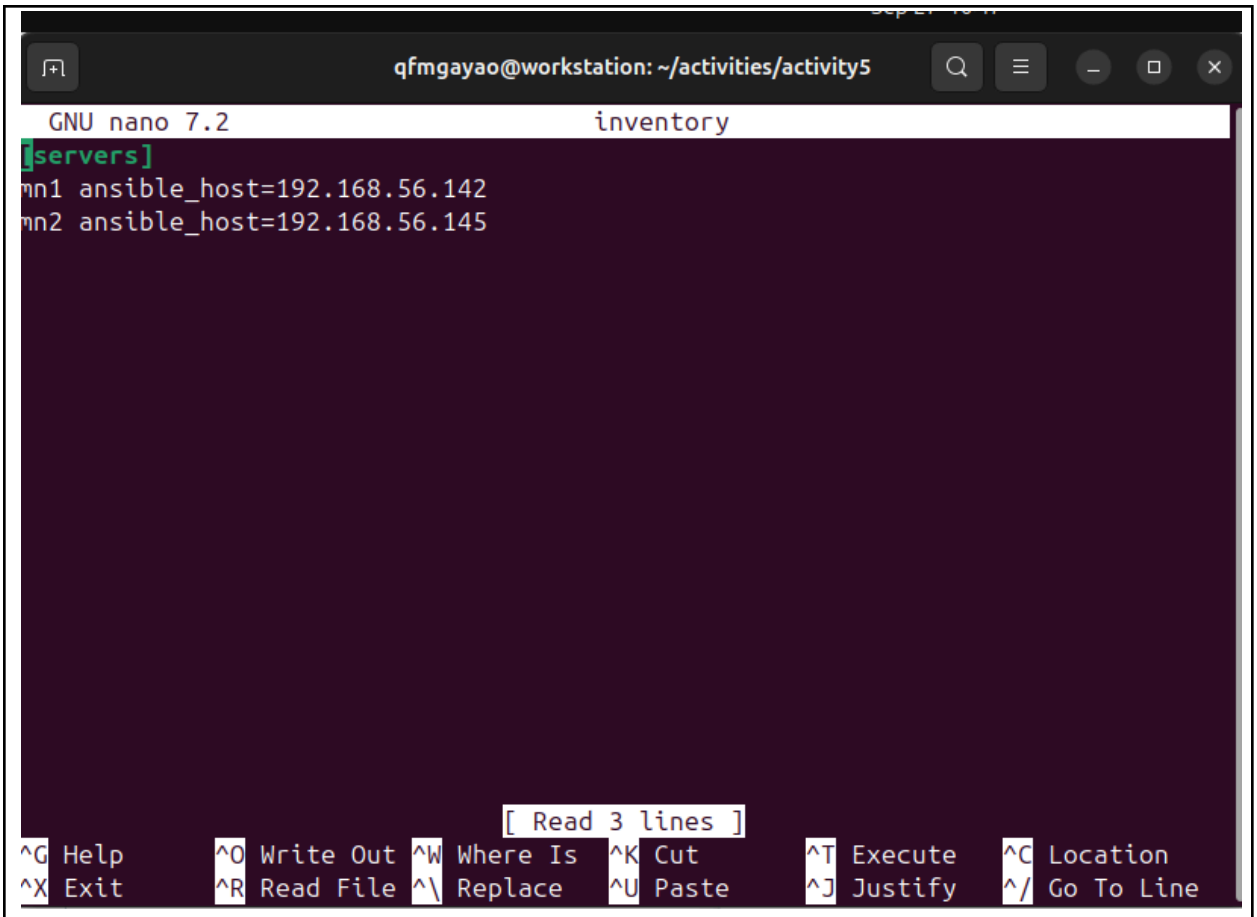
Task 1: Use when command for different distributions

