

Name: Pacinos, Angela Monique A.	Date Performed: 09-18-23
Course/Section: CPE232 - CPE31S4	Date Submitted: 09-19-23
Instructor: Dr. Jonathan V. Taylor	Semester and SY: 1st Sem / '23 - '24

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?

```
angela@workstation:~/HOA5_S4$ git pull
Username for 'https://github.com': Angela-Pacinos
Password for 'https://Angela-Pacinos@github.com':
Already up to date.
```

It updated the HOA_S5 repository with all the files from the github.

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."

```
angela@workstation: ~/HOA5_S4
File Edit View Search Terminal Tabs Help
angela@wor... x angela@wor... x angela@wor... x angela@wor... x
GNU nano 2.9.3 inventory Modified
192.168.56.114 ansible_python_interpreter=/usr/bin/python3
#192.168.56.115 ansible_python_interpreter=/usr/bin/python3
192.168.56.116 ansible_python_interpreter=/usr/bin/python

angela@workstation:~/HOA5_S4$ sudo nano install_apache.yml
angela@workstation:~/HOA5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.116]
ok: [192.168.56.114]

TASK [update repository index] *****
fatal: [192.168.56.116]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory", "rc": 2}
changed: [192.168.56.114]

TASK [install apache2 package] *****
ok: [192.168.56.114]

TASK [add PHP support for apache] *****
ok: [192.168.56.114]
to retry, use: --limit @/home/angela/HOA5_S4/install_apache.retry

PLAY RECAP *****
192.168.56.114      : ok=4    changed=1    unreachable=0    failed=0
192.168.56.116      : ok=1    changed=0    unreachable=0    failed=1
```

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.

```

angela@workstation:~/HOA5_S4$ sudo nano install_apache.yml
angela@workstation:~/HOA5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [update repository index] *****
skipping: [192.168.56.116]
changed: [192.168.56.114]

TASK [install apache2 package] *****
skipping: [192.168.56.116]
ok: [192.168.56.114]

TASK [add PHP support for apache] *****
skipping: [192.168.56.116]
ok: [192.168.56.114]

PLAY RECAP *****
192.168.56.114      : ok=4    changed=1    unreachable=0    failed=0
192.168.56.116      : ok=1    changed=0    unreachable=0    failed=0

```

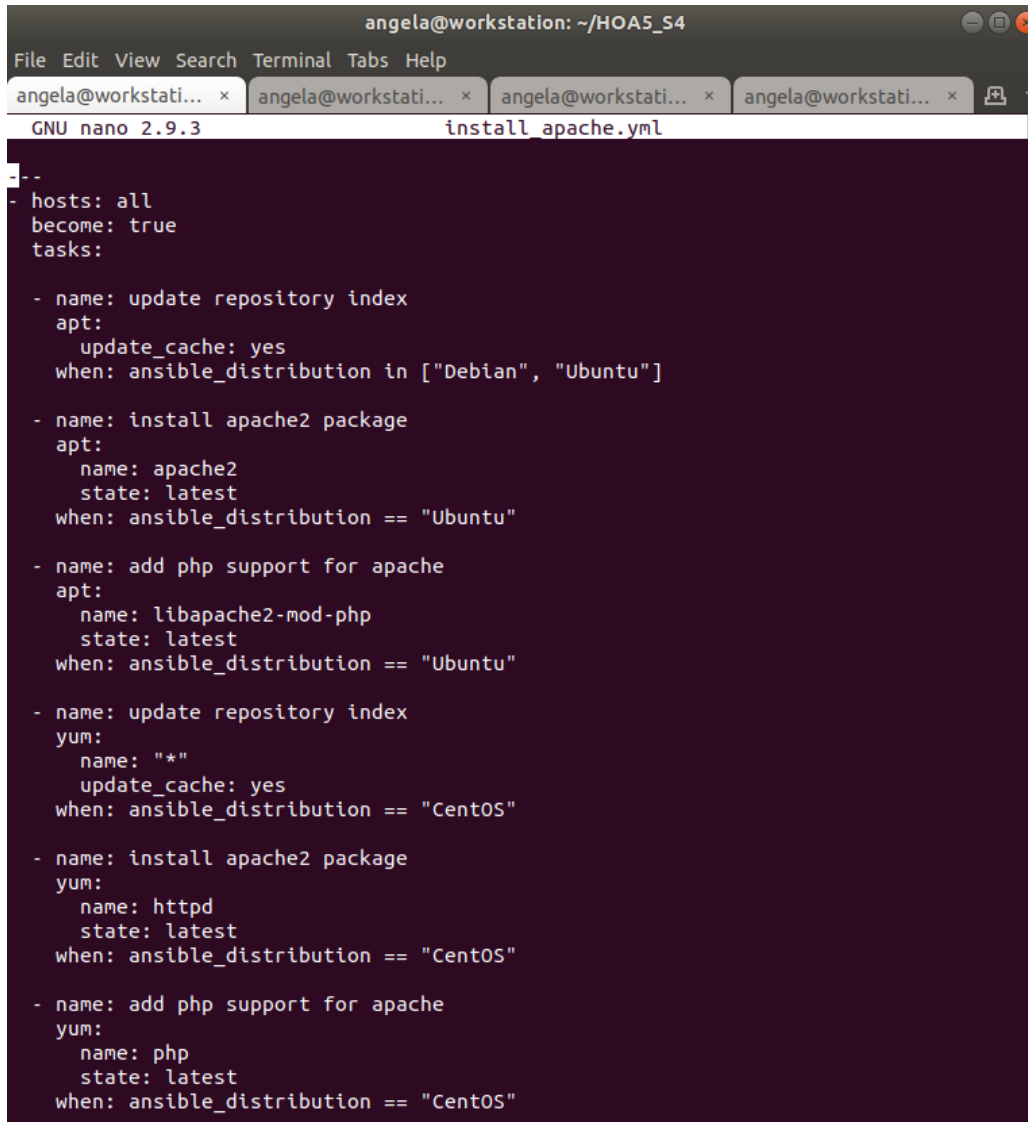
The server1 was working properly because it specified the 'when' and put Ubuntu to it.

