

Name: Tracey Dee Bringuela	Date Performed: 9/25/24
Course/Section: CPE31S2	Date Submitted: 9/25/24
Instructor:	Semester and SY:
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: <p>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.</p> <p>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.</p> <p>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.</p>	
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? <pre> [tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]\$ git pull Already up to date. [tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]\$ </pre>	

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."
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"
```

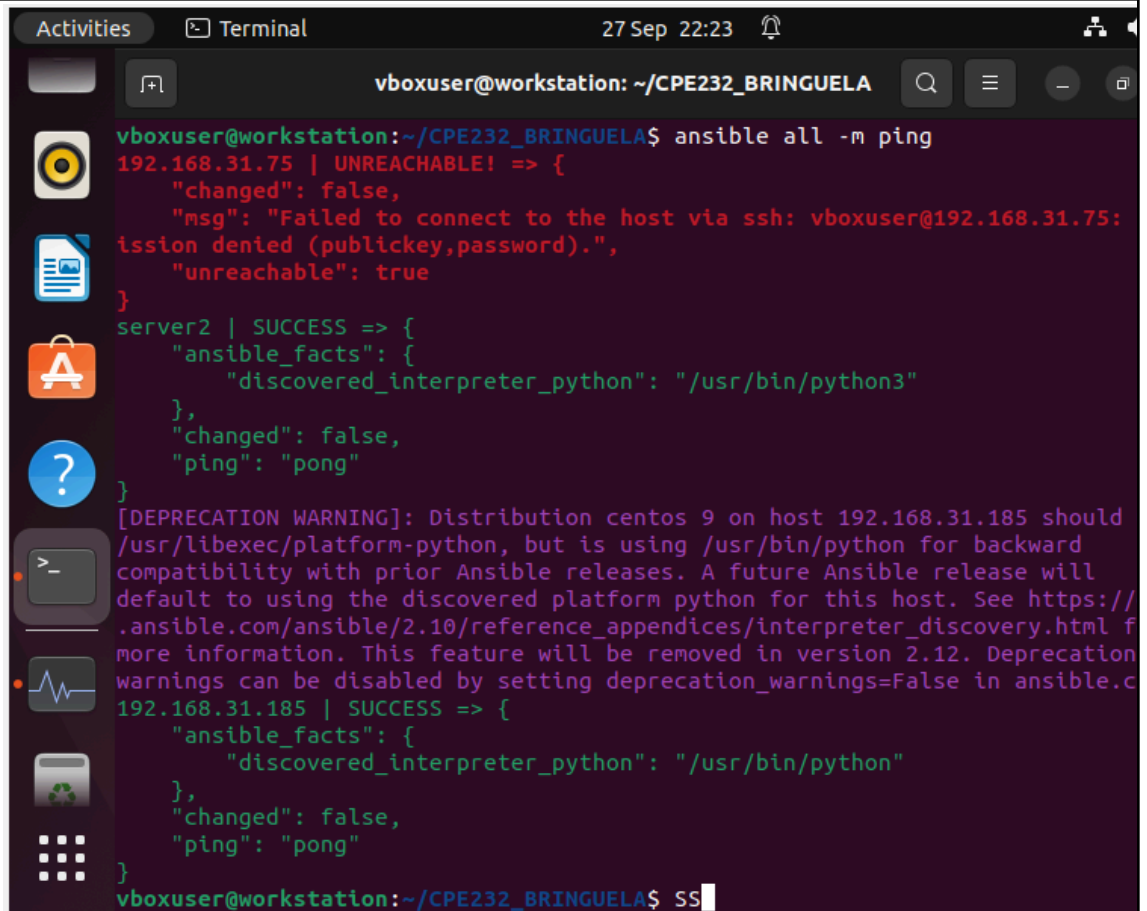
Make sure to save the file and exit.

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.

A terminal window titled 'Terminal' with a timestamp of '27 Sep 22:23'. The user is 'vboxuser@workstation' in the directory '~/CPE232_BRINGUELA'. The command 'ansible all -m ping' has been executed. The output shows that the host '192.168.31.75' is 'UNREACHABLE!' with a message: 'Failed to connect to the host via ssh: vboxuser@192.168.31.75: Permission denied (publickey,password).' The host 'server2' is 'SUCCESS' and returns 'pong'. A deprecation warning is shown for host '192.168.31.185', stating that CentOS 9 should use '/usr/libexec/platform-python' instead of '/usr/bin/python'. The terminal also shows the start of the 'ss' command.

