Name: Karlo Santos	**Date Performed:09/22/23**
Course/Section: CPE31S5	**Date Submitted:09/26/23**
Instructor: Engr. Roman Richard	**Semester and SY: 1st (2023-2024)**

Activity 5: Consolidating Playbook plays

1. **Objectives:**

1.1 Use **when** command in playbook for different OS distributions

1.2 Apply refactoring techniques in cleaning up the playbook codes

2. **Discussion**:

We are going to look at a way that we can differentiate a playbook by a host in terms of which distribution the host is running. It's very common in most Linux shops to run multiple distributions, for example, Ubuntu shop or Debian shop and you need a different distribution for a one off-case or perhaps you want to run plays only on certain distributions.

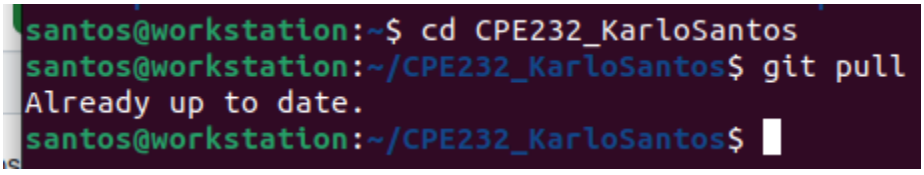
It is a best practice in ansible when you are working in a collaborative environment to use the command git pull. git pull is a Git command used to update the local version of a repository from a remote. By default, git pull does two things. Updates the current local working branch (currently checked out branch) and updates the remote-tracking branches for all other branches. git pull essentially pulls down any changes that may have happened since the last time you worked on the repository.

Requirement:

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

Task 1: Use when command for different distributions

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



It shows the message “Already up to date” after I issue the command git pull. It means it updates the local repository and matches its contents.

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."

```
  GNU nano 6.2                          inventory
[ubuntu_server]
192.168.100.122
192.168.100.123

[centos_server]
192.168.100.124
```

This screenshot shows the IP address that I put in inventory

```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass insta
ll_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.100.122]
ok: [192.168.100.124]
ok: [192.168.100.123]

TASK [update repository index] *************************************************
[WARNING]: Updating cache and auto-installing missing dependency: python-apt
fatal: [192.168.100.124]: FAILED! => {"changed": false, "cmd": "apt-get update",
"msg": "[Errno 2] No such file or directory", "rc": 2}
changed: [192.168.100.122]
changed: [192.168.100.123]

TASK [install apache2 package] *************************************************
ok: [192.168.100.122]
changed: [192.168.100.123]

TASK [add PHP support for apache] **********************************************
ok: [192.168.100.122]
changed: [192.168.100.123]

PLAY RECAP *********************************************************************
192.168.100.122            : ok=4    changed=1    unreachable=0    failed=0    sk
ipped=0    rescued=0    ignored=0
192.168.100.123            : ok=4    changed=3    unreachable=0    failed=0    sk
ipped=0    rescued=0    ignored=0
192.168.100.124            : ok=1    changed=0    unreachable=0    failed=1    sk
ipped=0    rescued=0    ignored=0
```

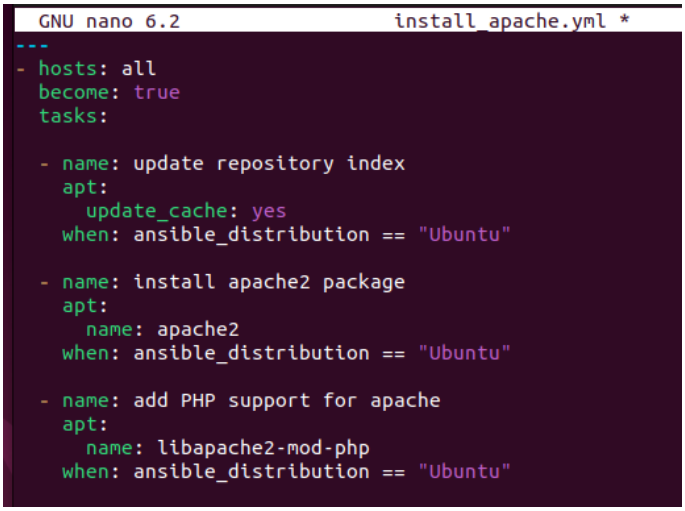
This screenshot shows the output after executing the command. We can see that the IP address in CentOS has failed=1 since it didn't support the apt package.

