

Name: Atienza, Stephen Lhay C.	Date Performed:02/16/24
Course/Section:CPE231S1	Date Submitted:02/20/24
Instructor: Dr. Jonathan V. Taylar	Semester and SY: 2nd and 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: <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>stephen@workstation:~/CPE232_SLAtienza\$ git pull Already up to date.</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."

```
[DEPRECATION WARNING]: Ansible will require Python 3.6 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:40:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.111]
ok: [192.168.56.109]

TASK [update repository index] *****
[WARNING]: Updating cache and auto-installing missing dependency: python-apt
fatal: [192.168.56.111]: FAILED! => ('changed': false, 'cmd': 'apt-get update', 'msg': "[Errno 2] No such file or directory", 'rc': 2, 'stderr': '', 'stderr_lines': [], 'stdout_lines': [])
changed: [192.168.56.109]

TASK [install apache2 package] *****
ok: [192.168.56.109]

TASK [add PHP support for apache] *****
ok: [192.168.56.109]

PLAY RECAP *****
192.168.56.109 : ok=4 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
192.168.56.111 : 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.

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.

```

[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.109]
ok: [192.168.56.111]

TASK [update repository index] *****
skipping: [192.168.56.111]
changed: [192.168.56.109]

TASK [install apache2 package] *****
skipping: [192.168.56.111]
ok: [192.168.56.109]

TASK [add PHP support for apache] *****
skipping: [192.168.56.111]
ok: [192.168.56.109]

PLAY RECAP *****
192.168.56.109      : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    t
192.168.56.111      : ok=1    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    t

```

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.

```

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor
  t: 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

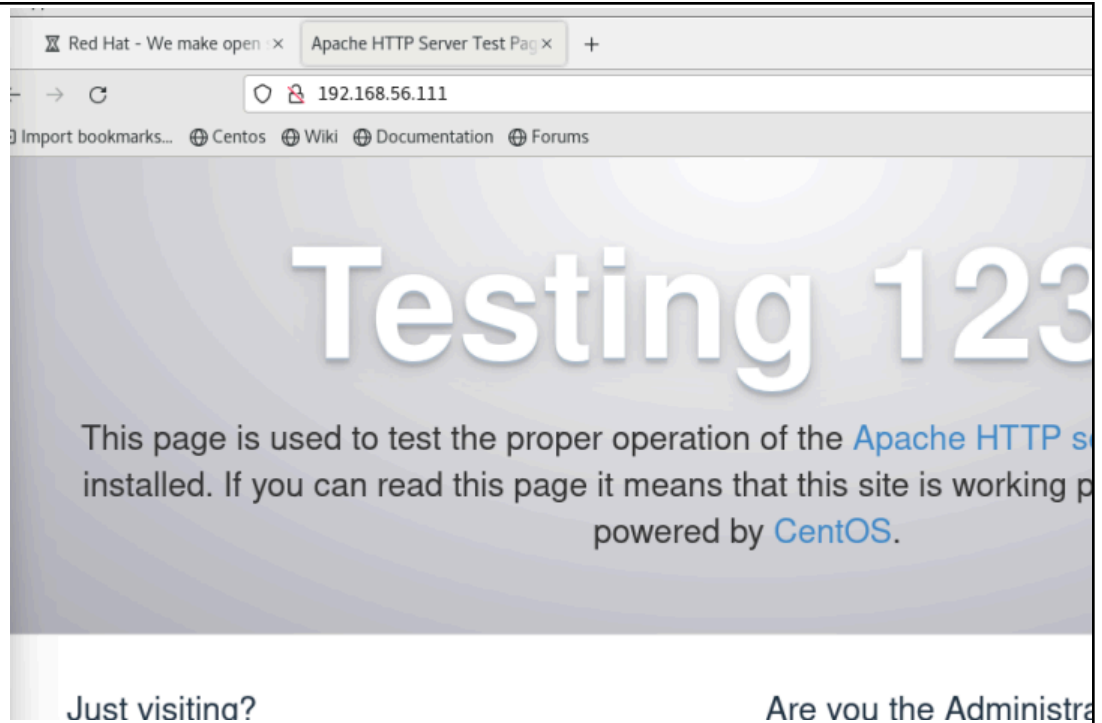
(The result should be a success)

```

i@localhost ~]$ sudo firewall-cmd --add-port=80/tcp
i@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 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.

