Name: DEAN LENARD PEREZ	**Date Performed : 13/09/24**
Course/Section: CPE31S21	**Date Submitted: 14/09/24**
Instructor:	**Semester and SY: 202024-2025**

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.

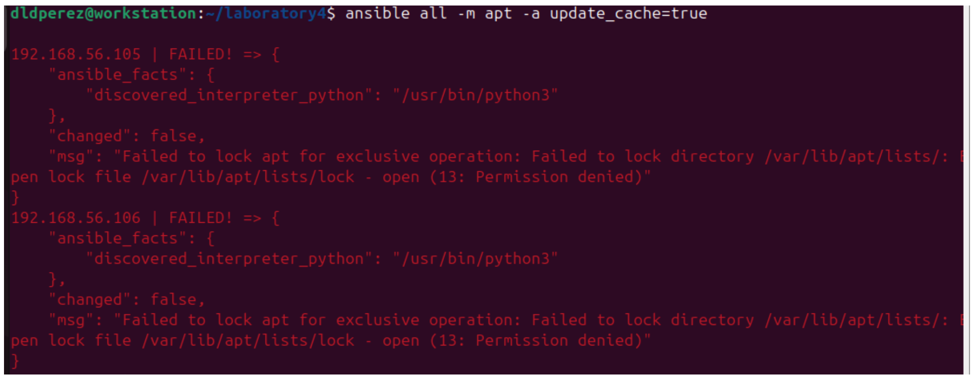
Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, 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

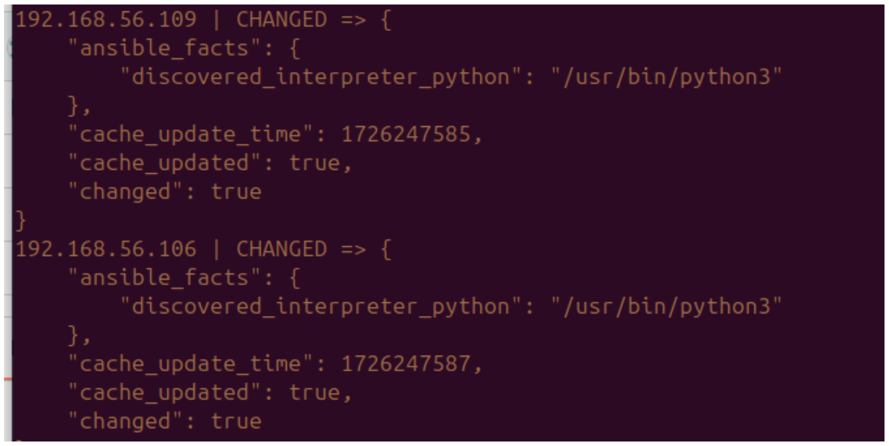


```
dldperez@workstation:~/laboratory4$ ansible all -m apt -a update_cache=true

192.168.56.105 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/lists/: 
pen lock file /var/lib/apt/lists/lock - open (13: Permission denied)"
}
192.168.56.106 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/lists/: 
pen lock file /var/lib/apt/lists/lock - open (13: Permission denied)"
}
```

What is the result of the command? Is it successful? **No, it was not.**

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.

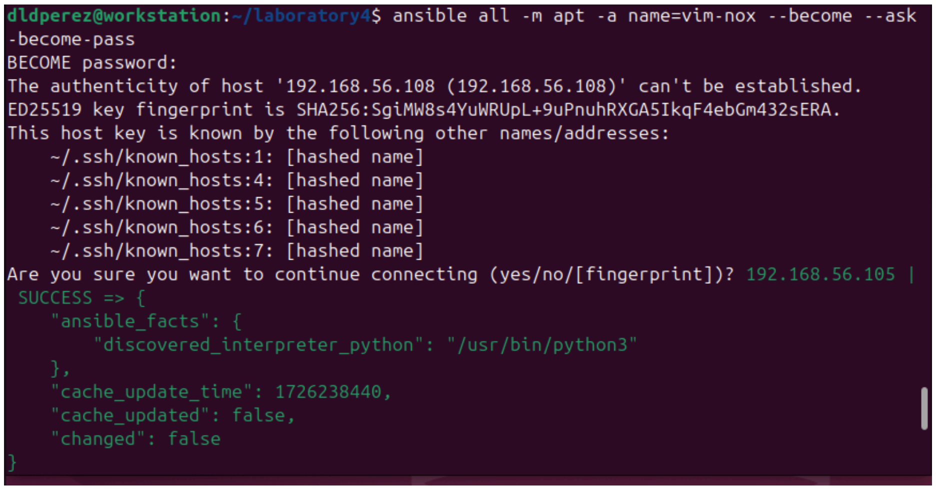
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.



