Name: Ruud Van G. Apostol	**Date Performed:** 09/05/25
Course/Section: CPE212 / CPE31S4	**Date Submitted:** 09/05/25
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Sem / 2025-2026

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

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

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

Task 1: Run elevated ad hoc commands

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

   ansible all -m apt -a update_cache=true

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

```
[Apostol@vbox ~]$ ansible all -m apt -a update_cache=true
bash: ansible: command not found...
Install package 'ansible-core' to provide command 'ansible'? [N/y] y


 * Waiting in queue...
The following packages have to be installed:
 ansible-core-1:2.14.18-1.el9.x86_64    SSH-based configuration management, depl
oyment, and task execution system
 git-core-2.47.3-1.el9.x86_64   Core package of git with minimal functionality
 python3-cffi-1.14.5-5.el9.x86_64       Foreign Function Interface for Python 3
to call C code
 python3-cryptography-36.0.1-5.el9.x86_64       PyCA's cryptography library
 python3-packaging-20.9-5.el9.noarch    Core utilities for Python packages
 python3-ply-3.11-14.el9.noarch Python Lex-Yacc
 python3-pycparser-2.20-6.el9.noarch    C parser and AST generator written in Py
```

   - **It is successful**

     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.

```
[Apostol@vbox ~]$ ansible all -m apt -a update_cache=true --become --ask-become-
pass
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'
[Apostol@vbox ~]$
```

     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.

```
[Apostol@vbox ~]$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'
[Apostol@vbox ~]$
```

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?

```
[Apostol@vbox ~]$ dnf search vim-nox
Not root, Subscription Management repositories not updated
CentOS Stream 9 - BaseOS                        35 kB/s | 8.8 MB     04:14
CentOS Stream 9 - AppStream                    142 kB/s |  25 MB     02:58
CentOS Stream 9 - Extras packages              5.6 kB/s |  19 kB     00:03
No matches found.
[Apostol@vbox ~]$
```

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.

```
Apostol@ApostolCN:~$ cd /var/log
Apostol@ApostolCN:/var/log$ cd apt
Apostol@ApostolCN:/var/log/apt$ ls
eipp.log.xz  history.log  history.log.1.gz  term.log  term.log.1.gz
Apostol@ApostolCN:/var/log/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*

```
[Apostol@vbox log]$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'
[Apostol@vbox log]$
```

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

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.

```
[Apostol@vbox log]$ ansible all -m apt -a "name=snapd state=latest" --become --a
sk-become-pass
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'
[Apostol@vbox log]$
```

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

Task 2: Writing our First Playbook

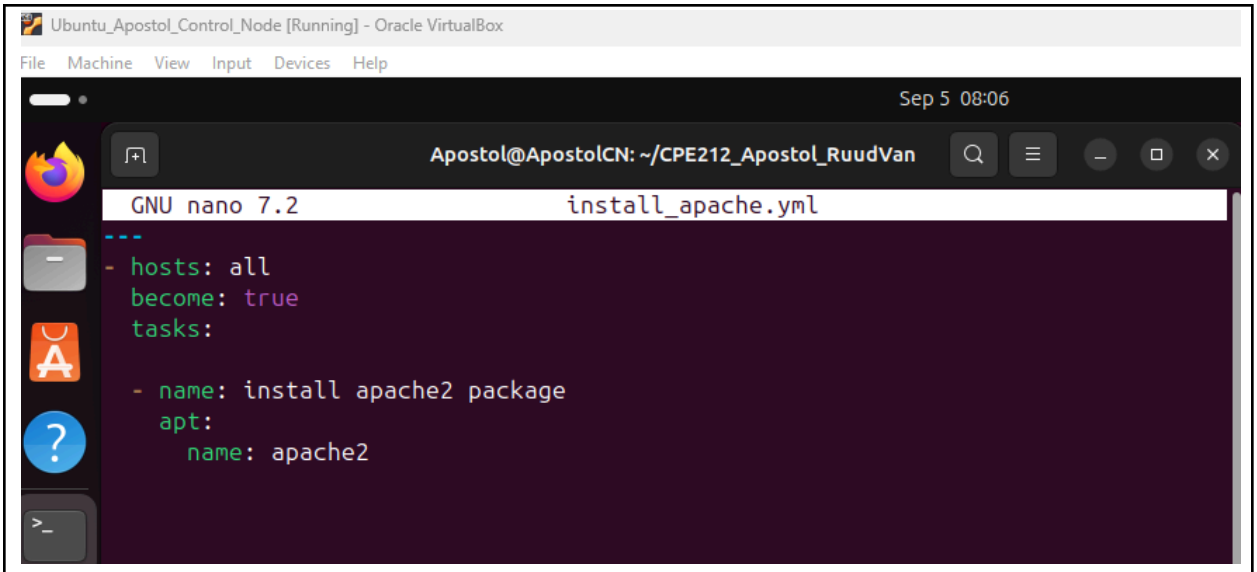
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and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

