

<b>Name:</b> Aldwin Joseph B. Revilla	<b>Date Performed:</b> 09/11/2023
<b>Course/Section:</b> CPE31S5	<b>Date Submitted:</b> 09/12/2023
<b>Instructor:</b> Richard Roman	<b>Semester and SY:</b> 2023-2024
<b>Activity 4: Running Elevated Ad hoc Commands</b>	
<b>1. Objectives:</b> 1.1 Use commands that makes changes to remote machines 1.2 Use playbook in automating ansible commands	
<b>2. Discussion:</b>  <i>Provide screenshots for each task.</i>  <b>Elevated Ad hoc commands</b> 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.  <b>Playbooks</b> record and execute <b>Ansible</b> '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. <a href="#">Working with playbooks — Ansible Documentation</a>  Ansible setup: sudo apt update - in local machine, and 2 servers create new repository for ansible - git clone repository in local machine - git clone [http link]  sudo mkdir -p /etc/ansible cd /etc/ansible  creating ansible.cfg	

```
sudo nano ansible.cfg
```

```
[defaults]
```

```
inventory=/etc/ansible/hosts
```

```
creating hosts
```

```
sudo nano hosts
```

```
[localhosts]
```

```
127.0.0.1
```

```
ansible_connection=local
```

## 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:

```
qabrevilla@workstation:~$ sudo apt update
Hit:1 http://ph.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ph.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [319 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [764 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [165 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [826 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [133 kB]
Fetched 2,317 kB in 14s (165 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
qabrevilla@workstation:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  ansible
Learn more about Ubuntu Pro at https://ubuntu.com/pro
The following packages have been kept back:
  gjs libgjs0g
The following packages will be upgraded:
  libsmclient libwbclient0 samba-libs
3 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
Need to get 6,612 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsmclient
amd64 2:4.15.13+dfsg-0ubuntu1.4 [65.9 kB]
```

*ansible all -m apt -a update\_cache=true*

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

```
gabrevilla@workstation:~$ ansible all -m apt -a update_cache=true
192.0.0.1 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock
/var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock
LibreOffice Writer denied)"
}
```

The command was unsuccessful because ansible was not installed in the local machine.

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 sudo 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.

```
gabrevilla@workstation:~$ ansible all -m apt -a update_cache=true --become
-become-pass
BECOME password:
192.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694521744,
  "cache_updated": true,
  "changed": true
}
```

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.

```
gabrevilla@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694521744,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following additional packages will be installed:\n fonts-lato liblua5.2-0 libruby3.0 rake ruby ruby-net-telnet ruby-rubygems\n ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration vim-runtime\nSuggested packages:\n ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n fonts-lato liblua5.2-0 libruby3.0 rake ruby ruby-net-telnet ruby-rubygems\n ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration vim-nox vim-runtime\n0 upgraded, 13 newly installed, 0 to remove and 2 not upgraded.\nNeed to get 17.1 MB of archives.\nAfter this operation, 75.6 MB of additional disk space will be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato a
11.2-0.2.1 [2296 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy/universe a
```

- 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?

```
gabrevilla@workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,automatic]
  Vi IMproved - enhanced vi editor - compact version

gabrevilla@workstation:~$
```

```
gabrevilla@server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,automatic]
  Vi IMproved - enhanced vi editor - compact version

gabrevilla@server1:~$
```

```
gabrevilla@server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,automatic]
  Vi IMproved - enhanced vi editor - compact version

gabrevilla@server2:~$
```

Yes it was successful, it was able to search when typing the command search vim-nox

