Name: Davonn Escobilla	Date Performed: 14/09/2022
Course/Section: CPE31S24	Date Submitted: 16/09/2022
Instructor: Dr. Jonathan Taylar	Semester and SY: 1st Sem, 2022-2023

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?

```
davonn@workstation:~$ ansible all -m apt -a update_cache=true
192.168.56.101 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
192.168.56.103 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
```

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.

```
davonn@workstation:~$ ansible all -m apt -a update_cache=true --become --ask-be
come-pass
BECOME password:
192.168.56.103 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249433,
    "cache_updated": true,
    "changed": true
}
192.168.56.101 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249481,
    "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.

```
davonn@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-
pass
BECOME password:
192.168.56.101 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249481,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libfwupdplugin1\nUse 'sudo apt autoremove' to remove it.\
nThe following additional packages will be installed:\n  fonts-lato javascript-
common libjs-jquery liblua5.2-0 libruby2.7 libtcl8.6\n  rake ruby ruby-minitest
 ruby-net-telnet ruby-power-assert ruby-test-unit\n  ruby-xmlrpc ruby2.7 rubyge
ms-integration vim-common vim-runtime vim-tiny\nSuggested packages:\n  apache2
| lighttpd | httpd tcl8.6 ri ruby-dev bundler cscope vim-doc indent\nThe follow
ing NEW packages will be installed:\n  fonts-lato javascript-common libjs-jquer
y liblua5.2-0 libruby2.7 libtcl8.6\n  rake ruby ruby-minitest ruby-net-telnet r
uby-power-assert ruby-test-unit\n  ruby-xmlrpc ruby2.7 rubygems-integration vim
-nox vim-runtime\nThe following packages will be upgraded:\n  vim-common vim-ti
ny\n2 upgraded, 17 newly installed, 0 to remove and 24 not upgraded.\nNeed to g
et 15.8 MB of archives.\nAfter this operation, 70.9 MB of additional disk space
 will be used.\nGet:1 http://security.ubuntu.com/ubuntu focal-security/main amd
```

```
192.168.56.103 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249433,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libfwupdplugin1\nUse 'sudo apt autoremove' to remove it.\
nThe following additional packages will be installed:\n  fonts-lato javascript-
common libjs-jquery liblua5.2-0 libruby2.7 libtcl8.6\n  rake ruby ruby-minitest
 ruby-net-telnet ruby-power-assert ruby-test-unit\n  ruby-xmlrpc ruby2.7 rubyge
ms-integration vim-common vim-runtime vim-tiny\nSuggested packages:\n  apache2
| lighttpd | httpd tcl8.6 ri ruby-dev bundler cscope vim-doc indent\nThe follow
ing NEW packages will be installed:\n  fonts-lato javascript-common libjs-jquer
y liblua5.2-0 libruby2.7 libtcl8.6\n  rake ruby ruby-minitest ruby-net-telnet r
uby-power-assert ruby-test-unit\n  ruby-xmlrpc ruby2.7 rubygems-integration vim
-nox vim-runtime\nThe following packages will be upgraded:\n  vim-common vim-ti
ny\n2 upgraded, 17 newly installed, 0 to remove and 8 not upgraded.\nNeed to ge
t 15.8 MB of archives.\nAfter this operation, 70.9 MB of additional disk space
```

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?

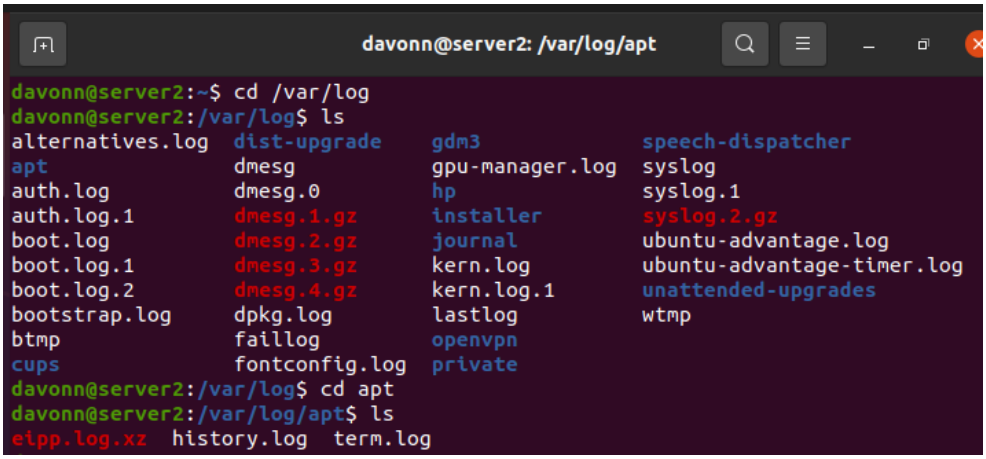
```
davonn@server1: ~
davonn@server1:~$ which vim
/usr/bin/vim
davonn@server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/focal-updates,focal-security,now 2:8.1.2269-1ubuntu5.8 amd64 [installed
]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/focal-updates,focal-security,now 2:8.1.2269-1ubuntu5.8 amd64 [installe
d,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

```
davonn@server2: ~
davonn@server2:~$ which vim
/usr/bin/vim
davonn@server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/focal-security,now 2:8.1.2269-1ubuntu5.8 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/focal-security,now 2:8.1.2269-1ubuntu5.8 amd64 [installed,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

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.