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.



```
angela@workstation: ~/HOA5_S4
File Edit View Search Terminal Tabs Help
angela@workstati... x angela@workstati... x angela@workstati... x angela@workstati... x
GNU nano 2.9.3 install_apache.yml

--
- hosts: all
  become: true
  tasks:

    - name: update repository index
      apt:
        update_cache: yes
        when: ansible_distribution in ["Debian", "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
      yum:
        name: "*"
        update_cache: yes
        when: ansible_distribution == "CentOS"

    - name: install apache2 package
      yum:
        name: httpd
        state: latest
        when: ansible_distribution == "CentOS"

    - name: add php support for apache
      yum:
        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.

```
angela@workstation:~/H0A5_S4$ sudo nano install_apache.yml
angela@workstation:~/H0A5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [update repository index] *****
skipping: [192.168.56.116]
changed: [192.168.56.114]

TASK [install apache2 package] *****
skipping: [192.168.56.116]
ok: [192.168.56.114]

TASK [add php support for apache] *****
skipping: [192.168.56.116]
ok: [192.168.56.114]

TASK [update repository index] *****
skipping: [192.168.56.114]
ok: [192.168.56.116]

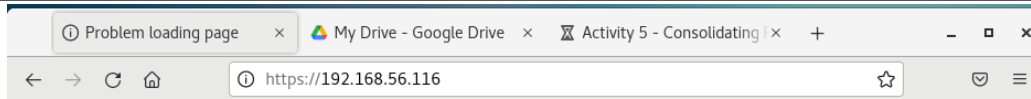
TASK [install apache2 package] *****
skipping: [192.168.56.114]
changed: [192.168.56.116]

TASK [add php support for apache] *****
skipping: [192.168.56.114]
changed: [192.168.56.116]

PLAY RECAP *****
192.168.56.114      : ok=4    changed=1    unreachable=0    failed=0
192.168.56.116      : ok=4    changed=2    unreachable=0    failed=0
```

The CentOS was also working properly now that we have modified and add the 'when' = CestOS. I also modified some parts so that both station is working properly.

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.



Unable to connect

An error occurred during a connection to 192.168.56.116.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again

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.

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

5.2 Issue the following command to start the service:

sudo systemctl start httpd

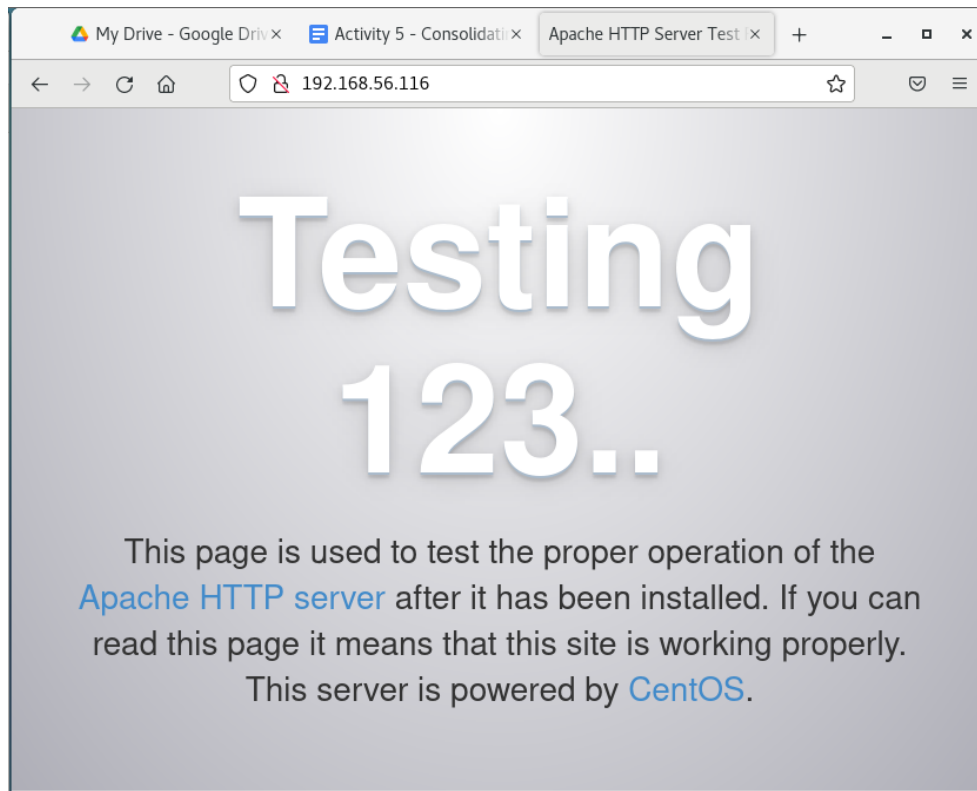
(When prompted, enter the sudo password)

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

(The result should be a success)

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



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.


