Name: Escosia, Jerico James	**Date Performed: 09-29-24**
Course/Section: CPE212- CPE31S21	**Date Submitted:09-29-24**
Instructor: Engr. Valenzuela	**Semester and SY: 1st Sem- 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:

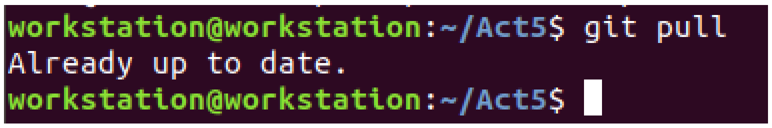
We are going to look at a way that we can differentiate a playbook by a host in terms of which distribution the host is running. It's very common in most Linux shops to run multiple distributions, for example, Ubuntu shop or Debian shop and you need a different distribution for a one off-case or perhaps 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

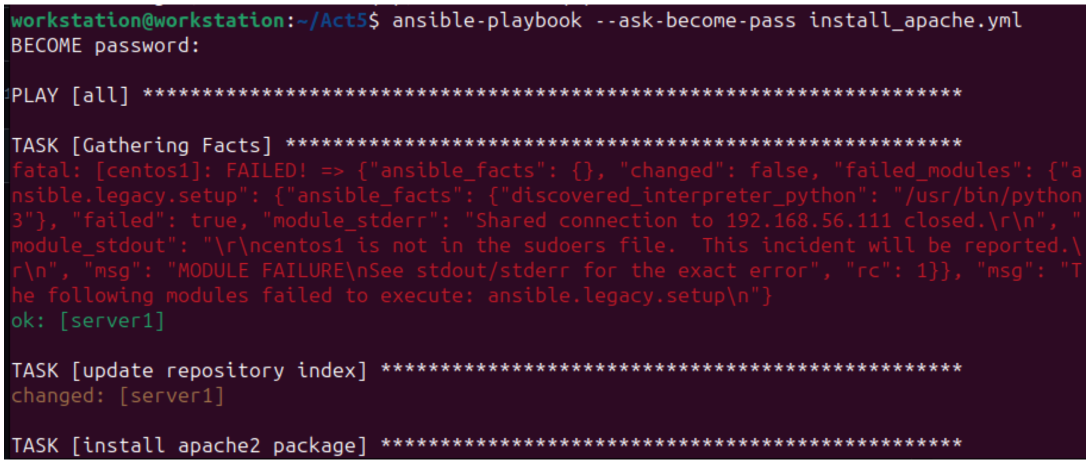
1. In the local machine, make sure you are in the local repository directory (*CPE232_yourname*). Issue the command git pull. When prompted, enter the correct passphrase or password. Describe what happened when you issue this command. Did something happen? Why?



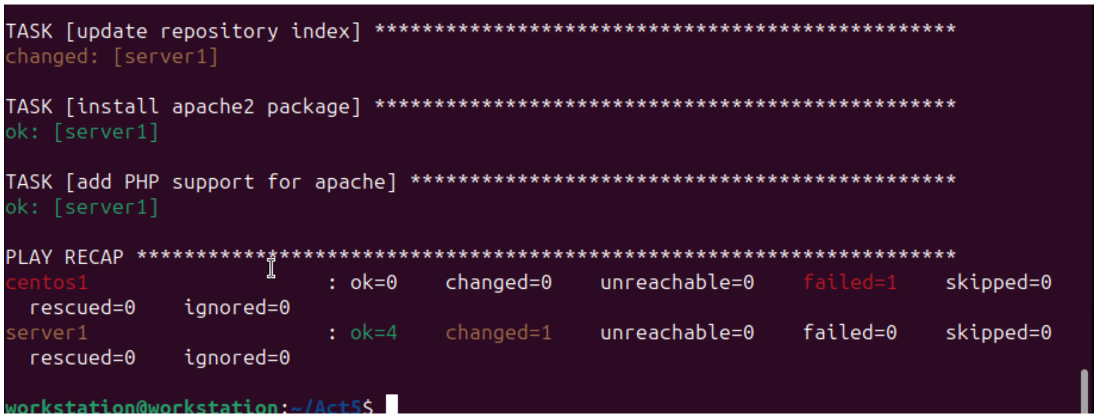
```
workstation@workstation:~/Act5$ git pull
Already up to date.
workstation@workstation:~/Act5$ 
```

Nothing happened because there is no file that was added in the github repo.

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."

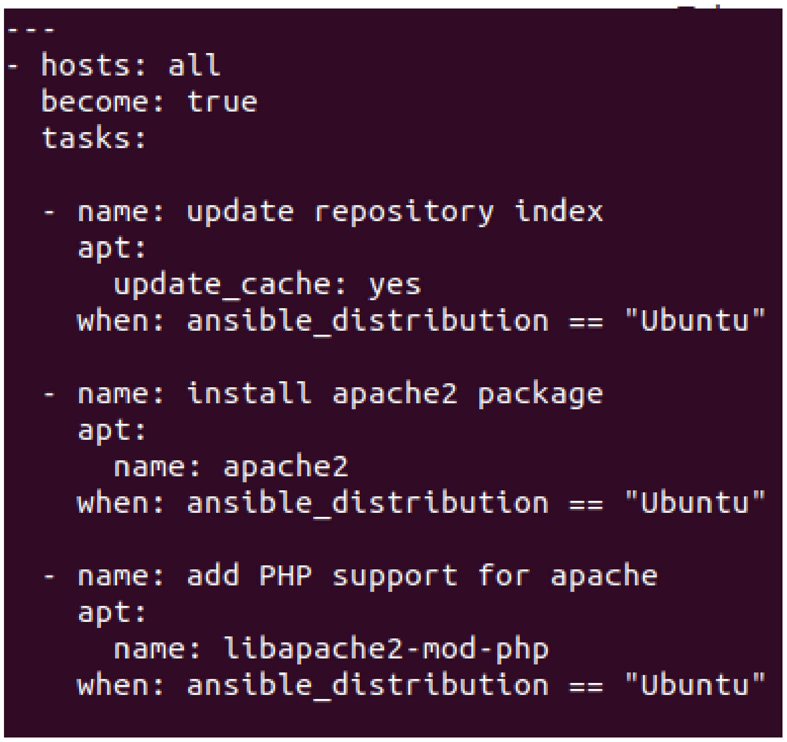


```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
fatal: [centos1]: FAILED! => {"ansible_facts": {}, "changed": false, "failed_modules": {"a
nsible.legacy.setup": {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python
3"}, "failed": true, "module_stderr": "Shared connection to 192.168.56.111 closed.\r\n", "
module_stdout": "\r\ncentos1 is not in the sudoers file.  This incident will be reported.\
r\n", "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error", "rc": 1}}, "msg": "T
he following modules failed to execute: ansible.legacy.setup\n"}
ok: [server1]

TASK [update repository index] *************************************************
changed: [server1]

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



```
TASK [update repository index] *************************************************
changed: [server1]

TASK [install apache2 package] *************************************************
ok: [server1]