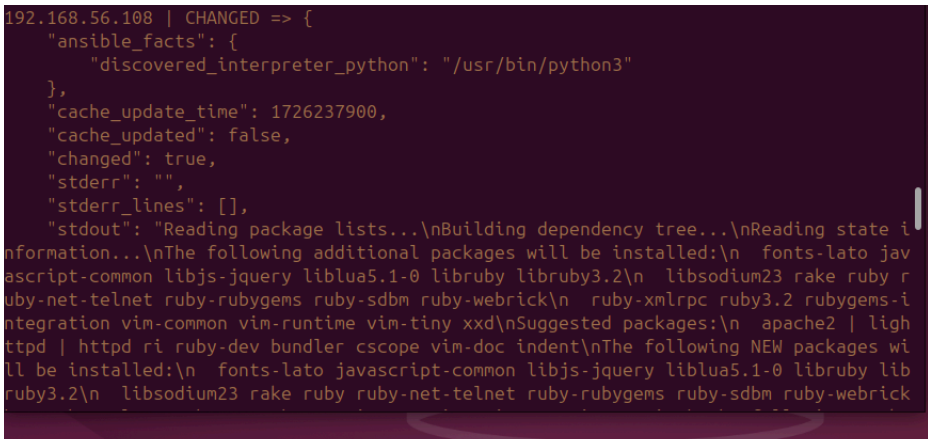
```
192.168.56.109 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247585,
    "cache_updated": true,
    "changed": true
}
192.168.56.106 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247587,
    "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.

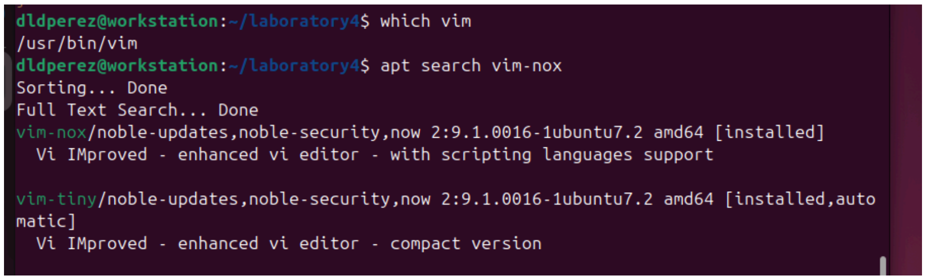


```
dldperez@workstation:~/laboratory4$ ansible all -m apt -a name=vim-nox --become --ask
-become-pass
BECOME password:
The authenticity of host '192.168.56.108 (192.168.56.108)' can't be established.
ED25519 key fingerprint is SHA256:SgiMW8s4YuWRUpL+9uPnuhRXGA5IkqF4ebGm432sERA.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
    ~/.ssh/known_hosts:4: [hashed name]
    ~/.ssh/known_hosts:5: [hashed name]
    ~/.ssh/known_hosts:6: [hashed name]
    ~/.ssh/known_hosts:7: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? 192.168.56.105 |
 SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726238440,
    "cache_updated": false,
    "changed": false
}
```



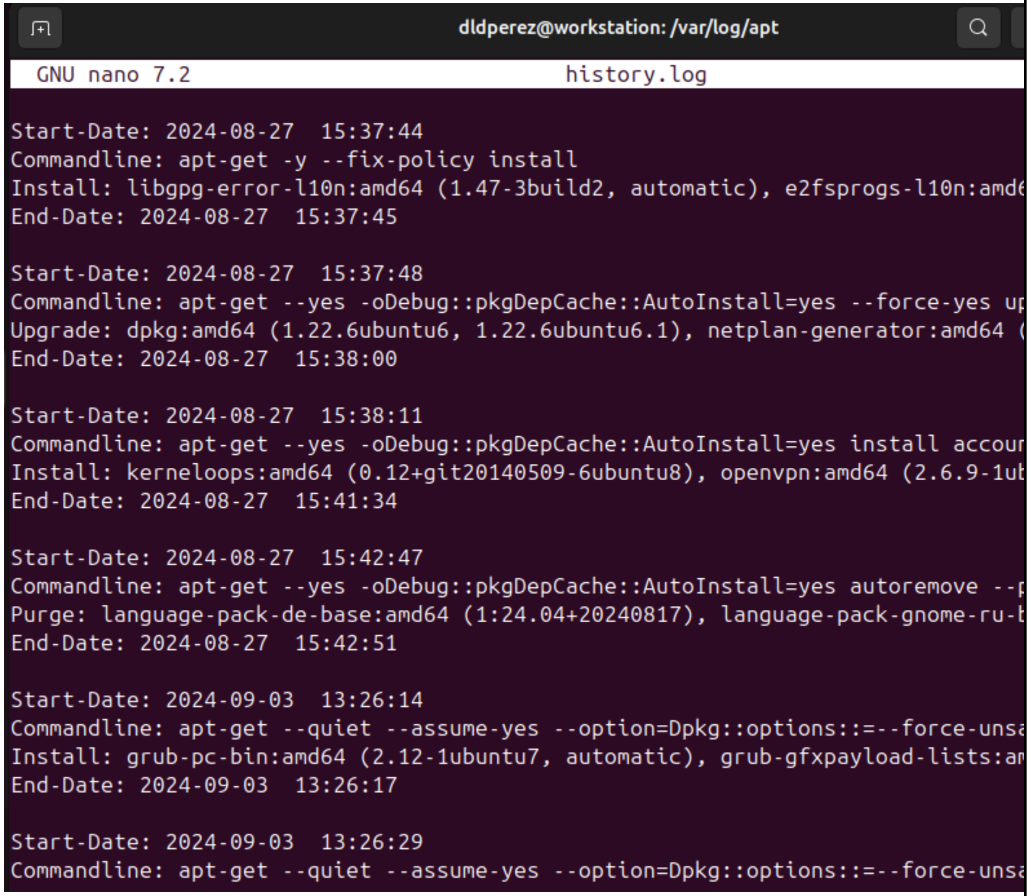
```
192.168.56.108 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726237900,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state i
nformation...\nThe following additional packages will be installed:\n  fonts-lato jav
ascript-common libjs-jquery liblua5.1-0 libruby libruby3.2\n  libsodium23 rake ruby r
uby-net-telnet ruby-rubygems ruby-sdbm ruby-webrick\n  ruby-xmlrpc ruby3.2 rubygems-i
ntegration vim-common vim-runtime vim-tiny xxd\nSuggested packages:\n  apache2 | ligh
ttpd | httpd ri ruby-dev bundler cscope vim-doc indent\nThe following NEW packages wi
ll be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.1-0 libruby lib
ruby3.2\n  libsodium23 rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-webrick
```

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? **Yes.**



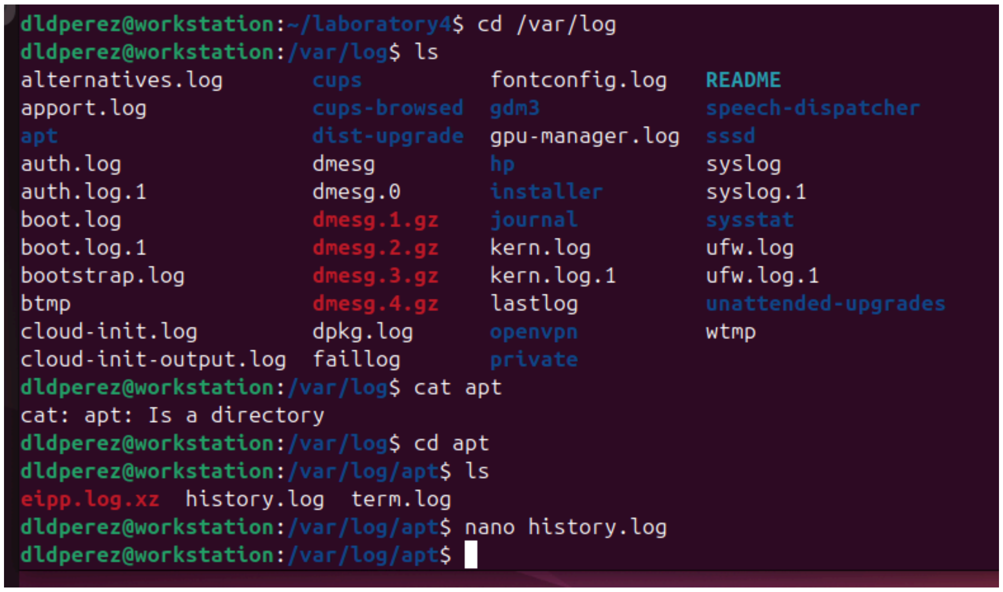
```
dldperez@workstation:~/laboratory4$ which vim
/usr/bin/vim
dldperez@workstation:~/laboratory4$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.2 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.2 amd64 [installed,auto
matic]
  Vi IMproved - enhanced vi editor - compact version
```

2.1 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.



dldperez@workstation: /var/log/apt

