Name: Renier L. Lope	**Date Performed: 09/15/2024**
Course/Section: CPE212 – CPE31S2	**Date Submitted: 09/16/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 1st Sem(2024-2025)**

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

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible all -m apt -a update_cache=true
server1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /v
ar/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: P
ermission denied)"
}
server2 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /v
ar/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: P
ermission denied)"
}
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

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

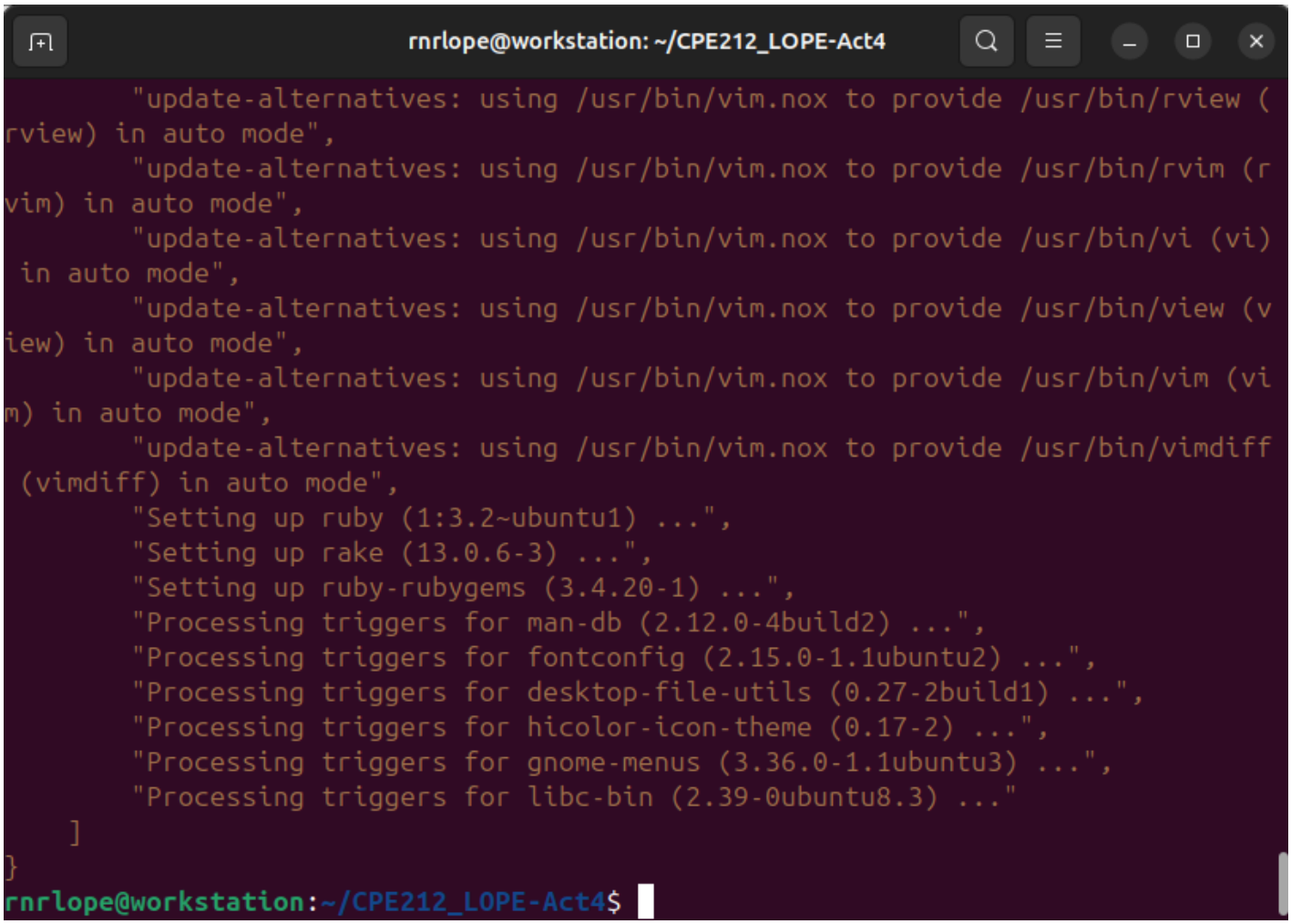
- **No, because I don't have the permission.**

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.