TASK [add PHP support for apache] **********************************************
ok: [server1]

PLAY RECAP *********************************************************************
centos1                    : ok=0    changed=0    unreachable=0    failed=1    skipped=0
  rescued=0    ignored=0
server1                    : ok=4    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0

workstation@workstation:~/Act5$
```

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"
```

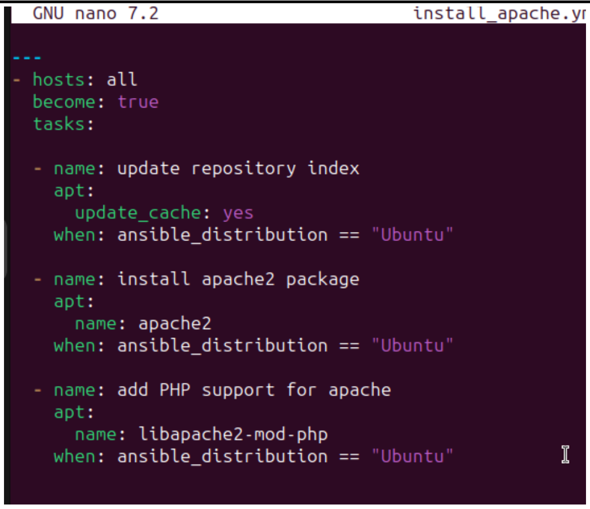
Make sure to save the file and exit.

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

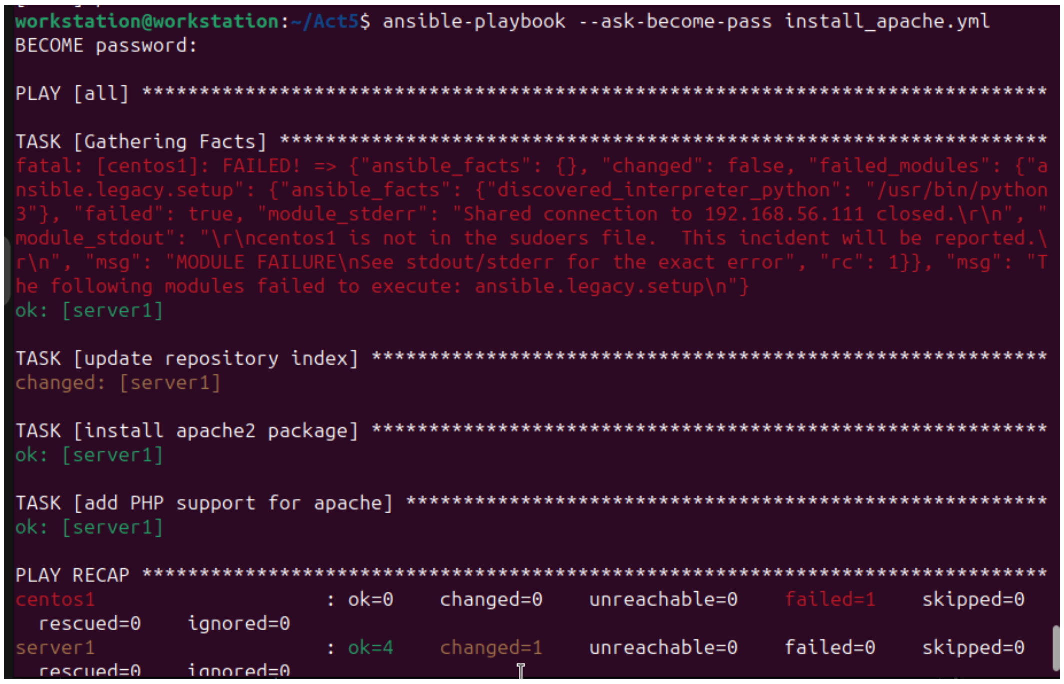
If you have a mix of Debian and Ubuntu servers, you can change the configuration of your playbook like this.

```
-    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.

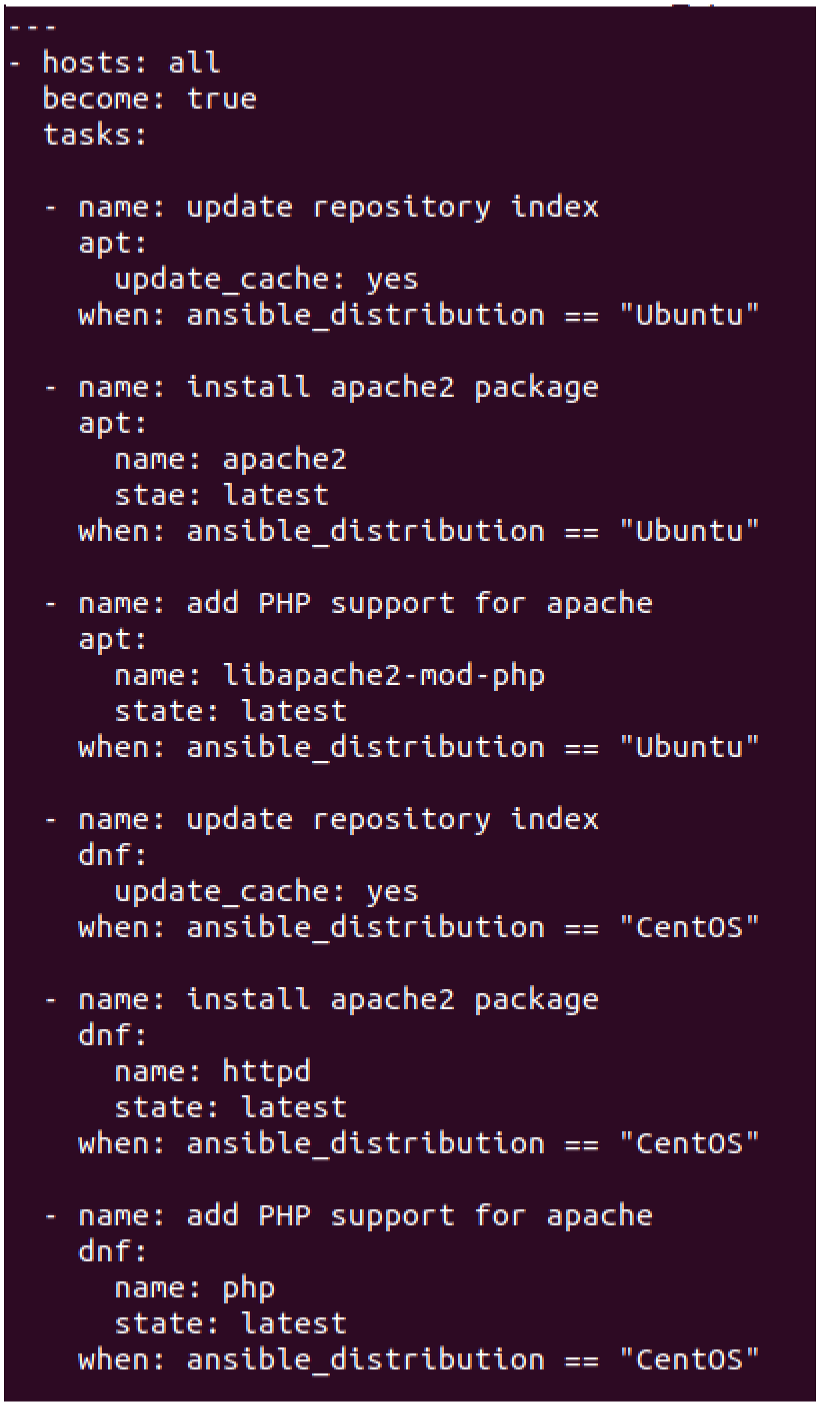