```
 GNU nano 7.2                               history.log

Start-Date: 2024-08-27  15:37:44
Commandline: apt-get -y --fix-policy install
Install: libgpg-error-l10n:amd64 (1.47-3build2, automatic), e2fsprogs-l10n:amd6
End-Date: 2024-08-27  15:37:45

Start-Date: 2024-08-27  15:37:48
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes up
Upgrade: dpkg:amd64 (1.22.6ubuntu6, 1.22.6ubuntu6.1), netplan-generator:amd64 (
End-Date: 2024-08-27  15:38:00

Start-Date: 2024-08-27  15:38:11
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install accour
Install: kerneloops:amd64 (0.12+git20140509-6ubuntu8), openvpn:amd64 (2.6.9-1ut
End-Date: 2024-08-27  15:41:34

Start-Date: 2024-08-27  15:42:47
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes autoremove --p
Purge: language-pack-de-base:amd64 (1:24.04+20240817), language-pack-gnome-ru-t
End-Date: 2024-08-27  15:42:51

Start-Date: 2024-09-03  13:26:14
Commandline: apt-get --quiet --assume-yes --option=Dpkg::options::=--force-unsa
Install: grub-pc-bin:amd64 (2.12-1ubuntu7, automatic), grub-gfxpayload-lists:an
End-Date: 2024-09-03  13:26:17