```

angela@workstation:~/H0A5_S4$ sudo nano install_apache.yml
angela@workstation:~/H0A5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

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

TASK [Gathering Facts] *****
****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [update repository index Ubuntu] *****
****
skipping: [192.168.56.116]
changed: [192.168.56.114]

TASK [install apache2 and php packages for Ubuntu] *****
****
skipping: [192.168.56.116]
ok: [192.168.56.114]

TASK [update repository index CentOS] *****
****
skipping: [192.168.56.114]
ok: [192.168.56.116]

TASK [install apache and php packages for CentOS] *****
****
skipping: [192.168.56.114]
ok: [192.168.56.116]

PLAY RECAP *****
****
192.168.56.114      : ok=3    changed=1    unreachable=0    failed=0
192.168.56.116      : ok=3    changed=0    unreachable=0    failed=0

```

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.

```

angela@workstation:~/H0A5_S4$ sudo nano install_apache.yml
angela@workstation:~/H0A5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

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

TASK [Gathering Facts] *****
****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [install apache2 and php packages for Ubuntu] *****
****
skipping: [192.168.56.116]
ok: [192.168.56.114]

TASK [install apache and php packages for CentOS] *****
****
skipping: [192.168.56.114]
ok: [192.168.56.116]

PLAY RECAP *****
****
192.168.56.114      : ok=2    changed=0    unreachable=0    failed=0
192.168.56.116      : ok=2    changed=0    unreachable=0    failed=0

```

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.

```
angela@workstation:~/H0A5_S4$ sudo nano install_apache.yml
angela@workstation:~/H0A5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

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

TASK [Gathering Facts] *****
****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [install apache and php] *****
****
fatal: [192.168.56.114]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined\n\nThe error appears to have been in '/home/angela/H0A5_S4/install_apache.yml': line 79, 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 apache and php\n    ^ here\n"}
fatal: [192.168.56.116]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'apache_package' is undefined\n\nThe error appears to have been in '/home/angela/H0A5_S4/install_apache.yml': line 79, 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 apache and php\n    ^ here\n"}
      to retry, use: --limit @/home/angela/H0A5_S4/install_apache.retry

PLAY RECAP *****
****
192.168.56.114          : ok=1    changed=0    unreachable=0    failed=1
192.168.56.116          : ok=1    changed=0    unreachable=0    failed=1
```

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

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

```
angela@workstation:~/H0A5_S4$ sudo nano install_apache.yml
angela@workstation:~/H0A5_S4$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.114]
ok: [192.168.56.116]

TASK [install apache and php] *****
ok: [192.168.56.114]
ok: [192.168.56.116]

PLAY RECAP *****
192.168.56.114      : ok=2    changed=0    unreachable=0    failed=0
192.168.56.116      : ok=2    changed=0    unreachable=0    failed=0
```

Reflections:

Answer the following:

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

Refactoring a playbook depends on the output that we want to achieve. It can also depend on the type of stations that we are adding to our workstation. We refactor a playbook so that we can simplify, organize or add more contents to it. We also do this to update the contents of our playbook to have a better performance and can output the desired results.

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

We use the command when it depends on the situation that we encounter. For this activity we used it as a conditional command. Since we are using two different types of OS there can be connection issues if not used by any conditional commands. For this activity we used it to install the ansible based on the ansible distribution- Ubuntu or CentOS.

Conclusion:

For this activity it is connected to the last one which is also about ansible. The difference is that for this one we are using an ubuntu as well as CentOS. For this activity we used a new command which is the ‘when’ command. This command is used as a conditional command for our two different OS. For the playbook to recognize the OS and connect in the playbook. We also did refactoring on the playbook. We refactor or modify our playbook often to better, update, and optimize the playbook. We used this for the installation of the apache and php in both Ubuntu and CentOS. Overall, this activity was okay and allowed us to be more familiar with Ansible and the playbook. Although there are times that I get confused with the steps, reading the guides helped with the problem. I also often had errors every time the script inside the playbook all have correct spellings, indentation is correct, as well as spaces and the commands itself.

"I affirm that I will not give or receive any unauthorized help on this activity and that all work will be my own."