| | |
|---|---|
| **Name:** Quizon, Nowell Gabriel C. | **Date Performed:** 09/11/2023 |
| **Course/Section:** CPE31S5 | **Date Submitted:** 09/12/2023 |
| **Instructor:** Engr. Roman Richard | **Semester and SY:** 1st and 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:**

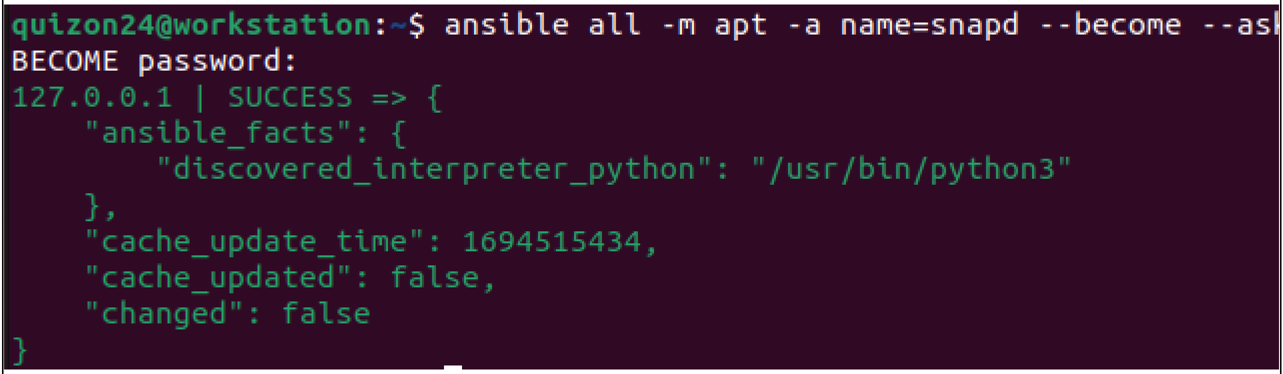
*Provide screenshots for each task*.

**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

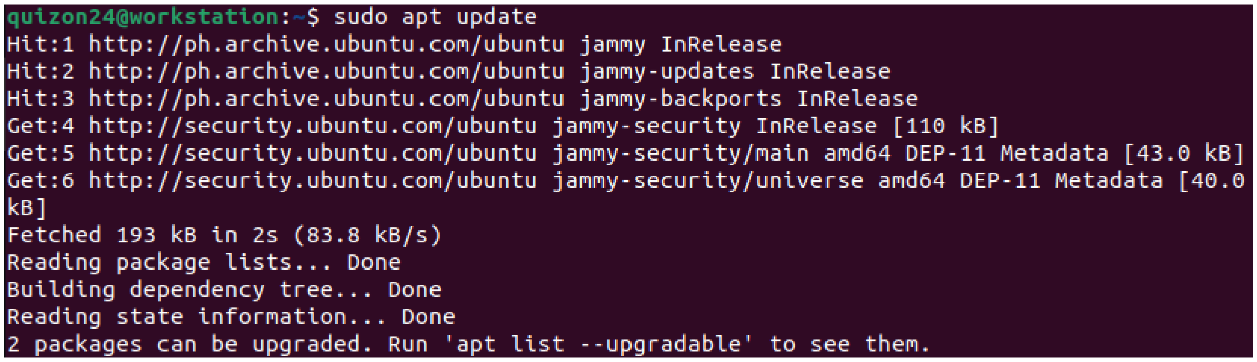
## Task 1: Run elevated ad hoc commands



```
quizon24@workstation:~$ ansible all -m apt -a name=snapd --become --as
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694515434,
    "cache_updated": false,
    "changed": false
}
```

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:

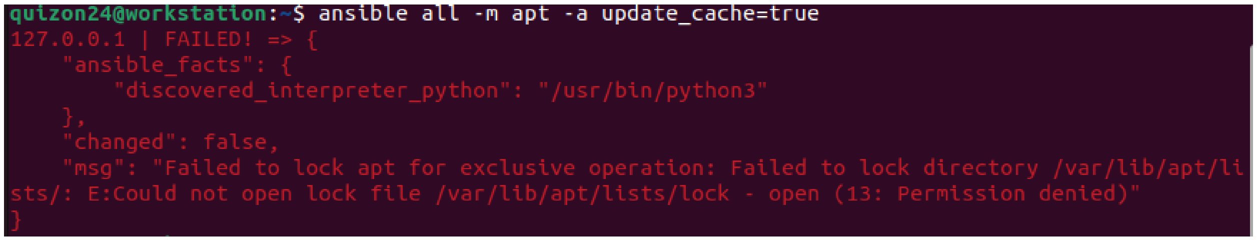


```
quizon24@workstation:~$ sudo apt update
Hit:1 http://ph.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ph.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.0 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [40.0
kB]
Fetched 193 kB in 2s (83.8 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
2 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

*ansible all -m apt -a update_cache=true*

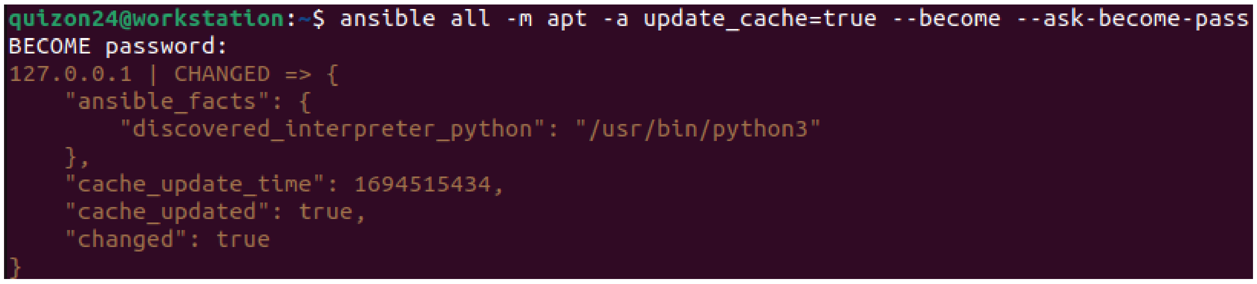
What is the result of the command? Is it successful?

It was not successful



```
quizon24@workstation:~$ ansible all -m apt -a update_cache=true
127.0.0.1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/li
sts/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)"
}
```

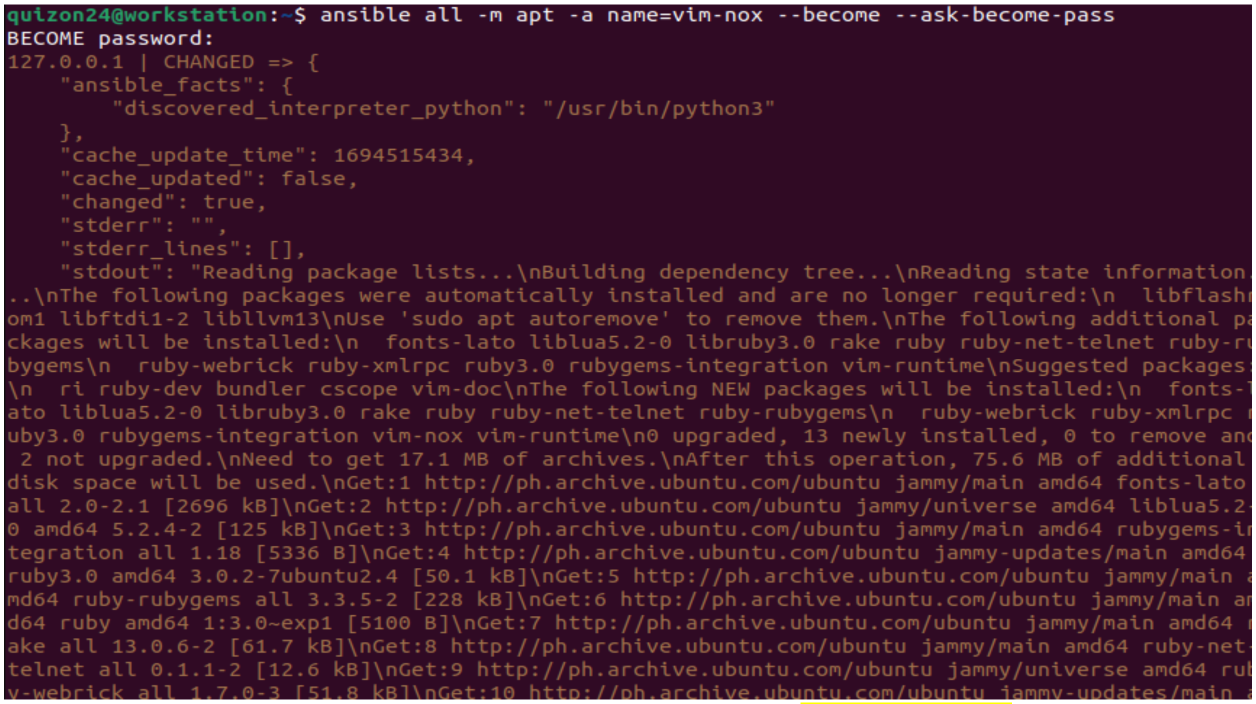
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same 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.



```
quizon24@workstation:~$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694515434,
    "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.

