Name: Dela cruz, Ivan Kenneth B.	**Date Performed:** Sep 11, 2023
Course/Section: CPE31S5	**Date Submitted:** Sep 12, 2023
Instructor: Engr. Roman Richard	**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:

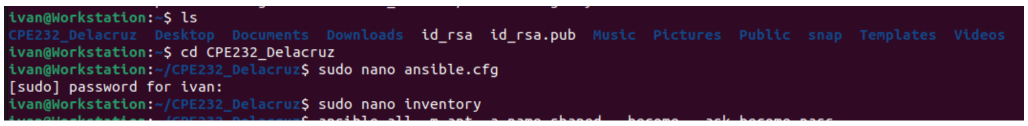
Elevated Ad hoc commands

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

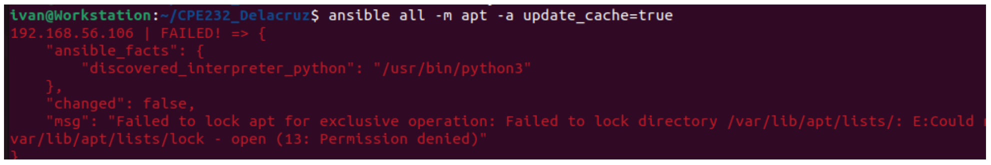
1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package 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:



```
ivan@Workstation:~$ ls
CPE232_Delacruz  Desktop  Documents  Downloads  id_rsa  id_rsa.pub  Music  Pictures  Public  snap  Templates  Videos
ivan@Workstation:~$ cd CPE232_Delacruz
ivan@Workstation:~/CPE232_Delacruz$ sudo nano ansible.cfg
[sudo] password for ivan:
ivan@Workstation:~/CPE232_Delacruz$ sudo nano inventory
```

ansible all -m apt -a update_cache=true

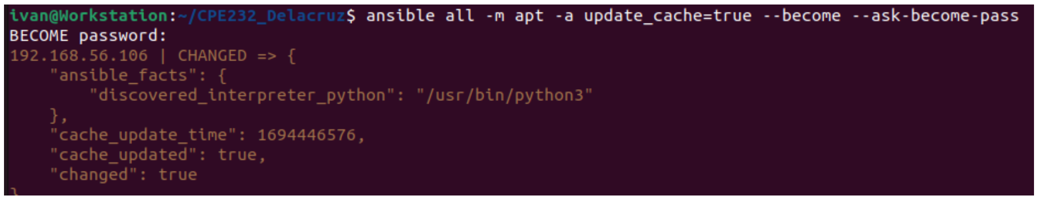
What is the result of the command? Is it successful? **No it didn't success**



```
ivan@Workstation:~/CPE232_Delacruz$ ansible all -m apt -a update_cache=true
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 directory /var/lib/apt/lists/: E:Could 
var/lib/apt/lists/lock - open (13: Permission denied)"
```

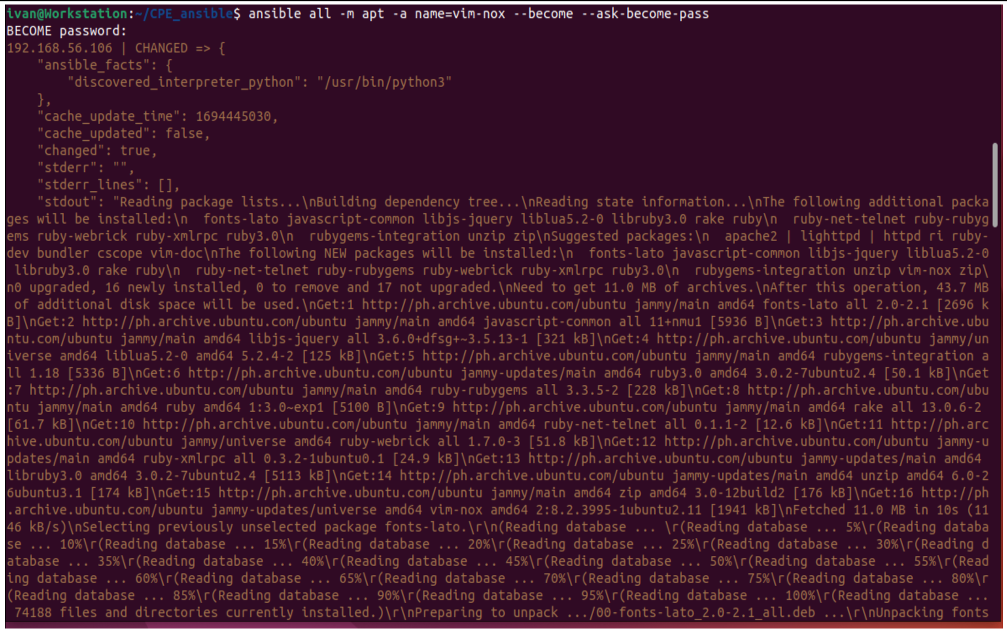
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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