Start-Date: 2024-09-03  13:26:29
Commandline: apt-get --quiet --assume-yes --option=Dpkg::options::=--force-unsa
```

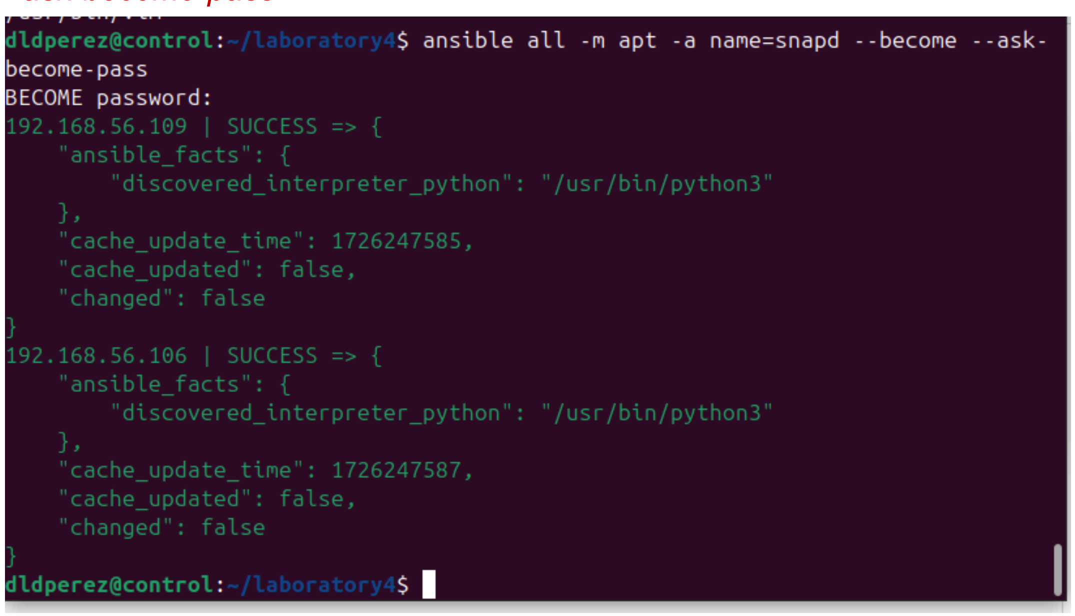


```
dldperez@workstation:~/laboratory4$ cd /var/log
dldperez@workstation:/var/log$ ls
alternatives.log       cups             fontconfig.log   README
apport.log             cups-browsed     gdm3             speech-dispatcher
apt                    dist-upgrade     gpu-manager.log  sssd
auth.log               dmesg            hp               syslog
auth.log.1             dmesg.0          installer        syslog.1
boot.log               dmesg.1.gz       journal          sysstat
boot.log.1             dmesg.2.gz       kern.log         ufw.log
bootstrap.log          dmesg.3.gz       kern.log.1       ufw.log.1
btmp                   dmesg.4.gz       lastlog          unattended-upgrades
cloud-init.log         dpkg.log         openvpn          wtmp
cloud-init-output.log  faillog          private
dldperez@workstation:/var/log$ cat apt
cat: apt: Is a directory
dldperez@workstation:/var/log$ cd apt
dldperez@workstation:/var/log/apt$ ls
eipp.log.xz  history.log  term.log
dldperez@workstation:/var/log/apt$ nano history.log
dldperez@workstation:/var/log/apt$
```

It shows the different installations' information and the date when it was installed.

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*

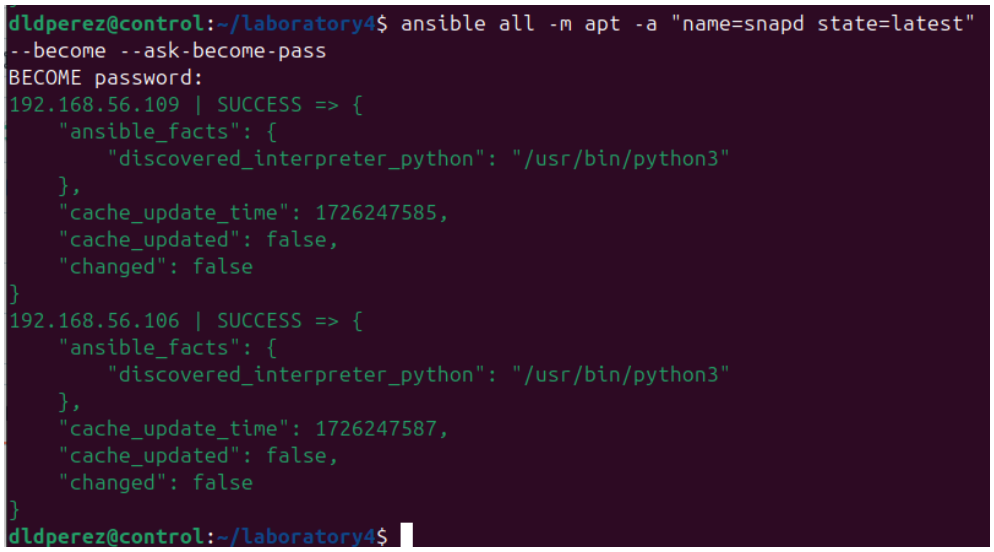