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`.

```
qabrevilla@workstation:/var/log$ ls
alternatives.log      dist-upgrade          kern.log.4.gz
alternatives.log.1    dmesg                 lastlog
alternatives.log.2.gz dmesg.0               openvpn
apt                   dmesg.1.gz           private
auth.log              dmesg.2.gz           speech-dispatcher
auth.log.1            dmesg.3.gz           syslog
auth.log.2.gz         dmesg.4.gz           syslog.1
auth.log.3.gz         dpkg.log              syslog.2.gz
auth.log.4.gz         dpkg.log.1            syslog.3.gz
boot.log              dpkg.log.2.gz         syslog.4.gz
boot.log.1            faillog               ubuntu-advantage.log
boot.log.2            fontconfig.log        ubuntu-advantage.log.1
boot.log.3            gdm3                  ubuntu-advantage.log.2.gz
boot.log.4            gpu-manager.log       ubuntu-advantage-timer.log
boot.log.5            hp                    ubuntu-advantage-timer.log.1
boot.log.6            installer             ubuntu-advantage-timer.log.2.gz
boot.log.7            journal               ufw.log
bootstrap.log         kern.log              ufw.log.1
btmtp                 kern.log.1            ufw.log.2.gz
btmtp.1               kern.log.2.gz         unattended-upgrades
cups                  kern.log.3.gz         wtmp
```

```
qabrevilla@workstation:/var/log/apt$ cat history.log

Start-Date: 2023-09-05  22:14:18
Commandline: apt install ssh
Requested-By: qabrevilla (1000)
Install: ssh:amd64 (1:8.9p1-3ubuntu0.3)
End-Date: 2023-09-05  22:14:19

Start-Date: 2023-09-05  22:39:17
Commandline: /usr/bin/unattended-upgrade
Install: linux-modules-extra-6.2.0-32-generic:amd64 (6.2.0-32.32~22.04.1, automatic), linux-hwe-6.2-headers-6.2.0-32:amd64 (6.2.0-32.32~22.04.1, automatic), linux-image-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic), linux-headers-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic), linux-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic)
Upgrade: linux-image-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic), linux-headers-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic), linux-generic-hwe-22.04:amd64 (6.2.0-26.26~22.04.7, 6.2.0-32.32~22.04.1, automatic)
End-Date: 2023-09-05  22:39:53

Start-Date: 2023-09-05  22:39:56
Commandline: /usr/bin/unattended-upgrade
```

After the command `cat history.log` we can see the different packages installed in our local machine where it was installed by admin or the user of the device.

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.

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

```
qabrevilla@workstation:/var/log/apt$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.0.0.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694521744,
  "cache_updated": false,
  "changed": false
}
```

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

There was no changes applied on the target servers and the snapd package was already installed.

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

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

```
qabrevilla@workstation:/var/log/apt$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.0.0.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694521744,
  "cache_updated": false,
  "changed": false
}
```

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

```

qabrevilla@workstation:~/CPE232_Revilla$ touch example
qabrevilla@workstation:~/CPE232_Revilla$ git add example
qabrevilla@workstation:~/CPE232_Revilla$ ls
example  README.md
qabrevilla@workstation:~/CPE232_Revilla$ git commit -m "Changes"
[main 088018d] Changes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 example
qabrevilla@workstation:~/CPE232_Revilla$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (3/3), 272 bytes | 272.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qabrevilla/CPE232_Revilla.git
 79aace1..088018d  main -> main
qabrevilla@workstation:~/CPE232_Revilla$

```

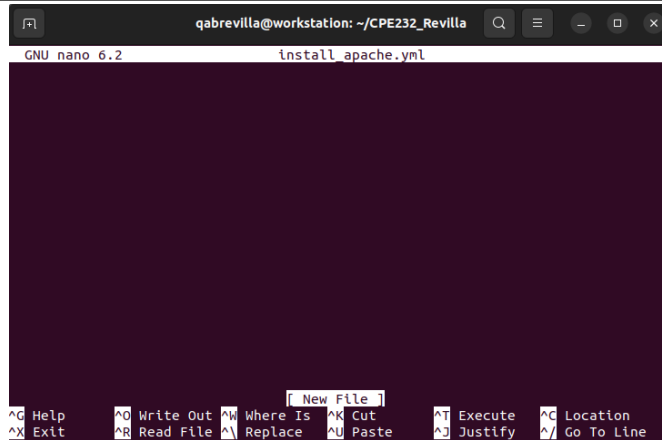
The screenshot shows a web browser window with the URL [https://github.com/qabrevilla/CPE232\\_Revilla](https://github.com/qabrevilla/CPE232_Revilla). The repository is public and has 1 branch (main) and 0 tags. The commit history table shows the following entries:

File	Author	Commit Message	Time
README.md	Lopit men		2 weeks ago
example	Changes		2 minutes ago

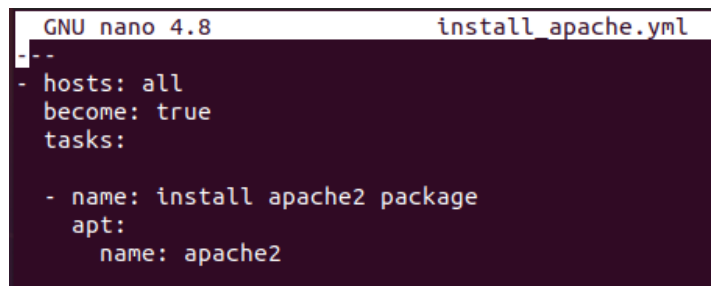
The README.md file content is displayed below the table, showing the title **CPE232\_Revilla** and the text `hi`.

## Task 2: Writing our First Playbook

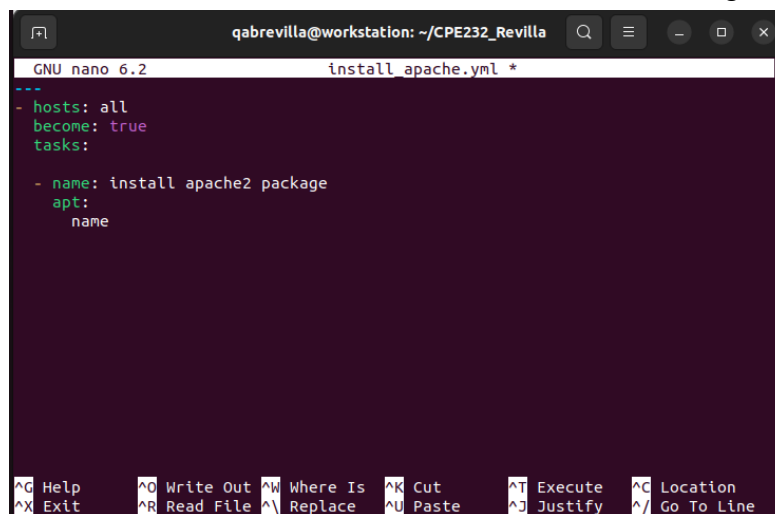
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and 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:



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.



```

qabrevilla@workstation:~/CPE232_Revilla$ ansible-playbook --ask-become-pass inst
all_apache.yml
BECOME password:

PLAY [all] *****

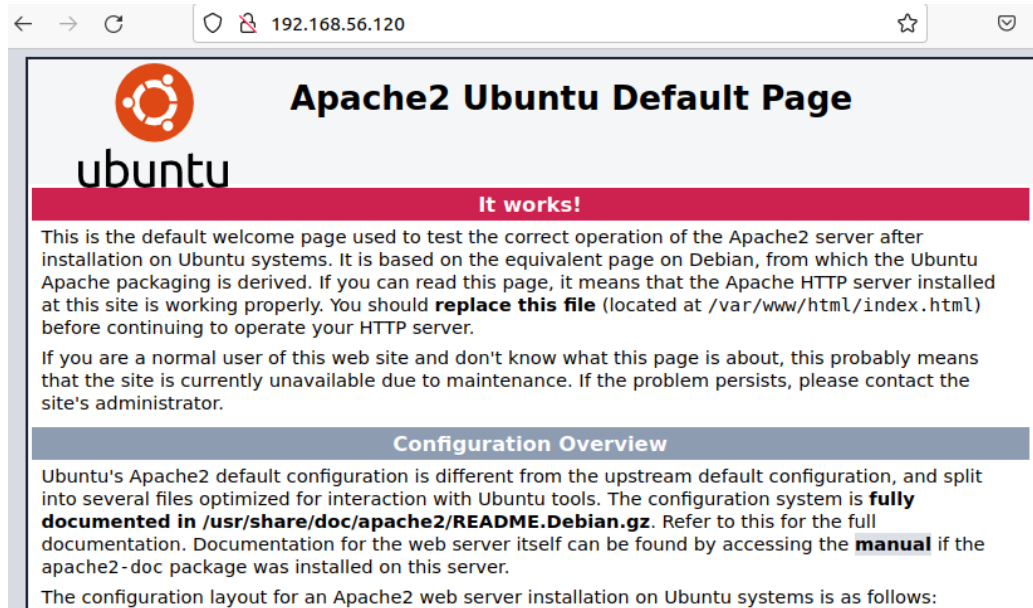
TASK [Gathering Facts] *****
ok: [192.0.0.1]

TASK [install apache2 package] *****
changed: [192.0.0.1]

PLAY RECAP *****
192.0.0.1 : ok=2 changed=1 unreachable=0 failed=0 s
kipped=0 rescued=0 ignored=0