```
davonn@server1:~$ cd /var/log
davonn@server1:/var/log$ ls
alternatives.log  dist-upgrade    gdm3              speech-dispatcher
apt               dmesg           gpu-manager.log   syslog
auth.log          dmesg.0         hp                syslog.1
auth.log.1        dmesg.1.gz      installer         syslog.2.gz
boot.log          dmesg.2.gz      journal           ubuntu-advantage.log
boot.log.1        dmesg.3.gz      kern.log          ubuntu-advantage-timer.log
boot.log.2        dmesg.4.gz      kern.log.1        unattended-upgrades
bootstrap.log     dpkg.log        lastlog           wtmp
btmp              faillog         openvpn
cups              fontconfig.log  private
davonn@server1:/var/log$ cd apt
davonn@server1:/var/log/apt$ ls
eipp.log.xz  history.log  term.log
```


davonn@server1: /var/log/apt
GNU nano 4.8 history.log
Start-Date: 2022-02-23 08:47:33
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install ubunt>
Upgrade: ubuntu-keyring:amd64 (2020.02.11.2, 2020.02.11.4)
End-Date: 2022-02-23 08:47:33
Start-Date: 2022-02-23 08:47:36
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes u>
Upgrade: libpam0g:amd64 (1.3.1-5ubuntu4, 1.3.1-5ubuntu4.3), fdisk:amd64 (2.34->
End-Date: 2022-02-23 08:47:50
Start-Date: 2022-02-23 08:47:50
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes d>
Install: iso-codes:amd64 (4.4-1, automatic), python-apt-common:amd64 (2.0.0ubu>
Upgrade: ubuntu-advantage-tools:amd64 (20.3, 27.6~20.04.1)
End-Date: 2022-02-23 08:47:52
Start-Date: 2022-02-23 08:48:24
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install linux>
Install: speech-dispatcher-espeak-ng:amd64 (0.9.1-4), speech-dispatcher-audio->
End-Date: 2022-02-23 08:51:24


davonn@server2: /var/log/apt
davonn@server2:~$ cd /var/log
davonn@server2:/var/log$ ls
alternatives.log dist-upgrade gdm3 speech-dispatcher
apt dmesg gpu-manager.log syslog
auth.log dmesg.0 hp syslog.1
auth.log.1 dmesg.1.gz installer syslog.2.gz
boot.log dmesg.2.gz journal ubuntu-advantage.log
boot.log.1 dmesg.3.gz kern.log ubuntu-advantage-timer.log
boot.log.2 dmesg.4.gz kern.log.1 unattended-upgrades
bootstrap.log dpkg.log lastlog wtmp
btmp faillog openvpn
cups fontconfig.log private
davonn@server2:/var/log$ cd apt
davonn@server2:/var/log/apt$ ls
eipp.log.xz history.log term.log

