Name: Hermitano, Johnny C.	**Date Performed: September 15, 2022**
Course/Section: CPE31S23	**Date Submitted: September 15, 2022**
Instructor: Engr. Jonathan Taylar	**Semester and SY: 1st sem sy2022-2023**

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:

```
Get:16 http://security.ubuntu.com/ubuntu jammy-security/univers
etadata [10.4 kB]
Fetched 2,093 kB in 3s (769 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see
jhermitano@Workstation:~$
```

```
jhermitano@Workstation:~$ /etc/apt/sources.list
bash: /etc/apt/sources.list: Permission denied
jhermitano@Workstation:~$ /etc/apt/sources.list.d/
bash: /etc/apt/sources.list.d/: Is a directory
jhermitano@Workstation:~$
```

ansible all -m apt -a update_cache=true

```
jhermitano@Workstation:~/hermitano$ 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 director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (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 director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (13: Permission denied)"
}
```

What is the result of the command? Is it 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.

```
jhermitano@Workstation:~/hermitano$ ansible all -m apt -a update_cache=true --b
ecome --ask-become-pass
BECOME password:
192.168.56.105 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249998,
    "cache_updated": true,
    "changed": true
}
192.168.56.106 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249999,
    "cache_updated": true,
    "changed": true
}
```

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.

```
jhermitano@Workstation:~/hermitano$ ansible all -m apt -a name=vim-nox --become
 --ask-become-pass
BECOME password:
192.168.56.106 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249999,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following additional packages will be installed:\n  fo
nts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  rub
y-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integra
tion vim-common vim-runtime vim-tiny\nSuggested packages:\n  apache2 | lighttpd
 | httpd ri ruby-dev bundler cscope vim-doc indent\nThe following NEW packages
will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 lib
ruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc rub
y3.0\n  rubygems-integration vim-nox vim-runtime\nThe following packages will b
e upgraded:\n  vim-common vim-tiny\n2 upgraded, 15 newly installed, 0 to remove
 and 9 not upgraded.\nNeed to get 18.2 MB of archives.\nAfter this operation, 7
6.3 MB of additional disk space will be used.\nGet:1 http://ph.archive.ubuntu.c
om/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\nGet:2 http://ph.ar
chive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5936 B]
\nGet:3 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 libjs-jquery all 3
.6.0+dfsg+~3.5.13-1 [321 kB]\nGet:4 http://ph.archive.ubuntu.com/ubuntu jammy/u
```

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?

```
jhermitano@Workstation:~/hermitano$ which vim
jhermitano@Workstation:~/hermitano$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy 2:8.2.3995-1ubuntu2 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy,now 2:8.2.3995-1ubuntu2 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.

```
jhermitano@Workstation:~/hermitano$ cd /var/log
jhermitano@Workstation:/var/log$ ls
alternatives.log  dmesg       fontconfig.log   openvpn
apt               dmesg.0     gdm3             private
auth.log          dmesg.1.gz  gpu-manager.log  speech-dispatcher
boot.log          dmesg.2.gz  hp               syslog
bootstrap.log     dmesg.3.gz  installer        ubuntu-advantage.log
btmp              dmesg.4.gz  journal          ubuntu-advantage-timer.log
cups              dpkg.log    kern.log         unattended-upgrades
dist-upgrade      faillog     lastlog          wtmp
jhermitano@Workstation:/var/log$ apt
apt 2.4.7 (amd64)
Usage: apt [options] command

apt is a commandline package manager and provides commands for
searching and managing as well as querying information about packages.
It provides the same functionality as the specialized APT tools,
like apt-get and apt-cache, but enables options more suitable for
interactive use by default.

Most used commands:
  list - list packages based on package names
  search - search in package descriptions
```

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?

   Fortunately it successfully installed the package.

