

Name: Bacong, El Cid A.	Date Performed: 9/5/2025
Course/Section: CPE212 - CPE31S4	Date Submitted: 9/5/2025
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2025-2026
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 <ol style="list-style-type: none"> 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? 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.

```
GNU nano 2.9.3                                install_apache.yml
---
- 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"
```

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

```
cidee@workstation:~/CPE212$ ansible-playbook --ask-become-pass install_apache.y
ml
SUDO password:

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [192.168.56.105]

TASK [update repository index] *****
*
changed: [192.168.56.105]

TASK [install apache2 package] *****
*
ok: [192.168.56.105]

TASK [add PHP support for apache] *****
*
ok: [192.168.56.105]

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

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.

```
idee@workstation:~/CPE212$ ansible-playbook --ask-become-pass install_apache.yml
SUDO password:
```

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

```
TASK [Gathering Facts] *****
*****
```

```
ok: [192.168.56.105]
```

```
TASK [update repository index] *****
*****
```

```
changed: [192.168.56.105]
```

```
TASK [install apache2 package] *****
*****
```

```
ok: [192.168.56.105]
```

```
TASK [add PHP support for apache] *****
*****
```

```
ok: [192.168.56.105]
```

```
TASK [add PHP support for apache] *****
*****
```

```
ok: [192.168.56.105]
```

```
TASK [update repository index] *****
*****
```

```
changed: [192.168.56.105]
```

```
TASK [install apache2 package] *****
*****
```

```
ok: [192.168.56.105]
```

```
TASK [add PHP support for apache] *****
*****
```

```
ok: [192.168.56.105]
```

```
PLAY RECAP *****
*****
```

```
192.168.56.105 : ok=7    changed=2    unreachable=0    failed=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.

```
[cidee@vbox ~]$ systemctl status httpd
o httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; pres
   Active: inactive (dead)
   Docs: man:httpd.service(8)
lines 1-4/4 (END)
```

5.2 Issue the following command to start the service:

sudo systemctl start httpd

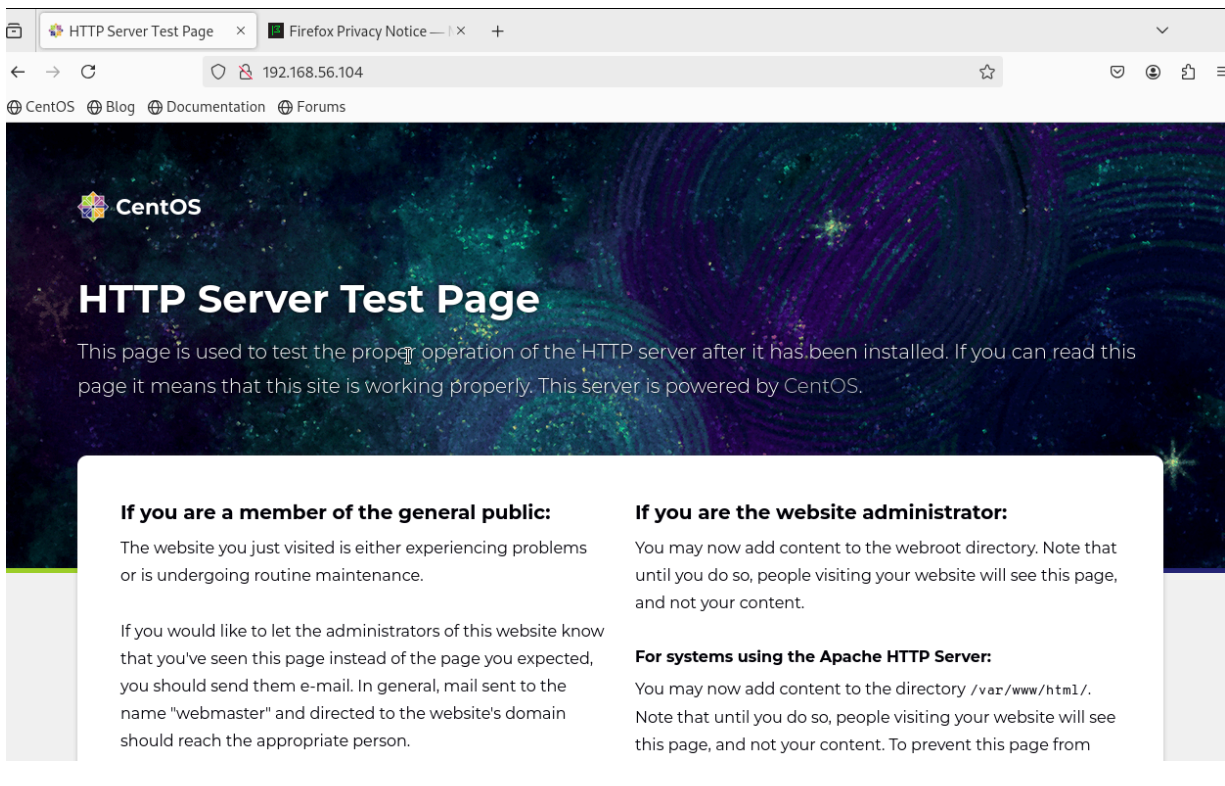
(When prompted, enter the sudo password)

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

(The result should be a success)

```
[cidee@vbox ~]$ sudo systemctl start httpd
[cidee@vbox ~]$ 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.