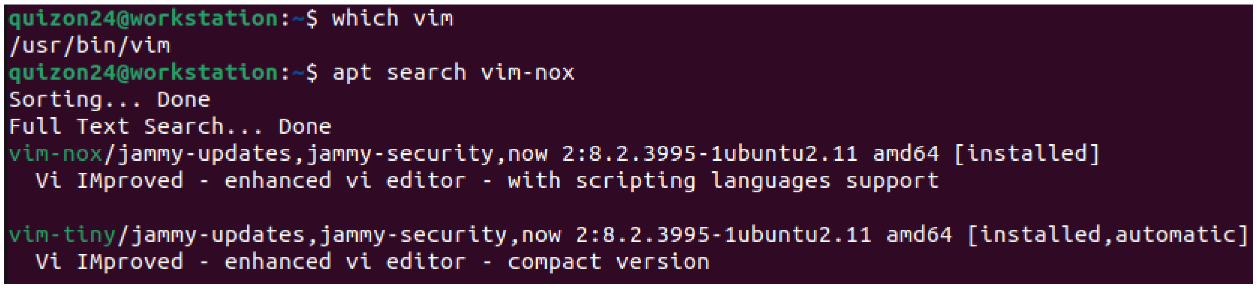
2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction.



```
quizon24@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694515434,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information
..\nThe following packages were automatically installed and are no longer required:\n  libflashr
om1 libftdi1-2 libllvm13\nUse 'sudo apt autoremove' to remove them.\nThe following additional pa
ckages will be installed:\n  fonts-lato liblua5.2-0 libruby3.0 rake ruby ruby-net-telnet ruby-ru
bygems\n  ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration vim-runtime\nSuggested packages:
\n  ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n  fonts-l
ato liblua5.2-0 libruby3.0 rake ruby ruby-net-telnet ruby-rubygems\n  ruby-webrick ruby-xmlrpc r
uby3.0 rubygems-integration vim-nox vim-runtime\n0 upgraded, 13 newly installed, 0 to remove and
 2 not upgraded.\nNeed to get 17.1 MB of archives.\nAfter this operation, 75.6 MB of additional
disk space will be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato
all 2.0-2.1 [2696 kB]\nGet:2 http://ph.archive.ubuntu.com/ubuntu jammy/universe amd64 liblua5.2-
0 amd64 5.2.4-2 [125 kB]\nGet:3 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-in
tegration all 1.18 [5336 B]\nGet:4 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64
ruby3.0 amd64 3.0.2-7ubuntu2.4 [50.1 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu jammy/main a
md64 ruby-rubygems all 3.3.5-2 [228 kB]\nGet:6 http://ph.archive.ubuntu.com/ubuntu jammy/main am
d64 ruby amd64 1:3.0~exp1 [5100 B]\nGet:7 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 r
ake all 13.0.6-2 [61.7 kB]\nGet:8 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-
telnet all 0.1.1-2 [12.6 kB]\nGet:9 http://ph.archive.ubuntu.com/ubuntu jammy/universe amd64 rub
y-webrick all 1.7.0-3 [51.8 kB]\nGet:10 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main a
```

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.

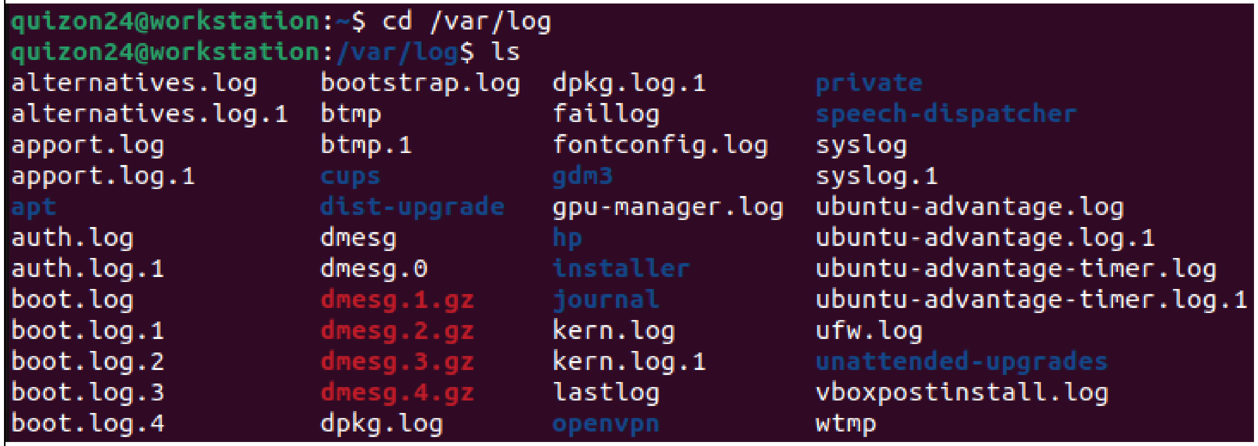
2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful? **Yes**



