

Name: Baltazar, Paul Eimar R.	Date Performed: Sep 25, 2024
Course/Section: CPE 212 - CPE31S2	Date Submitted: Sep 27, 2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2024-2025
Activity 5: Consolidating Playbook plays	
<p>1. Objectives:</p> <p>1.1 Use when command in playbook for different OS distributions</p> <p>1.2 Apply refactoring techniques in cleaning up the playbook codes</p>	
<p>2. Discussion:</p> <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:</p> <p>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 <i>ssh-copy-id</i> to copy the public key to CentOS. Verify if you can successfully SSH to CentOS VM.</p>	
<p>Task 1: Use when command for different distributions</p> <ol style="list-style-type: none"> 1. In the local machine, make sure you are in the local repository directory (<i>CPE232_yourname</i>). 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? 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): <i>ansible-playbook --ask-become-pass install_apache.yml</i>. 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." 	

```

aperbaltazar@workstation:~/Activity_3$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.102]
ok: [192.168.56.104]

TASK [install apache2 package] *****
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
fatal: [192.168.56.104]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory: b'apt-get'", "rc": 2, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
ok: [192.168.56.102]

TASK [install apache2 package] *****
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
ok: [192.168.56.102]

PLAY RECAP *****
192.168.56.102      : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.104      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

```

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 completed all the tasks, unlike the first attempt which has resulted in an error*

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"
```

Make sure to save and exit.

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

- Some tasks are skipped because of the `[when: ansible_distribution == "distribution"]` argument. The same tasks are later successful when the condition is met

```
TASK [Gathering Facts] *****
ok: [192.168.56.104]
ok: [192.168.56.102]

TASK [update repository index] *****
skipping: [192.168.56.104]
changed: [192.168.56.102]

TASK [install apache2 package] *****
skipping: [192.168.56.104]
ok: [192.168.56.102]

TASK [install apache2 package] *****
skipping: [192.168.56.104]
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
skipping: [192.168.56.104]
ok: [192.168.56.102]

TASK [update repository index] *****
skipping: [192.168.56.102]
ok: [192.168.56.104]

TASK [install apache2 package] *****
skipping: [192.168.56.102]
changed: [192.168.56.104]

TASK [add PHP support for apache] *****
skipping: [192.168.56.102]
changed: [192.168.56.104]

PLAY RECAP *****
192.168.56.102      : ok=5    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
192.168.56.104      : ok=4    changed=2    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
```

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:

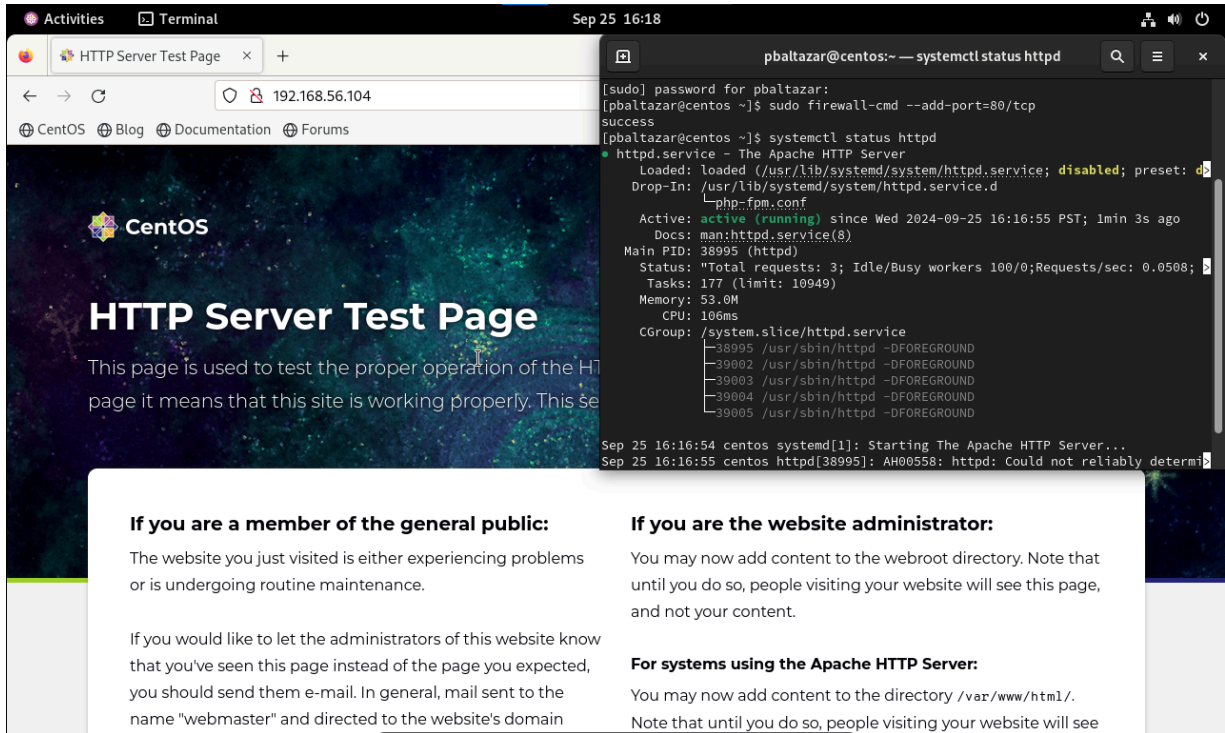
sudo systemctl start httpd

(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)



The screenshot shows a CentOS VM interface. On the left, a web browser window displays the 'HTTP Server Test Page' at IP 192.168.56.104. The page has a dark background with the CentOS logo and text: 'HTTP Server Test Page', 'This page is used to test the proper operation of the HTTP server.', and 'page it means that this site is working properly. This se...'. Below the browser, there are two columns of text. On the right, a terminal window titled 'pbaltazar@centos:~ -- systemctl status httpd' shows the command 'systemctl status httpd' and its output. The output indicates that the httpd service is 'active (running)' and lists its loaded files, drop-in files, and active processes. The terminal also shows the command 'sudo firewall-cmd --add-port=80/tcp' and its successful execution.

If you are a member of the general public:

The website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain

If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

For systems using the Apache HTTP Server:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see

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 make it 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.

– *The playbook log has become shorter*

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.

– *Just like the previous edit, the log has become shorter*

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.

– It has resulted in an error

```

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.104]
ok: [192.168.56.102]

TASK [install apache2 package and php] *****
fatal: [192.168.56.102]: 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/qperbaltazar/Activity_5/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  - name: install apache2 package and php\n    ^ here\n"]
fatal: [192.168.56.104]: 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/qperbaltazar/Activity_5/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  - name: install apache2 package and php\n    ^ here\n"]

```

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

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

– consolidating the commands has resulted in a shorter and much cleaner playbook log.

```
PLAY [all] *****
TASK [Gathering Facts] *****
ok: [192.168.56.102]
ok: [192.168.56.104]
TASK [install apache2 package and php] *****
ok: [192.168.56.104]
ok: [192.168.56.102]
PLAY RECAP *****
192.168.56.102      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.56.104      : ok=2    changed=0    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.

Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?
 - Refactoring of playbook codes is important because it makes ansible more efficient in running tasks. As shown in the images, there are many tasks, including checking the distribution on the managed nodes. When you simplify the code, you'll notice the playbook runs faster compared to a longer, more complex version.
2. When do we use the “when” command in playbook?
 - The “when” command is used in playbook when there are multiple managed nodes that run on different distributions. Its function is to check whether or not the indicated distribution matches with the current managed node since there are variations in commands between different distributions.