Name:	**Date Performed:**
Course/Section:	**Date Submitted:**
Instructor:	**Semester and SY:**

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

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible all -m apt -a update_ca
true
192.168.56.102 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock dire
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock -
n (13: Permission denied)"
}
192.168.56.103 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock dire
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock -
n (13: Permission denied)"
}
valenzuela@workstation:~/CPE232_Valenzuela$ s
```

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

- The result of the command is a failure because it does not have access without become.

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.

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible all -m apt -a update_ca
true --become --ask-become-pass
BECOME password:

192.168.56.102 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "cache_updated": true,
    "changed": true
}
192.168.56.103 | CHANGED => {
    "ansible_facts": {
Ubuntu Software  ered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "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.

```
valenzuela@workstation:/$ ansible all -m apt -a name=vim-nox --become --asl
ome-pass
BECOME password:
server1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663299196,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nRead
tate information...\nThe following additional packages will be installed:\
nts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n
y-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-in
tion vim-common vim-runtime vim-tiny\nSuggested packages:\n  apache2 | ligl
 | httpd ri ruby-dev bundler cscope vim-doc indent\nThe following NEW packa
will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-
ruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrp
y3.0\n  rubygems-integration vim-nox vim-runtime\nThe following packages w
e upgraded:\n  vim-common vim-tiny\n2 upgraded, 15 newly installed, 0 to r
 and 55 not upgraded.\nNeed to get 18.2 MB of archives.\nAfter this operat
76.3 MB of additional disk space will be used.\nGet:1 http://ph.archive.ub
com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\nGet:2 http:/
rchive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-tiny amd64 2:8.2.399
untu2.1 [704 kB]\nGet:3 http://ph.archive.ubuntu.com/ubuntu jammy-updates/
```

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?

- Yes, the command was successful.

```
valenzuela@workstation:~/CPE232_Valenzuela$ which vim
valenzuela@workstation:~/CPE232_Valenzuela$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.1 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.1 amd64 [upgradable f
rom: 2:8.2.3995-1ubuntu2]
  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.

- The history log shows the history in the ubuntu

```
valenzuela@workstation:/var/log$ cd apt
valenzuela@workstation:/var/log/apt$ ls
eipp.log.xz  history.log  term.log
valenzuela@workstation:/var/log/apt$ cat history.log

Start-Date: 2022-08-09  11:48:47
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --forc
grade
Upgrade: dpkg:amd64 (1.21.1ubuntu2, 1.21.1ubuntu2.1), networkd-dispatch
 (2.1-2, 2.1-2ubuntu0.22.04.2), udev:amd64 (249.11-0ubuntu3, 249.11-0ub
                                64 (3.10.4-3, 3.10.4-3ubuntu0.1), python3-gi:amd64 (3.4
                                ntu1), libext2fs2:amd64 (1.46.5-2ubuntu1, 1.46.5-2ubunt
pt:amd64 (2.4.5, 2.4.6), systemd-timesyncd:amd64 (249.11-0ubuntu3, 249.
tu3.4), libtirpc-common:amd64 (1.3.2-2build1, 1.3.2-2ubuntu0.1), libpcr
 (2:8.39-13build5, 2:8.39-13ubuntu0.22.04.1), libpam-systemd:amd64 (249
ntu3, 249.11-0ubuntu3.4), libpython3.10-minimal:amd64 (3.10.4-3, 3.10.4
0.1), libapt-pkg6.0:amd64 (2.4.5, 2.4.6), libfribidi0:amd64 (1.0.8-2ubu
0.8-2ubuntu3.1), libpython3.10-stdlib:amd64 (3.10.4-3, 3.10.4-3ubuntu0.
ystemd0:amd64 (249.11-0ubuntu3, 249.11-0ubuntu3.4), libnss-systemd:amd6
1-0ubuntu3, 249.11-0ubuntu3.4), logrotate:amd64 (3.19.0-1ubuntu1, 3.19.
u1.1), libapparmor1:amd64 (3.0.4-2ubuntu2, 3.0.4-2ubuntu2.1), libxml2:a
9.13+dfsg-1build1, 2.9.13+dfsg-1ubuntu0.1), python-apt-common:amd64 (2.
u2, 2.3.0ubuntu2.1), systemd:amd64 (249.11-0ubuntu3, 249.11-0ubuntu3.4)
v1:amd64 (249.11-0ubuntu3, 249.11-0ubuntu3.4), isc-dhcp-common:amd64 (4
ubuntu2, 4.4.1-2.3ubuntu2.1), libcom-err2:amd64 (1.46.5-2ubuntu1, 1.46.
```

LibreOffice Writer

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*

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible all -m apt -a name=snap
become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "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?