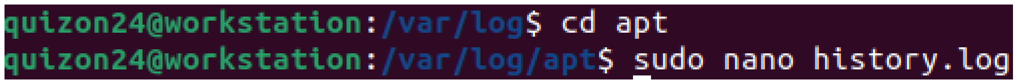
```
quizon24@workstation:~$ which vim
/usr/bin/vim
quizon24@workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 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.

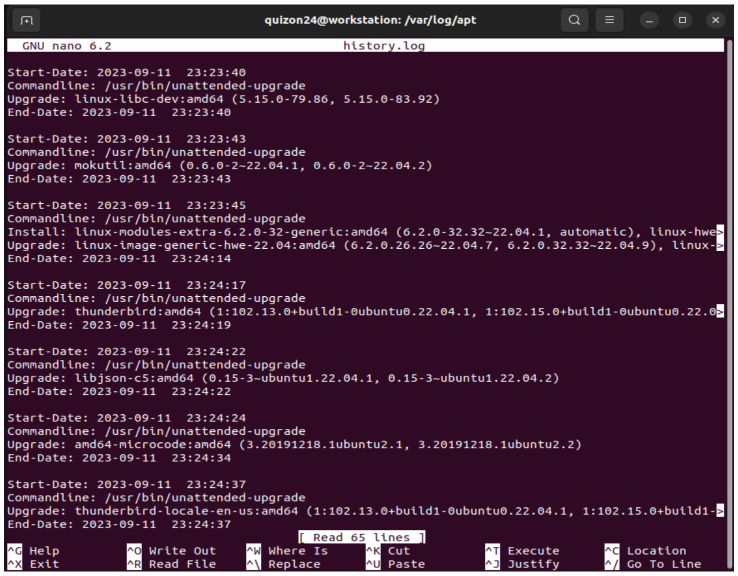
**From what I can understand, the texts that I can see in the history.log are the past updates, upgrades and installations.**



```
quizon24@workstation:~$ cd /var/log
quizon24@workstation:/var/log$ ls
alternatives.log    bootstrap.log  dpkg.log.1        private
alternatives.log.1  btmp           faillog           speech-dispatcher
apport.log          btmp.1         fontconfig.log    syslog
apport.log.1        cups           gdm3              syslog.1
apt                 dist-upgrade   gpu-manager.log   ubuntu-advantage.log
auth.log            dmesg          hp                ubuntu-advantage.log.1
auth.log.1          dmesg.0        installer         ubuntu-advantage-timer.log
boot.log            dmesg.1.gz     journal           ubuntu-advantage-timer.log.1
boot.log.1          dmesg.2.gz     kern.log          ufw.log
boot.log.2          dmesg.3.gz     kern.log.1        unattended-upgrades
boot.log.3          dmesg.4.gz     lastlog           vboxpostinstall.log
boot.log.4          dpkg.log       openvpn           wtmp
```



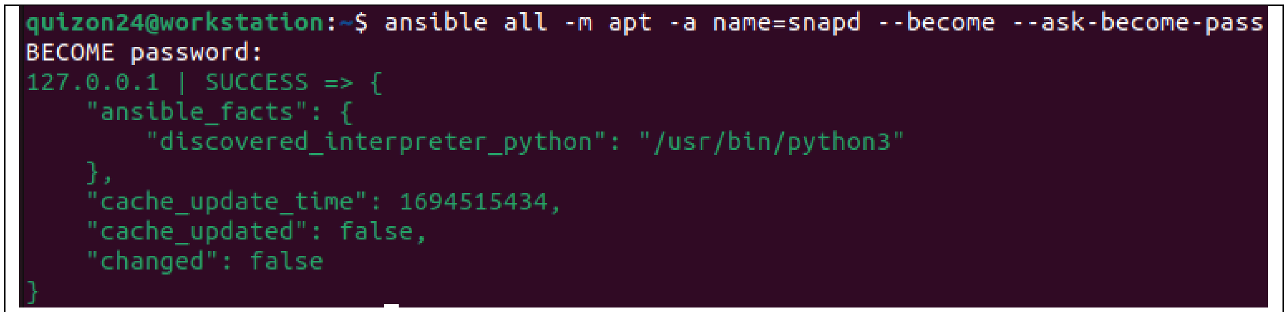
```
quizon24@workstation:/var/log$ cd apt
quizon24@workstation:/var/log/apt$ sudo nano history.log
```



```
quizon24@workstation: /var/log/apt

  GNU nano 6.2                          history.log

Start-Date: 2023-09-11  23:23:40
Commandline: /usr/bin/unattended-upgrade
Upgrade: linux-libc-dev:amd64 (5.15.0-79.86, 5.15.0-83.92)
End-Date: 2023-09-11  23:23:40

Start-Date: 2023-09-11  23:23:43
Commandline: /usr/bin/unattended-upgrade
Upgrade: mokutil:amd64 (0.6.0-2~22.04.1, 0.6.0-2~22.04.2)
End-Date: 2023-09-11  23:23:43

Start-Date: 2023-09-11  23:23:45
Commandline: /usr/bin/unattended-upgrade
Install: linux-modules-extra-6.2.0-32-generic:amd64 (6.2.0-32.32~22.04.1, automatic), linux-hwe>
Upgrade: linux-image-generic-hwe-22.04:amd64 (6.2.0.26.26~22.04.7, 6.2.0.32.32~22.04.9), linux->
End-Date: 2023-09-11  23:24:14