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.



```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass insta
ll_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.100.122]
ok: [192.168.100.124]
ok: [192.168.100.123]

TASK [update repository index] *************************************************
skipping: [192.168.100.124]
changed: [192.168.100.123]
changed: [192.168.100.122]

TASK [install apache2 package] *************************************************
skipping: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [add PHP support for apache] **********************************************
skipping: [192.168.100.124]
ok: [192.168.100.123]
ok: [192.168.100.122]

PLAY RECAP *********************************************************************
192.168.100.122            : ok=4    changed=1    unreachable=0    failed=0    sk
ipped=0    rescued=0    ignored=0
192.168.100.123            : ok=4    changed=1    unreachable=0    failed=0    sk
ipped=0    rescued=0    ignored=0
192.168.100.124            : ok=1    changed=0    unreachable=0    failed=0    sk
ipped=3    rescued=0    ignored=0
```

The screenshot shows that it updated the repository and installed the apache2 package successfully in the ubuntu server. While in CentOS it shows that it is skipped.

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.

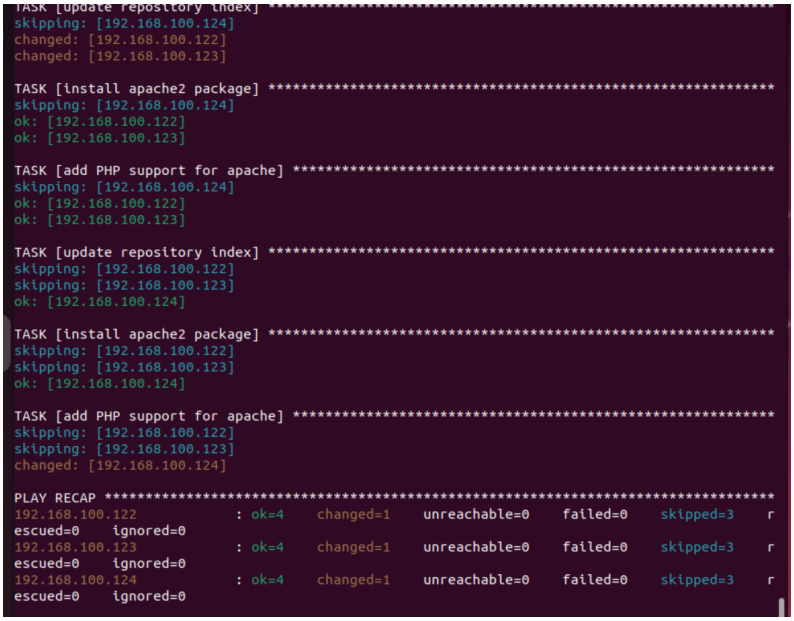
4. Edit the *install_apache.yml* file and insert the lines shown below.

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

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

  - name: install apache2 package
    apt:
      name: apache2
      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.

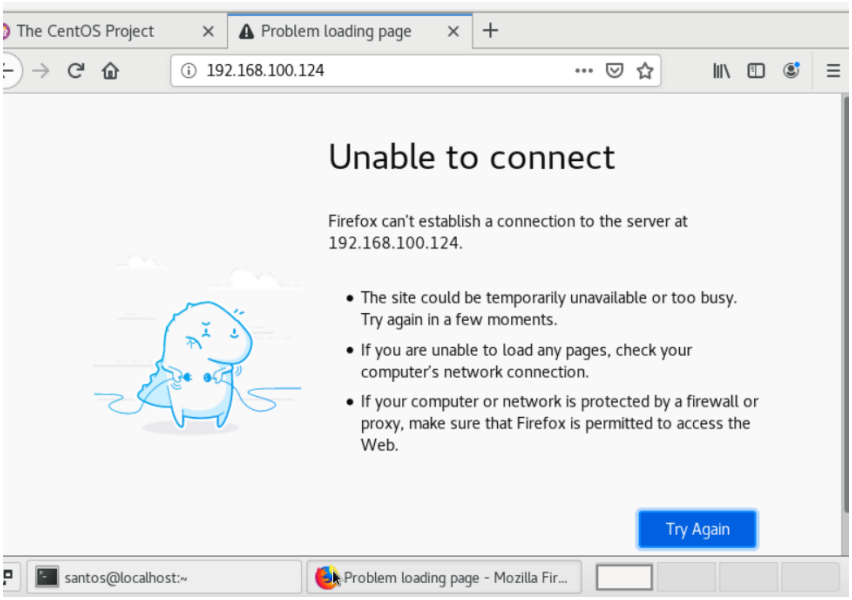
```
  GNU nano 6.2                        install_apache.
---
- hosts: all
  become: true
  tasks:

  - 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
```