You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

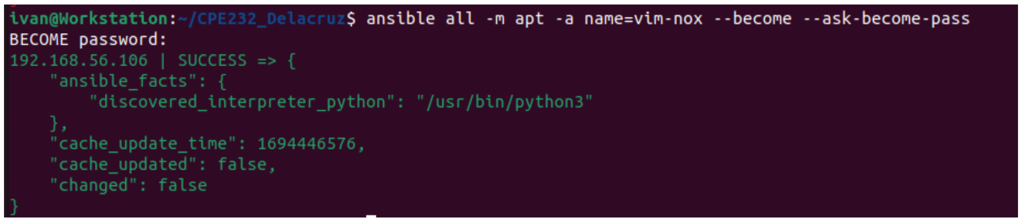


```
ivan@Workstation:~/CPE232_Delacruz$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
192.168.56.106 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694446576,
    "cache_updated": 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.

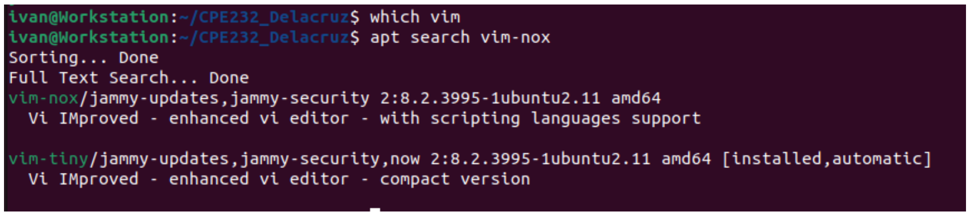


```
ivan@Workstation:~/CPE_ansible$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.106 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694445030,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following additional packa
ges will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  ruby-net-telnet ruby-rubyg
ems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration unzip zip\nSuggested packages:\n  apache2 | lighttpd | httpd ri ruby-
dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0
 libruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration unzip vim-nox zip\
n0 upgraded, 16 newly installed, 0 to remove and 17 not upgraded.\nNeed to get 11.0 MB of archives.\nAfter this operation, 43.7 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 k
B]\nGet:2 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5936 B]\nGet:3 http://ph.archive.ubu
ntu.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/un
iverse amd64 liblua5.2-0 amd64 5.2.4-2 [125 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration a
ll 1.18 [5336 B]\nGet:6 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64 3.0.2-7ubuntu2.4 [50.1 kB]\nGet
:7 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all 3.3.5-2 [228 kB]\nGet:8 http://ph.archive.ubuntu.com/ubu
ntu jammy/main amd64 ruby amd64 1:3.0~exp1 [5100 B]\nGet:9 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2
[61.7 kB]\nGet:10 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all 0.1.1-2 [12.6 kB]\nGet:11 http://ph.arc
hive.ubuntu.com/ubuntu jammy/universe amd64 ruby-webrick all 1.7.0-3 [51.8 kB]\nGet:12 http://ph.archive.ubuntu.com/ubuntu jammy-u
pdates/main amd64 ruby-xmlrpc all 0.3.2-1ubuntu0.1 [24.9 kB]\nGet:13 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64
libruby3.0 amd64 3.0.2-7ubuntu2.4 [5113 kB]\nGet:14 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-2
6ubuntu3.1 [174 kB]\nGet:15 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 zip amd64 3.0-12build2 [176 kB]\nGet:16 http://ph
.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 vim-nox amd64 2:8.2.3995-1ubuntu2.11 [1941 kB]\nFetched 11.0 MB in 10s (11
46 kB/s)\nSelecting previously unselected package fonts-lato.\r\n(Reading database ... \r(Reading database ... 5%\r(Reading databa
se ... 10%\r(Reading database ... 15%\r(Reading database ... 20%\r(Reading database ... 25%\r(Reading database ... 30%\r(Reading d
atabase ... 35%\r(Reading database ... 40%\r(Reading database ... 45%\r(Reading database ... 50%\r(Reading database ... 55%\r(Read
ing database ... 60%\r(Reading database ... 65%\r(Reading database ... 70%\r(Reading database ... 75%\r(Reading database ... 80%\r
(Reading database ... 85%\r(Reading database ... 90%\r(Reading database ... 95%\r(Reading database ... 100%\r(Reading database ...
 74188 files and directories currently installed.)\r\nPreparing to unpack .../00-fonts-lato_2.0-2.1_all.deb ...\r\nUnpacking fonts
```



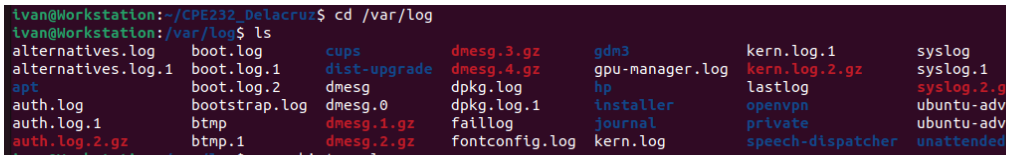
```
ivan@Workstation:~/CPE232_Delacruz$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694446576,
    "cache_updated": false,
    "changed": false
}
```

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?

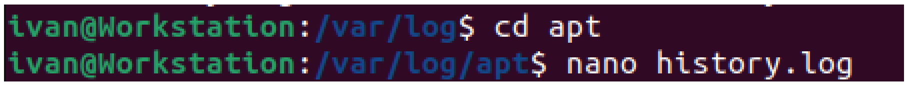