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?

```
qfmgayao@workstation:~$ cd activities
qfmgayao@workstation:~/activities$ ls
activity-4.1-new  Gayao_PrelimExam
qfmgayao@workstation:~/activities$ sudo nano /etc/hosts
[sudo] password for qfmgayao:
qfmgayao@workstation:~/activities$ git pull git@github.com:PooKYZZZ/activity5.gi
t
fatal: not a git repository (or any of the parent directories): .git
qfmgayao@workstation:~/activities$ ls
activity-4.1-new  Gayao_PrelimExam
qfmgayao@workstation:~/activities$ git clone git@github.com:PooKYZZZ/activity5.g
it
Cloning into 'activity5'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.
qfmgayao@workstation:~/activities$ ls
```

Instead of using git pull, i just make a new repository which Instead of git pull, I use git clone then I also made ansible.cfg and inventory inside the github and I just clone it to my workstation

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."



I inserted my CentOS ip address to my workstation inventory for it to connect to ansible.

```
qfmgayao@workstation:~/activities/activity5$ ansible-playbook --ask-become-pass
apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [mn1]
ok: [mn2]

TASK [Update package index] ****************************************************
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
fatal: [mn2]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Er
rno 2] No such file or directory: b'apt-get'", "rc": 2, "stderr": "", "stderr_li
nes": [], "stdout": "", "stdout_lines": []}
changed: [mn1]

TASK [Install apache2 package] *************************************************
ok: [mn1]

TASK [add PHP support for apache] **********************************************
ok: [mn1]

PLAY RECAP *********************************************************************
```

This failed because “apt” command doesn’t work in CentOS instead we need to use yum command.

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"
```

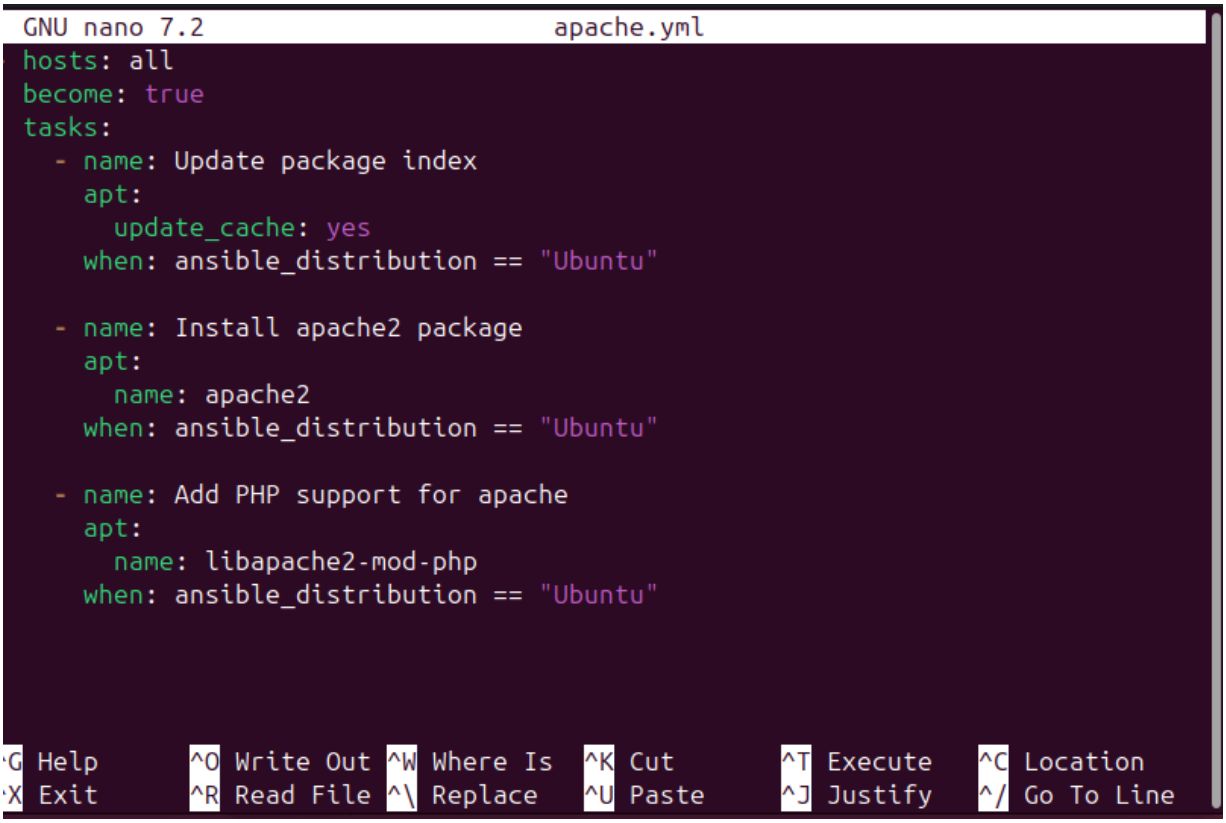
Make sure to save the file and exit.

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]

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