```
  GNU nano 7.2                       install_apache.yr

---
- 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"
```



```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ******************************************************************
fatal: [centos1]: FAILED! => {"ansible_facts": {}, "changed": false, "failed_modules": {"a
nsible.legacy.setup": {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python
3"}, "failed": true, "module_stderr": "Shared connection to 192.168.56.111 closed.\r\n", "
module_stdout": "\r\ncentos1 is not in the sudoers file.  This incident will be reported.\
r\n", "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error", "rc": 1}}, "msg": "T
he following modules failed to execute: ansible.legacy.setup\n"}
ok: [server1]

TASK [update repository index] **********************************************************
changed: [server1]

TASK [install apache2 package] **********************************************************
ok: [server1]

TASK [add PHP support for apache] *******************************************************
ok: [server1]

PLAY RECAP ******************************************************************************
centos1                    : ok=0    changed=0    unreachable=0    failed=1    skipped=0
  rescued=0    ignored=0
server1                    : ok=4    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
```

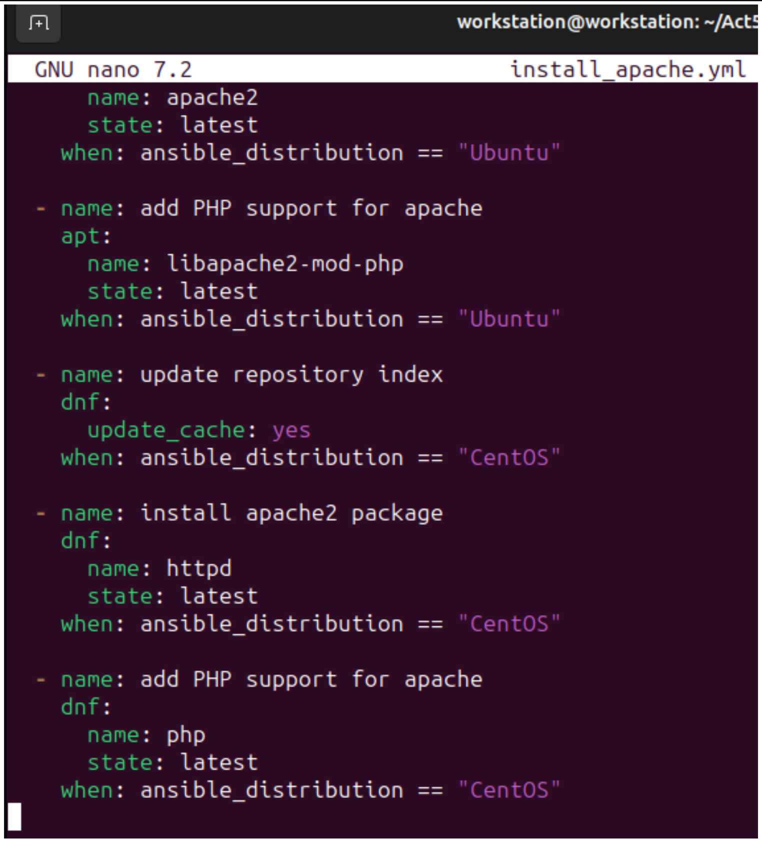
4. Edit the *install_apache.yml* file and insert the lines shown below.



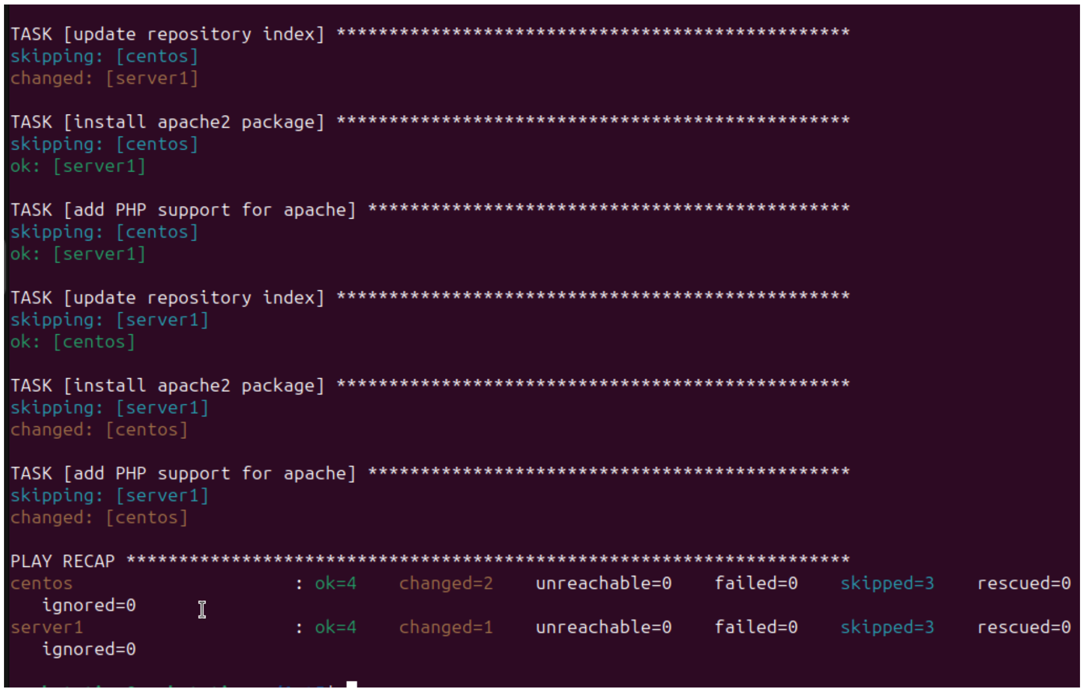
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 package
    apt:
      name: apache2
      stae: 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.



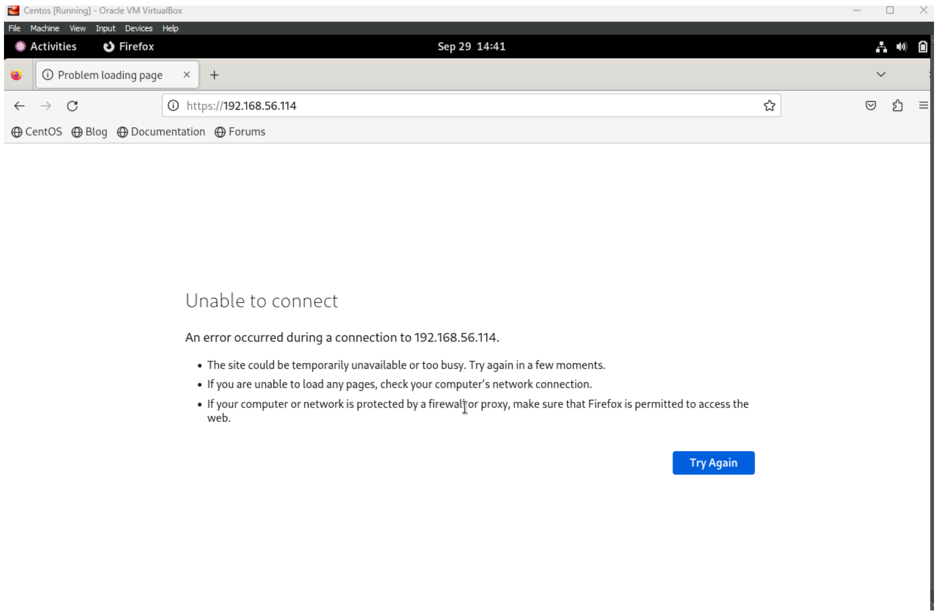
```
workstation@workstation: ~/Act5
GNU nano 7.2                          install_apache.yml
      name: apache2
      state: latest
    when: ansible_distribution == "Ubuntu"