```
davonn@server2: /var/log/apt
GNU nano 4.8                    history.log

Start-Date: 2022-02-23  08:47:33
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install ubunt>
Upgrade: ubuntu-keyring:amd64 (2020.02.11.2, 2020.02.11.4)
End-Date: 2022-02-23  08:47:33

Start-Date: 2022-02-23  08:47:36
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes u>
Upgrade: libpam0g:amd64 (1.3.1-5ubuntu4, 1.3.1-5ubuntu4.3), fdisk:amd64 (2.34->
End-Date: 2022-02-23  08:47:50

Start-Date: 2022-02-23  08:47:50
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes d>
Install: iso-codes:amd64 (4.4-1, automatic), python-apt-common:amd64 (2.0.0ubu>
Upgrade: ubuntu-advantage-tools:amd64 (20.3, 27.6~20.04.1)
End-Date: 2022-02-23  08:47:52

Start-Date: 2022-02-23  08:48:24
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install linux>
Install: speech-dispatcher-espeak-ng:amd64 (0.9.1-4), speech-dispatcher-audio->
End-Date: 2022-02-23  08:51:24

Start-Date: 2022-02-23  08:51:29
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install lupin>
Install: hunspell-en-gb:amd64 (1:6.4.3-1), cryptsetup-bin:amd64 (2:2.2.2-3ubun>
```

All the records on the logs are the installation dates of some packages and their respective status.

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

```
davonn@workstation:~$ ansible all -m apt -a name=snapd --become --ask-become-pa
ss
BECOME password:
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249433,
    "cache_updated": false,
    "changed": false
}
192.168.56.101 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249481,
    "cache_updated": false,
    "changed": false
}
```

While the command seems to be successful, we can see the “changed” value with false, it means it does not change anything in the remote servers. Since as from the description, it says that snapd is already installed in ubuntu.

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.

```
davonn@workstation:~$ ansible all -m apt -a "name=snapd state=latest" --become
--ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249433,
    "cache_updated": false,
    "changed": false
}
192.168.56.101 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249481,
    "cache_updated": false,
    "changed": false
}
```

The added command “state=latest” is to add another argument with a meaning of updating the package to the latest version. However, the snapd is already in its latest version that is why the output of the command is only the same as before.

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

```
davonn@workstation:~/CPE232_Davonn$ git add -A
davonn@workstation:~/CPE232_Davonn$ git commit
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
davonn@workstation:~/CPE232_Davonn$ git push
Everything up-to-date
```

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.

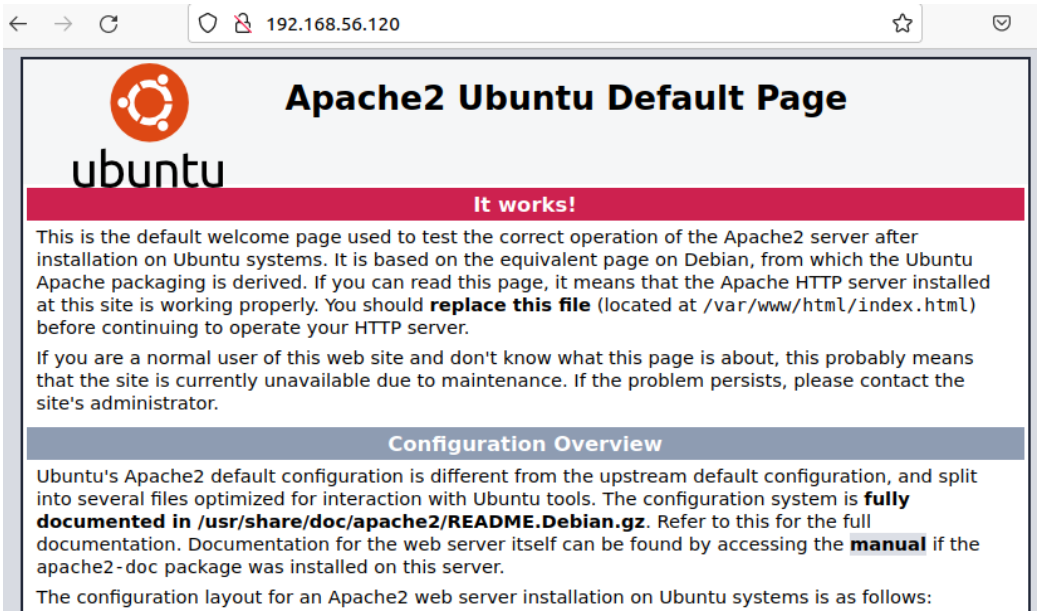
```
davonn@workstation: ~/CPE232_Davonn
  GNU nano 4.8                    install_apache.yml                    Modified
---
- hosts: all
  become: true
  tasks:

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

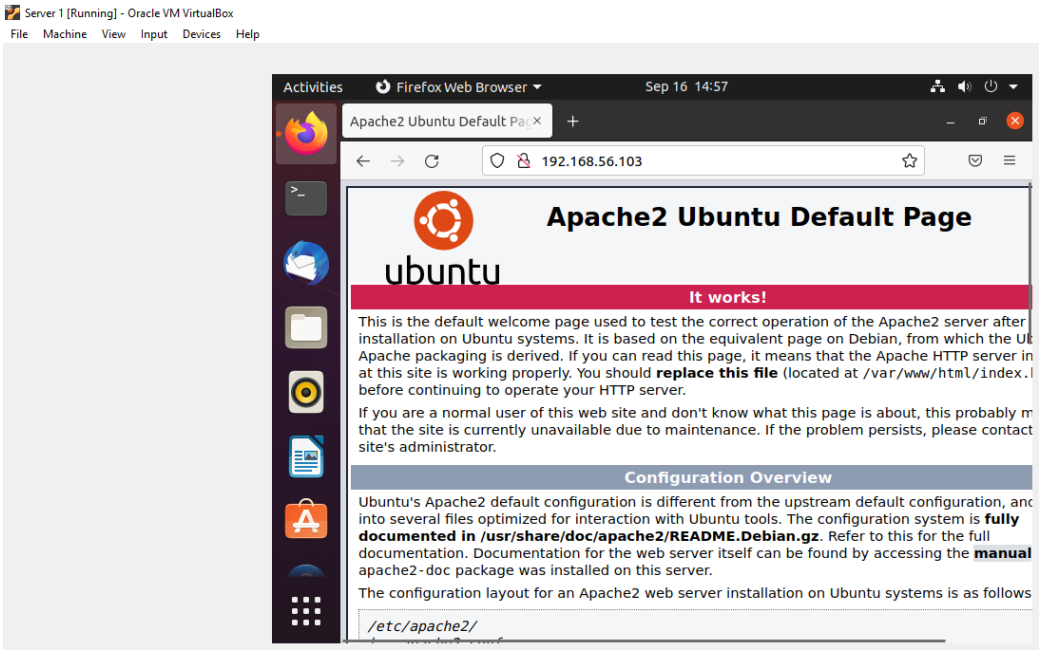
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