```
qfmgayao@workstation:~/activities/activity5$ ansible-playbook --ask-become-pass apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [mn1]
ok: [mn2]

TASK [Update package index] ****************************************************
skipping: [mn2]
changed: [mn1]

TASK [Install apache2 package] *************************************************
skipping: [mn2]
ok: [mn1]

TASK [Add PHP support for apache] **********************************************
skipping: [mn2]
ok: [mn1]

PLAY RECAP *********************************************************************
mn1                        : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
mn2                        : ok=1    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```

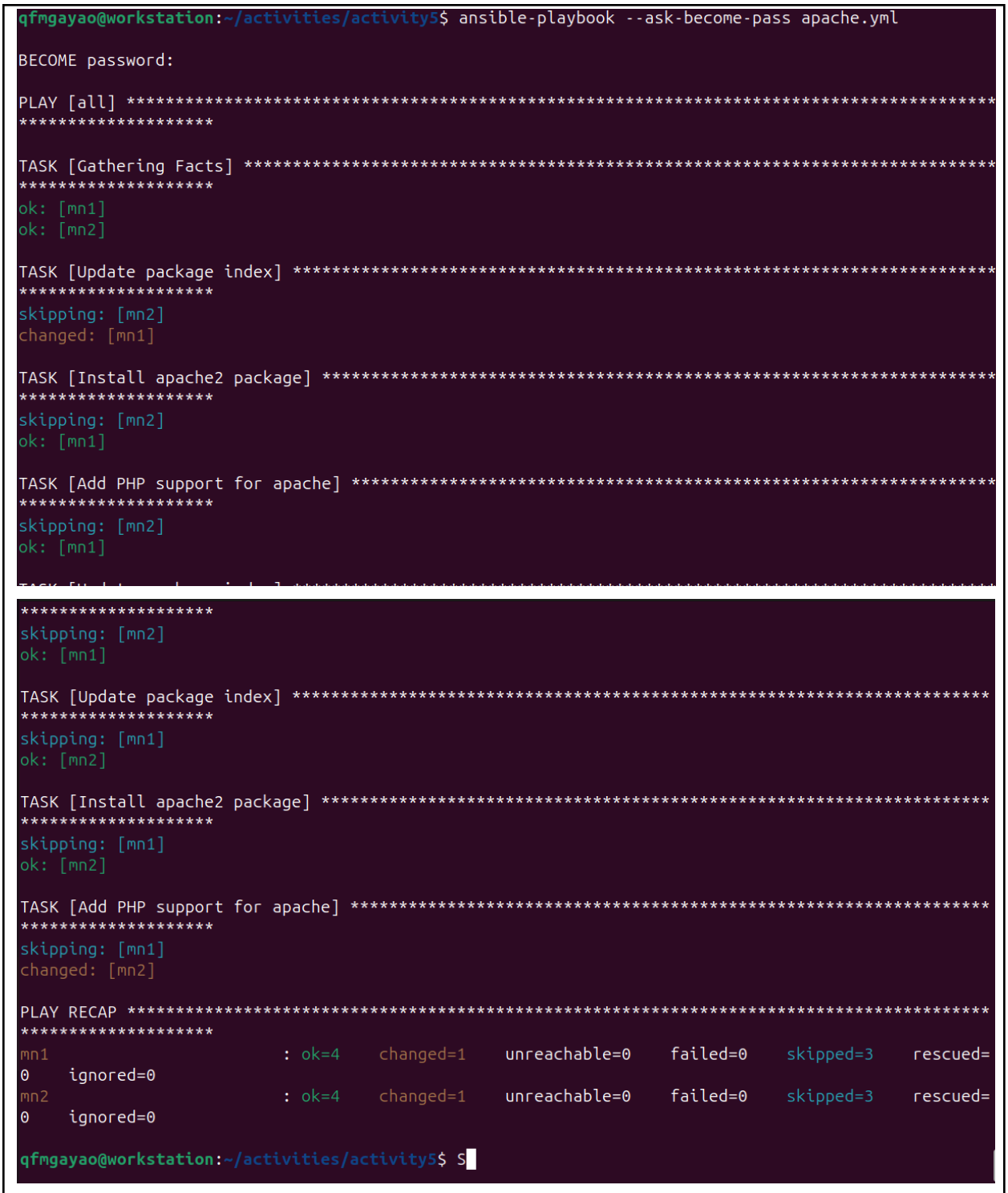
It skip mn2 because in my playbook yml I specified that it will only install in ubuntu system.