  - 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"
```



```
TASK [update repository index] *************************************************
skipping: [centos]
changed: [server1]

TASK [install apache2 package] *************************************************
skipping: [centos]
ok: [server1]

TASK [add PHP support for apache] **********************************************
skipping: [centos]
ok: [server1]

TASK [update repository index] *************************************************
skipping: [server1]
ok: [centos]

TASK [install apache2 package] *************************************************
skipping: [server1]
changed: [centos]

TASK [add PHP support for apache] **********************************************
skipping: [server1]
changed: [centos]

PLAY RECAP *********************************************************************
centos                     : ok=4    changed=2    unreachable=0    failed=0    skipped=3    rescued=0
   ignored=0
server1                    : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0
   ignored=0
```

By putting the specific installation on the specific OS the ansible skips the content and proceeds to the next task that the centos was given to.

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.



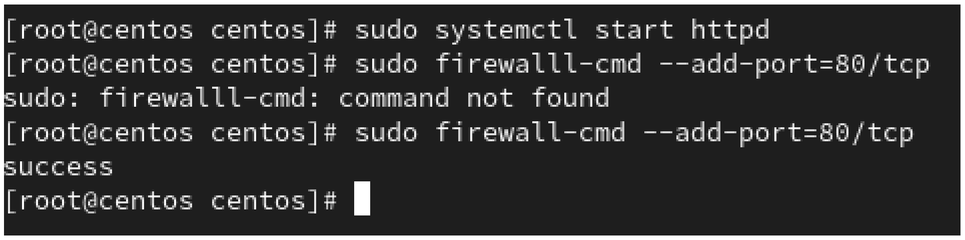
Unable to connect

An error occurred during a connection to 192.168.56.114.

- 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.

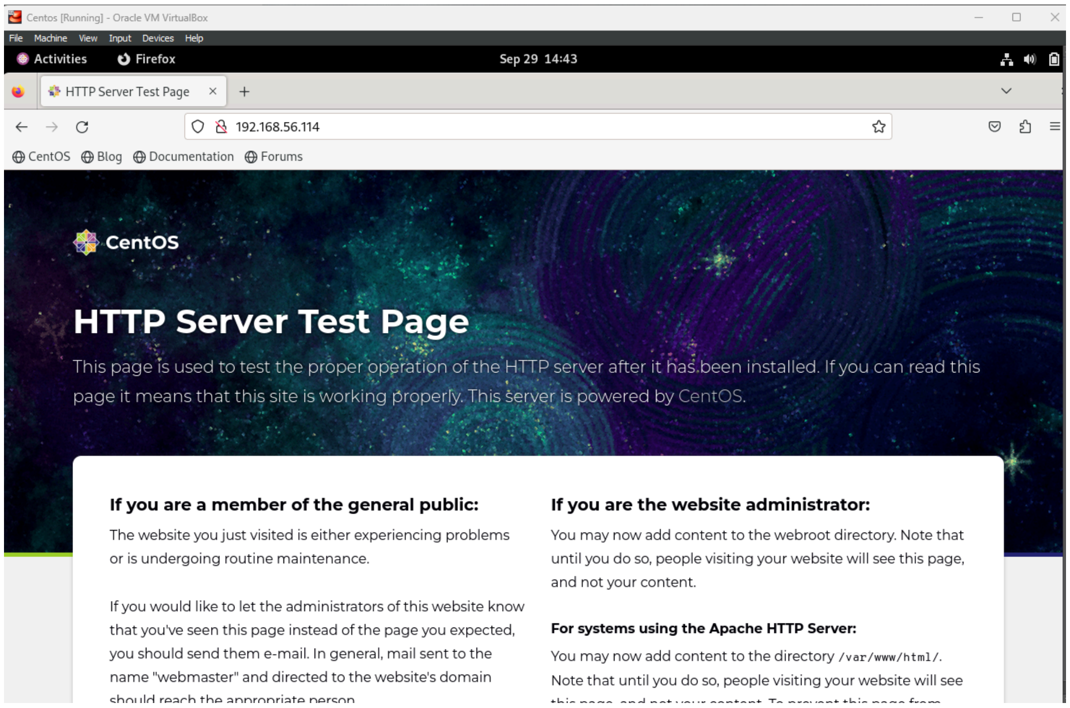
5.1 To activate, go to the CentOS VM terminal and enter the following:
systemctl status httpd
The result of this command tells you that the service is inactive.

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)