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



It shows that it successfully executes in the 3 server, but it also shows that it has 3 skipped. SInce, some is for Ubuntu server and the other which uses “dnf” for 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.

```
[santos@localhost ~]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disa
bled)
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)
[santos@localhost ~]$
```

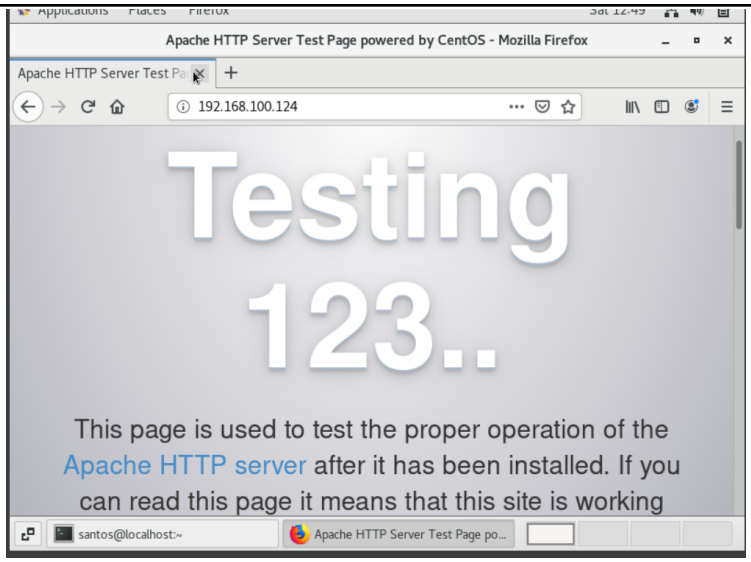
5.2 Issue the following command to start the service:
sudo systemctl start httpd
(When prompted, enter the sudo password)

```
[santos@localhost ~]$ sudo systemctl start httpd
[sudo] password for santos:
```

sudo firewall-cmd --add-port=80/tcp
(The result should be a success)

```
[santos@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)



This screenshot shows that after the activation, I successfully run the IP address.

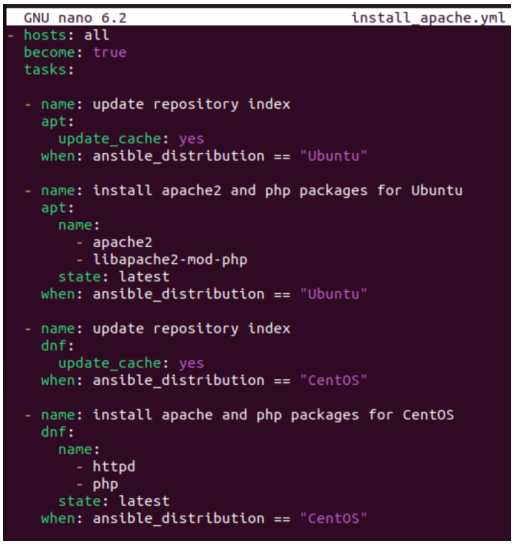
Task 2: Refactoring playbook

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.