```
jhermitano@Workstation:~/hermitano$ ansible all -m apt -a name=snapd --become -
-ask-become-pass
BECOME password:
192.168.56.105 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249998,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249999,
    "cache_updated": false,
    "changed": false
}
```

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

```
jhermitano@Workstation:~/hermitano$ ansible all -m apt -a "name=snapd
est" --become --ask-become-pass
BECOME password:
192.168.56.105 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249998,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663249999,
    "cache_updated": false,
    "changed": false
}
```

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

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

```
jhermitano@Workstation:~/hermitano$ state=latest
jhermitano@Workstation:~/hermitano$
```

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:

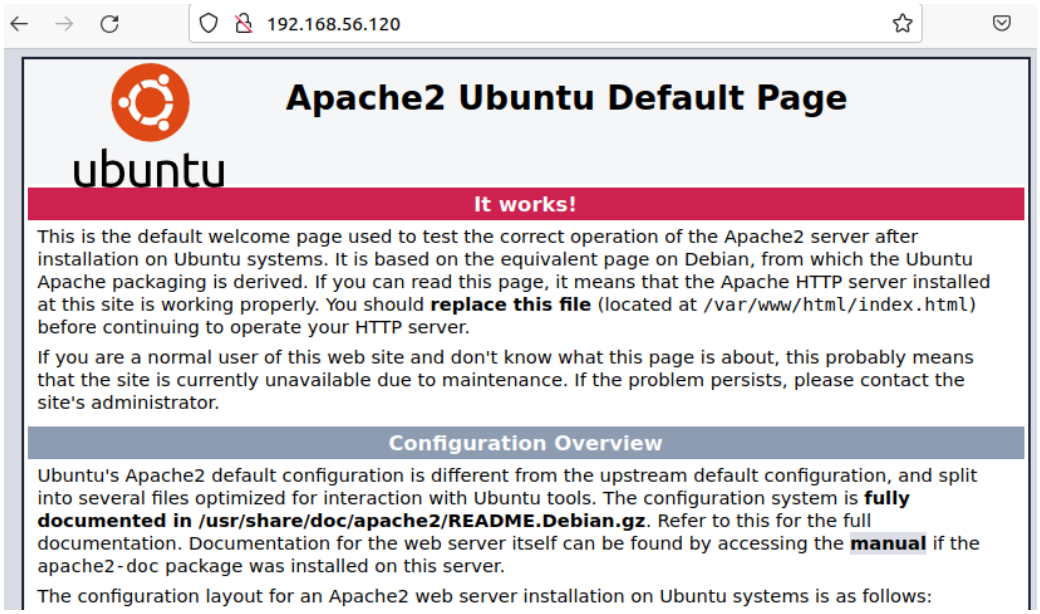
```
  GNU nano 4.8                  install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

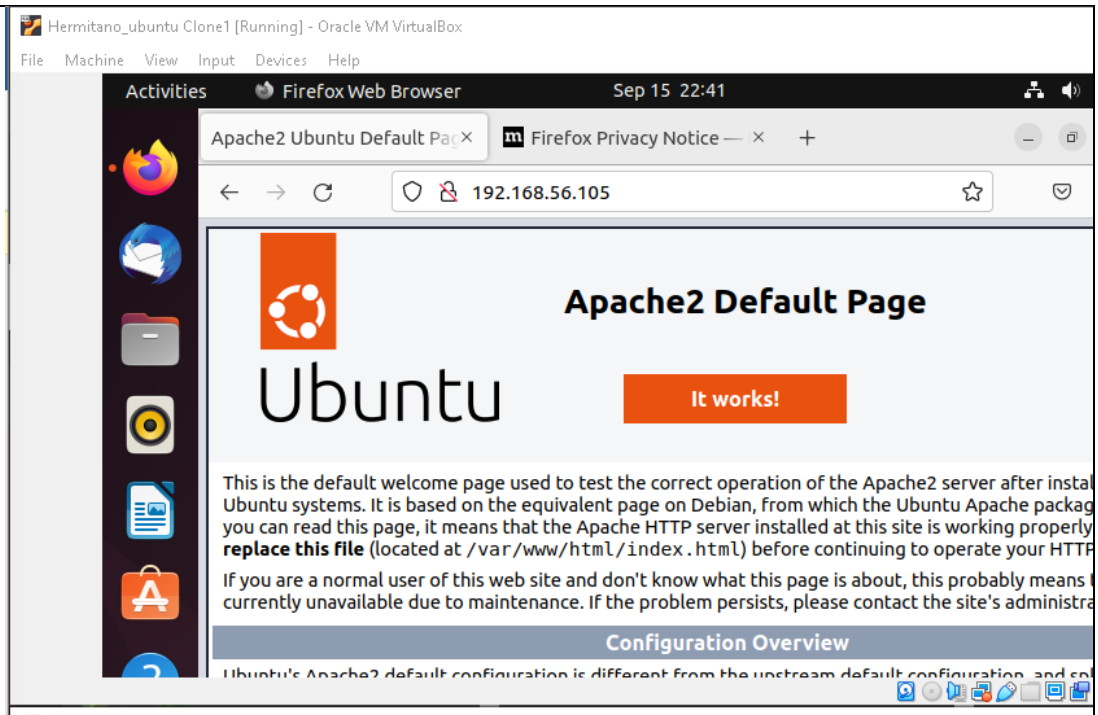
Make sure to save the file. Take note also of the alignments of the texts.

