

<b>Name: Kevin Roi A. Sumaya</b>	<b>Date Performed: September 18 2023</b>
<b>Course/Section: CPE232 31S6</b>	<b>Date Submitted: September 18 2023</b>
<b>Instructor: Engr. Jonathan Taylar</b>	<b>Semester and SY: 2023-2024</b>
<b>Activity 5: Consolidating Playbook plays</b>	
<b>1. Objectives:</b> 1.1 Use <b>when</b> command in playbook for different OS distributions 1.2 Apply refactoring techniques in cleaning up the playbook codes	
<b>2. Discussion:</b>  <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><b>Requirement:</b>  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 <b>ssh-copy-id</b> to copy the public key to CentOS. Verify if you can successfully SSH to CentOS VM.</p>	
<b>Task 1: Use when command for different distributions</b>  1. In the local machine, make sure you are in the local repository directory ( <b>CPE232_yourname</b> ). 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?	

```

kevin@Workstation:~$ cd sysads6
kevin@Workstation:~/sysads6$ ls
ansible.cfg      inventory        main.py         sysads6
install_apache.yml  inventory.pyml  README.md
kevin@Workstation:~/sysads6$ gitpull
gitpull: command not found
kevin@Workstation:~/sysads6$ git pull
Already up to date.
kevin@Workstation:~/sysads6$ █

```

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."

```

vin@192.168.56.110: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,
assword).", "unreachable": true}

TASK [update repository index] *****
*
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] *****
*
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
*
ok: [192.168.56.103]
ok: [192.168.56.102]

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

```

TASK [update repository index] *****
*
skipping: [sumaya@192.168.56.110]
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

PLAY RECAP *****
*
192.168.56.102      : ok=4    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=4    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
sumaya@192.168.56.110 : ok=1    changed=0    unreachable=0    failed=0
skipped=3    rescued=0    ignored=0

kevin@Workstation:~/sysads6$

```

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.

```
Kevin@Workstation: ~/sysad50
File Edit View Search Terminal Help
*
ok: [192.168.56.102]
ok: [sumaya@192.168.56.110]
ok: [192.168.56.103]

TASK [update repository index] *****
*
skipping: [sumaya@192.168.56.110]
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [add PHP support for apache] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [update repository index] *****
*
skipping: [192.168.56.102]
skipping: [192.168.56.103]
```

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*

File Edit View Search Terminal Help

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

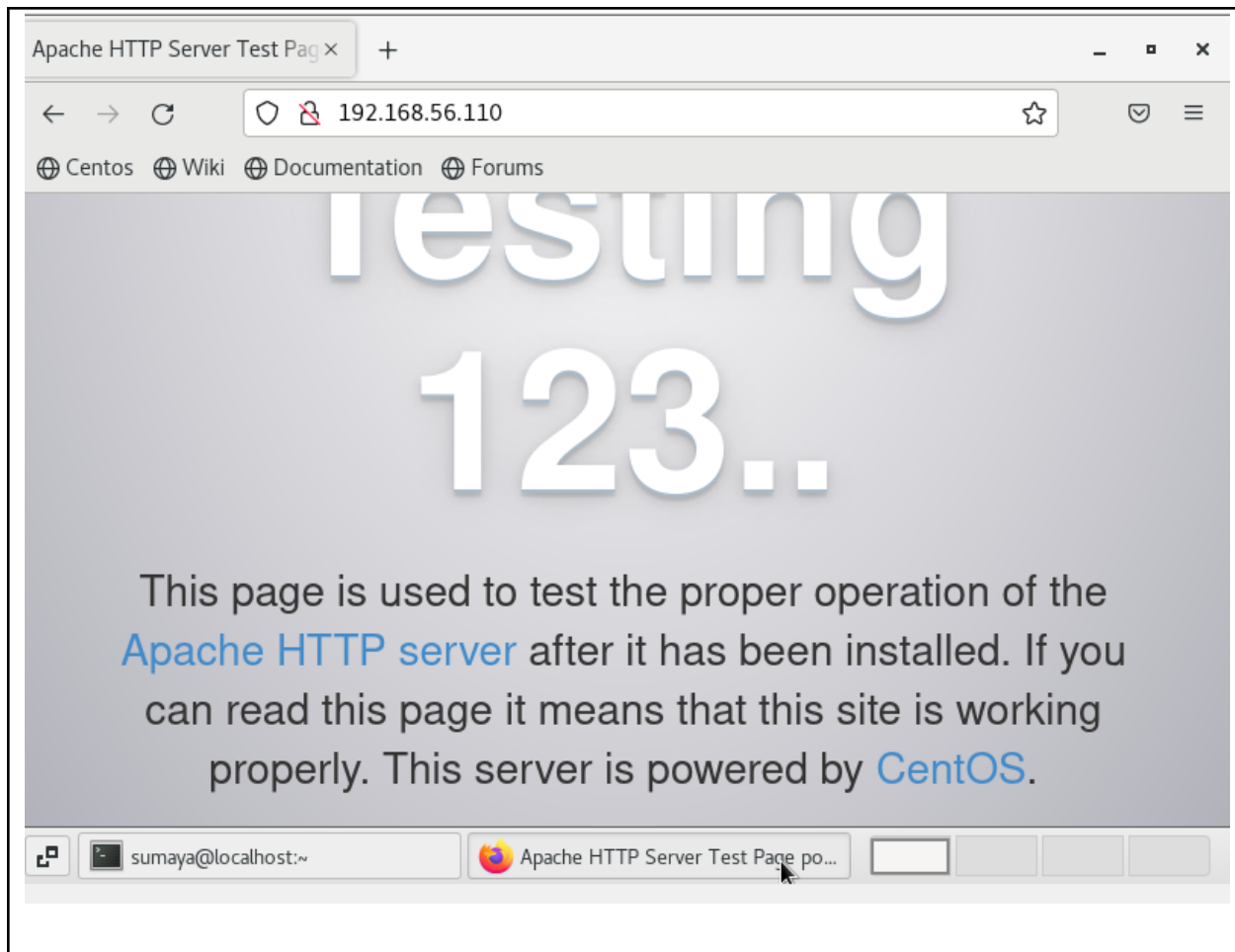
(When prompted, enter the sudo password)

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

(The result should be a success)