```
vboxuser@workstation:~/CPE232_BRINGUELA$ ansible all -m ping
192.168.31.75 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: vboxuser@192.168.31.75:
ission denied (publickey,password).",
  "unreachable": true
}
server2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
[DEPRECATION WARNING]: Distribution centos 9 on host 192.168.31.185 should
/usr/libexec/platform-python, but is using /usr/bin/python for backward
compatibility with prior Ansible releases. A future Ansible release will
default to using the discovered platform python for this host. See https://
.ansible.com/ansible/2.10/reference_appendices/interpreter_discovery.html f
more information. This feature will be removed in version 2.12. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.c
192.168.31.185 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
vboxuser@workstation:~/CPE232_BRINGUELA$ ss
```

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.

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.

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

TASK [Gathering Facts] *****
*
fatal: [192.168.31.75]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: vboxuser@192.168.31.75: Permission denied (publickey,password).", "unreachable": true}
ok: [server2]
[DEPRECATION WARNING]: Distribution centos 9 on host 192.168.31.185 should use /usr/libexec/platform-python, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.10/reference_appendices/interpreter_discovery.html for more information. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.31.185]

TASK [update repository index] *****
*
skipping: [192.168.31.185]
changed: [server2]

TASK [install apache2 package] *****
*
skipping: [192.168.31.185]
ok: [server2]
```

6.

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.

```
[tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]$ systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset/enabled)
   Drop-In: /usr/lib/systemd/system/httpd.service.d
            └─php-fpm.conf
   Active: inactive (dead)
     Docs: man:httpd.service(8)
```

6.2 Issue the following command to start the service:

sudo systemctl start httpd

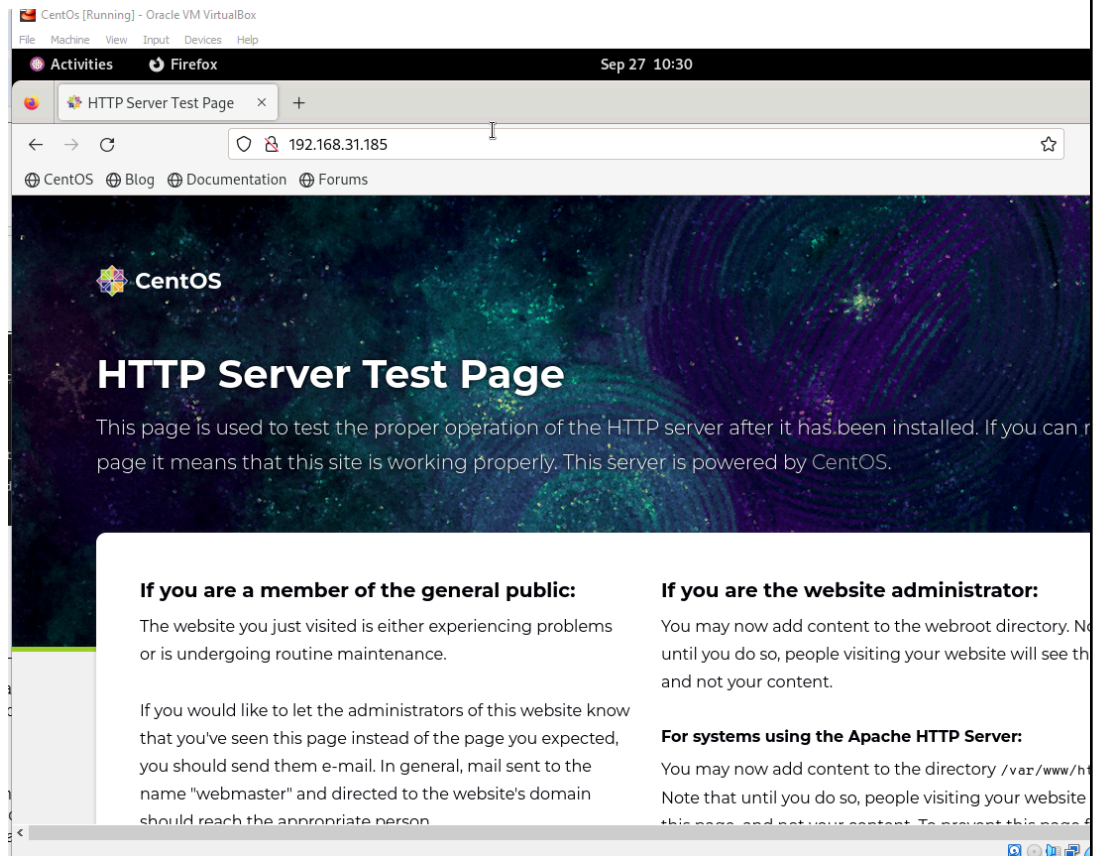
(When prompted, enter the sudo password)