```
dldperez@control:~/laboratory4$ ansible all -m apt -a name=snapd --become --ask-
become-pass
BECOME password:
192.168.56.109 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247585,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247587,
    "cache_updated": false,
    "changed": false
}
dldperez@control:~/laboratory4$ 
```

   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?
   It was a success but it didn't change anything, it just installed the package.

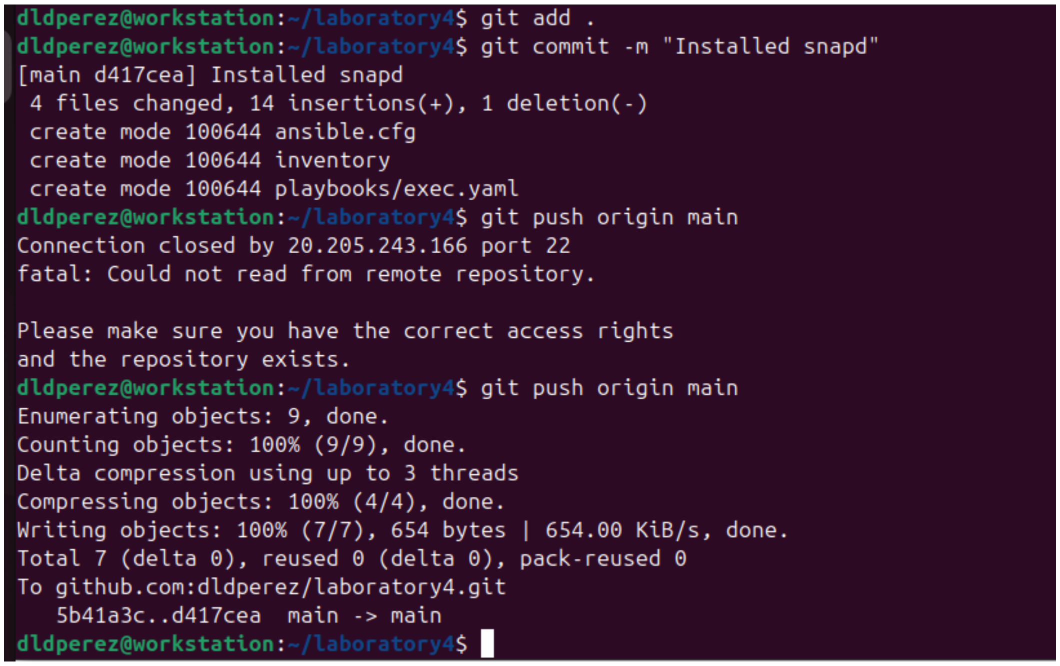
   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*

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