```
ivan@Workstation:~/CPE232_Delacruz$ which vim
ivan@Workstation:~/CPE232_Delacruz$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

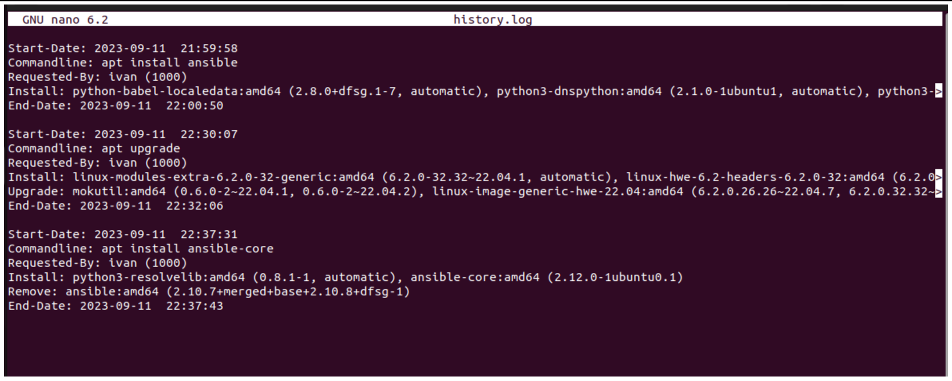
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.



```
ivan@Workstation:~/CPE232_Delacruz$ cd /var/log
ivan@Workstation:/var/log$ ls
alternatives.log    boot.log       cups          dmesg.3.gz     gdm3              kern.log.1         syslog
alternatives.log.1  boot.log.1     dist-upgrade  dmesg.4.gz     gpu-manager.log   kern.log.2.gz      syslog.1
apt                 boot.log.2     dmesg         dpkg.log       hp                lastlog            syslog.2.g
auth.log            bootstrap.log  dmesg.0       dpkg.log.1     installer         openvpn            ubuntu-adv
auth.log.1          btmp           dmesg.1.gz    faillog        journal           private            ubuntu-adv
auth.log.2.gz       btmp.1         dmesg.2.gz    fontconfig.log kern.log          speech-dispatcher  unattended
```