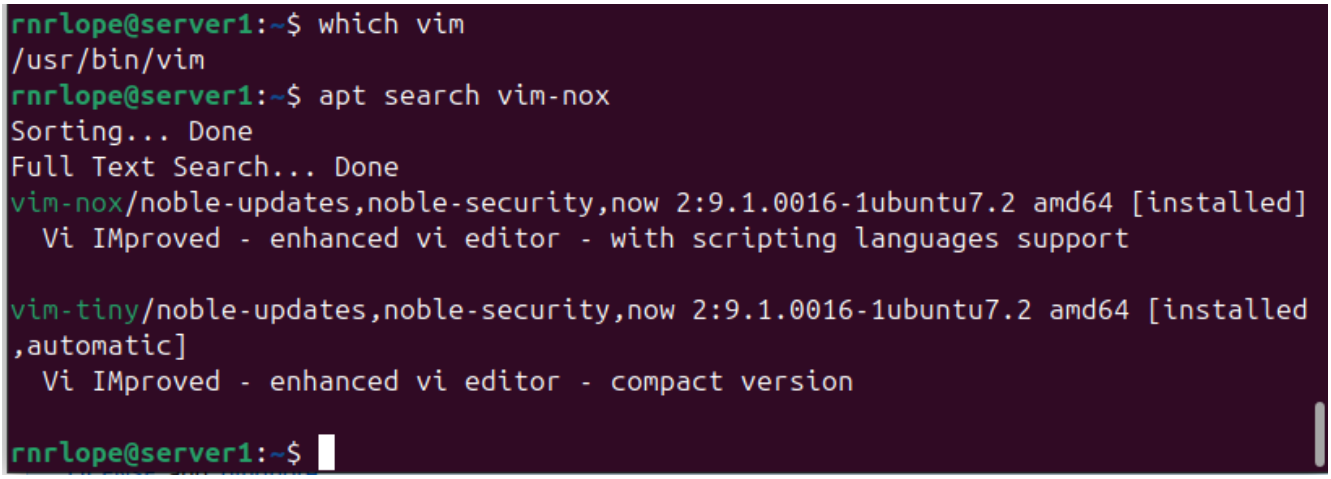
```
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible all -m apt -a update_cache=true
--become --ask-become-pass
BECOME password:
server2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415032,
    "cache_updated": true,
    "changed": true
}
server1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415034,
    "cache_updated": true,
    "changed": true
}
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

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

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.



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.



```
rnrlope@server2:~$ which vim
/usr/bin/vim
rnrlope@server2:~$ 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
,automatic]
  Vi IMproved - enhanced vi editor - compact version

rnrlope@server2:~$
```

Was the command successful? **- Yes.**

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.

```
rnrlope@server1:~$ cd /var/log
rnrlope@server1:/var/log$ ls
alternatives.log  cloud-init.log         faillog          README
apport.log        cloud-init-output.log  fontconfig.log   speech-dispatcher
apport.log.1      cups                   gdm3             sssd
apt               cups-browsed           gpu-manager.log  syslog
auth.log          dist-upgrade           hp               syslog.1
auth.log.1        dmesg                  installer        sysstat
boot.log          dmesg.0                journal          ubuntu-advantage.log
boot.log.1        dmesg.1.gz             kern.log         ufw.log
boot.log.2        dmesg.2.gz             kern.log.1       ufw.log.1
boot.log.3        dmesg.3.gz             lastlog          unattended-upgrades
bootstrap.log     dmesg.4.gz             openvpn          wtmp
btmp              dpkg.log               private
rnrlope@server1:/var/log$ cd apt
rnrlope@server1:/var/log/apt$ cat history.log

Start-Date: 2024-04-24  10:47:34
Commandline: apt-get -y --fix-policy install
Install: libgpg-error-l10n:amd64 (1.47-3build2, automatic), e2fsprogs-l10n:amd64
 (1.47.0-2.4~exp1ubuntu4, automatic), libgpm2:amd64 (1.20.7-11, automatic), psmi
```