Start-Date: 2023-09-11  23:24:17
Commandline: /usr/bin/unattended-upgrade
Upgrade: thunderbird:amd64 (1:102.13.0+build1-0ubuntu0.22.04.1, 1:102.15.0+build1-0ubuntu0.22.0>
End-Date: 2023-09-11  23:24:19

Start-Date: 2023-09-11  23:24:22
Commandline: /usr/bin/unattended-upgrade
Upgrade: libjson-c5:amd64 (0.15-3~ubuntu1.22.04.1, 0.15-3~ubuntu1.22.04.2)
End-Date: 2023-09-11  23:24:22

Start-Date: 2023-09-11  23:24:24
Commandline: /usr/bin/unattended-upgrade
Upgrade: amd64-microcode:amd64 (3.20191218.1ubuntu2.1, 3.20191218.1ubuntu2.2)
End-Date: 2023-09-11  23:24:34

Start-Date: 2023-09-11  23:24:37
Commandline: /usr/bin/unattended-upgrade
Upgrade: thunderbird-locale-en-us:amd64 (1:102.13.0+build1-0ubuntu0.22.04.1, 1:102.15.0+build1->
End-Date: 2023-09-11  23:24:37
                                  [ Read 65 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

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.

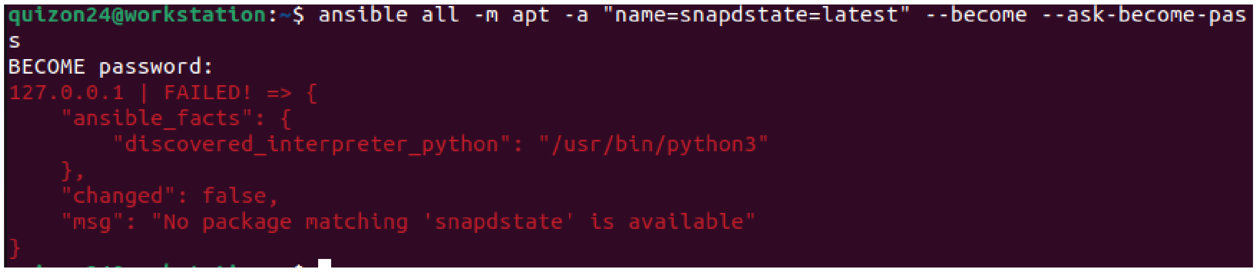
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*



```
quizon24@workstation:~$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694515434,
    "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? It was a success.

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.



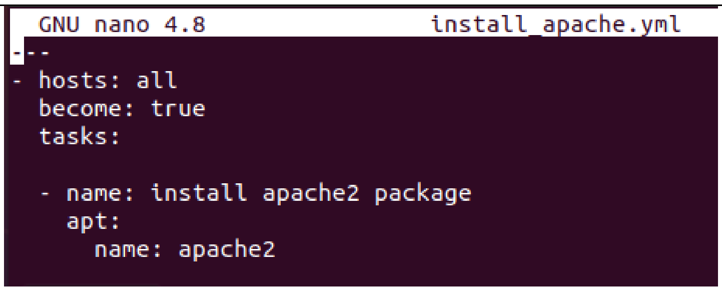
```
quizon24@workstation:~$ ansible all -m apt -a "name=snapdstate=latest" --become --ask-become-pas
s
BECOME password:
127.0.0.1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "No package matching 'snapdstate' is available"
}
```

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.

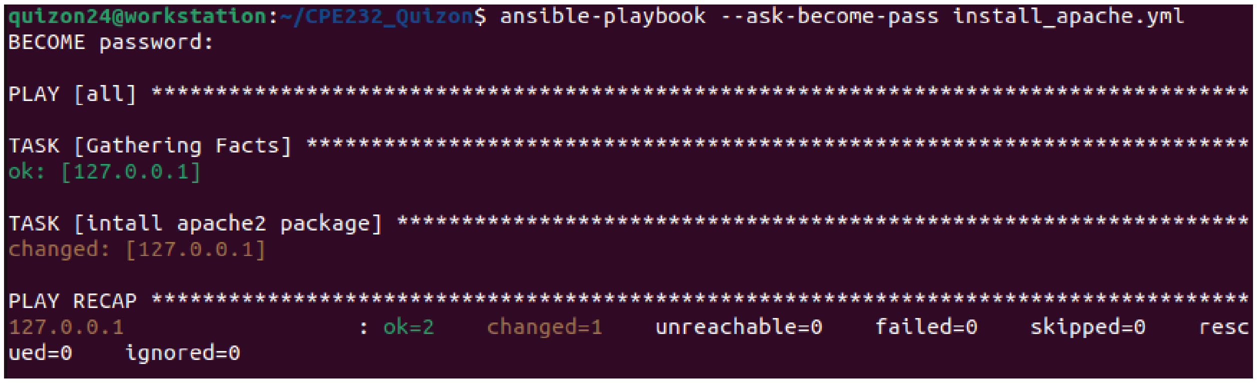
   When the editor appears, type the following:



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

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

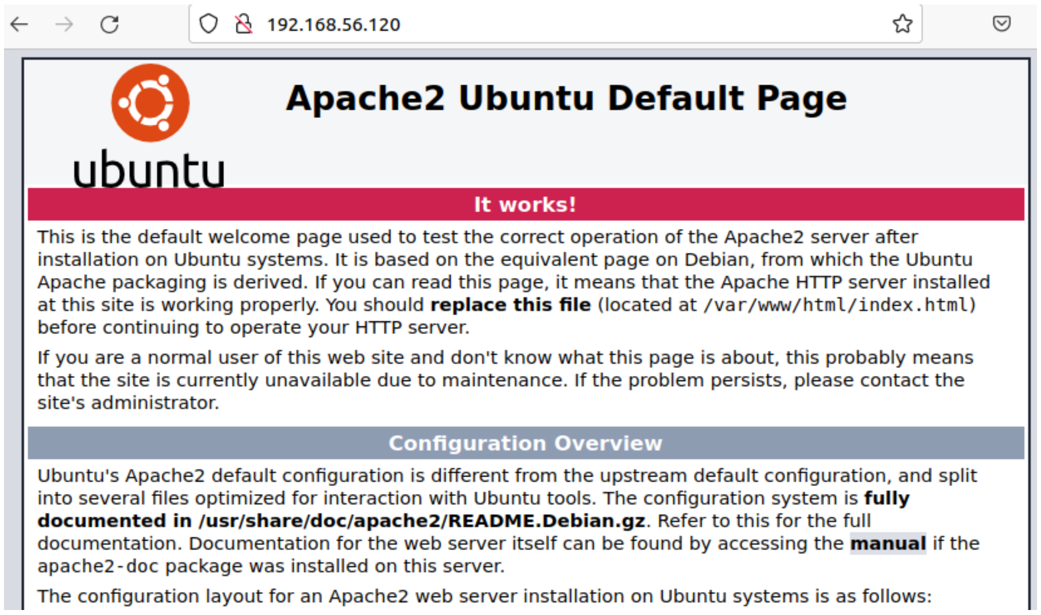
Make sure to save the file. Take note also of the alignments of the texts.

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