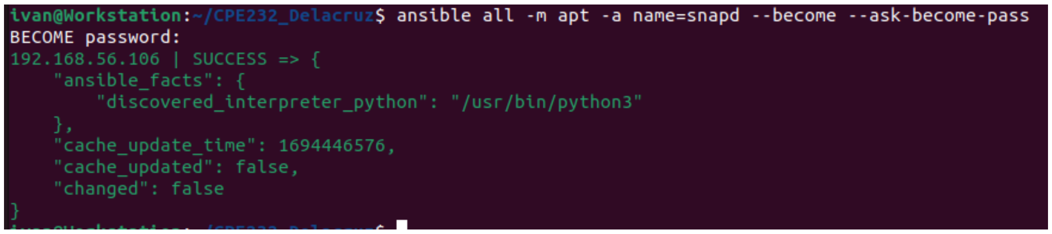
```
ivan@Workstation:/var/log$ cd apt
ivan@Workstation:/var/log/apt$ nano history.log
```



```
  GNU nano 6.2                                   history.log

Start-Date: 2023-09-11  21:59:58
Commandline: apt install ansible
Requested-By: ivan (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64 (2.1.0-1ubuntu1, automatic), python3->
End-Date: 2023-09-11  22:00:50

Start-Date: 2023-09-11  22:30:07
Commandline: apt upgrade
Requested-By: ivan (1000)
Install: linux-modules-extra-6.2.0-32-generic:amd64 (6.2.0-32.32~22.04.1, automatic), linux-hwe-6.2-headers-6.2.0-32:amd64 (6.2.0>
Upgrade: mokutil:amd64 (0.6.0-2~22.04.1, 0.6.0-2~22.04.2), linux-image-generic-hwe-22.04:amd64 (6.2.0.26.26~22.04.7, 6.2.0.32.32~>
End-Date: 2023-09-11  22:32:06

Start-Date: 2023-09-11  22:37:31
Commandline: apt install ansible-core
Requested-By: ivan (1000)
Install: python3-resolvelib:amd64 (0.8.1-1, automatic), ansible-core:amd64 (2.12.0-1ubuntu0.1)
Remove: ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1)
End-Date: 2023-09-11  22:37:43
```

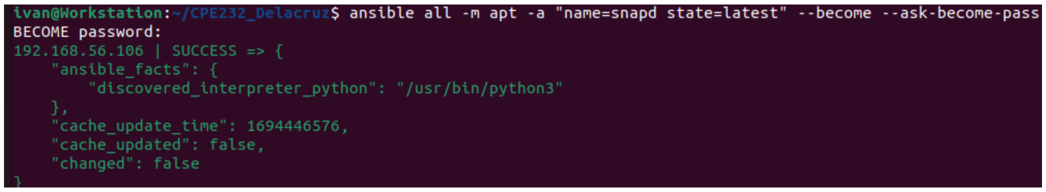
3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

   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?



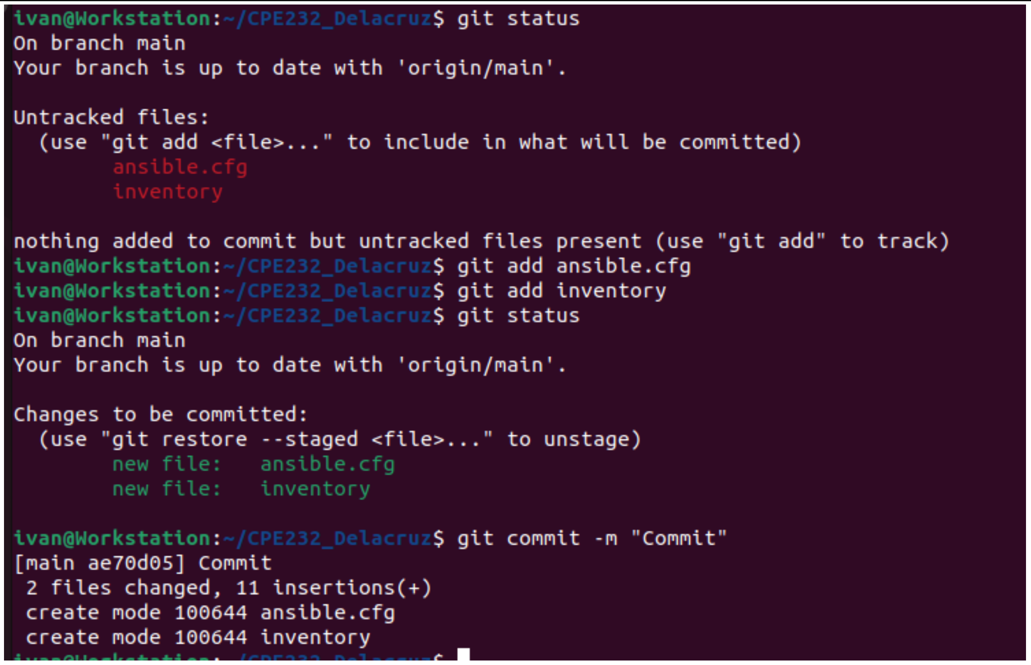
```
ivan@Workstation:~/CPE232_Delacruz$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694446576,
    "cache_updated": false,
    "changed": false
}
```