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

(The result should be a success)

```
[tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]$ systemctl status httpd
o httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; pr
   Drop-In: /usr/lib/systemd/system/httpd.service.d
           └─php-fpm.conf
   Active: inactive (dead)
   Docs: man:httpd.service(8)
[tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]$ sudo systemctl start httpd
[sudo] password for tdee:
[tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]$ sudo firewall-cmd --add-port=80/
success
[tdee@MiWiFi-R4A-srv CPE232_BRINGUELA]$
```

6.3 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.

```
GNU nano 6.2                                install_apache.yml *
```

```
- name: install apache2 and php packages for ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    when: ansible_distribution == "Ubuntu"

- name: update repository index
  dnf:
    update_cache: yes
    when: ansible_distribution == "CentOS"

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

```
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

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


```

ok: [server2]
[DEPRECATION WARNING]: Distribution centos 9 on host 192.168.31.185 should
/usr/libexec/platform-python, but is using /usr/bin/python for backward
compatibility with prior Ansible releases. A future Ansible release will
default to using the discovered platform python for this host. See https:
.ansible.com/ansible/2.10/reference_appendices/interpreter_discovery.html
more information. This feature will be removed in version 2.12. Deprecati
warnings can be disabled by setting deprecation_warnings=False in ansible
ok: [192.168.31.185]

TASK [update repository index] *****
*
skipping: [192.168.31.185]
changed: [server2]

TASK [install apache2 and php packages for ubuntu] *****
*
skipping: [192.168.31.185]
ok: [server2]

TASK [update repository index] *****
*
skipping: [server2]
ok: [192.168.31.185]

TASK [install apache2 and php packages for CentOS] *****
*
skipping: [server2]

```

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 *
apt:
  update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: install apache2 and php packages for ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    state: latest
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: update repository index
  dnf:
    update_cache: yes
  when: ansible_distribution == "CentOS"

- name: install apache2 and php packages for CentOS
  dnf:
    name:
      - httpd
      - php
    state: latest
    update_cache: yes

```

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

```
*
skipping: [192.168.31.185]
changed: [server2]

TASK [install apache2 and php packages for ubuntu] *****
*
skipping: [192.168.31.185]
ok: [server2]

TASK [update repository index] *****
*
skipping: [server2]
ok: [192.168.31.185]

TASK [install apache2 and php packages for CentOS] *****
*
skipping: [server2]
ok: [192.168.31.185]

PLAY RECAP *****
*
192.168.31.185      : ok=3    changed=0    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
192.168.31.75      : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
server2           : ok=3    changed=1    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0

ubuntu@workstation: ~/CD5222-88TNCUELAS$
```

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.

