Name: Erebete, Jan Kenneth F.	Date Performed: 9/5/25
Course/Section: CPE22S4	Date Submitted: 9/5/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 2025-2026

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages in the remote machines. We will also create a playbook that will be used for automations.

Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. You can check this documentation if you want to learn more about playbooks. Working with playbooks — Ansible Documentation

Task 1: Run elevated ad hoc commands

1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package information from the Internet. It is useful to get info on an updated version of packages or their dependencies. We can only run

an apt update command in a remote machine. Issue the following command:

ansible all -m apt -a update_cache=true

What is the result of the command? Is it successful?

```
erebete@workstation:~$ ansible all -m apt -a update_cache=true
192.168.56.101 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "E: Could not open lock file /var/lib/apt/lists/lock - open (13: Per
mission denied)\nE: Unable to lock directory /var/lib/apt/lists/\nW: Problem un
linking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission den
ied)\nW: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCach
es (13: Permission denied)",
    "rc": 100,
    "stderr": "E: Could not open lock file /var/lib/apt/lists/lock - open (13:
Permission denied)\nE: Unable to lock directory /var/lib/apt/lists/\nW: Problem
 unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission
denied)\nW: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveC
aches (13: Permission denied)\n",
    "stderr_lines": [
        "E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permis
sion denied)",
        "E: Unable to lock directory /var/lib/apt/lists/",
        "W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCach
es (13: Permission denied)",
        "W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveC
aches (13: Permission denied)"
```

Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sud –o password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same thing as running *sudo apt update*. The --become command elevate the privileges and the *--ask-become-pass* asks for the password. For now, even if we only have changed the packaged index, we were able to change something on the remote server.

You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

```
erebete@workstation:~$ ansible all -m apt -a update_cache=true --become --ask-b
ecome-pass
SUDO password:
192.168.56.102 | FAILED! => {
    "changed": false,
    "cmd": "apt-get install python-apt -y -q",
    "msg": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/
python-apt/python-apt_1.6.6_amd64.deb  Could not connect to ph.archive.ubuntu.c
om:80 (202.79.180.254). - connect (111: Connection refused) [IP: 202.79.180.254
 80]\nE: Unable to fetch some archives, maybe run apt-get update or try with --
fix-missing?",
    "rc": 100,
    "stderr": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main
/p/python-apt/python-apt_1.6.6_amd64.deb  Could not connect to ph.archive.ubunt
u.com:80 (202.79.180.254). - connect (111: Connection refused) [IP: 202.79.180.
254 80]\nE: Unable to fetch some archives, maybe run apt-get update or try with
 --fix-missing?\n",
    "stderr_lines": [
        "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/pyt
hon-apt/python-apt_1.6.6_amd64.deb  Could not connect to ph.archive.ubuntu.com:
80 (202.79.180.254). - connect (111: Connection refused) [IP: 202.79.180.254 80
]",
        "E: Unable to fetch some archives, maybe run apt-get update or try with
 --fix-missing?"
    ],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
```

2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction. Here is the command: *ansible all -m apt -a name=vim-nox --become --ask-become-pass.* The command would take some time after typing the password because the local machine instructed the remote servers to actually install the package.

```
erebete@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become
-pass
SUDO password:
192.168.56.101 | FAILED! => {
    "changed": false,
    "cmd": "apt-get install python-apt -y -q",
    "msg": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/
python-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254 80
]\nE: Unable to fetch some archives, maybe run apt-get update or try with --fix
-missing?",
    "rc": 100,
    "stderr": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main
/p/python-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254
 80]\nE: Unable to fetch some archives, maybe run apt-get update or try with --
fix-missing?\n",
    "stderr_lines": [
        "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/pyt
hon-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254 80]",

        "E: Unable to fetch some archives, maybe run apt-get update or try with
 --fix-missing?"
    ],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nSugges
```

2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?

```
erebete@workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security 2:8.0.1453-1ubuntu1.13 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled]
  Vi IMproved - enhanced vi editor - compact version

erebete@workstation:~$ which vim
```