   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*
   Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.



```
ivan@Workstation:~/CPE232_Delacruz$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694446576,
    "cache_updated": false,
    "changed": false
}
```

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

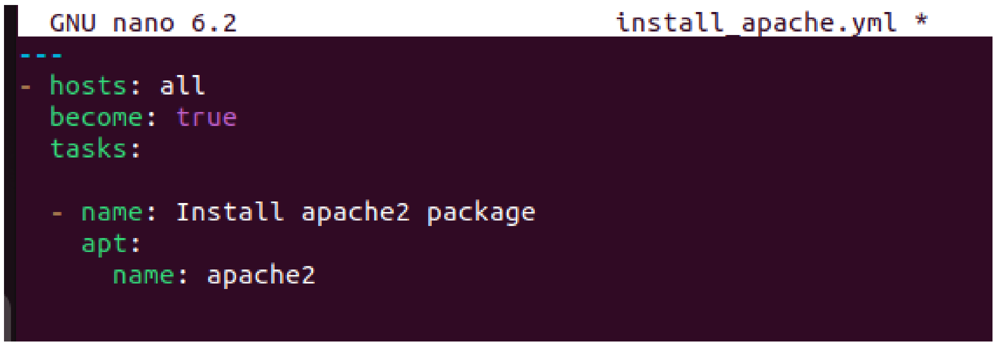


```
ivan@Workstation:~/CPE232_Delacruz$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ansible.cfg
        inventory

nothing added to commit but untracked files present (use "git add" to track)
ivan@Workstation:~/CPE232_Delacruz$ git add ansible.cfg
ivan@Workstation:~/CPE232_Delacruz$ git add inventory
ivan@Workstation:~/CPE232_Delacruz$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   ansible.cfg
        new file:   inventory

ivan@Workstation:~/CPE232_Delacruz$ git commit -m "Commit"
[main ae70d05] Commit
 2 files changed, 11 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 inventory
```

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. g

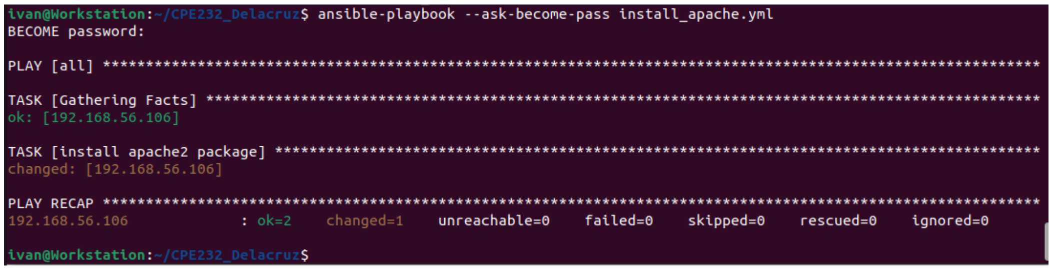
   When the editor appears, type the following:



```
  GNU nano 6.2                    install_apache.yml *
---
- hosts: all
  become: true
  tasks:

  - name: Install apache2 package
    apt:
      name: apache2
```