```
dldperez@control:~/laboratory4$ ansible all -m apt -a "name=snapd state=latest"
--become --ask-become-pass
BECOME password:
192.168.56.109 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247585,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726247587,
    "cache_updated": false,
    "changed": false
}
dldperez@control:~/laboratory4$
```

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



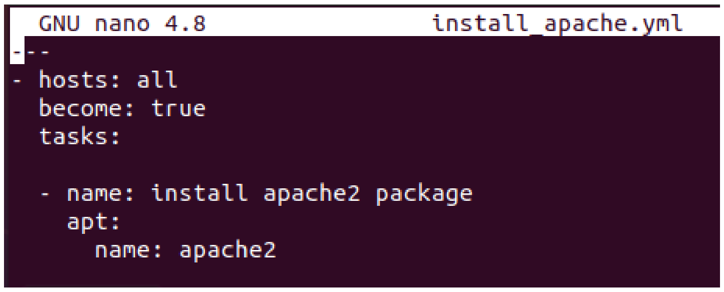
```
dldperez@workstation:~/laboratory4$ git add .
dldperez@workstation:~/laboratory4$ git commit -m "Installed snapd"
[main d417cea] Installed snapd
 4 files changed, 14 insertions(+), 1 deletion(-)
 create mode 100644 ansible.cfg
 create mode 100644 inventory
 create mode 100644 playbooks/exec.yaml
dldperez@workstation:~/laboratory4$ git push origin main
Connection closed by 20.205.243.166 port 22
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
dldperez@workstation:~/laboratory4$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 3 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 654 bytes | 654.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:dldperez/laboratory4.git
   5b41a3c..d417cea  main -> main
dldperez@workstation:~/laboratory4$
```

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

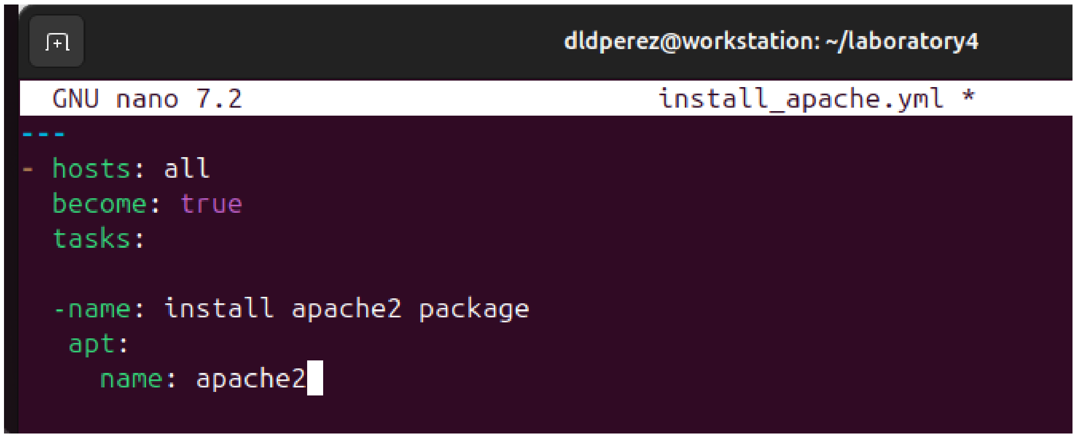
in and the place or commands that ansible will carry out to bring to that state. You can use an editor to create a playbook. Before we proceed, make sure that you are in the directory of the repository that we use in the previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. 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:



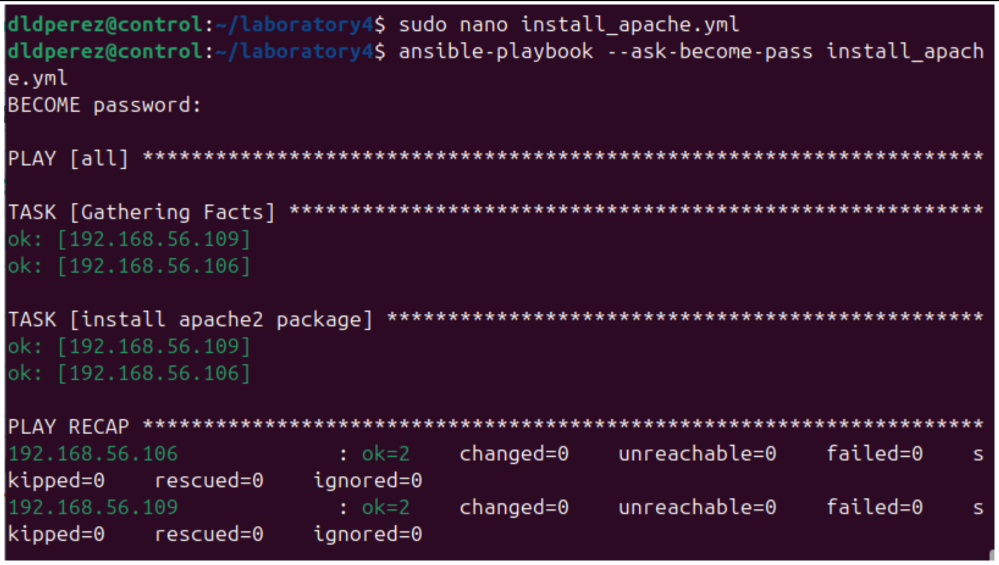
```
 GNU nano 4.8                       install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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



```
 GNU nano 7.2                       install_apache.yml *
---
- hosts: all
  become: true
  tasks:

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

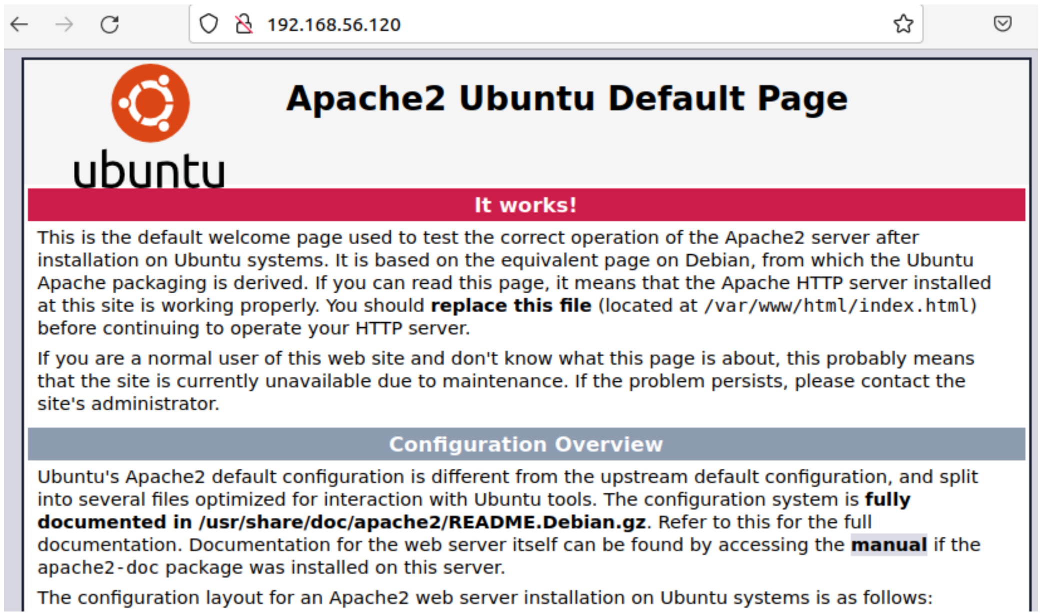
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.



```
dldperez@control:~/laboratory4$ sudo nano install_apache.yml
dldperez@control:~/laboratory4$ ansible-playbook --ask-become-pass install_apach
e.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [install apache2 package] *************************************************
ok: [192.168.56.109]
ok: [192.168.56.106]

PLAY RECAP *********************************************************************
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.109             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

It showed the different servers ip addresses and it was installed. There are two tasks: gathering the facts and doing the installation. And it also shows the details or recap of the installed servers.

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.