```
rnrlope@server2:~$ cd /var/log
rnrlope@server2:/var/log$ ls
alternatives.log  btmp                 dpkg.log         private
apport.log        cloud-init.log       faillog          README
apport.log.1      cloud-init-output.log  fontconfig.log   speech-dispatcher
apport.log.2.gz   cups                 gdm3             sssd
apt               cups-browsed         gpu-manager.log  syslog
auth.log          dist-upgrade         hp               syslog.1
auth.log.1        dmesg                installer        sysstat
boot.log          dmesg.0              journal          ubuntu-advantage.log
boot.log.1        dmesg.1.gz           kern.log         ufw.log
boot.log.2        dmesg.2.gz           kern.log.1       ufw.log.1
boot.log.3        dmesg.3.gz           lastlog          unattended-upgrades
bootstrap.log     dmesg.4.gz           openvpn          wtmp
rnrlope@server2:/var/log$ cd apt
rnrlope@server2:/var/log/apt$ cat history.log

Start-Date: 2024-04-24  10:47:34
Commandline: apt-get -y --fix-policy install
Install: libgpg-error-l10n:amd64 (1.47-3build2, automatic), e2fsprogs-l10n:amd64
 (1.47.0-2.4~exp1ubuntu4, automatic), libgpm2:amd64 (1.20.7-11, automatic), psmi
sc:amd64 (23.7-1build1, automatic), uuid-runtime:amd64 (2.39.3-9ubuntu6, automat
ic), bash-completion:amd64 (1:2.11-8, automatic), bsdextrautils:amd64 (2.39.3-9u
buntu6, automatic)
```

Describe what you see in the history.log.

- **It displays all package changes that were made to the system via apt.**

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*

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible all -m apt -a name=snapd --becom
e --ask-become-pass
BECOME password:
server1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415168,
    "cache_updated": false,
    "changed": false
}
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415169,
    "cache_updated": false,
    "changed": false
}
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

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

- **It installed the snapd. Yes, it is a Success. It didn't change anything in the remotes servers because the package is already installed.**

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

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible all -m apt -a "name=snapd state=
latest" --become --ask-become-pass
BECOME password:
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415169,
    "cache_updated": false,
    "changed": false
}
server1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726415168,
    "cache_updated": false,
    "changed": false
}
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

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

- **This command would install the latest version of snapd.**

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

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ git add ansible.cfg
rnrlope@workstation:~/CPE212_LOPE-Act4$ git add inventory
rnrlope@workstation:~/CPE212_LOPE-Act4$ git commit -m "Activity 4.1"
[main (root-commit) 9452dd9] Activity 4.1
 2 files changed, 7 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 inventory
rnrlope@workstation:~/CPE212_LOPE-Act4$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 347 bytes | 347.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:RenierCode/CPE212_LOPE-Act4.git
 * [new branch]      main -> main
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

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.

```
rnrlope@workstation: ~/CPE212_LOPE-Act4
 GNU nano 7.2                    install_apache.yml
---
- hosts: all
  become: true
  task:


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

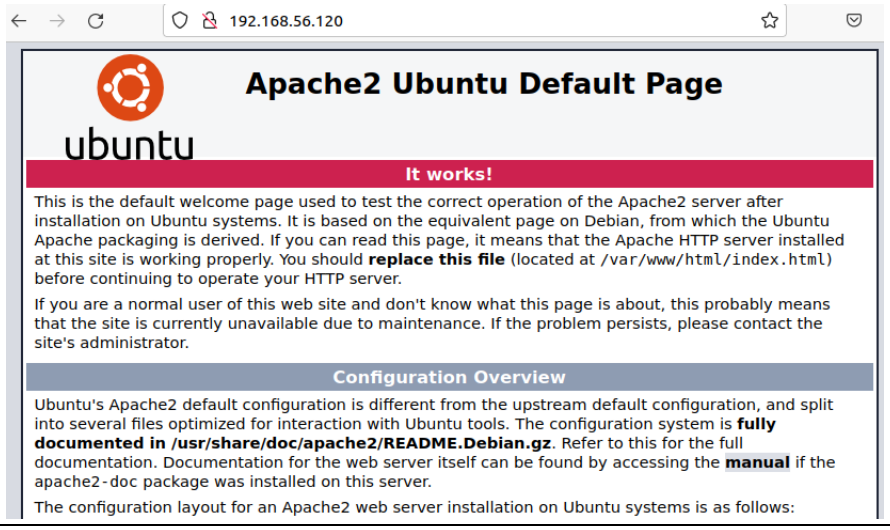
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.*