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"
```

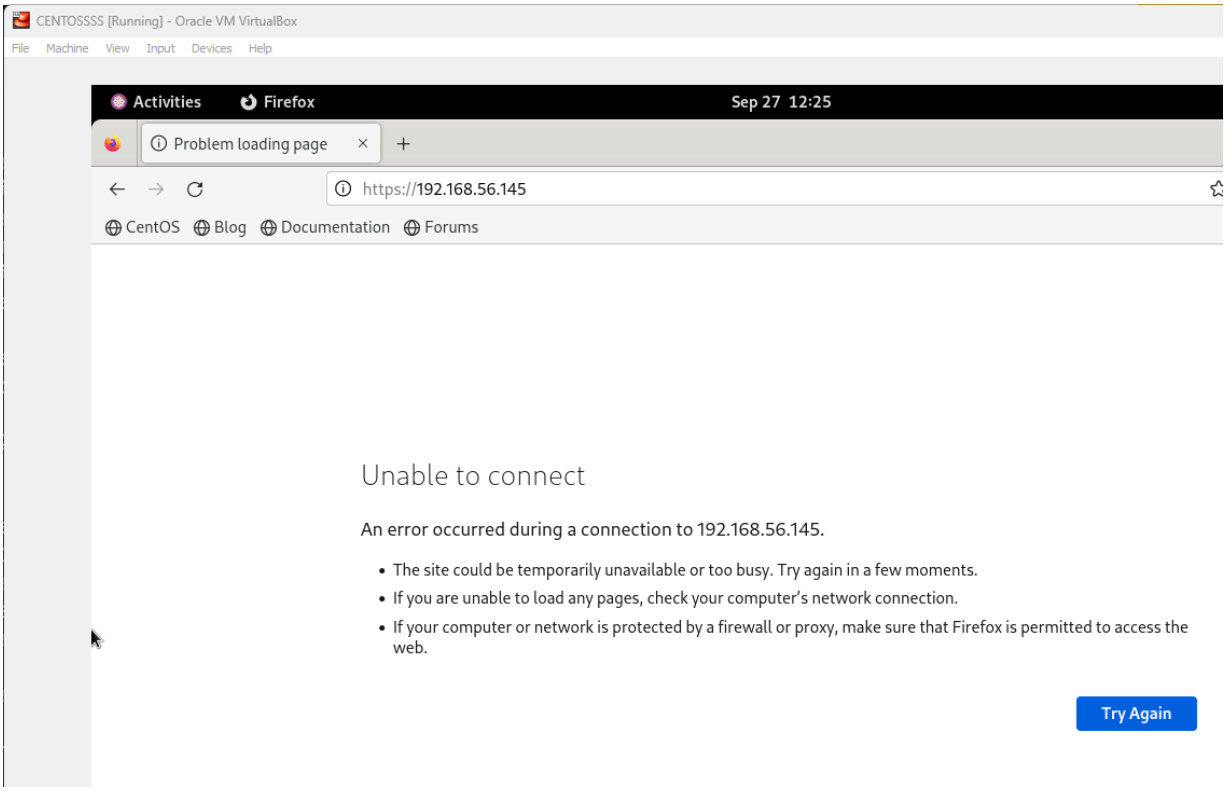
Make sure to save and exit.

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



Now that we have specified to install into both Ubuntu and CentOS, it functions in both directions. It also shows that we have six lines of tasks, which they