```

cidee@workstation:~/CPE212_elcid$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

TASK [update repository index Ubuntu] *****
skipping: [192.168.56.104]
changed: [192.168.56.103]

TASK [install apache2 and php packages for Ubuntu] *****
skipping: [192.168.56.104]
ok: [192.168.56.103]

TASK [update repository index CentOS] *****
skipping: [192.168.56.103]
ok: [192.168.56.104]

TASK [install apache2 and php packages for CentOS] *****
skipping: [192.168.56.103]
changed: [192.168.56.104]

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

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

```

cidee@workstation:~/CPE212_elcid$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

TASK [install apache2 and php packages for Ubuntu] *****
skipping: [192.168.56.104]
ok: [192.168.56.103]

TASK [install apache2 and php packages for CentOS] *****
skipping: [192.168.56.103]
ok: [192.168.56.104]

PLAY RECAP *****
192.168.56.103      : ok=2    changed=0    unreachable=0    failed=0    skipped=
1    rescued=0    ignored=0
192.168.56.104      : ok=2    changed=0    unreachable=0    failed=0    skipped=
1    rescued=0    ignored=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.

```
cidee@workstation:~/CPE212_elcid$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

TASK [install apache2 and php] *****
fatal: [192.168.56.103]: FAILED! => {"msg": "The task includes an option with an unde-
fined variable.. 'apache_package' is undefined\n\nThe error appears to be in '/home/cidee
/CPE212_elcid/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 - na
me: install apache2 and php\n    ^ here\n"}
fatal: [192.168.56.104]: FAILED! => {"msg": "The task includes an option with an unde-
fined variable.. 'apache_package' is undefined\n\nThe error appears to be in '/home/cidee
/CPE212_elcid/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 - na
me: install apache2 and php\n    ^ here\n"}

PLAY RECAP *****
192.168.56.103      : ok=1    changed=0    unreachable=0    failed=1    skipped=
0    rescued=0    ignored=0
192.168.56.104      : 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.

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.

```
cidee@workstation:~/CPE212_elcid$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

TASK [install apache2 and php] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

PLAY RECAP *****
192.168.56.103      : ok=2    changed=0    unreachable=0    failed=0    skipped=0
0   rescued=0    ignored=0
192.168.56.104      : ok=2    changed=0    unreachable=0    failed=0    skipped=0
0   rescued=0    ignored=0
```

Supplementary Activity:

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

None

```
- name: Install Apache and PHP on mixed environments
  hosts: all
```

```
become: yes
vars:
  apache_package: "{{ 'apache2' if ansible_os_family == 'Debian' else
'httpd' }}"
  php_package: php
  apache_service: "{{ 'apache2' if ansible_os_family == 'Debian' else
'httpd' }}"

tasks:
  - name: Install Apache and PHP packages
    package:
      name:
        - "{{ apache_package }}"
        - "{{ php_package }}"
      state: latest

  - name: Start and enable Apache service
    systemd:
      name: "{{ apache_service }}"
      state: started
      enabled: yes

  - name: Configure firewall (Red Hat systems)
    firewalld:
      service: http
      permanent: yes
      state: enabled
    when: ansible_os_family == "RedHat"
    notify:
      - reload firewall

  - name: Configure firewall (Debian systems)
    ufw:
      rule: allow
      port: '80'
      proto: tcp
    when: ansible_os_family == "Debian"

handlers:
  - name: reload firewall
    command: firewall-cmd --reload
    when: ansible_os_family == "RedHat"
```

```

BECOME password:

PLAY [Install Apache and PHP on Red Hat systems] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.104]

TASK [Install Apache and PHP packages] *****
fatal: [192.168.56.103]: FAILED! => {"ansible_facts": {"pkg_mgr": "apt"}, "changed": false, "msg": ["Could not detect which major revision of dnf is in use, which is required to determine module backend.", "You should manually specify use_backend to tell the module whether to use the dnf4 or dnf5 backend"]}
ok: [192.168.56.104]

TASK [Start and enable Apache service] *****
changed: [192.168.56.104]

TASK [Configure firewall to allow HTTP traffic] *****
changed: [192.168.56.104]

TASK [Verify Apache is running] *****
fatal: [192.168.56.104]: FAILED! => {"accept_ranges": "bytes", "changed": false, "connection": "close", "content_length": "2713881", "content_type": "text/html; charset=UTF-8", "date": "Fri, 12 Sep 2025 15:48:55 GMT", "elapsed": 0, "etag": "\"296919-61a185ec88200\"", "failed_when_result": true, "last_modified": "Tue, 04 Jun 2024 22:57:12 GMT", "msg": "Status code was 403 and not [200]: HTTP Error 403: Forbidden", "redirected": false, "server": "Apache/2.4.62 (CentOS Stream)", "status": 403, "url": "http://localhost"}

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

192.168.56.104      : ok=4    changed=2    unreachable=0    failed=1    skipped=
0      rescued=0    ignored=0

```

Reflections:

Answer the following:

- Why do you think refactoring of playbook codes is important?
Refactoring improves playbook maintainability, readability, and reusability. It eliminates redundancy, reduces errors, and makes code easier to update and share across teams. Well-structured playbooks run more efficiently and are simpler to debug.
- When do we use the “when” command in playbook?
We use 'when' for conditional task execution, primarily to handle different operating systems (like Ubuntu vs CentOS), environment-specific configurations, optional features, or to ensure tasks only run when specific requirements are met. It allows playbooks to adapt to diverse target systems.