```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [update repository index] *************************************************
skipping: [192.168.100.124]
changed: [192.168.100.122]
changed: [192.168.100.123]

TASK [install apache2 and php packages for Ubuntu] *****************************
skipping: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [update repository index] *************************************************
skipping: [192.168.100.123]
skipping: [192.168.100.122]
ok: [192.168.100.124]

TASK [install apache and php packages for CentOS] ******************************
skipping: [192.168.100.122]
skipping: [192.168.100.123]
ok: [192.168.100.124]

PLAY RECAP *********************************************************************
192.168.100.122            : ok=3    changed=1    unreachable=0    failed=0    skipped=2    r
escued=0    ignored=0
192.168.100.123            : ok=3    changed=1    unreachable=0    failed=0    skipped=2    r
escued=0    ignored=0
192.168.100.124            : ok=3    changed=0    unreachable=0    failed=0    skipped=2    r
escued=0    ignored=0
```

As we can see, it shows changes in ubuntu servers but no changes in the CenOS.

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.

```
  GNU nano 6.2                                 install_apache.yml
---
- 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"
```

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

```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass install_apache.ym
l
BECOME password:

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

TASK [Gathering Facts] *********************************************************************
ok: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [install apache2 and php packages for Ubuntu] *****************************************
skipping: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [install apache and php packages for CentOS] ******************************************
skipping: [192.168.100.122]
skipping: [192.168.100.123]
ok: [192.168.100.124]

PLAY RECAP *********************************************************************************
192.168.100.122            : ok=2    changed=0    unreachable=0    failed=0    skipped=1    r
escued=0    ignored=0
192.168.100.123            : ok=2    changed=0    unreachable=0    failed=0    skipped=1    r
escued=0    ignored=0
192.168.100.124            : ok=2    changed=0    unreachable=0    failed=0    skipped=1    r
escued=0    ignored=0
```

It shows that It successfully installed apache2 and php packages in Ubuntu server and also in CentOS. We can also see skipped in all of the server since it didn't match in some command some task in 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
```

```
 GNU nano 6.2
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 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.

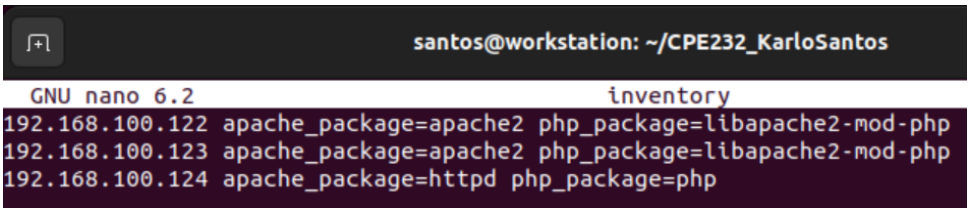
```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************************
ok: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [install apache2 and php] *************************************************************
fatal: [192.168.100.122]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined\n\nThe error appears to be in '/home/santos/CPE232_KarloSantos/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\n  - name: install apache2 and php\n    ^ here\n"}
fatal: [192.168.100.124]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined\n\nThe error appears to be in '/home/santos/CPE232_KarloSantos/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\n  - name: install apache2 and php\n    ^ here\n"}
fatal: [192.168.100.123]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined\n\nThe error appears to be in '/home/santos/CPE232_KarloSantos/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\n  - name: install apache2 and php\n    ^ here\n"}

PLAY RECAP *********************************************************************************
192.168.100.122            : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.100.123            : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.100.124            : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

We can see it fails in both Ubuntu server and CentOS.