```
[sudo] password for sumaya:
[sumaya@localhost ~]$ sudo firewall-cmd --add-port=80/tcp
success
[sumaya@localhost ~]$ █
```

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.

```

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [sumaya@192.168.56.110]
ok: [192.168.56.103]

TASK [update repository index] *****
*
skipping: [sumaya@192.168.56.110]
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package and php packages for Ubuntu] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [update repository index for CentOS] *****
*
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [sumaya@192.168.56.110]

```

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.

```

TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [sumaya@192.168.56.110]
ok: [192.168.56.103]

TASK [install apache2 package and php packages for Ubuntu] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [install apache package and php packages for CentOS] *****
*
skipping: [192.168.56.102]
skipping: [192.168.56.103]
ok: [sumaya@192.168.56.110]

PLAY RECAP *****
*
192.168.56.102      : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
192.168.56.103      : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
sumaya@192.168.56.110 : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

```

- 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.

```
TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]
ok: [sumaya@192.168.56.110]

TASK [install apache package and php] *****
*
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
fatal: [sumaya@192.168.56.110]: FAILED! => {"changed": false, "cmd": "apt-get u
pdate", "msg": "[Errno 2] No such file or directory: b'apt-get': b'apt-get'", "
rc": 2, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
ok: [192.168.56.103]
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
192.168.56.103      : ok=2    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
sumaya@192.168.56.110 : ok=1    changed=0    unreachable=0    failed=1
skipped=0    rescued=0    ignored=0
```

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.

```
GNU nano 2.9.3 inventory

192.168.56.102 ansible_python_interpreter=/usr/bin/python3
192.168.56.102 apache_package=apache2 php_package=libapache2-mod-php

192.168.56.103 ansible_python_interpreter=/usr/bin/python3
192.168.56.103 apache_package=apache2 php_package=libapache2-mod-php

192.168.56.110 ansible_python_interpreter=/usr/bin/python3
192.168.56.110 apache_package=httpd php_package=php
# you can put other ip address below
```

**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.

```
File Edit View Search Terminal Help
TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]
ok: [sumaya@192.168.56.110]

TASK [install apache2 package and php packages for Ubuntu] *****
*
skipping: [sumaya@192.168.56.110]
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [install apache and php packages for CentOS] *****
*
skipping: [192.168.56.102]
skipping: [192.168.56.103]
changed: [sumaya@192.168.56.110]

PLAY RECAP *****
*
192.168.56.102      : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
192.168.56.103      : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
sumaya@192.168.56.110 : ok=2    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

kevin@Workstation:~/sysads6$ S
```

### Reflections:

Answer the following:

1. Why do you think refactoring of playbook codes is important?
  - Refactoring playbook code is a proactive approach that leads to a cleaner, more maintainable, and efficient codebase, ultimately improving the effectiveness of the playbook. This will make the playbook faster and cleaner to run.

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

- We use when commands in the playbook to give us the flexibility to execute different tasks, making our playbook more dynamic and adaptable to various scenarios and conditions within your environment. when command can be use on task execution, variable based condition, combining conditions, and using ansible facts