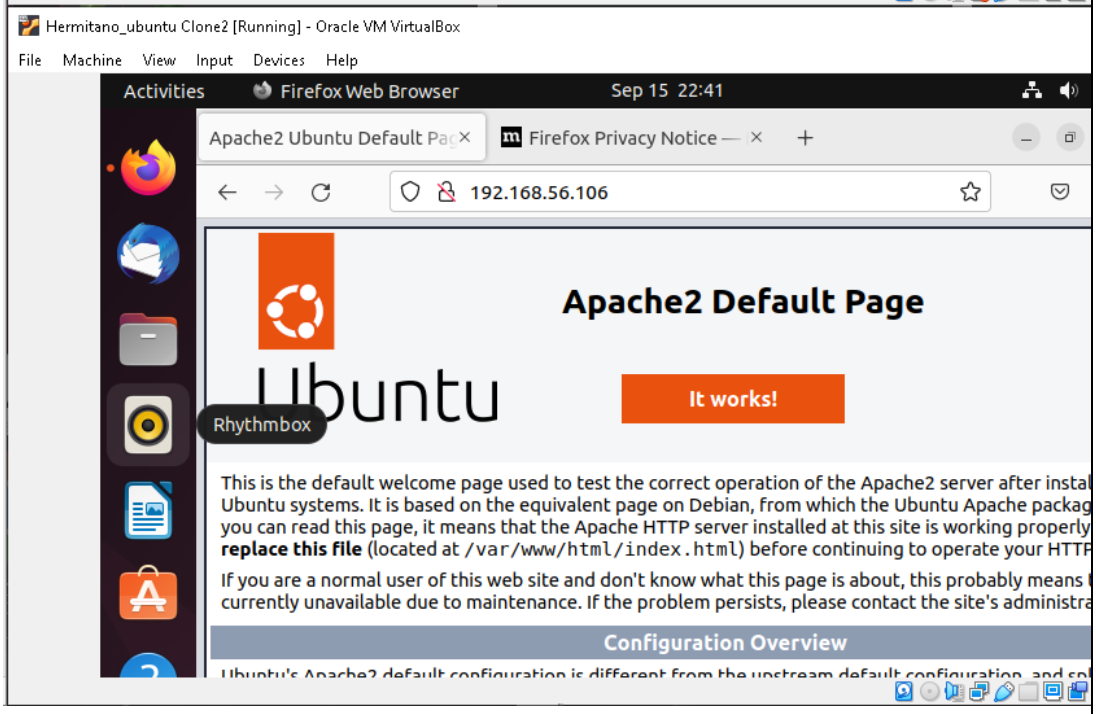
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

```
jhermitano@Workstation:~/CPE232_Hermitano$ ansible-playbook --ask-bec
nstall_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************
*
ok: [192.168.56.106]
ok: [192.168.56.105]

TASK [install apache2 package] *************************************
*
changed: [192.168.56.105]
changed: [192.168.56.106]

PLAY RECAP *********************************************************
*
192.168.56.105             : ok=2    changed=1    unreachable=0    fa
skipped=0    rescued=0    ignored=0
192.168.56.106             : ok=2    changed=1    unreachable=0    fa
skipped=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?