```
[root@centos centos]# sudo systemctl start httpd
[root@centos centos]# sudo firewalll-cmd --add-port=80/tcp
sudo: firewalll-cmd: command not found
[root@centos centos]# sudo firewall-cmd --add-port=80/tcp
success
[root@centos centos]# 
```

5.3 To verify the service is already running, go to CentOS VM and type its IP address on the browser. Was it successful? YES (Screenshot the browser)



HTTP Server Test Page

This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

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 should reach the appropriate person.

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.

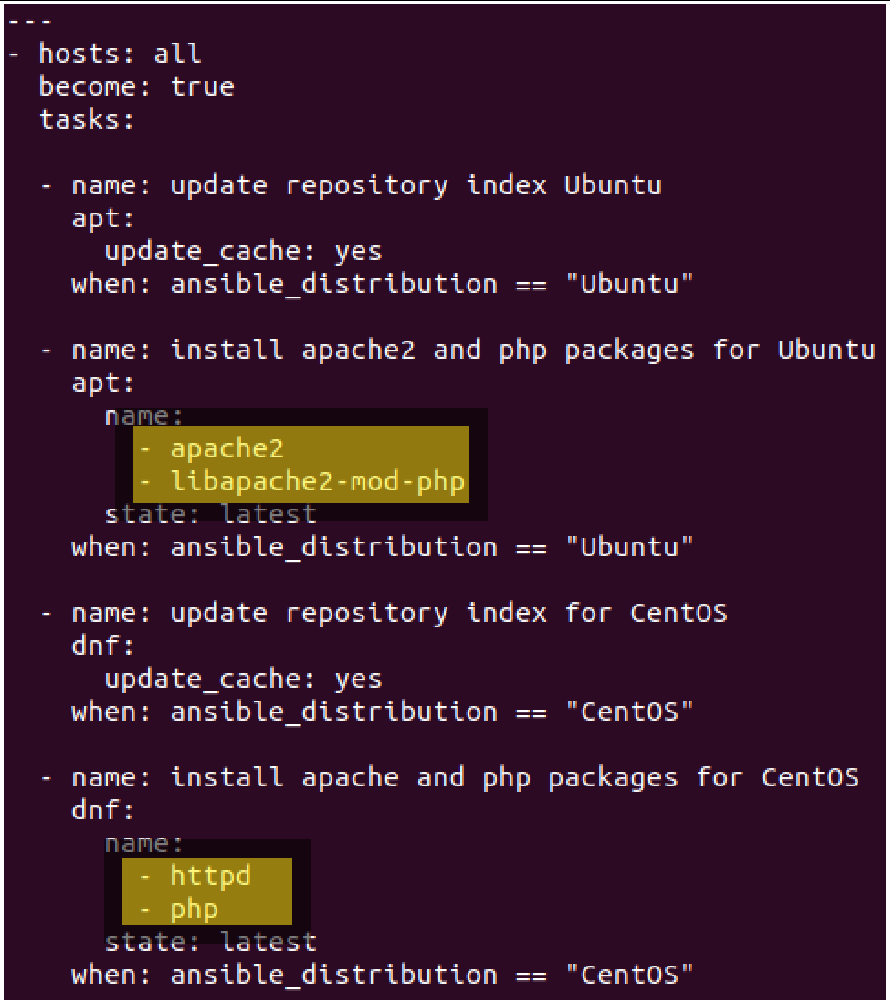
For systems using the Apache HTTP Server:

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 makes run ansible more quickly if it has to execute fewer tasks to do the same thing.

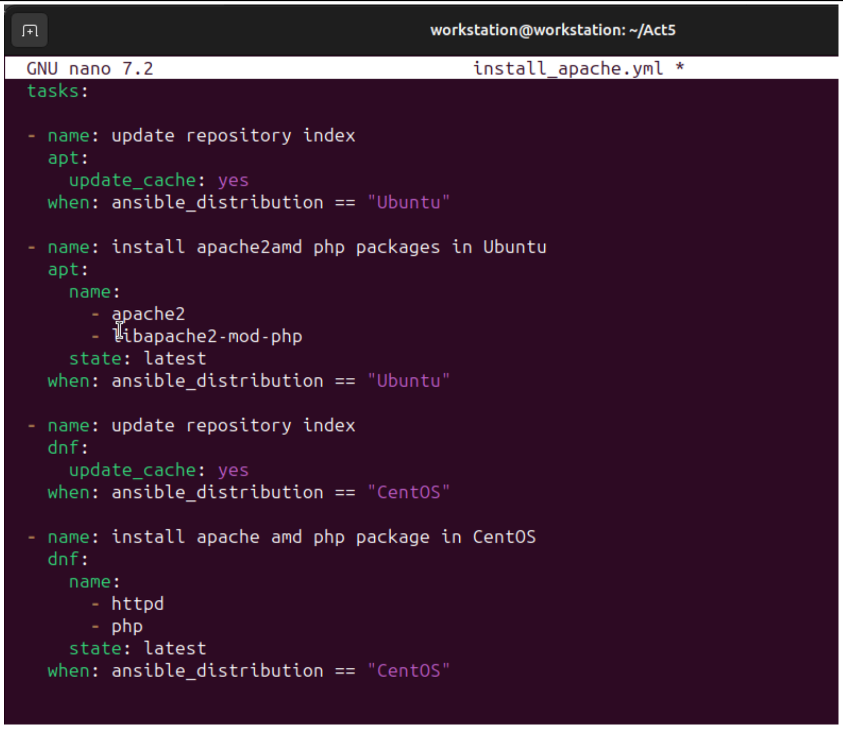
1. Edit the playbook *install_apache.yml*. Currently, we have three tasks targeting our Ubuntu machines and 3 tasks targeting our CentOS machine. Right now, we try to consolidate some tasks that are typically the same. For example, we can consolidate two plays that install packages. We can do that by creating a list of installation packages as shown below:



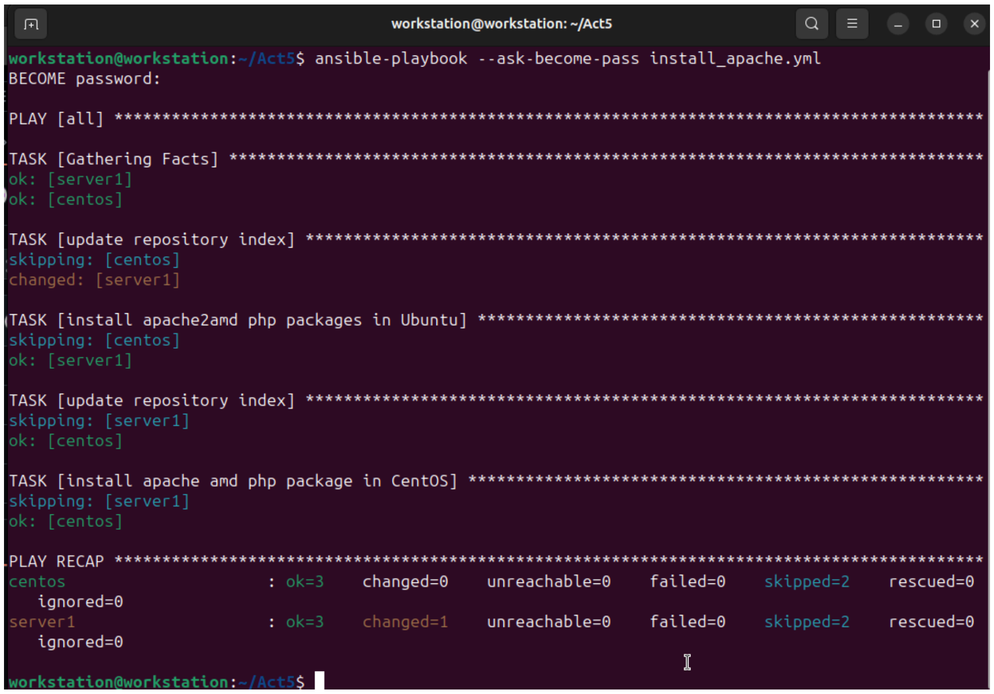
```
---
- hosts: all
  become: true
  tasks:

  - 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.



