Name: Daniela Marie D. Rabang	**Date Performed:** 09/11/2023
Course/Section: CPE232/CPE31S4	**Date Submitted:** 09/12/2023
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem 2023-2024

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

```
daniela@workstation:~/ansibleS4$ ansible all -m apt -a update_cache=true
192.168.56.111 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
192.168.56.110 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
```

What is the result of the command? Is it successful? The results of the command that is being asked to issue, failed. It is not successful and it is shown.

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 changed the packaged index, we were able to change something on the remote server.

```
daniela@workstation:~/ansibleS4$ ansible all -m apt -a update_cache=true --becom
e --ask-become-pass
SUDO password:
192.168.56.111 | SUCCESS => {
    "cache_update_time": 1694406716,
    "cache_updated": true,
    "changed": true
}
192.168.56.110 | SUCCESS => {
    "cache_update_time": 1694406716,
    "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.

```
daniela@workstation:~/ansibleS4$ ansible all -m apt -a name=vim-nox --become --a
sk-become-pass
SUDO password:
192.168.56.110 | SUCCESS => {
    "cache_update_time": 1694406716,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nThe following package was automatically installed and is no
longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe follo
wing additional packages will be installed:\n  fonts-lato javascript-common libj
s-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ruby-mi
nitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygems-inte
gration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd tcl8.6 ri
 ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\
n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n
```

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.

```
daniela@workstation:~/ansibleS4$ which vim
daniela@workstation:~/ansibleS4$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security 2:8.0.1453-1ubuntu1.13 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - compact version
```

Was the command successful? The command that is asked to be run, is successfully done.

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. In the history.log, I saw the creation date, the command line that is used, the person that requested it, the install information, and the end date.

```
daniela@server1:~$ cd /var/log
daniela@server1:/var/log$ cd apt
daniela@server1:/var/log/apt$ sudo nano history.log
[sudo] password for daniela:
```

```
  GNU nano 2.9.3                  history.log                       Modified

Start-Date: 2023-09-11  12:36:16
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::O$
Requested-By: daniela (1000)
Install: javascript-common:amd64 (11, automatic), ruby2.5:amd64 (2.5.1-1ubuntu$
End-Date: 2023-09-11  12:36:24
```

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*

```
daniela@workstation:~/ansibleS4$ ansible all -m apt -a name=snapd --become --ask
-become-pass
SUDO password:
192.168.56.111 | SUCCESS => {
    "cache_update_time": 1694406716,
    "cache_updated": false,
    "changed": false
}
192.168.56.110 | SUCCESS => {
    "cache_update_time": 1694406716,
    "cache_updated": false,
    "changed": false
}
```

Can you describe the result of this command? Is it a success? Did it change anything in the remote servers? This command pings and displays all the servers that are connected with the main server. It is successfully done since the message is displayed in a green colored text. It also does not 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*

```
daniela@workstation:~/ansibleS4$ ansible all -m apt -a "name=snapd state=latest"
 --become --ask-become-pass
SUDO password:
192.168.56.111 | SUCCESS => {
    "cache_update_time": 1694406716,
    "cache_updated": false,
    "changed": false
}
192.168.56.110 | SUCCESS => {
    "cache_update_time": 1694406716,
    "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. The latest update for the two other servers are displayed. Since it was the state=latest is added.

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:

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

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

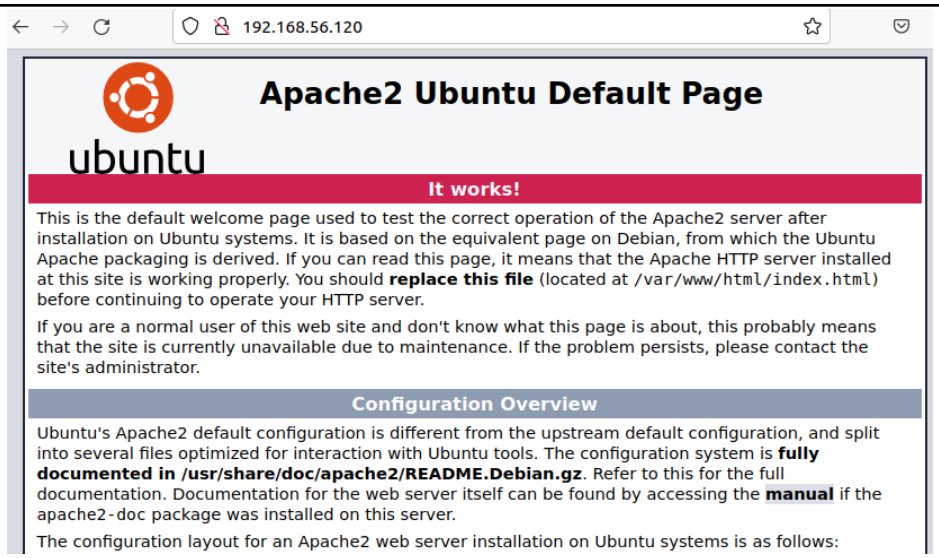
```
  GNU nano 2.9.3              install_apache.yml              Modified
---
- hosts: all
  become: true
  tasks:

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

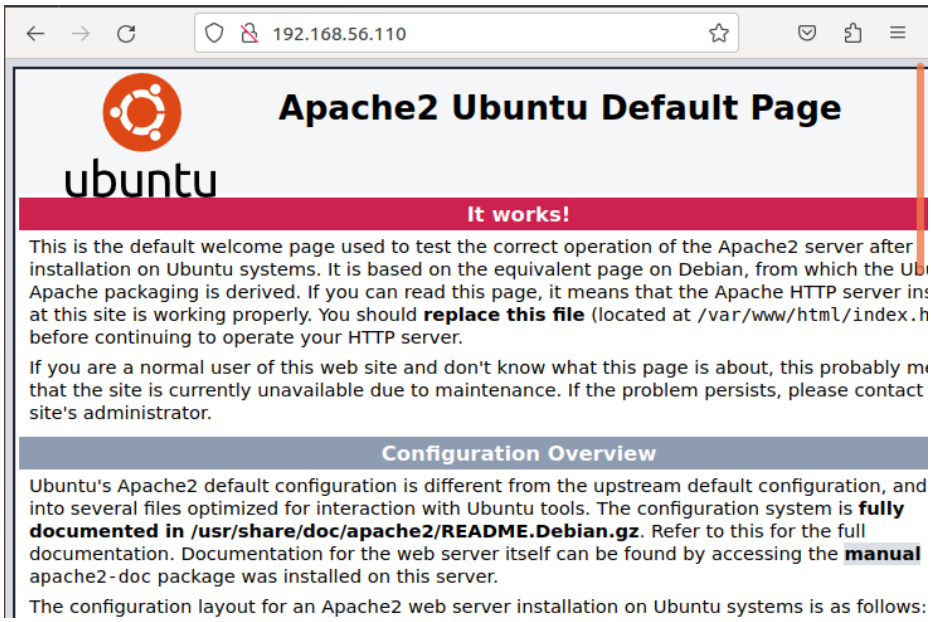
Make sure to save the file. Take note also of the alignments of the texts.

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

```
daniela@workstation:~/CPE232_Rabang$ ansible-playbook --ask-become-pass install_
apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.110]
ok: [192.168.56.111]

TASK [install apache2 package] *************************************************
ok: [192.168.56.110]
changed: [192.168.56.111]

PLAY RECAP *********************************************************************
192.168.56.110             : ok=2    changed=0    unreachable=0    failed=0
192.168.56.111             : ok=2    changed=1    unreachable=0    failed=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? The output after changing the name, failed since the name is not recognized and there are no packages with the same name.

```
daniela@workstation:~/CPE232_Rabang$ ansible-playbook --ask-become-pass install_
apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.111]
ok: [192.168.56.110]

TASK [install apache2 package] *************************************************
fatal: [192.168.56.111]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'apache' is available"}
fatal: [192.168.56.110]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'apache' is available"}
        to retry, use: --limit @/home/daniela/CPE232_Rabang/install_apache.retry

PLAY RECAP *********************************************************************
192.168.56.110             : ok=1    changed=0    unreachable=0    failed=1
192.168.56.111             : ok=1    changed=0    unreachable=0    failed=1
```