```
jhermitano@Workstation:~/CPE232_Hermitano$ ansible-playbook --ask-become-pass d
ownload_apache.yml
BECOME password:

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

TASK [Gathering Facts] ********************************************************
*
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [install apache2 package] ************************************************
*
ok: [192.168.56.105]
ok: [192.168.56.106]

PLAY RECAP ********************************************************************
*
192.168.56.105             : ok=2    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Yes, it still recognized the package.

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.

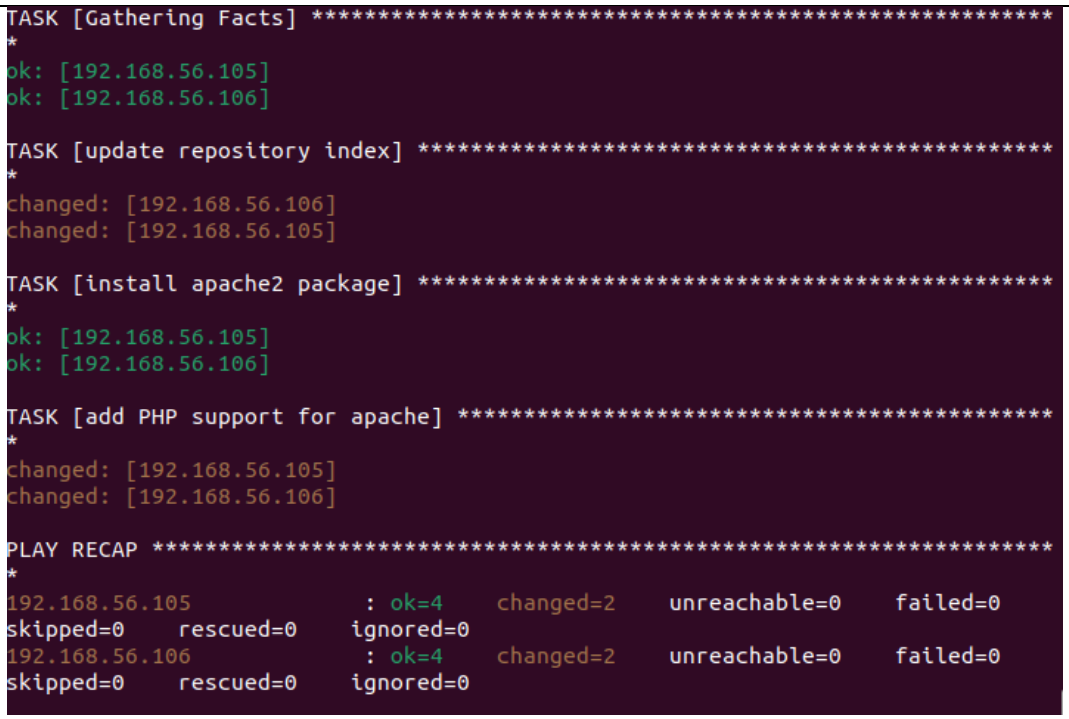
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?

```
nstall_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [update repository index] *************************************************
*
changed: [192.168.56.106]
changed: [192.168.56.105]

TASK [install apache2 package] *************************************************
*
ok: [192.168.56.105]
Welcome to Ubuntu
PLAY RECAP *********************************************************************
*
192.168.56.105             : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.106             : ok=3    changed=1    unreachable=0    failed=0
skipped=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.

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   The playbook assists the team in visualizing goals, comprehending the continuous improvement paradigm, and understanding what is required to succeed. The main workflow steps are specified, and the specific tasks within those sections are listed.
2. Summarize what we have done on this activity.
   In order to perform this task, we must first install Ansible. After that, we began studying the fundamentals of Ansible, which included learning about passwords. Finally, we write ourselves a playbook and save it to the long-ago constructed github.