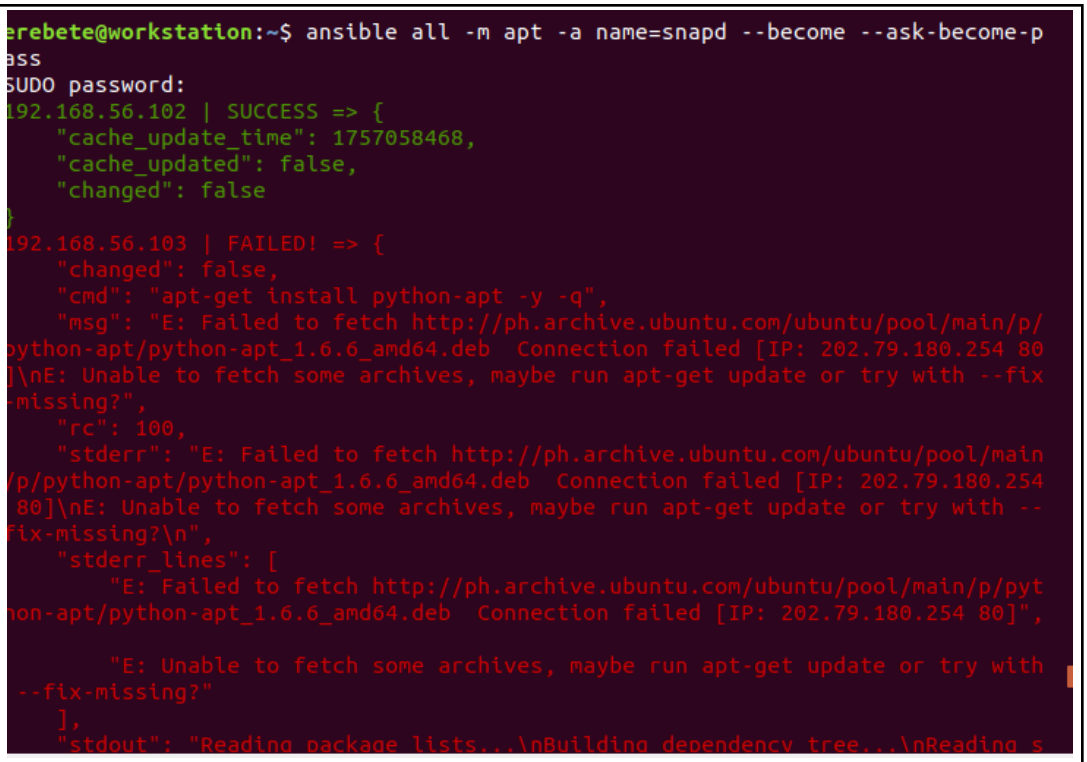
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.

```
erebete@workstation:/var/log$ apt
apt 1.6.17 (amd64)
Usage: apt [options] command

apt is a commandline package manager and provides commands for
searching and managing as well as querying information about packages.
It provides the same functionality as the specialized APT tools,
like apt-get and apt-cache, but enables options more suitable for
interactive use by default.

Most used commands:
  list - list packages based on package names
  search - search in package descriptions
  show - show package details
  install - install packages
  remove - remove packages
  autoremove - Remove automatically all unused packages
  update - update list of available packages
  upgrade - upgrade the system by installing/upgrading packages
  full-upgrade - upgrade the system by removing/installing/upgrading packages
  edit-sources - edit the source information file

See apt(8) for more information about the available commands.
Configuration options and syntax is detailed in apt.conf(5).
Information about how to configure sources can be found in sources.list(5).
Package and version choices can be expressed via apt_preferences(5).
Security details are available in apt-secure(8).
```

- It shows the different commands in installing packages

3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package. cd

   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*



Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?

- It only gets the Server 1 ping while workstation and server 2 can't be seen.