```

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.





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?  
Changing the name of the package that will not be recognized will result in an error. The error may tell the problem and give instruction on fixing the error.
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
```

Save the changes to this file and exit.

```
GNU nano 6.2                                install_apache.yml *
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

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

```
gabrevilla@workstation:~/CPE232_Revilla$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.0.0.1]

TASK [install apache2 package] *****
changed: [192.0.0.1]

PLAY RECAP *****
192.0.0.1          : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=
0               ignored=0
```

After executing the playbook, changes occur in the server. Also some packages are updated

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

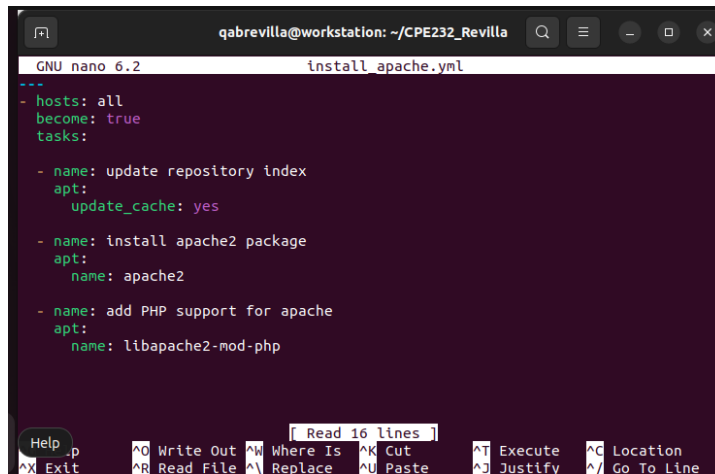
```
---
- hosts: all
  become: true
  tasks:

    - 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.



```
qabrevilla@workstation: ~/CPE232_Revilla
GNU nano 6.2      install apache.yml
---
- 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
```

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

```

qabrevilla@workstation:~/CPE232_Revilla$ ansible-playbook --ask-become-pass inst
all_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.0.0.1]

TASK [update repository index] *****
changed: [192.0.0.1]

TASK [install apache2 package] *****
ok: [192.0.0.1]

TASK [add PHP support for apache] *****
changed: [192.0.0.1]

PLAY RECAP *****
192.0.0.1 : ok=4    changed=2    unreachable=0    failed=0    s
Help

```

The command did change on the remote server allowing PHP support for apache.

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

[https://github.com/qabrevilla/CPE232\\_Revilla](https://github.com/qabrevilla/CPE232_Revilla)

### Reflections:

Answer the following:

1. What is the importance of using a playbook?

Using a playbook as a system administrator enables them to automate tasks reducing time and errors. It enhances the organization of the task by being consistent and ensuring that all tasks are performed efficiently. Playbooks are also good for its adaptability, meaning it can develop and improve over time. Playbook's documents are not static and can be changed by the user.

2. Summarize what we have done on this activity.

In this activity, I learned how to install ansible in Ubuntu and configure it as we connect it to the local servers and git repository. The activity gives us a new experience of exploring playbooks and ansible. The activity challenged us on how deep our understanding in installing plugins in ubuntu. The activity did not give any instructions on creating a setup for ansible because it assumed that we already have knowledge in basic configuration. I also experience writing playbooks and executing different variations of commands. In conclusion, the activity was able to introduce us to a new platform of ansible and git. They are used to remotely change

servers, git repositories, update, and upgrade packages. Playbooks are used to manage configuration and deploy changes on remote servers. And Ansible are tools that can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.