skip when you indicate in the code that you wish to install in both Ubuntu and CentOS.

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.



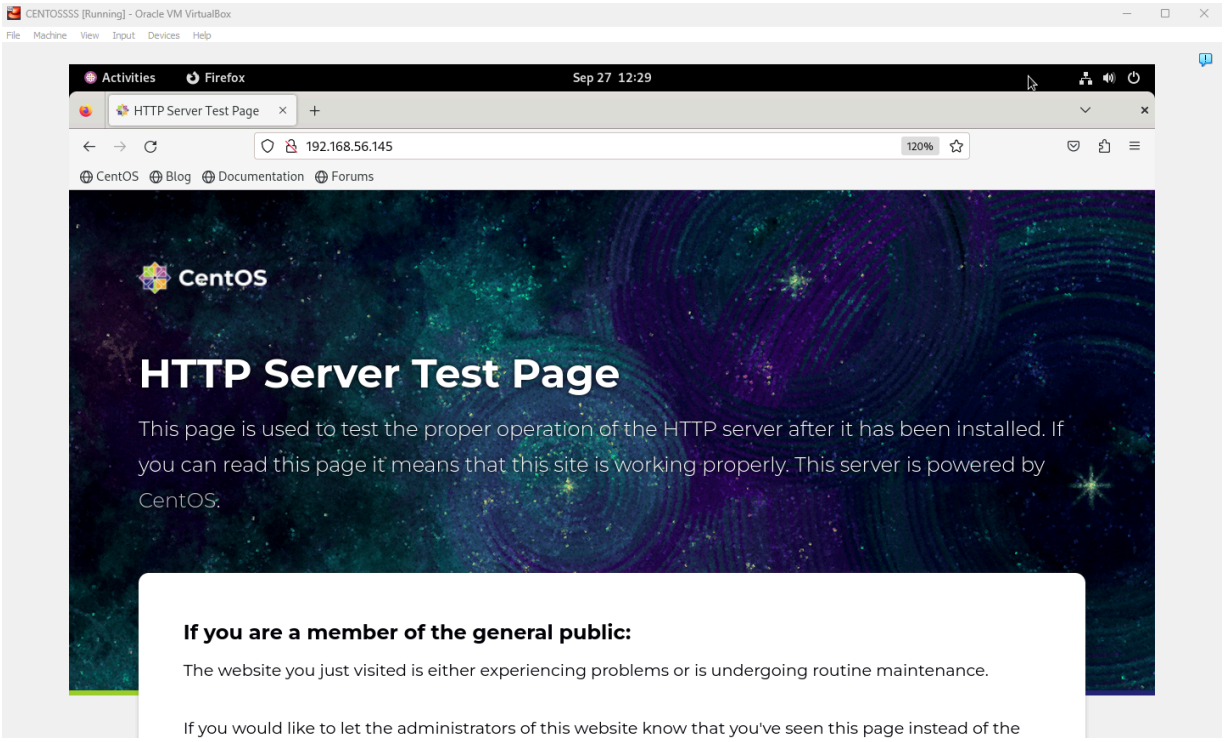
5.1 To activate, go to the CentOS VM terminal and enter the following:
systemctl status httpd
The result of this command tells you that the service is inactive.