3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*

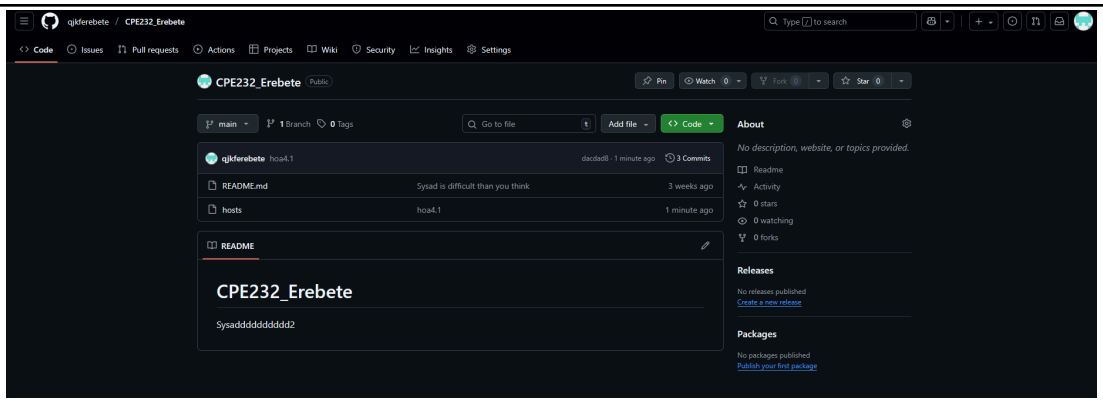
```
erebete@workstation:~$ ansible all -m apt -a "name=snapd state=latest" --become
 --ask-become-pass
SUDO password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1757058468,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | FAILED! => {
    "changed": false,
    "cmd": "apt-get install python-apt -y -q",
    "msg": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/
python-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254 80
]\nE: Unable to fetch some archives, maybe run apt-get update or try with --fix
-missing?",
    "rc": 100,
    "stderr": "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main
/p/python-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254
 80]\nE: Unable to fetch some archives, maybe run apt-get update or try with --
fix-missing?\n",
    "stderr_lines": [
        "E: Failed to fetch http://ph.archive.ubuntu.com/ubuntu/pool/main/p/pyt
hon-apt/python-apt_1.6.6_amd64.deb  Connection failed [IP: 202.79.180.254 80]",

        "E: Unable to fetch some archives, maybe run apt-get update or try with
 --fix-missing?"
    ],
```

Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.

The command uses state="latest" to ensure the specified package is installed and updated to the newest version, with quotes used to correctly interpret the value.

4. At this point, make sure to commit all changes to GitHub.

```
erebete@workstation:~/CPE232_Erebete$ git commit -m "hoa4.1"
[main dacdad8] hoa4.1
 1 file changed, 48 insertions(+)
 create mode 100644 hosts
erebete@workstation:~/CPE232_Erebete$ git push origin main
Warning: Permanently added the ECDSA host key for IP address '20.205.243.166' t
o the list of known hosts.
Counting objects: 3, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 794 bytes | 794.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:qjkferebete/CPE232_Erebete.git
   fbe9d34..dacdad8  main -> main
```



Task 2: Writing our First Playbook

1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and the place or commands that ansible will carry out to bring to that state. You can use an editor to create a playbook. Before we proceed, make sure that you are in the directory of the repository that we use in the previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. This will create a playbook file called *install_apache.yml*. The .yml is the basic standard extension for playbook files.