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [server2]
ok: [server1]

TASK [install apache2 package] *************************************************
changed: [server2]
changed: [server1]

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

server2                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


rnrlope@workstation:~/CPE212_LOPE-Act4$
```

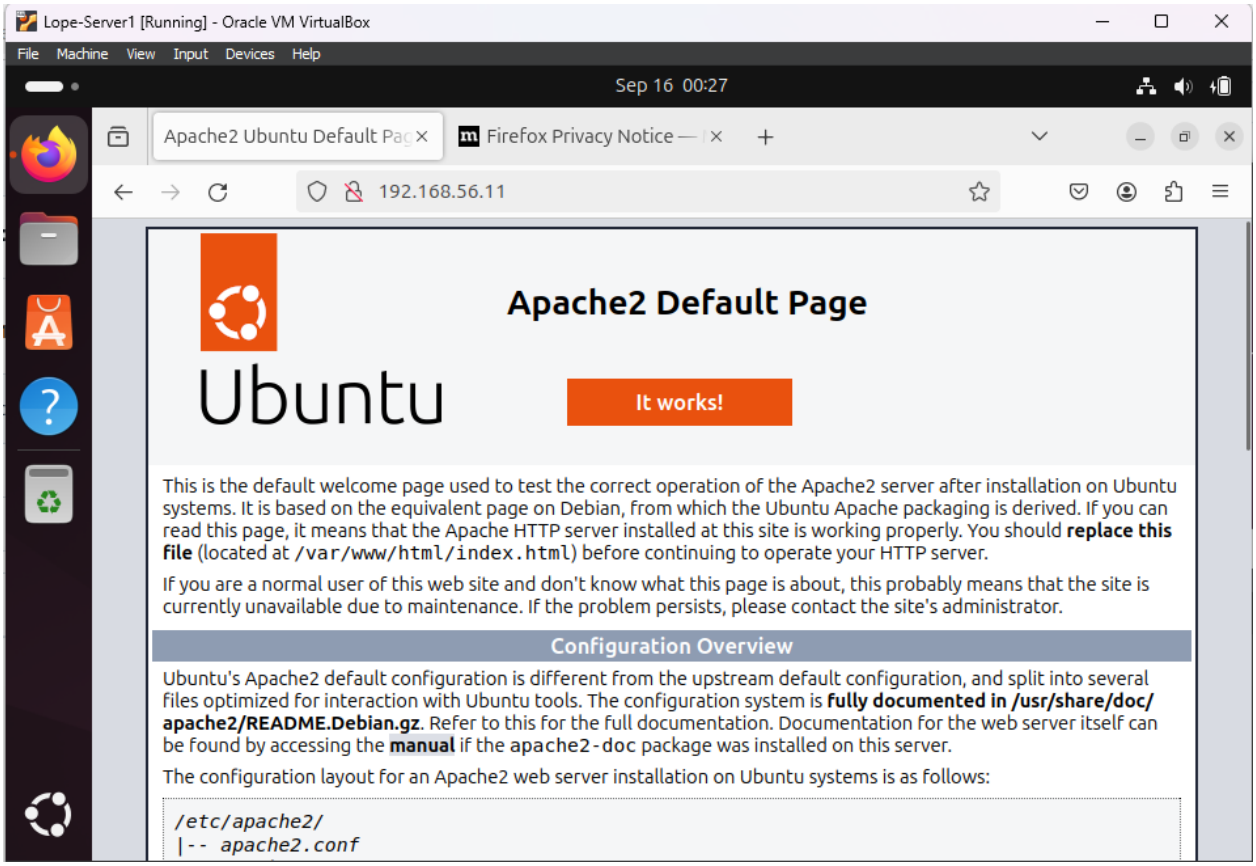
Describe the result of this command.

- **Firstly it gathered facts about the remote servers then it installed the apache2 package then it shows the status of remote servers.**

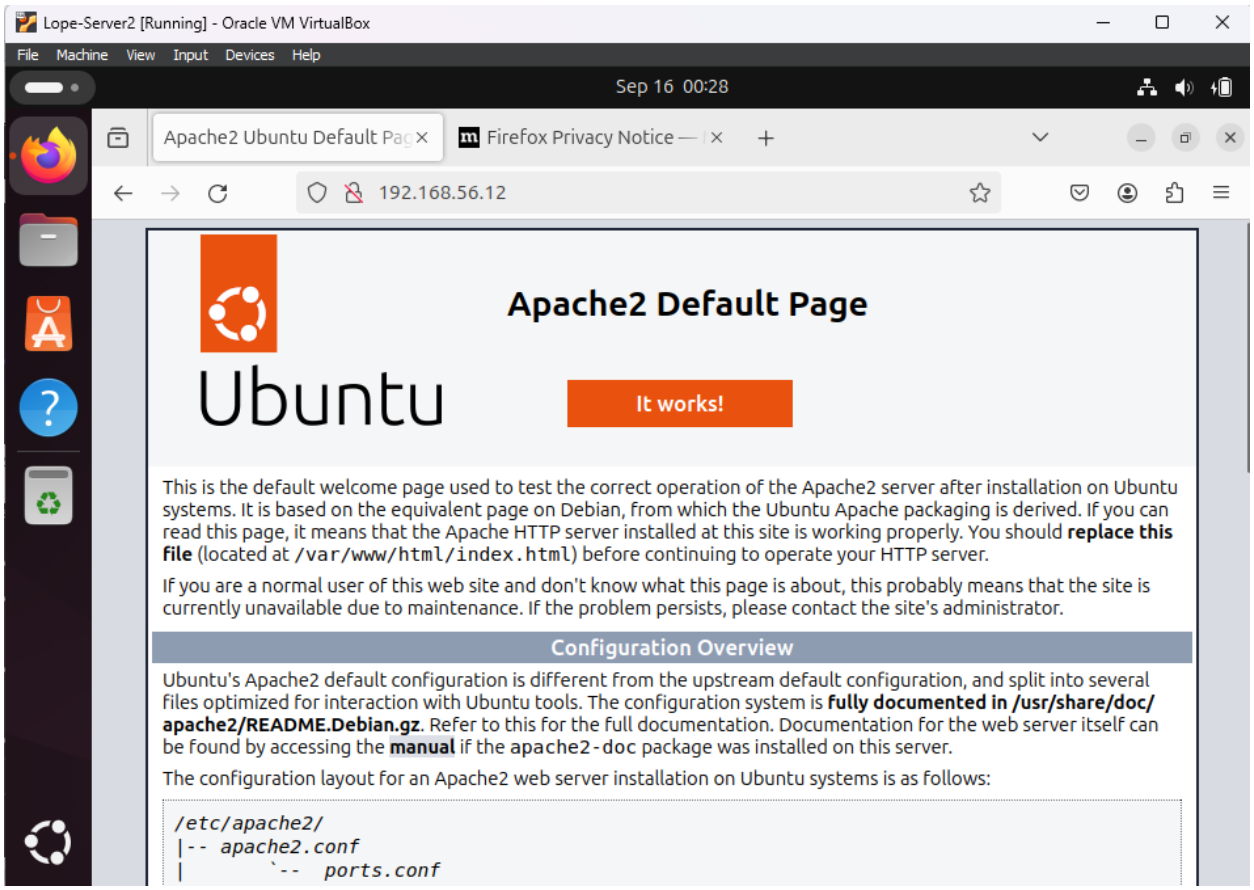
3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.



Server1:



Server2:



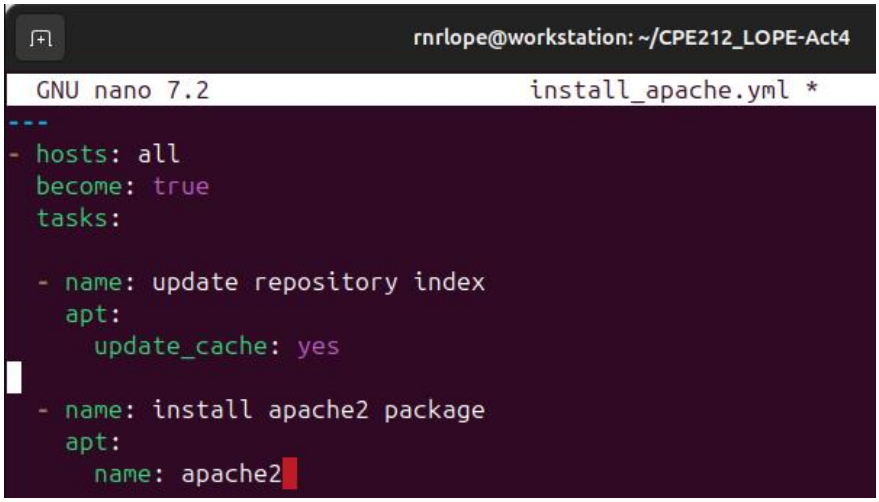
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized.

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ nano install_apache.yml
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible-playbook --ask-become-pass install_ap
ache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [server2]
ok: [server1]

TASK [hello world] *************************************************************
ok: [server1]
ok: [server2]

PLAY RECAP *********************************************************************
server1                    : ok=2    changed=0    unreachable=0    failed=0    skippe
d=0    rescued=0    ignored=0
server2                    : ok=2    changed=0    unreachable=0    failed=0    skippe
d=0    rescued=0    ignored=0

rnrlope@workstation:~/CPE212_LOPE-Act4$
```