```
qfmgayao@workstation:~/activities/activity5$ ssh qfmgayao@mn2
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Sep 27 12:00:13 2024 from 192.168.56.141
[qfmgayao@mn2 ~]$ systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[qfmgayao@mn2 ~]$
```

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)

```
[qfmgayao@mn2 ~]$ sudo systemctl start htttpd
[sudo] password for qfmgayao:
Sorry, try again.
[sudo] password for qfmgayao:
Failed to start htttpd.service: Unit htttpd.service not found.
[qfmgayao@mn2 ~]$ sudo systemctl start httpd
[qfmgayao@mn2 ~]$ sudo firewall-cmd --add-port=80/tcp
success
[qfmgayao@mn2 ~]$
```

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.

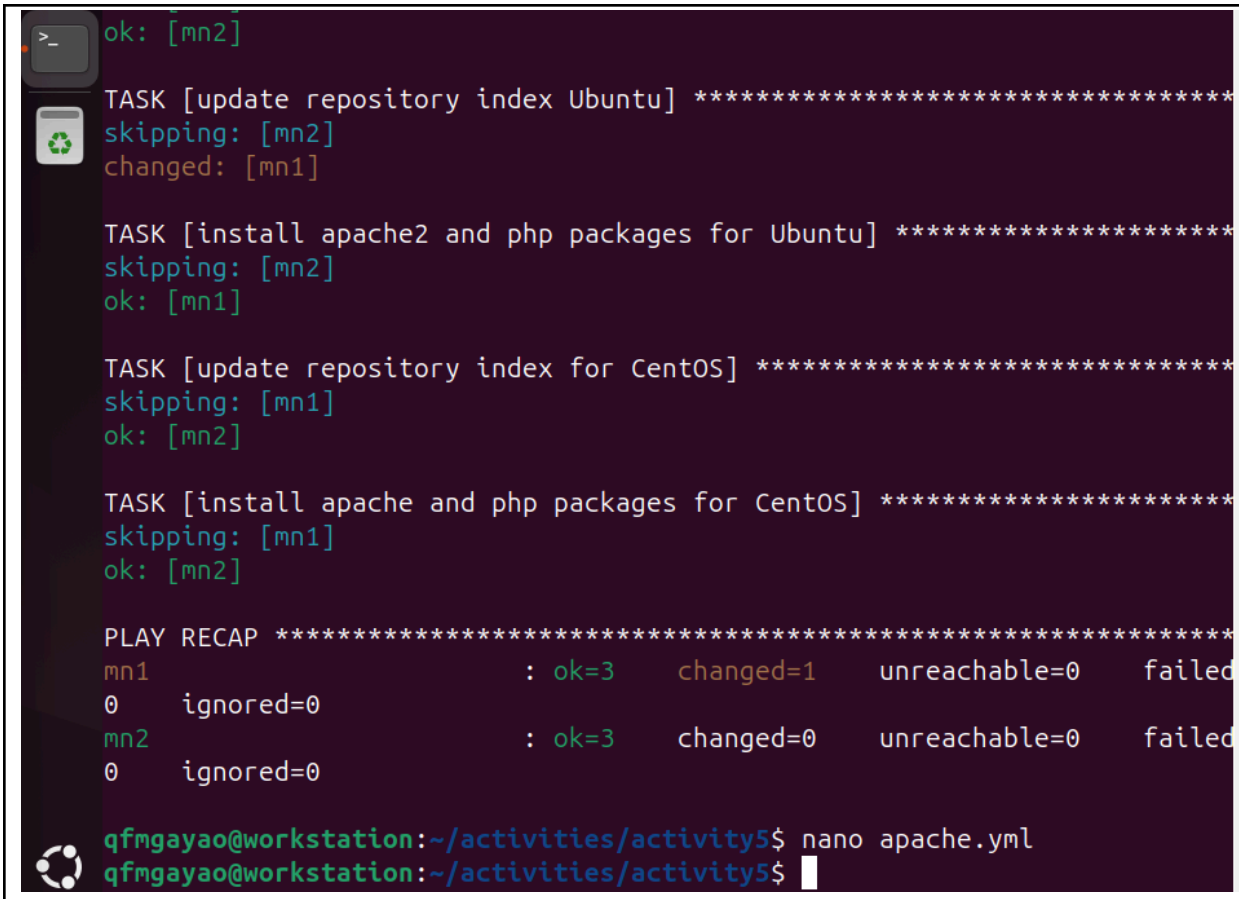
1. Edit the playbook *install_apache.yml*. Currently, we have three tasks targeting our Ubuntu machines and 3 tasks targeting our CentOS machine. Right now, we try to consolidate some tasks that are typically the same. For example, we

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.



This is possible because we integrated the install packages for PHP and Apache in our code, eliminating unnecessary parts and creating a more readable and organized code.

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.