   When the editor appears, type the following:

```
 GNU nano 4.8                  install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2
```

```
 GNU nano 2.9.3                  install_apache.yml

---
 hosts: all
 become: true
 tasks:

 - name: install apache2 package
   apt:
     name: apache2
```

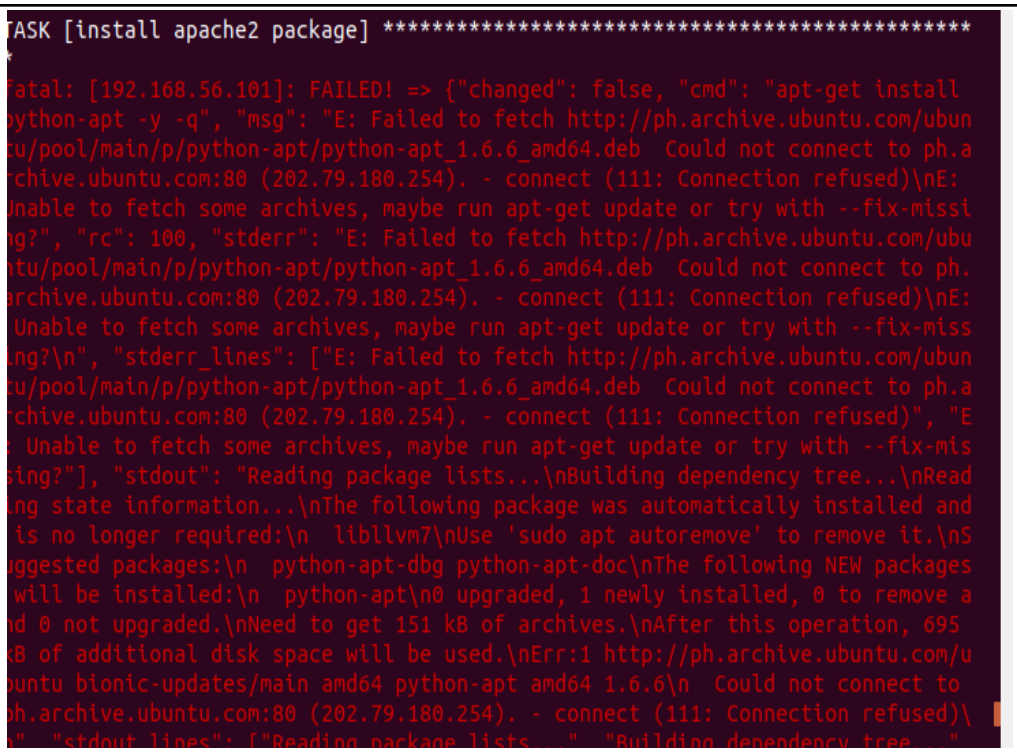
Make sure to save the file. Take note also of the alignments of the texts.

2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

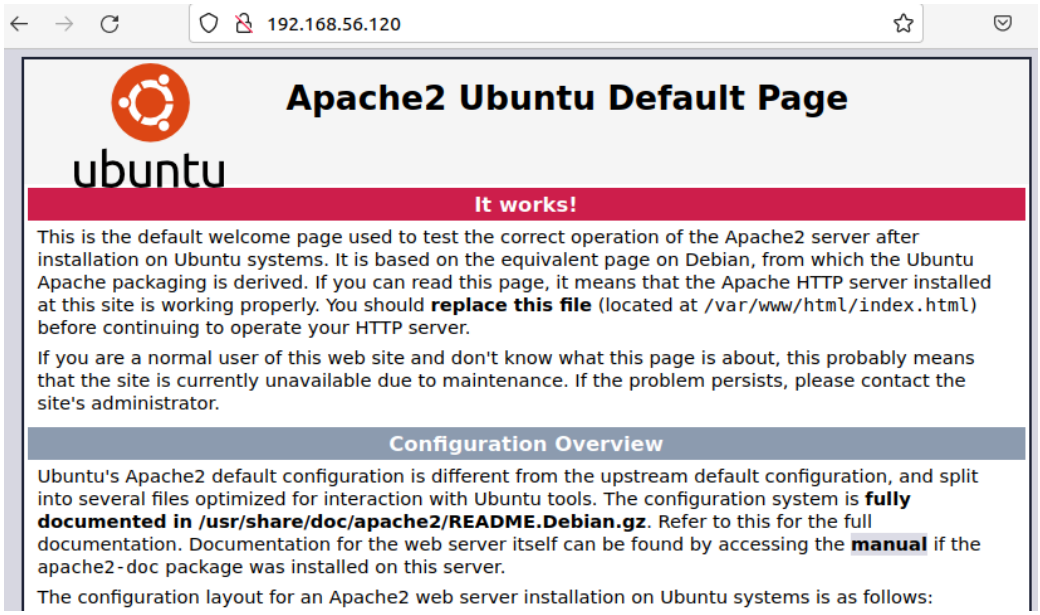
```
erebete@workstation:~/CPE232_Erebete$ ansible-playbook --ask-become-pass instal
l_apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.101]
ok: [192.168.56.102]
ok: [192.168.56.103]

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

- *It says it all ok and can be seen*

3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.



- The apache was error



4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?

5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.

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

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2
```

```
 GNU nano 2.9.3                  install_apache.yml                  Modified

--
 hosts: all
 become: true
 tasks:

 - name: update repository index
   apt:
     update_cache: yes

 - name: install apache2 package
   apt:
     name: apache2
```