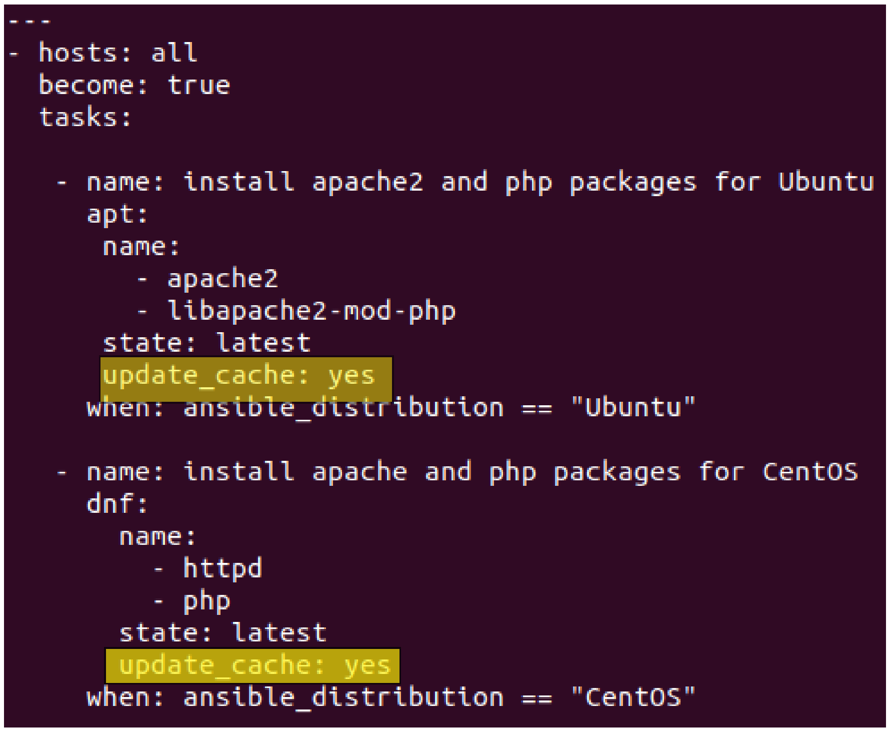
```
workstation@workstation: ~/Act5
  GNU nano 7.2                              install_apache.yml *
tasks:

- name: update repository index
  apt:
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: install apache2amd php packages in Ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    state: latest
  when: ansible_distribution == "Ubuntu"

- name: update repository index
  dnf:
    update_cache: yes
  when: ansible_distribution == "CentOS"

- name: install apache amd php package in CentOS
  dnf:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"
```



```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *****************************************************************************
ok: [server1]
ok: [centos]

TASK [update repository index] *********************************************************************
skipping: [centos]
changed: [server1]

TASK [install apache2amd php packages in Ubuntu] ***************************************************
skipping: [centos]
ok: [server1]

TASK [update repository index] *********************************************************************
skipping: [server1]
ok: [centos]

TASK [install apache amd php package in CentOS] ****************************************************
skipping: [server1]
ok: [centos]

PLAY RECAP *****************************************************************************************
centos                     : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
   ignored=0
server1                    : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0
   ignored=0

workstation@workstation:~/Act5$
```

By consolidating the 2 tasks in one, the code was shortened by still performing the same as the previous code.

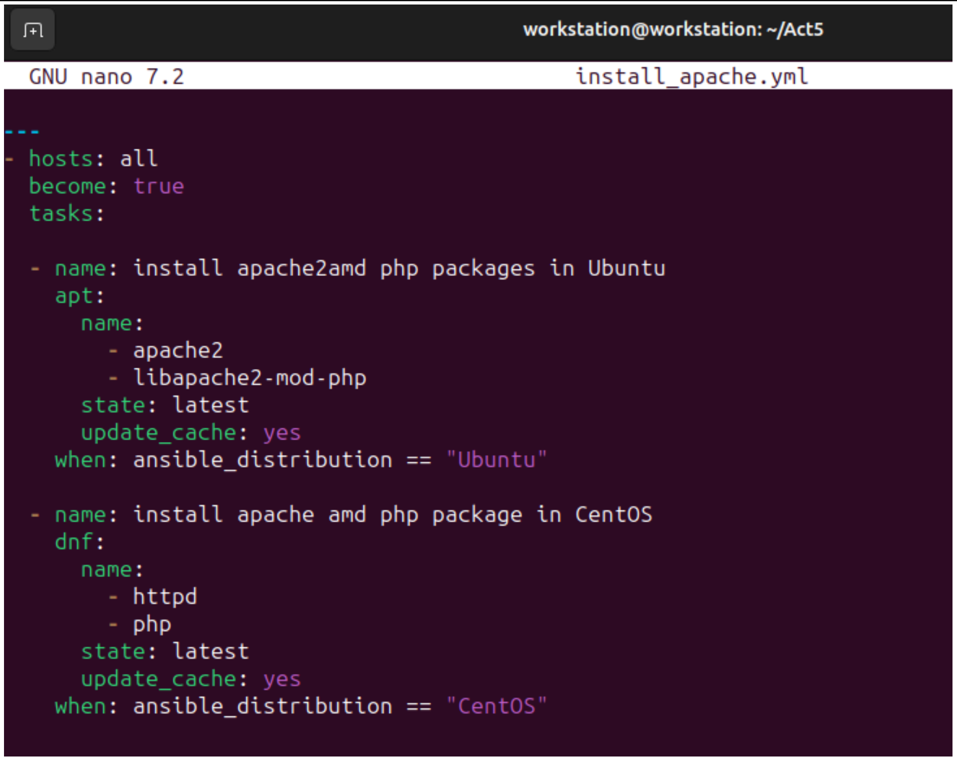
2. Edit the playbook *install_apache.yml* again. In task 2.1, we consolidated the plays into one play. This time we can actually consolidated everything in just 2 plays. This can be done by removing the update repository play and putting the command *update_cache: yes* below the command *state: latest*. See below for reference:



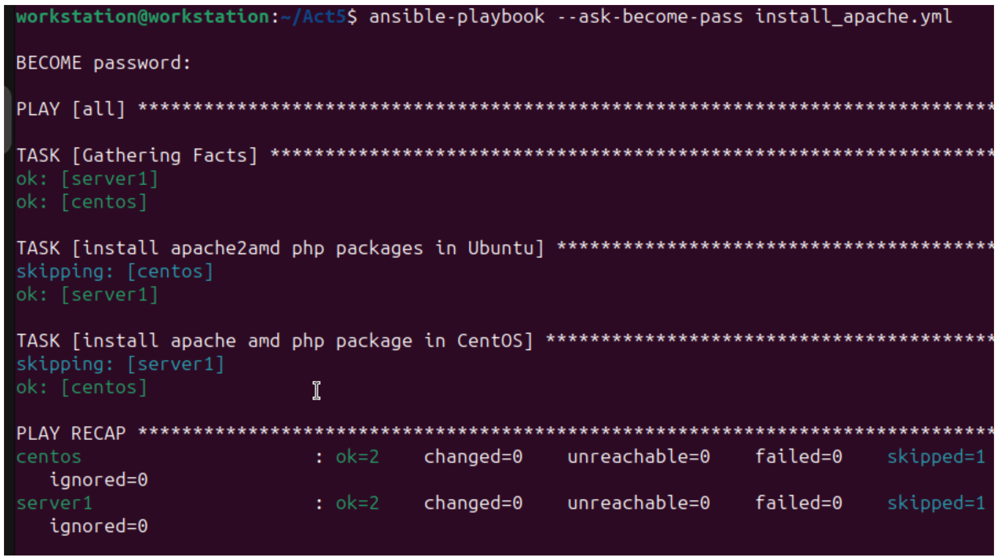
```
---
- hosts: all
  become: true
  tasks:

   - 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.



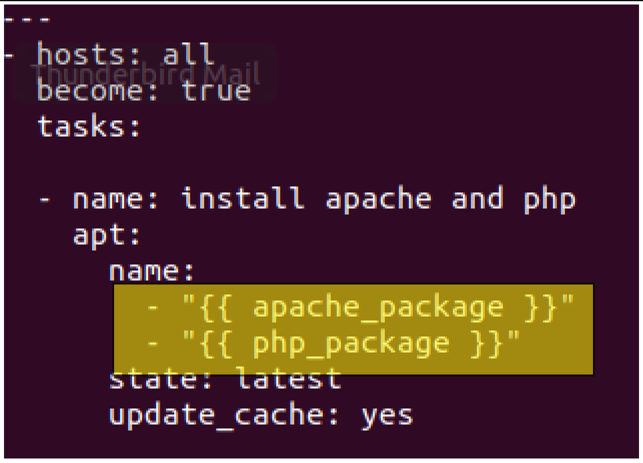
```
workstation@workstation: ~/Act5
GNU nano 7.2                              install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2amd php packages in Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache amd php package in CentOS
    dnf:
      name:
        - httpd
        - php
      state: latest
      update_cache: yes
    when: ansible_distribution == "CentOS"
```



```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml

BECOME password:

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

TASK [Gathering Facts] ********************************************************************
ok: [server1]
ok: [centos]

TASK [install apache2amd php packages in Ubuntu] ******************************************
skipping: [centos]
ok: [server1]

TASK [install apache amd php package in CentOS] *******************************************
skipping: [server1]
ok: [centos]

PLAY RECAP ********************************************************************************
centos                     : ok=2    changed=0    unreachable=0    failed=0    skipped=1
   ignored=0
server1                    : ok=2    changed=0    unreachable=0    failed=0    skipped=1
   ignored=0
```

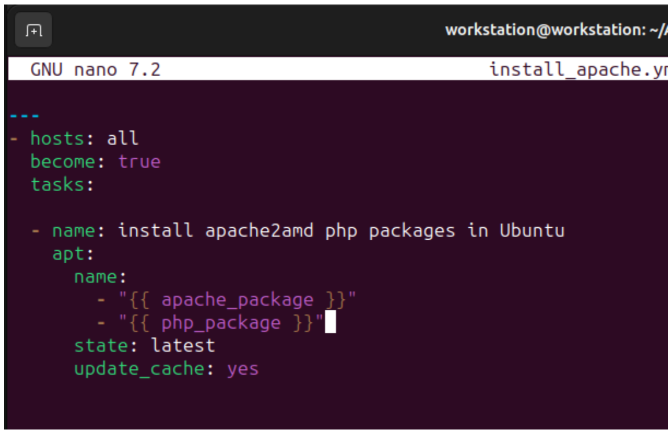
Consolidating the redundant syntax inside ansible makes the code much easier to read but still perform the task in the previous code.

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.

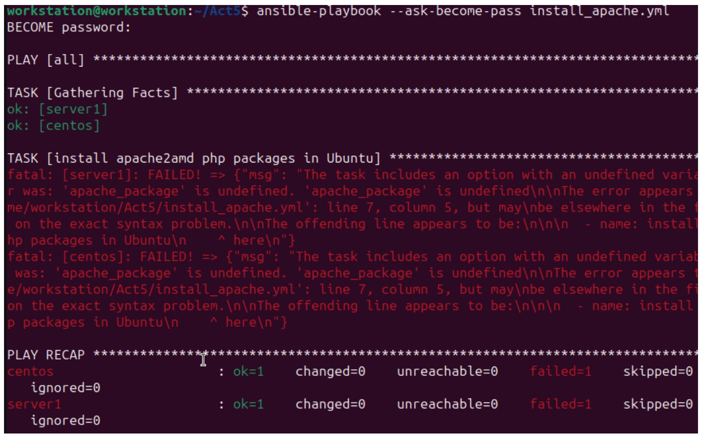


```
---
- hosts: all
  become: true
  tasks:

  - 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.



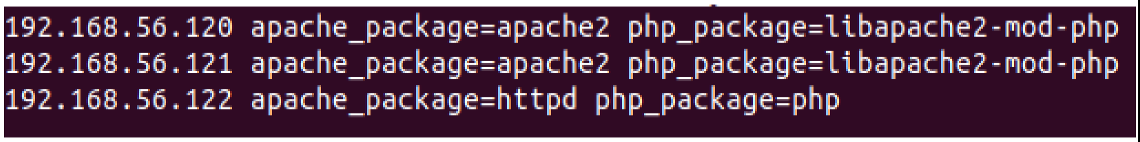
```
workstation@workstation: ~/
GNU nano 7.2                                  install_apache.y

---
- hosts: all
  become: true
  tasks:

  - name: install apache2amd php packages in Ubuntu
    apt:
      name:
        - "{{ apache_package }}"
        - "{{ php_package }}"
      state: latest
      update_cache: yes
```



```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] **************************************************************
ok: [server1]
ok: [centos]

TASK [install apache2amd php packages in Ubuntu] ************************************
fatal: [server1]: FAILED! => {"msg": "The task includes an option with an undefined varia
r was: 'apache_package' is undefined. 'apache_package' is undefined\n\nThe error appears 
me/workstation/Act5/install_apache.yml': line 7, column 5, but may\nbe elsewhere in the f
 on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n  - name: instal
hp packages in Ubuntu\n    ^ here\n"}
fatal: [centos]: FAILED! => {"msg": "The task includes an option with an undefined variat
 was: 'apache_package' is undefined. 'apache_package' is undefined\n\nThe error appears t
e/workstation/Act5/install_apache.yml': line 7, column 5, but may\nbe elsewhere in the fi
on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n  - name: install 
p packages in Ubuntu\n    ^ here\n"}

PLAY RECAP **************************************************************************
centos                     : ok=1    changed=0    unreachable=0    failed=1    skipped=0
   ignored=0
server1                    : ok=1    changed=0    unreachable=0    failed=1    skipped=0
   ignored=0
```

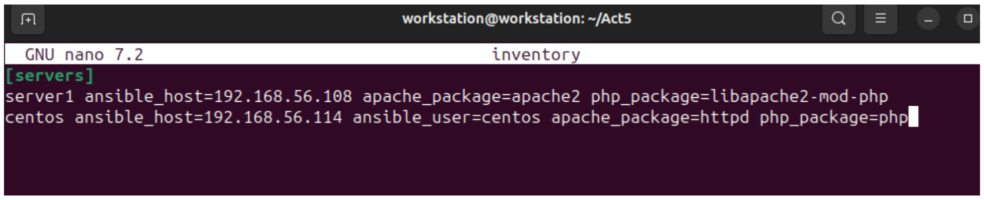
The playbook failed in doing the task because it includes an option with undefined variable, we need to change something in the inventory to fix this specific error.