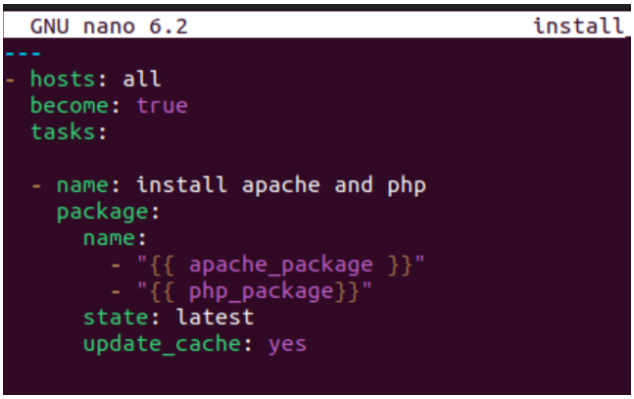
4. Unfortunately, task 2.3 was not successful. It's because we need to change something in the inventory file so that the variables we declared will be in place. Edit the *inventory* file and follow the below configuration:

```
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]



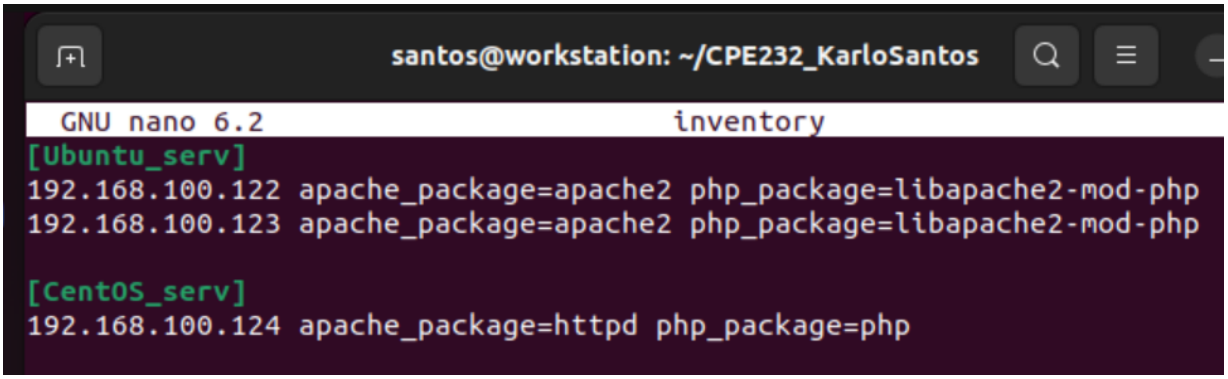
Run *ansible-playbook --ask-become-pass install_apache.yml* and describe the result.

```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************************
ok: [192.168.100.124]
ok: [192.168.100.122]
ok: [192.168.100.123]

TASK [install apache and php] **************************************************************
ok: [192.168.100.122]
ok: [192.168.100.123]
ok: [192.168.100.124]

PLAY RECAP *********************************************************************************
192.168.100.122            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    r
escued=0    ignored=0
192.168.100.123            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    r
escued=0    ignored=0
192.168.100.124            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    r
escued=0    ignored=0
```

As we can see in the results, it shows that it successfully installed all the packages in Ubuntu server and CentOS without any failure or skipped in the output.

Supplementary Activity:

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



First, I group the IP address in the inventory to distinguish which is for the Red hat OS. I put “CentOS_serv” as the group name

```
santos@workstation: ~/CPE232_KarloSantos
 GNU nano 6.2                                  cetossupp.yml
---
- hosts: CentOS_serv
  become: true
  tasks:

  - name: install apache, PHP support and Update repository
    package:
      name:
        - "{{ apache_package }}"
        - "{{ php_package}}"
      state: latest
      update_cache: yes
```

After that, I make a code that will install the apache package, give PHP support, and also update its repository index. I put the group name that I assigned earlier for CentOS in the hosts. Then I make the code to execute only in one play, maybe assigning variables in apache_package and php_package, and it is declared in the inventory to use httpd in apache and php in the php package. For the update of the repository index I used the update_cache and made it yes.

```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass cetossupp.yml
BECOME password:

PLAY [CentOS_serv] ************************************************************************

TASK [Gathering Facts] ********************************************************************
ok: [192.168.100.124]

TASK [install apache, PHP support and Update repository] **********************************
ok: [192.168.100.124]

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

Lastly, after I finalize the code I try to run the command to execute it. As shown in the screenshot above it successfully finished all the tasks in the CentOS.

Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?
   - **The refactoring of playbook codes are important in many different ways. One of these is to reduce code duplication, since having the code executed many times is not efficient and can create bugs. Another one, it**

can help in making the code easy to read and it will help in maintenance of the code in the future. In terms of error detection, refactoring also helps in detecting the different issues or even potential bugs and being able to prevent different problems from happening. All in all, refactoring the playbook gives benefits in terms of efficiency, being maintainable and debugging of the playbook so it is good to implement it.

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

   - **The “when” command is used in different tasks based on the specific conditions. It will check if that certain task is required before running it. It will help in allowing the different tasks if they will going to execute or run or it will be skipped based on the previous task, or other conditions.**