192.168.56.120

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

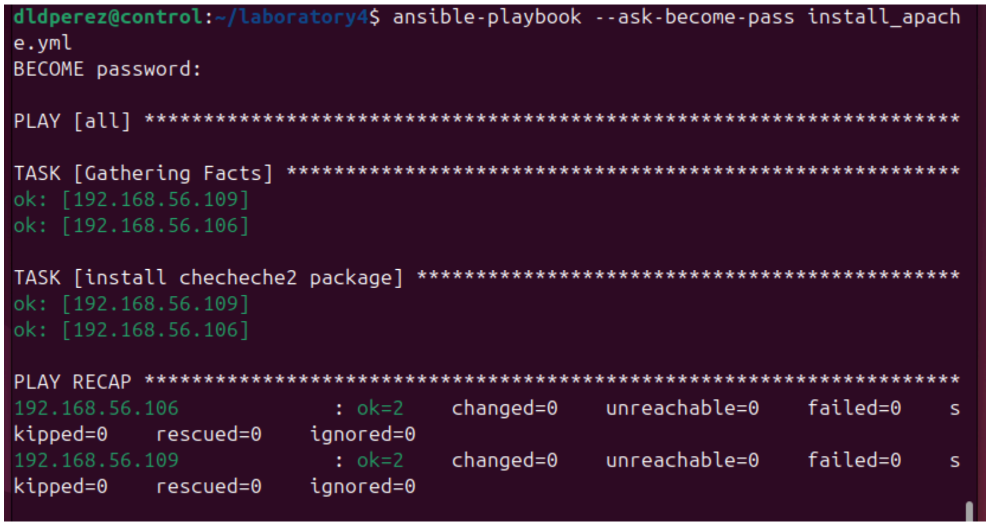
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

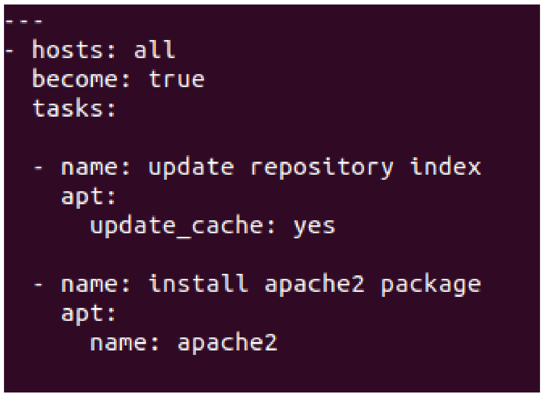
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?



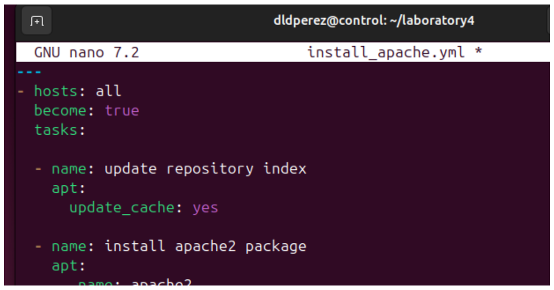
```
dldperez@control:~/laboratory4$ ansible-playbook --ask-become-pass install_apach
e.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [install checheche2 package] **********************************************
ok: [192.168.56.109]
ok: [192.168.56.106]

PLAY RECAP *********************************************************************
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.109             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Same result.

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
```

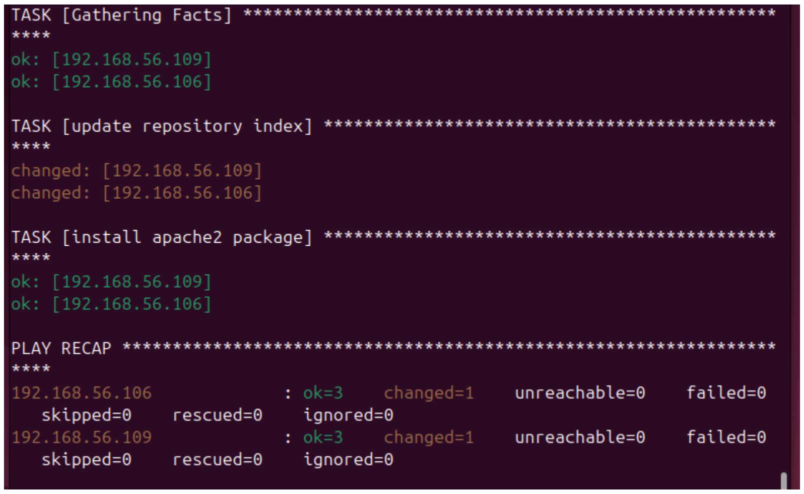


dldperez@control: ~/laboratory4

```
GNU nano 7.2                  install_apache.yml *
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
```

Save the changes to this file and exit.

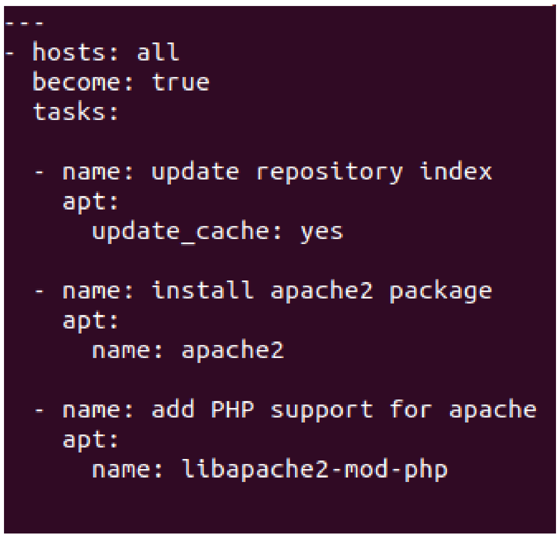
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
TASK [Gathering Facts] *******************************************************
****
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [update repository index] ***********************************************
****
changed: [192.168.56.109]
changed: [192.168.56.106]