Save the changes to this file and exit.

6. Run the playbook and describe the output. Did the new command change anything on the remote servers?

```
erebete@workstation:~/CPE232_Erebete$ ansible-playbook --ask-become-pass instal
l_apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.102]
ok: [192.168.56.101]
ok: [192.168.56.103]

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

7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.

```
 GNU nano 2.9.3                  install_apache.yml                  Modified

--
hosts: all
become: true
tasks:

- name: update repository index
  apt:
    update_cache: yes

- name: install apache2 package
  apt:
    name: apache2

- name: add PHP support for apache
  apt:
    name: libapache2-mod-php
```

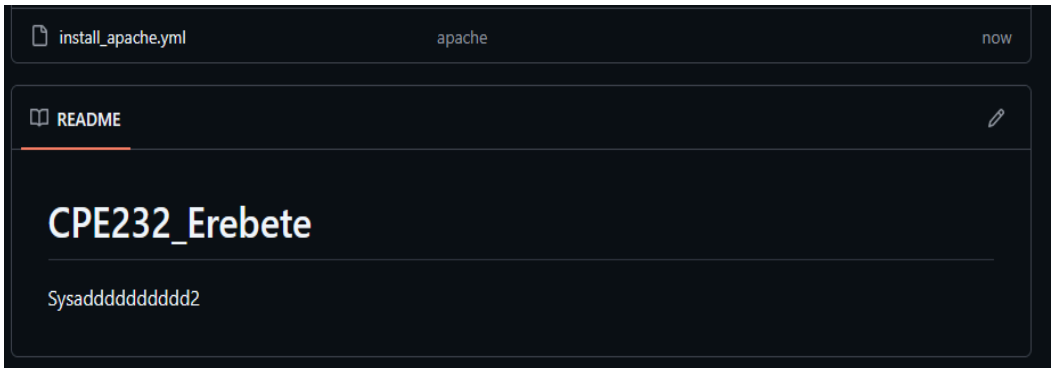
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

Save the changes to this file and exit.

8. Run the playbook and describe the output. Did the new command change anything on the remote servers?

```
erebete@workstation:~/CPE232_Erebete$ ansible-playbook --ask-become-pass instal
l_apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.102]
ok: [192.168.56.101]
ok: [192.168.56.103]

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

9. Finally, make sure that we are in sync with GitHub. Provide the link of your GitHub repository.




```
erebete@workstation:~/CPE232_Erebete$ git push origin
Counting objects: 3, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 450 bytes | 450.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:qjkferebete/CPE232_Erebete.git
   dacdad8..ceb1231  main -> main
erebete@workstation:~/CPE232_Erebete$ git push origin main
Everything up-to-date
```

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - using a playbook is important because it allows you to automate tasks in a consistent, repeatable, and organized
2. Summarize what we have done on this activity.
   - to summarize is we connect all of the server create and learn how the playbook works and understand every output