```
quizon24@workstation:~/CPE232_Quizon$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ***********************************************************************
ok: [127.0.0.1]

TASK [intall apache2 package] ****************************************************************
changed: [127.0.0.1]

PLAY RECAP ***********************************************************************************
127.0.0.1                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=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.



192.168.56.120

ubuntu

**Apache2 Ubuntu Default Page**

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

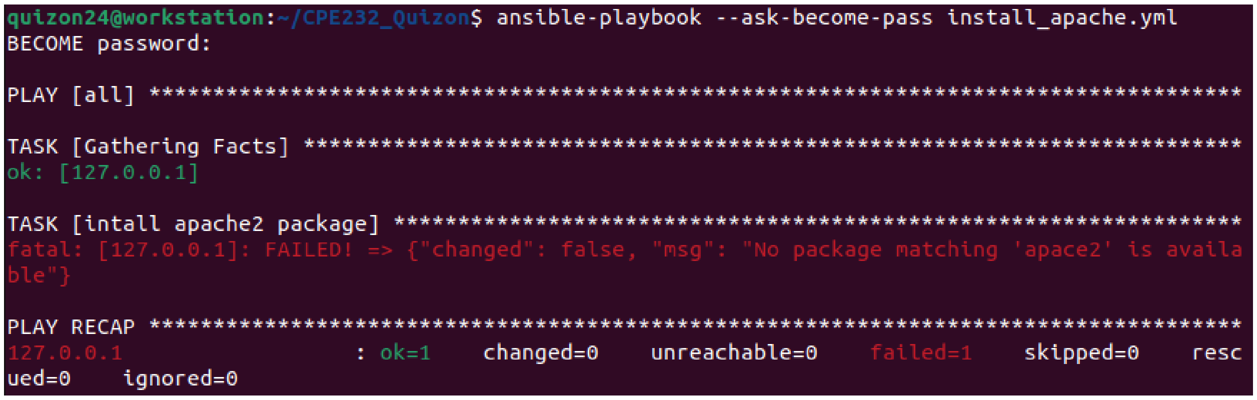
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

**Configuration Overview**

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

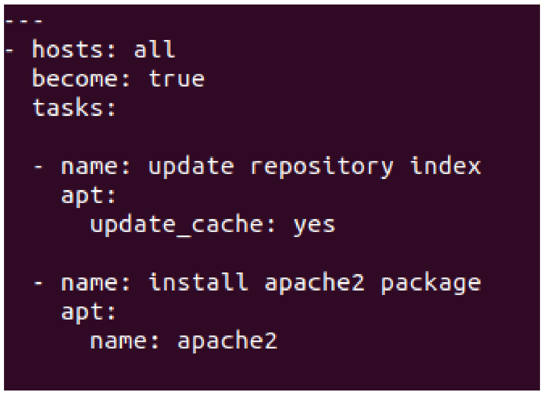
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