```

more information. This feature will be removed in version 2.12. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.c
ok: [192.168.31.185]

TASK [install apache2 and php] *****
*
fatal: [server2]: FAILED! => {"msg": "The task includes an option with an u
ned variable. The error was: 'apache2' is undefined\n\nThe error appears t
in '/home/vboxuser/CPE232_BRINGUELA/install_apache.yml': line 6, column 5,
may\nbe elsewhere in the file depending on the exact syntax problem.\n\nTh
fending line appears to be:\n\n\n - name: install apache2 and php\n    ^ h
n"}
fatal: [192.168.31.185]: FAILED! => {"msg": "The task includes an option wi
n undefined variable. The error was: 'apache2' is undefined\n\nThe error ap
s to be in '/home/vboxuser/CPE232_BRINGUELA/install_apache.yml': line 6, co
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"}

System Monitor *****
*
192.168.31.185      : ok=1    changed=0    unreachable=0    failed=1
skipped=0    rescued=0    ignored=0
192.168.31.75      : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
server2            : ok=1    changed=0    unreachable=0    failed=1
skipped=0    rescued=0    ignored=0

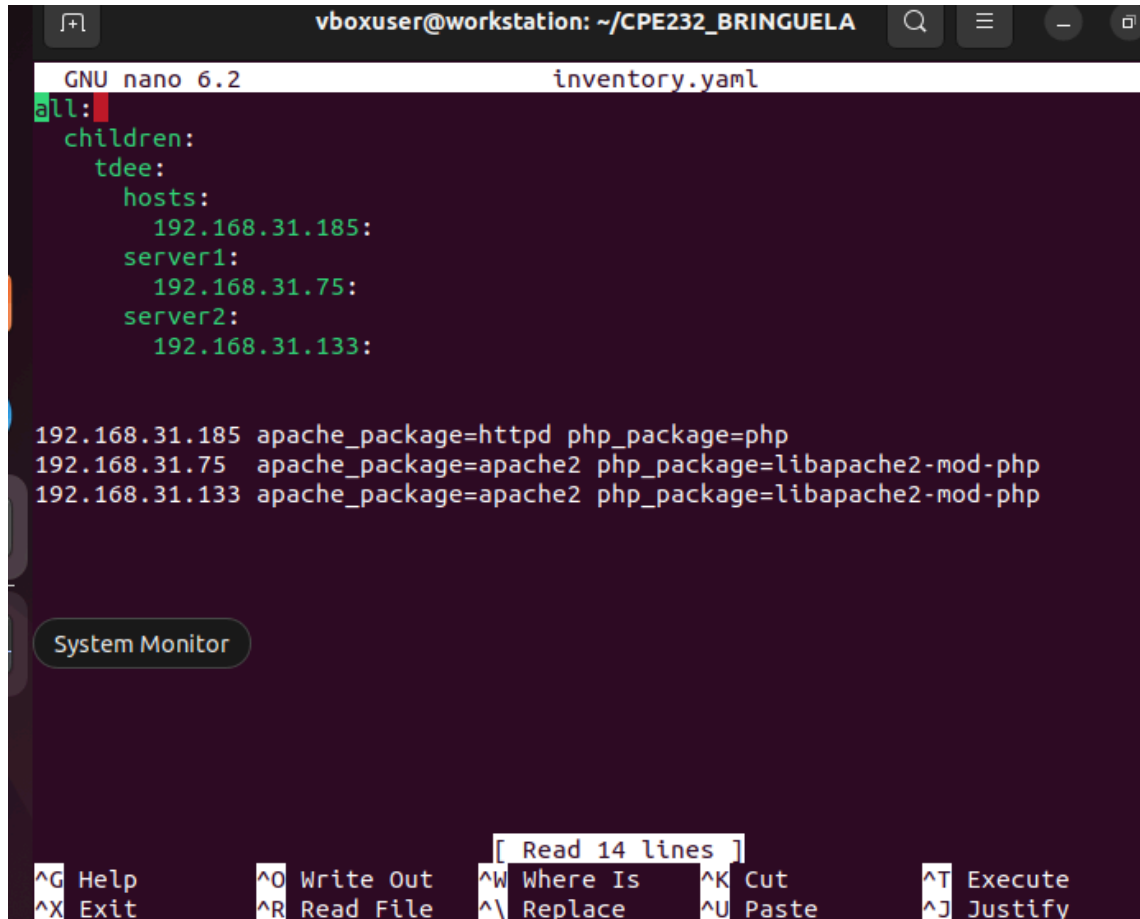
vboxuser@workstation:~/CPE232_BRINGUELA$

```

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.