What is the output? **– It changes the displayed package name while still being successful.**

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.

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ nano install_apache.yml
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible-playbook --ask-become-pass install_apache.yml

BECOME password:


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

TASK [Gathering Facts] *********************************************************
ok: [server2]
ok: [server1]

TASK [update repository index] *************************************************
changed: [server2]
changed: [server1]

TASK [install apache2 package] *************************************************
ok: [server1]
ok: [server2]

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

server2                    : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


rnrlope@workstation:~/CPE212_LOPE-Act4$
```

Did the new command change anything on the remote servers?

- **Yes, it changes something on the remote servers, specifically adds another task which is updating the 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.

```
---
- 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
```

rnrlope@workstation: ~/CPE212_LOPE-Act4

```
 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
```

Save the changes to this file and exit.

8. Run the playbook and describe the output.

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ nano install_apache.yml
rnrlope@workstation:~/CPE212_LOPE-Act4$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [server2]
ok: [server1]

TASK [update repository index] *************************************************
changed: [server1]
changed: [server2]

TASK [install apache2 package] *************************************************
ok: [server1]
ok: [server2]

TASK [add PHP support for apache] **********************************************
changed: [server1]
changed: [server2]

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

server2                    : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


rnrlope@workstation:~/CPE212_LOPE-Act4$
```

Did the new command change anything on the remote servers?

- **Yes, it changes something on the remote servers, specifically adds another task which is updating the repository index and also adding PHP support for apache.**
-

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

```
rnrlope@workstation:~/CPE212_LOPE-Act4$ git add install_apache.yml
rnrlope@workstation:~/CPE212_LOPE-Act4$ git commit -m "Playbook"
[main 6181ca0] Playbook
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
rnrlope@workstation:~/CPE212_LOPE-Act4$ git push origin main

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 458 bytes | 458.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:RenierCode/CPE212_LOPE-Act4.git
   9452dd9..6181ca0  main -> main
rnrlope@workstation:~/CPE212_LOPE-Act4$
rnrlope@workstation:~/CPE212_LOPE-Act4$
```

Github repository link: https://github.com/RenierCode/CPE212_LOPE-Act4.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
    - **The importance of using a playbook is defining the state that we want our remote servers to be in and the place and commands that ansible will run to bring our remote servers to that state.**
2. Summarize what we have done on this activity.
    - **In this activity, we have learned the basics of ansible; how to install ansible, creating ansible.cfg file, inventory file, and playbook.yml file, most importantly what is inside of those files. Using ad-hoc commands to simplify administration of remote servers. For example, inserting the IP addresses to the inventory, then using ansible command to install something in both servers at the same time possible. Then there is the playbook, in which can be used to make the servers to be in the state that we want it just by running the playbook.yml in turn running the scripts inside of it.**