- Yes it was a success, in the server, it was changed from true to false.

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.

- *It will update the package snapd on the server, in this case I already updated it.*

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible all -m apt -a "name=snapd s
tate=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1663326361,
    "cache_updated": false,
    "changed": false
}
```

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
```

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

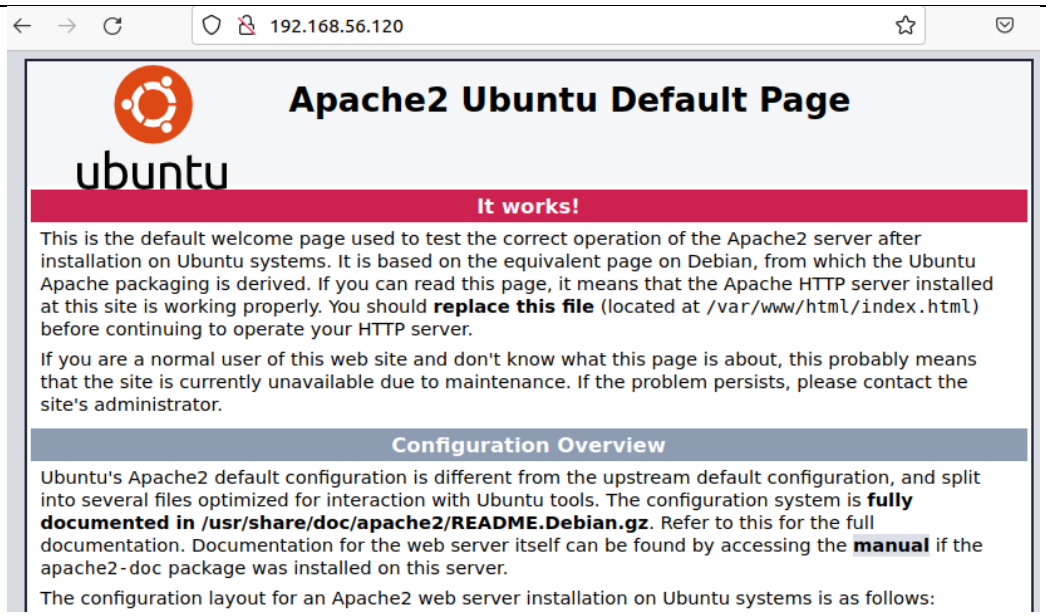
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible-playbook --ask-become-pass
install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.103]
ok: [192.168.56.102]
```

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?

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible-playbook --ask-become-pass
install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [install apache2 package] *************************************************
*
fatal: [192.168.56.102]: FAILED! => {"changed": false, "msg": "No package match
ing 'yowa' is available"}
fatal: [192.168.56.103]: FAILED! => {"changed": false, "msg": "No package match
ing 'yowa' is available"}

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

```
 GNU nano 6.2                    install_apache.yml
---

- hosts: all
  become: true
  tasks:

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

- The name was wrong and was not the same title that is why the installation failed.

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.

```
 GNU nano 6.2                    install_apache.yml
---

- hosts: all
  become: true
  tasks:

    - name: update repository index
      apt:
        update_cache: yes


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

6. Run the playbook and describe the output. Did the new command change anything on the remote servers?
   - The command updated the existing repository index.

```
valenzuela@workstation:~/CPE232_Valenzuela$ ansible-playbook --ask-beco
install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************
*
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update repository index] *************************************
*
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] *************************************
*
ok: [192.168.56.103]
ok: [192.168.56.102]

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

   - Yes, it update

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.

```
GNU nano 6.2                    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
```

8. Run the playbook and describe the output. Did the new command change anything on the remote servers?
   - The command added PHP support in the servers

```
TASK [install apache2 package] ************************************************
*
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [add PHP support for apache] *********************************************
*
changed: [192.168.56.102]
changed: [192.168.56.103]

PLAY RECAP ********************************************************************
*
192.168.56.102             : 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
```

git

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

https://github.com/qrevalenzuela/CPE232_Valenzuela

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - The importance of using a playbook is to record what we will do on remote servers so that it will be easier.

2. Summarize what we have done on this activity.
    - We installed ansible to create our playbook for remote servers.