5. This time, we are going to put additional tasks into 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
```

```
  GNU nano 2.9.3                  install_apache.yml

---
- 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? Yes, it does change things. It added a task that updates the repository index.

```
daniela@workstation:~/CPE232_Rabang$ ansible-playbook --ask-become-pass install_
apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.111]
ok: [192.168.56.110]

TASK [update repository index] *************************************************
changed: [192.168.56.111]
changed: [192.168.56.110]

TASK [install apache2 package] *************************************************
ok: [192.168.56.111]
ok: [192.168.56.110]

PLAY RECAP *********************************************************************
192.168.56.110             : ok=3    changed=1    unreachable=0    failed=0
192.168.56.111             : ok=3    changed=1    unreachable=0    failed=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.

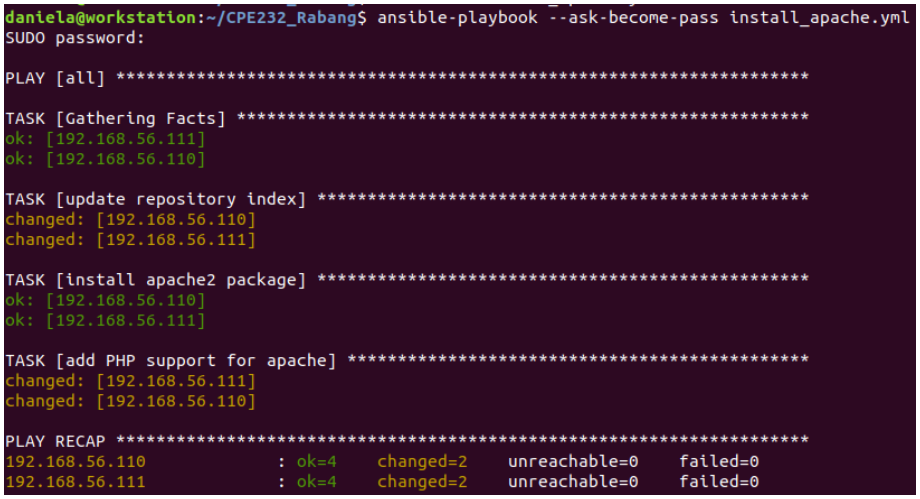
```
---
- hosts: all
  become: true
  tasks:

  - 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
```

```
 GNU nano 2.9.3                        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. Did the new command change anything on the remote servers? Yes, it does change something. It added a

new task to run which is to add PHP support for apache, and the results had changed.



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

   git@github.com:danielarabang/CPE232_Rabang.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - The importance of using a playbook is that the playbook helps the ansible to be informed on the certain tasks that are needed to be done in a specific device.
2. Summarize what we have done on this activity.
   - In this activity we had done many tasks. We had done the introduction to ansible. the installation of ansible in the workstation, checking of the hosts in the inventory by pinging them, creating inventory, creating playbooks that can be used in the activity and running them. Also we had runned all the commands that are used to update and play tasks. We installed apache2 and verified it by using the server1.

Conclusion:

In this hands-on activity, which is the hands-on 4 that is all about running elevated ad hoc commands. So we had done the installation of the ansible first, since this was the first thing to do to be able to do the other tasks. We had done the creation of files that we had inserted commands and this is done using sudo nano command. Then we had learned and executed the playbooks. After the introduction to ansible tasks to be executed. I had executed the command sudo apt update to update my workstation. Then we are asked to access the other two servers by running the given command. The next thing that I had done was to upgrade the command to elevate the privilege. This is used to access the two other servers and change the package index. Then I

installed the VIM, and verified that the package is installed in the system. After all the commands that are done and executed, I proceed to write my first playbook. By doing inside the repository CPE232_Rabang. When I had already issued the command sudo nano and edited the file and I had run the yml file. I already installed the apache2 package using the install_apache.yml file. Then I had verified if it is already installed. Then the last thing is that I had edited it again to put additional tasks that are asked to be runned.