```
Apostol@ApostolCN:~$ cd CPE212_Apostol_RuudVan
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ nano install_apache.
```

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

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

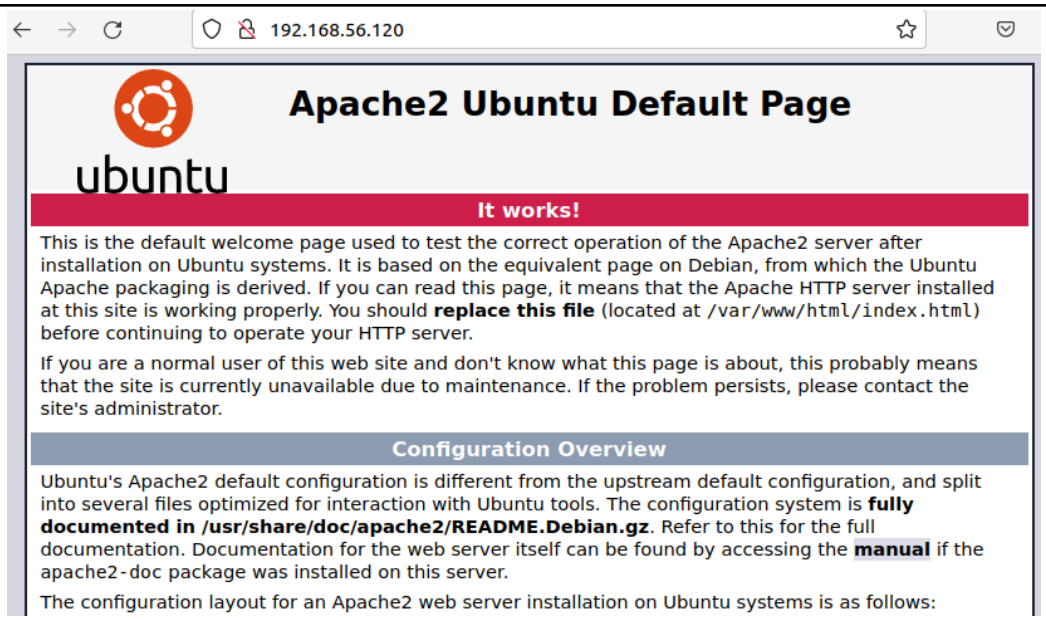
   Make sure to save the file. Take note also of the alignments of the texts.



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

```
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ ansible-playbook --ask-become-pass i
nstall_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
fatal: [server1]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect t
o the host via ssh: ssh: connect to host 192.168.56.106 port 22: No route to hos
t", "unreachable": true}
fatal: [server2]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect t
o the host via ssh: ssh: connect to host 192.168.56.108 port 22: No route to hos
t", "unreachable": true}

PLAY RECAP *********************************************************************
server1                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
server2                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0

Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$
```

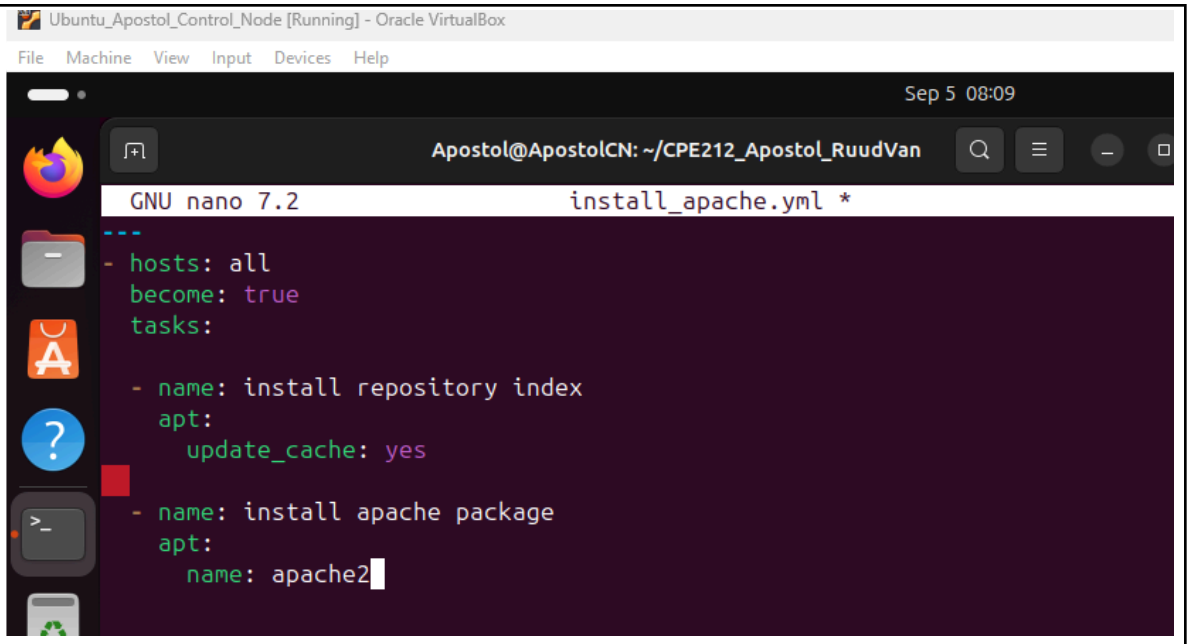
3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.



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?
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?

```
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ ansible-playbook --ask-become-pass i
nstall_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
fatal: [server1]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect t
o the host via ssh: ssh: connect to host 192.168.56.106 port 22: No route to hos
t", "unreachable": true}
fatal: [server2]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect t
o the host via ssh: ssh: connect to host 192.168.56.108 port 22: No route to hos
t", "unreachable": true}

PLAY RECAP *********************************************************************
server1                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
server2                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
```

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.

8. Run the playbook and describe the output. Did the new command change anything on the remote servers?
9. Finally, make sure that we are in sync with GitHub. Provide the link of your GitHub repository.

```
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ git add install_apache.yml
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ git commit -m "install_apache.yml"
[main 43ee516] install_apache.yml
 1 file changed, 12 insertions(+)
 create mode 100644 install_apache.yml
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$ git push origin main
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
Apostol@ApostolCN:~/CPE212_Apostol_RuudVan$
```

Reflections:

Answer the following:

1. **What is the importance of using a playbook?**
   - Playbooks are **the simplest way in Ansible to automate repeating tasks in the form of reusable and consistent configuration files**. Playbooks are scripts defined in YAML files and contain any ordered set of steps to be executed on our managed nodes. As mentioned, tasks in a playbook are executed from top to bottom.

2. Summarize what we have done on this activity.

- I learn how to use ansible and playbook in ubuntu, we also learn how to install ansible in our centos.