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
```

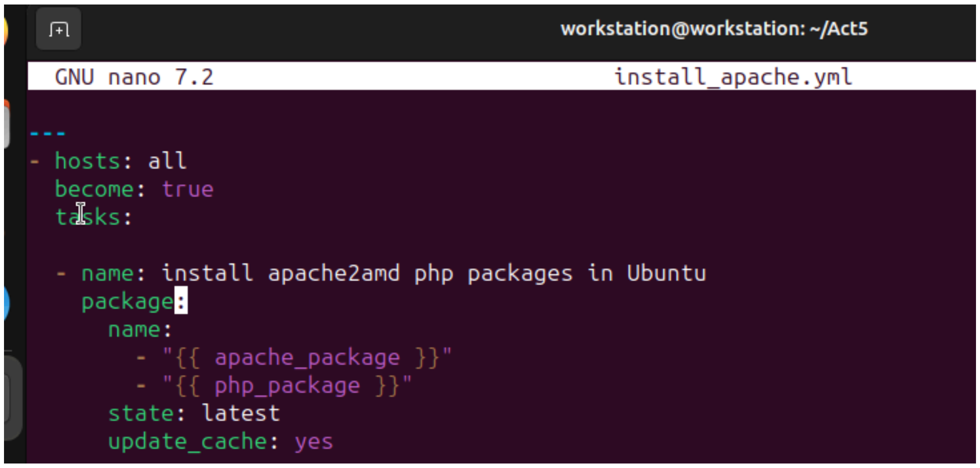
Make sure to save the *inventory* file and exit.



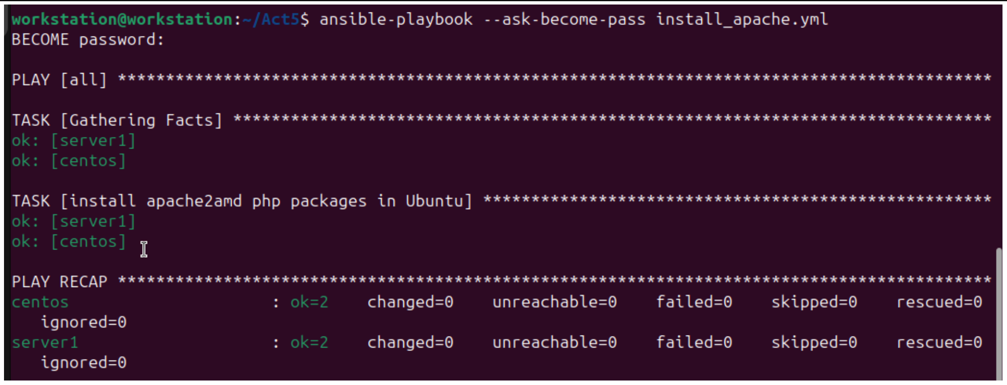
```
workstation@workstation: ~/Act5
  GNU nano 7.2                          inventory
[servers]
server1 ansible_host=192.168.56.108 apache_package=apache2 php_package=libapache2-mod-php
centos ansible_host=192.168.56.114 ansible_user=centos apache_package=httpd php_package=php
```

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.



```
workstation@workstation: ~/Act5
 GNU nano 7.2                          install_apache.yml

---
- hosts: all
  become: true
  tasks:

  - name: install apache2amd php packages in Ubuntu
    package:
      name:
        - "{{ apache_package }}"
        - "{{ php_package }}"
      state: latest
      update_cache: yes
```

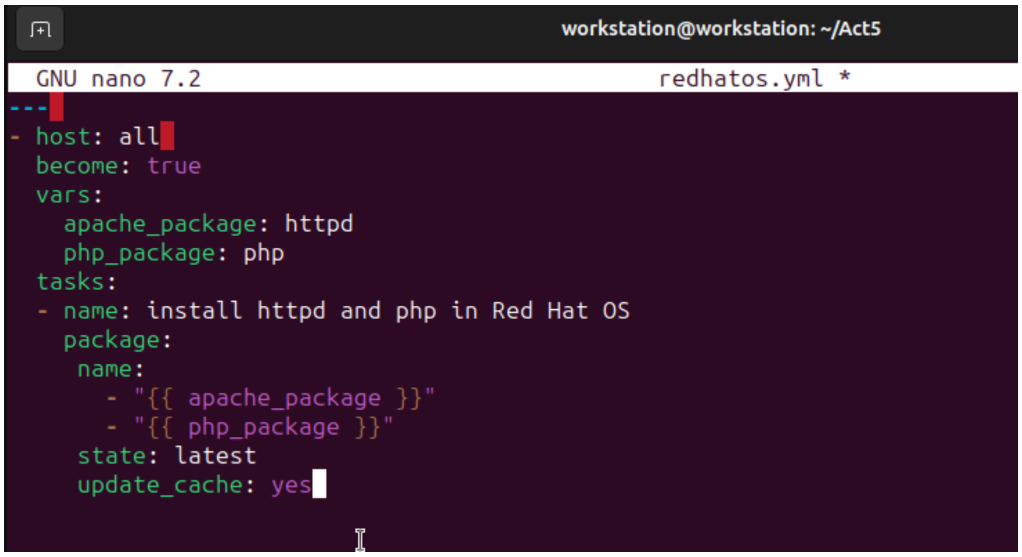


```
workstation@workstation:~/Act5$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *******************************************************************************
ok: [server1]
ok: [centos]

TASK [install apache2amd php packages in Ubuntu] *****************************************************
ok: [server1]
ok: [centos]

PLAY RECAP *******************************************************************************************
centos                     : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
   ignored=0
server1                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
   ignored=0
```

The playbook runs the same as the previous version, consolidating the redundant commands makes the playbook easier to read because it contains less lines of code but it still is able to perform intended tasks.

Supplementary Activity:

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



```
workstation@workstation: ~/Act5
  GNU nano 7.2                                   redhatos.yml *
---
- host: all
  become: true
  vars:
    apache_package: httpd
    php_package: php
  tasks:
  - name: install httpd and php in Red Hat OS
    package:
     name:
       - "{{ apache_package }}"
       - "{{ php_package }}"
     state: latest
     update_cache: yes
```

Reflections:

Answer the following:

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

   -Refactoring of the ansible playbook is important to eliminate the redundancy of the tasks it will perform. It also makes it shorter to code, doing the same thing as the longer code versions.

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

   - **We use when command in ansible playbook to specify the OS of the machine we want to configure, since different machines accepts different syntax this specifier is needed for the managed node to understand the task inside the ansible playbook.**