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? \



```
quizon24@workstation:~/CPE232_Quizon$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *************************************************************************
ok: [127.0.0.1]

TASK [intall apache2 package] ******************************************************************
fatal: [127.0.0.1]: FAILED! => {"changed": false, "msg": "No package matching 'apace2' is availa
ble"}

PLAY RECAP *************************************************************************************
127.0.0.1                  : ok=1    changed=0    unreachable=0    failed=1    skipped=0    resc
ued=0    ignored=0
```

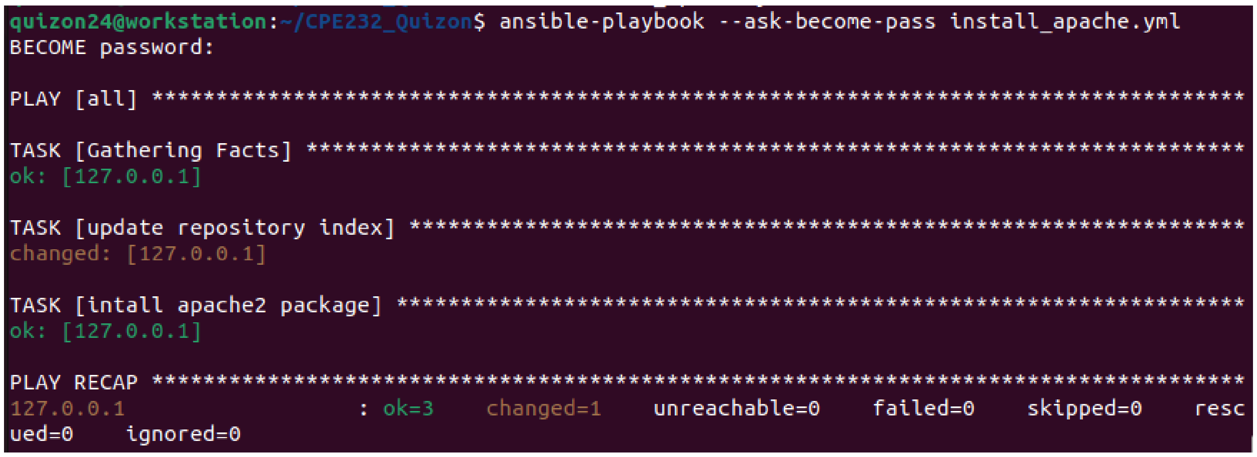
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache.* This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.



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

  - name: update repository index
    apt:
      update_cache: yes

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

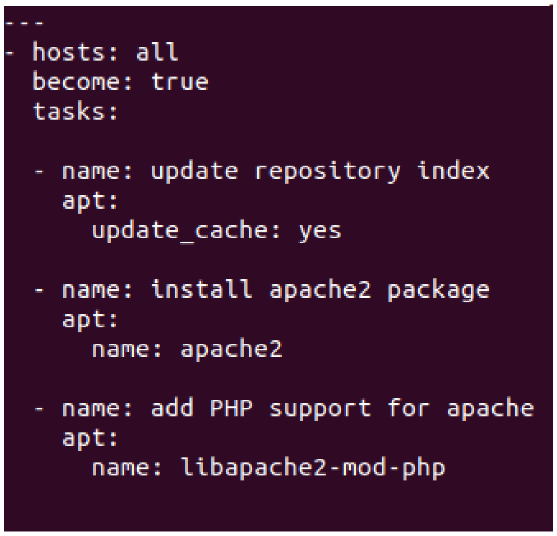
Save the changes to this file and exit.

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



```
quizon24@workstation:~/CPE232_Quizon$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ************************************************************************
ok: [127.0.0.1]

TASK [update repository index] ****************************************************************
changed: [127.0.0.1]

TASK [intall apache2 package] *****************************************************************
ok: [127.0.0.1]

PLAY RECAP ************************************************************************************
127.0.0.1                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
```