```
qfmgayao@workstation:~/activities/activity5$ ansible-playbook --ask-becom
BECOME password:

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

TASK [Gathering Facts] *********************************************
ok: [mn1]
ok: [mn2]

TASK [install apache2 and php packages for Ubuntu] *****************
skipping: [mn2]
ok: [mn1]

TASK [install apache and php packages for CentOS] ******************
skipping: [mn1]
ok: [mn2]

PLAY RECAP *********************************************************
mn1                        : ok=2    changed=0    unreachable=0    failed
0    ignored=0
mn2                        : ok=2    changed=0    unreachable=0    failed
0    ignored=0

qfmgayao@workstation:~/activities/activity5$
```

We simplify our code by removing the task of updating the cache index and placing it under the two tasks since it is a one-liner.

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.

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

TASK [Gathering Facts] *********************************************************************************
ok: [mn1]
ok: [mn2]

TASK [install apache and php] **************************************************************************
fatal: [mn1]: 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 '/ho
me/qfmgayao/activities/activity5/apache.yml': line 4, column 7, but may\nbe elsewhere in the file de
pending on the exact syntax problem.\n\nThe offending line appears to be:\n\n  tasks:\n    - name: i
nstall apache and php\n      ^ here\n"}
fatal: [mn2]: 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 '/ho
me/qfmgayao/activities/activity5/apache.yml': line 4, column 7, but may\nbe elsewhere in the file de
pending on the exact syntax problem.\n\nThe offending line appears to be:\n\n  tasks:\n    - name: i
nstall apache and php\n      ^ here\n"}

PLAY RECAP *********************************************************************************************
mn1                        : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=
0    ignored=0
mn2                        : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=
0    ignored=0

qfmgayao@workstation:~/activities/activity5$
```

according to the line of errors, the packages is not defined.