TASK [install apache2 package] ***********************************************
****
ok: [192.168.56.109]
ok: [192.168.56.106]

PLAY RECAP *******************************************************************
****
192.168.56.106             : ok=3    changed=1    unreachable=0    failed=0
   skipped=0    rescued=0    ignored=0
192.168.56.109             : ok=3    changed=1    unreachable=0    failed=0
   skipped=0    rescued=0    ignored=0
```

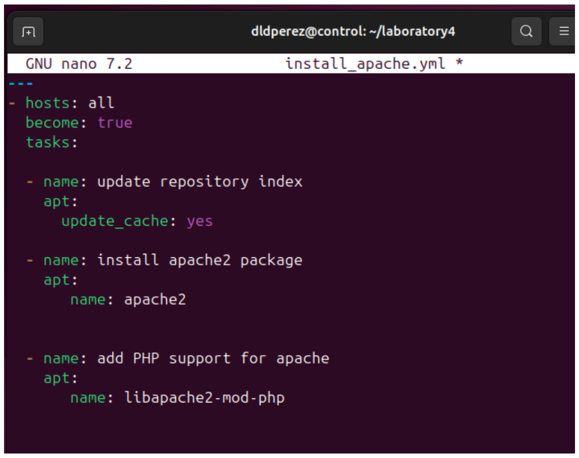
Yes, there are some changes. A new task appeared which is the update, and the recap also changed.

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



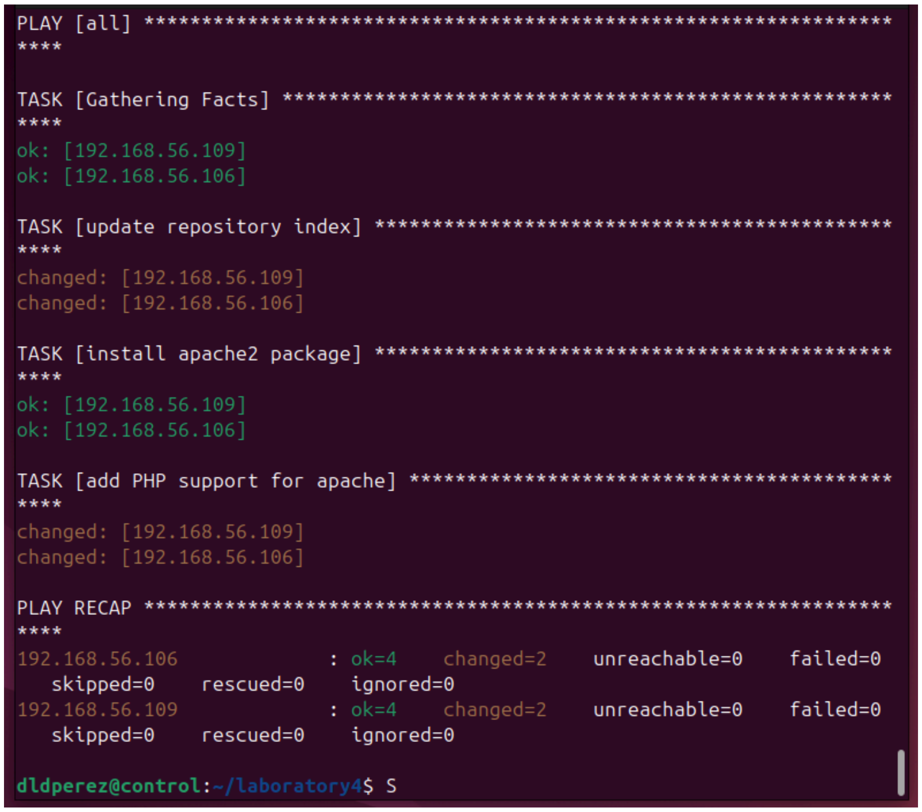
```
---
- hosts: all
  become: true
  tasks:

  - 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.



```
dldperez@control: ~/laboratory4
GNU nano 7.2                        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? Yes it did.



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

TASK [Gathering Facts] *****************************************************
****
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [update repository index] *********************************************
****
changed: [192.168.56.109]
changed: [192.168.56.106]

TASK [install apache2 package] *********************************************
****
ok: [192.168.56.109]
ok: [192.168.56.106]

TASK [add PHP support for apache] ******************************************
****
changed: [192.168.56.109]
changed: [192.168.56.106]

PLAY RECAP *****************************************************************
****
192.168.56.106             : ok=4    changed=2    unreachable=0    failed=0
   skipped=0    rescued=0    ignored=0
192.168.56.109             : ok=4    changed=2    unreachable=0    failed=0
   skipped=0    rescued=0    ignored=0

dldperez@control:~/laboratory4$ S
```

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

https://github.com/dldperez/laboratory4.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?

   It’s importance is to lessen the burden and to make fast download or update to numbers of users by just using a single phrase or command codes that will run to all users that are connected to it.

2. Summarize what we have done on this activity.

We used basic ansible commands to create changes to the remote server. We also learned about the use of playbook that can lessen our work significantly. We also installed different packages to remote servers.