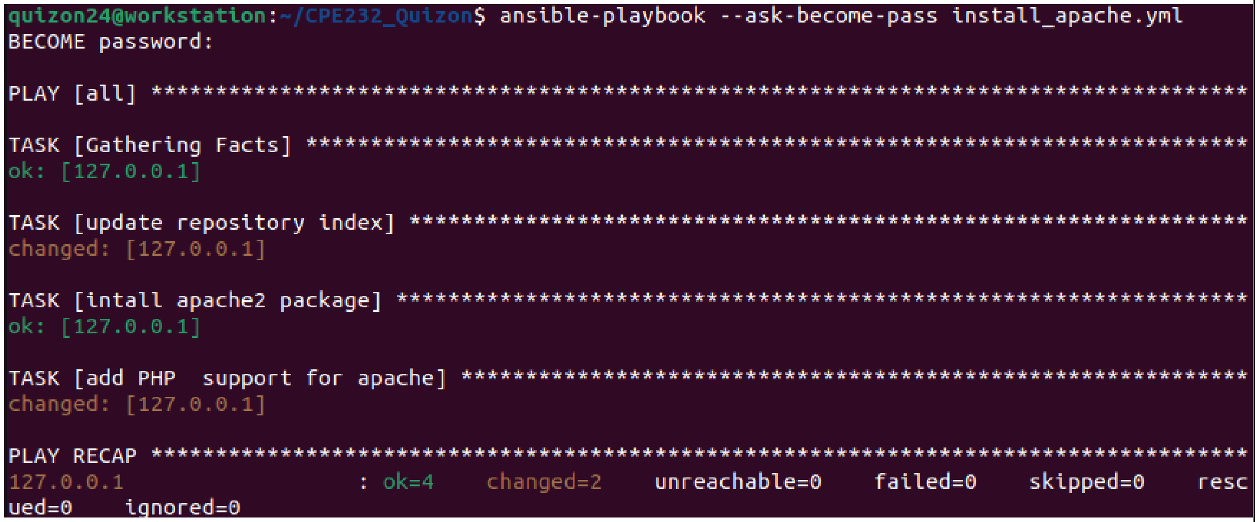
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
---
- 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
```

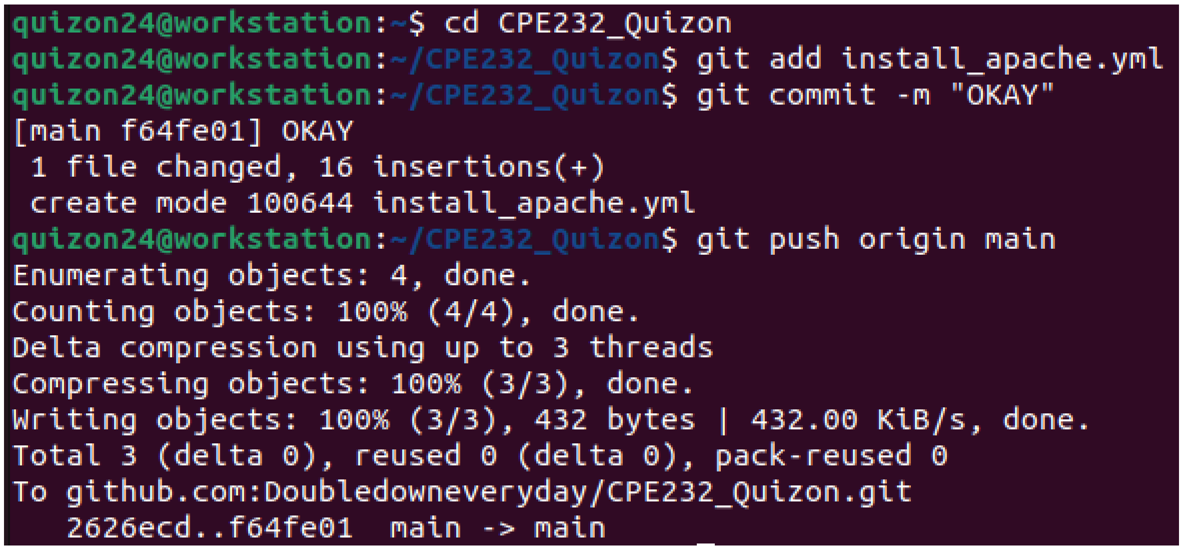
Save the changes to this file and exit.

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



```
quizon24@workstation:~/CPE232_Quizon$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *****************************************************************
ok: [127.0.0.1]

TASK [update repository index] *********************************************************
changed: [127.0.0.1]

TASK [intall apache2 package] **********************************************************
ok: [127.0.0.1]

TASK [add PHP  support for apache] *****************************************************
changed: [127.0.0.1]

PLAY RECAP *****************************************************************************
127.0.0.1                  : ok=4    changed=2    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
```

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



```
quizon24@workstation:~$ cd CPE232_Quizon
quizon24@workstation:~/CPE232_Quizon$ git add install_apache.yml
quizon24@workstation:~/CPE232_Quizon$ git commit -m "OKAY"
[main f64fe01] OKAY
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
quizon24@workstation:~/CPE232_Quizon$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 3 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 432 bytes | 432.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Doubledowneveryday/CPE232_Quizon.git
   2626ecd..f64fe01  main -> main
```

**Reflections:**

Answer the following:

1. What is the importance of using a playbook?
   It ensures consistent, efficient, and secure task execution.
2. Summarize what we have done on this activity.
   We created a repository for everything we did. Then, we setup the servers so that we can use ansible commands on the said remote servers. After that, we installed apache2 and made it work on the browser. With these, we completed creating the

playbook