```

DEPRECATION WARNING: Ansible will require Python 3.8 or newer on the Controller starting with Ansible 2.12. Current version: 3.8.9 (default, Mar 10 2023, 16:48:08) [GCC 8.4.0]. This feature will be removed
in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.109]
ok: [192.168.56.111]

TASK [update repository index Ubuntu] *****
skipping: [192.168.56.111]
changed: [192.168.56.109]

TASK [install apache2 and php packages for Ubuntu] *****
skipping: [192.168.56.111]
ok: [192.168.56.109]

TASK [update repository index for CentOS] *****
skipping: [192.168.56.109]
ok: [192.168.56.111]

TASK [install apache and php packages for CentOS] *****
skipping: [192.168.56.109]
ok: [192.168.56.111]

PLAY RECAP *****
192.168.56.109      : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
192.168.56.111      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=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.


```

- 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
    update_cache: yes
    when: ansible_distribution == "Ubuntu"

# - name: update repository index for CentOS
#   yum:
#     update_cache: yes
#   when: ansible_distribution == "CentOS"

- name: install apache and php packages for CentOS
  yum:
    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.

- 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

```

by setting deprecation_warnings=False in ansible.cfg.
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.109]
ok: [192.168.56.111]

TASK [install apache and php] *****
fatal: [192.168.56.109]: 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/eyssiii/OAS_54/install_apache.yml': line 77, column 6, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n  - name: install apache and php\n    ^ here\n")
fatal: [192.168.56.111]: 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/eyssiii/OAS_54/install_apache.yml': line 77, column 6, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n  - name: install apache and php\n    ^ here\n")

PLAY RECAP *****
192.168.56.109      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
192.168.56.111      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

```

- 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.

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?
 - The importance of refactoring playbook code cannot be overstated for a few compelling reasons. The primary benefit is an increase in the codebase's readability and maintainability, which in turn makes the code simpler to grasp, apply, and enhance. This is of utmost importance in the ever-changing world of information technology, where the exchange of data and assistance with problems is frequently required. Along with this, refactoring is a key step in code optimization since it finds and removes superfluous and repetitive code, which speeds up the program and reduces its resource usage. Consistent application of the automated code is another benefit, as is the enforcement of coding standards and good coding practices. Generally speaking, refactoring gives IT teams the tools they need to effectively regulate complex systems, which allows them to adapt to changing technology trends and flourish.
2. When do we use the “when” command in playbook?
 - According to automated work flows, or playbooks, this command is essential for conditionally performing activities or plays, since it makes them more flexible and accurate. Particularly helpful when you have to setup or execute anything depending on many criteria, such the kind of operating system, the version of the OS, or environment variables with custom values. Optimizing playbook execution, minimizing unnecessary activities, and making automation more adaptive according to varied infrastructures may be achieved by carefully employing "when" clauses. This will result in Ansible playbooks that are more durable and robust.

Conclusions:

- In conclusion, restructuring playbook code and accomplishing goals with the "when" command for diverse OS distributions are significant accomplishments towards Ansible Automation. Playbooks may be made to adapt to varied settings with the help of conditional execution on OSType. This gives our automation the flexibility and accuracy it needs. Simultaneously, refactoring enhances program readability and maintainability while optimizing execution time, leading to better performance and greater resource conservation. Our capacity to handle the complexity of IT infrastructure is enhanced by these aims, which allow us to design Ansible playbooks that are more efficient, adaptable, and succinct.