```
vboxuser@workstation: ~/CPE232_BRINGUELA
GNU nano 6.2 inventory.yaml
all:
  children:
    tdee:
      hosts:
        192.168.31.185:
        server1:
        192.168.31.75:
        server2:
        192.168.31.133:

192.168.31.185 apache_package=httpd php_package=php
192.168.31.75  apache_package=apache2 php_package=libapache2-mod-php
192.168.31.133 apache_package=apache2 php_package=libapache2-mod-php

System Monitor

[ Read 14 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

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](https://docs.ansible.com/ansible/latest/builtin/package_module.html)

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

```
GNU nano 6.2                                install_apache.yml *
```

```
---
- hosts: all
  become: true
  tasks:
    - name: Install Apache and PHP
      package:
        name:
          - "{{ apache_package }}"
          - "{{ php_package }}"
        state: latest
      vars:
        apache_package: "{{ 'httpd' if ansible_os_family == 'RedHat' else 'apach"
        php_package: "{{ 'php' if ansible_os_family == 'RedHat' else 'php' }}"
```

```
OK: [server2]
[DEPRECATION WARNING]: Distribution centos 9 on host 192.168.31.185 should use
/usr/libexec/platform-python, but is using /usr/bin/python for backward
compatibility with prior Ansible releases. A future Ansible release will
default to using the discovered platform python for this host. See https://docs
.ansible.com/ansible/2.10/reference_appendices/interpreter_discovery.html for
more information. This feature will be removed in version 2.12. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.31.185]

TASK [Install Apache and PHP] *****
*
Terminal 168.31.185]
changed: [server2]

PLAY RECAP *****
*
192.168.31.185      : ok=2    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.31.75      : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
server2            : ok=2    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Supplementary Activity:

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

Activities Terminal 27 Sep 23:28

vboxuser@workstation: ~/CPE232_BRINGUELA

GNU nano 6.2 install_apache.yml

```
--
- hosts: all
  become: true
  tasks:
    - name: Ensure the EPEL repository is enabled (for RHEL/CentOS)
      yum:
        name: epel-release
        state: present
        when: ansible_os_family == "RedHat"

    - name: Install Apache and PHP
      package:
        name:
          - "{{ apache_package }}"
          - "{{ php_package }}"
        state: latest
      vars:
        apache_package: "{{ 'httpd' if ansible_os_family == 'RedHat' else ' ' }}"
        php_package: "{{ 'php' if ansible_os_family == 'RedHat' else 'php' }}"

    - name: Start and enable Apache service
      systemd:
        name: "{{ 'httpd' if ansible_os_family == 'RedHat' else 'apache2' }}"
        state: started
        enabled: yes
```

[Read 37 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify

```

skipping: [server2]
ok: [192.168.31.185]

TASK [Install Apache and PHP] *****
*
ok: [server2]
ok: [192.168.31.185]

TASK [Start and enable Apache service] *****
*
ok: [server2]
Help 192.168.31.185]

TASK [Allow HTTP service through the firewall] *****
*
fatal: [server2]: FAILED! => {"changed": false, "msg": "Python Module not found: firewalld and its python module are required for this module, version 0.2.11 or newer required (0.3.9 or newer for offline operations)"}
ok: [192.168.31.185]

PLAY RECAP *****
*
192.168.31.185      : ok=5    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.31.75      : ok=0    changed=0    unreachable=1    failed=0
skipped=0    rescued=0    ignored=0
server2           : ok=3    changed=0    unreachable=0    failed=1
skipped=1    rescued=0    ignored=0

```

Reflections:

Answer the following:

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

Refactoring playbook code is important to improve readability, maintainability, and efficiency. By consolidating tasks, you reduce redundancy, making the playbook run faster and easier to troubleshoot or expand. It also ensures that the playbook can handle diverse environments more flexibly and uniformly.

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

The "when" command in a playbook is used to conditionally execute tasks based on certain criteria. It allows you to specify when a task should run based on factors like operating system, variables, or other system conditions. This is especially useful when dealing with heterogeneous environments, ensuring tasks are only performed where applicable.