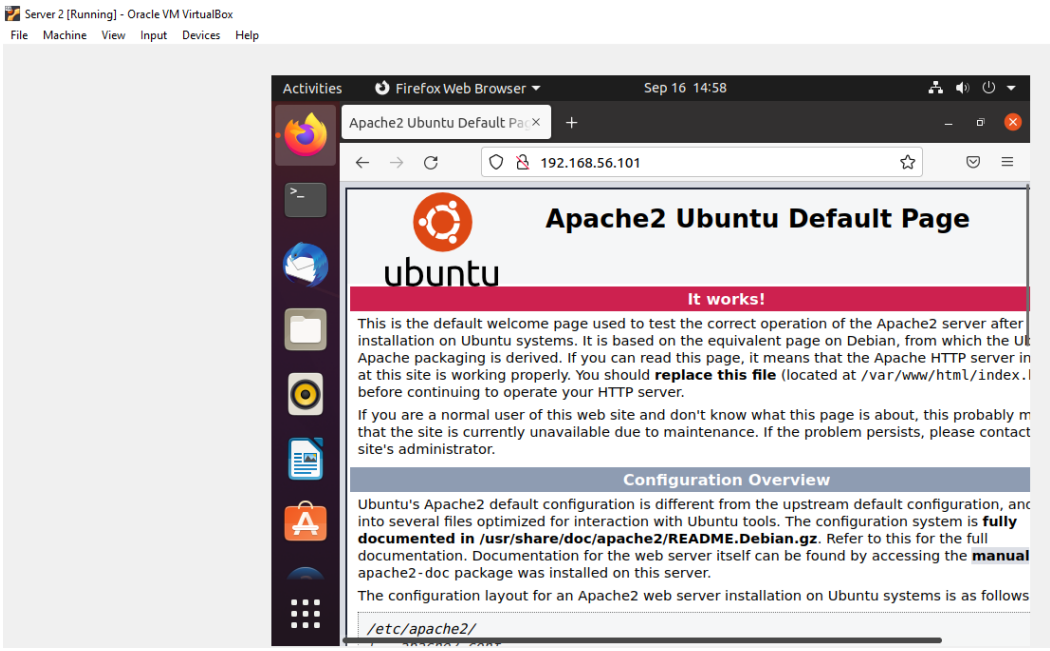
```
davonn@workstation:~/CPE232_Davonn$ ansible-playbook --ask-become-pass install_
apache.yml
BECOME password:

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

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

TASK [install apache2 package] ************************************************
*
changed: [192.168.56.101]
changed: [192.168.56.103]

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

The changes are applied on both servers successfully after connecting to them to determine if it is reachable.

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?

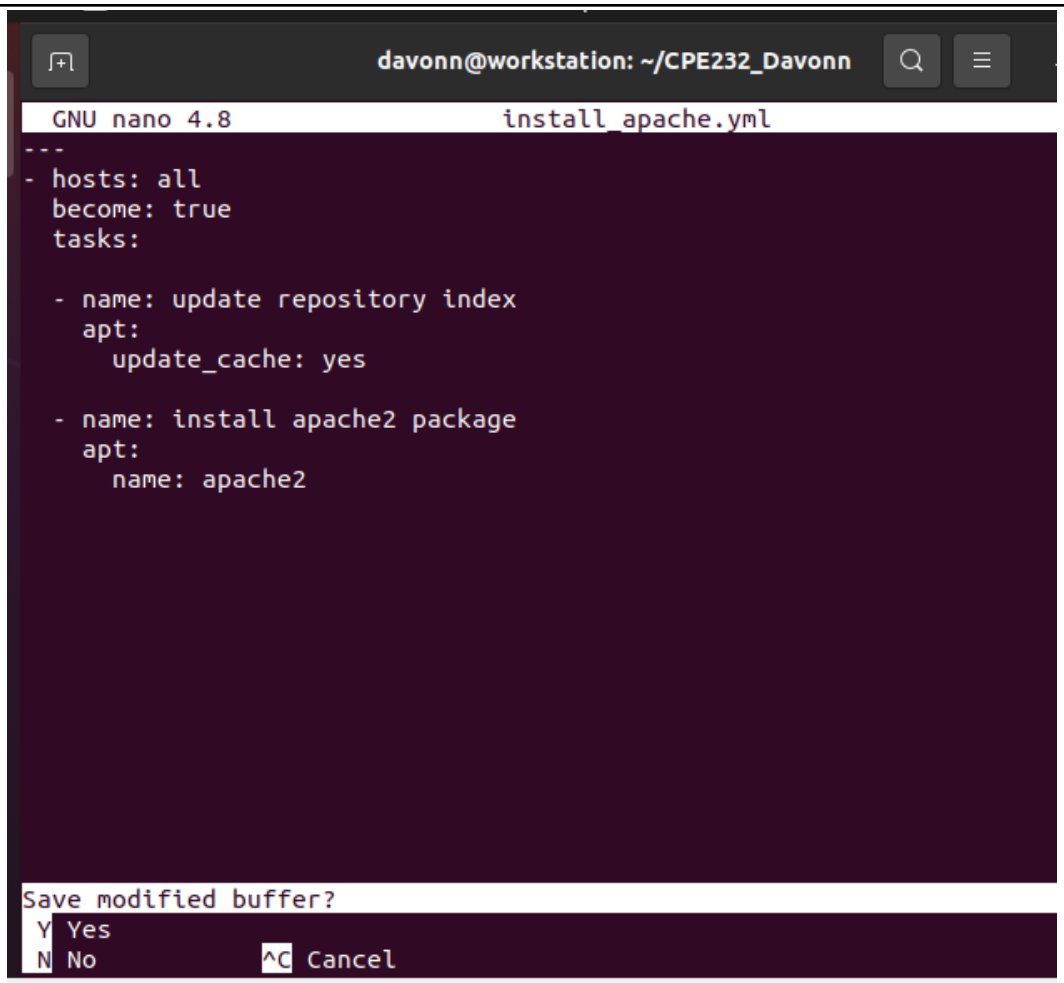
```
davonn@workstation:~/CPE232_Davonn$ nano install_apache.yml
davonn@workstation:~/CPE232_Davonn$ ansible-playbook --ask-become-pass install_
apache.yml
BECOME password:

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

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

TASK [install abc package] *****************************************************
*
ok: [192.168.56.103]
ok: [192.168.56.101]

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

There are no actual change as the previous package is installed.

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.



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

```
davonn@workstation: ~/CPE232_Davonn
davonn@workstation:~/CPE232_Davonn$ nano install_apache.yml
davonn@workstation:~/CPE232_Davonn$ ansible-playbook --ask-become-pass install_
apache.yml
BECOME password:

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

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

TASK [update repository index] ***************************************************
*
changed: [192.168.56.101]
changed: [192.168.56.103]

TASK [install apache2 package] ***************************************************
*
ok: [192.168.56.103]
ok: [192.168.56.101]

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

The repository from the remote servers are updated as we can see in the screenshot the status are changed together with the respective ip addresses of the servers.

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.

```
davonn@workstation: ~/CPE232_Davonn
GNU nano 4.8                    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
```

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

```
davonn@workstation: ~/CPE232_Davonn
*

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

TASK [update repository index] ***************************************************
*
changed: [192.168.56.101]
changed: [192.168.56.103]

TASK [install apache2 package] ***************************************************
*
ok: [192.168.56.101]
ok: [192.168.56.103]

TASK [add PHP support for apache] ************************************************
*
changed: [192.168.56.101]
changed: [192.168.56.103]

PLAY RECAP ***********************************************************************
*
192.168.56.101             : ok=4    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.103             : ok=4    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

The new command added PHP support for the existing package which is the apache2 package.

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

```
davonn@workstation:~/CPE232_Davonn$ git add -A
davonn@workstation:~/CPE232_Davonn$ git commit
Aborting commit due to empty commit message.
davonn@workstation:~/CPE232_Davonn$ git commit -m "Playbook added"
[main 7d4b22d] Playbook added
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
davonn@workstation:~/CPE232_Davonn$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 501 bytes | 501.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:DavonnEscobilla/CPE232_Davonn.git
   6a3fc65..7d4b22d  main -> main
```

Reflections:

Answer the following:

1. What is the importance of using a playbook?

   Playbooks can play a significant role in rolling out updates, changes, and configuration to the remote servers. It can also serve as a manual for serving automation.

2. Summarize what we have done on this activity.

   This activity is a building factor on controlling the changes and updates on the remote servers in order to deploy system configuration and functions. The ansible modules help to execute these commands on the remote servers which is a tool that is very useful. While the playbooks are the manuals that instruct the servers. It can be used for configurations or even in an advanced level of modification.

I affirm that I shall not give or receive any unauthorized help on this assignment and that all work shall be my own.