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.

```
[servers]
mn1 ansible_host=192.168.56.142 ansible_become_pass="dikoalam1991" apache_package=apache2 php_packa>
mn2 ansible_host=192.168.56.145 ansible_become_pass="qfmgayao" apache_package=httpd php_package=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 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.

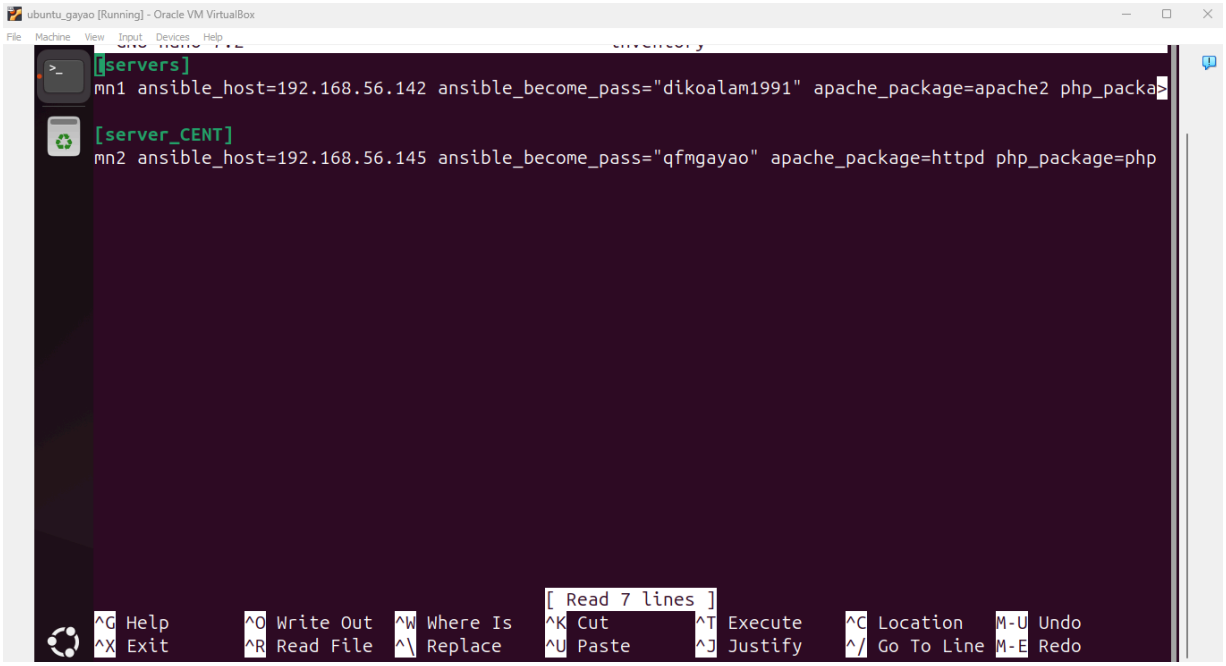
```yaml
- hosts: all
  become: true
  tasks:
    - name: install apache and php
      package:
        name:
          - "{{ apache_package }}"
          - "{{ php_package }}"
        state: latest
        update_cache: yes
```

```
qfmgayao@workstation:~/activities/activity5$ ansible-playbook --ask-become-pass apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [mn1]
ok: [mn2]

TASK [install apache and php] **************************************************
ok: [mn1]
ok: [mn2]

PLAY RECAP *********************************************************************
mn1                        : ok=2    changed=0    unreachable=0    failed=0    skipped=0    r
0    ignored=0
mn2                        : ok=2    changed=0    unreachable=0    failed=0    skipped=0    r
```

Now this code adds more lines of code to the inventory where the Apache package is assigned, in which it simplifies the task that needs to be installed using packages.

Supplementary Activity:

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



in this code, I separate the Cent OS to the ubuntu server which i can use in the playbook yml.

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

after separating ubuntu and centOS, I use the serverCENT to the hosts.

```
qfmgayao@workstation:~/activities/activity5$ ansible-playbook --ask-become-pass apache.yml
BECOME password:

PLAY [server_CENT] *************************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [mn2]

TASK [install apache and php] **************************************************************
ok: [mn2]

PLAY RECAP *********************************************************************************
mn2                        : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=
0    ignored=0

qfmgayao@workstation:~/activities/activity5$
```

this shows the result where it successfully worked.

Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?
   - the refactoring of playbook codes makes it cleaner and easy to read the codes which we can use to fix the bugs if there's one.
2. When do we use the "when" command in playbook?

- We only use the command in the playbook when we need to use task run only if it's true, it checks the condition before doing a task, if the condition turns false, it skips the task.