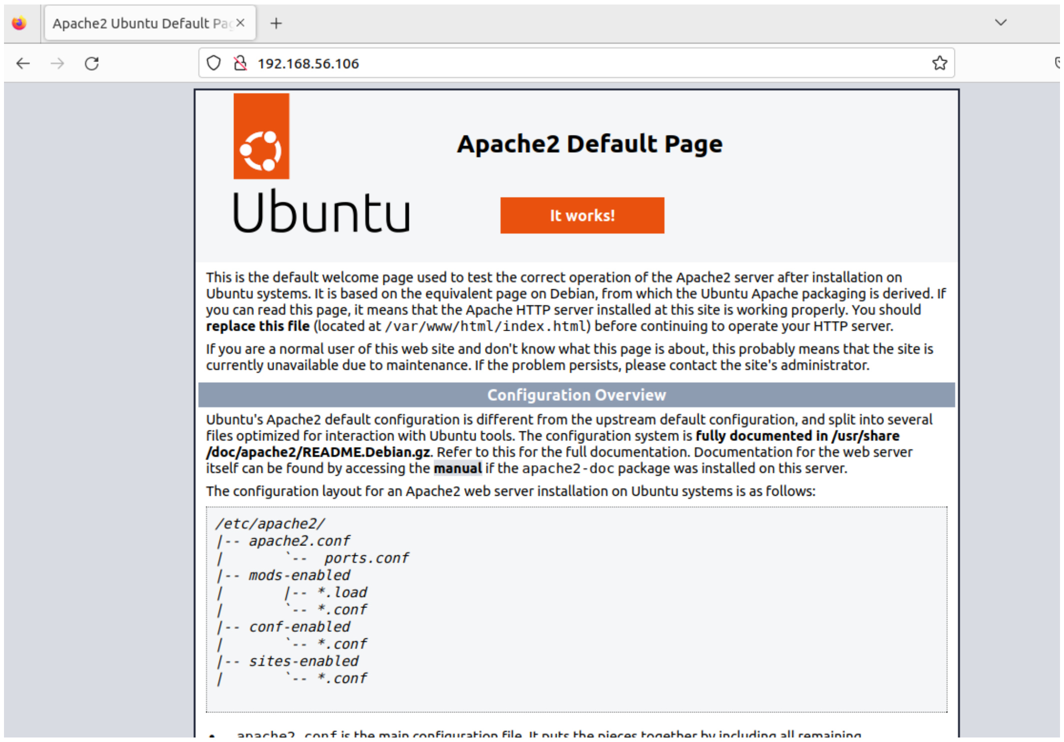
Make sure to save the file. Take note also of the alignments of the texts.

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



```
ivan@Workstation:~/CPE232_Delacruz$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

PLAY RECAP ***********************************************************************************************
192.168.56.106             : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ivan@Workstation:~/CPE232_Delacruz$
```

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.



Apache2 Default Page

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:

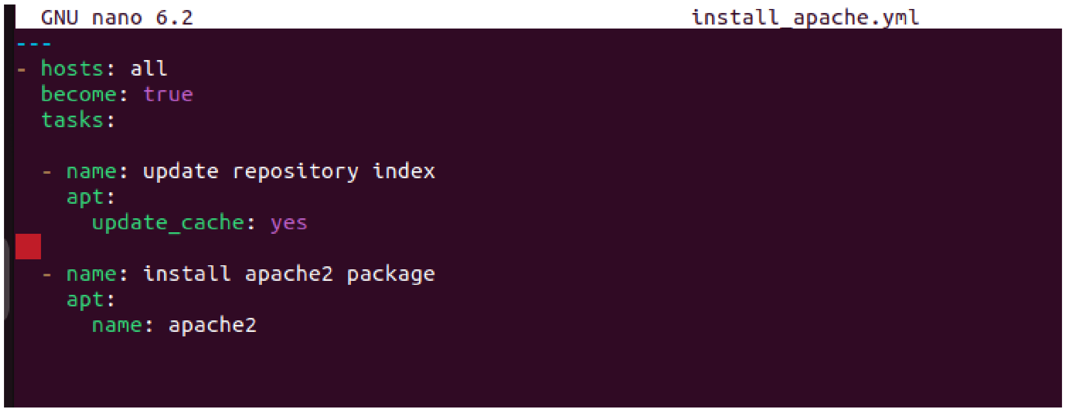
```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining

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?

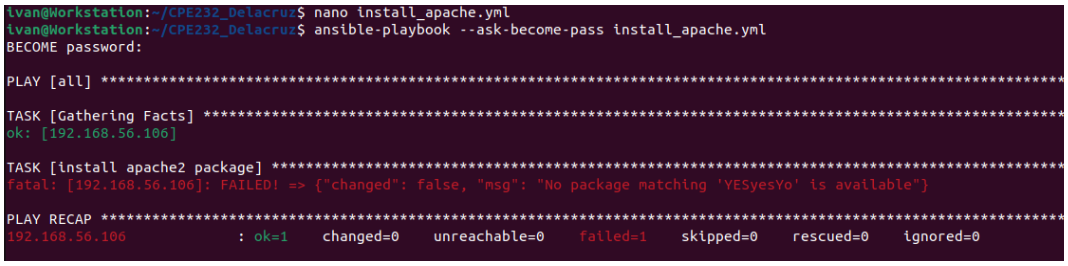
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache.* This command updates existing

package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.



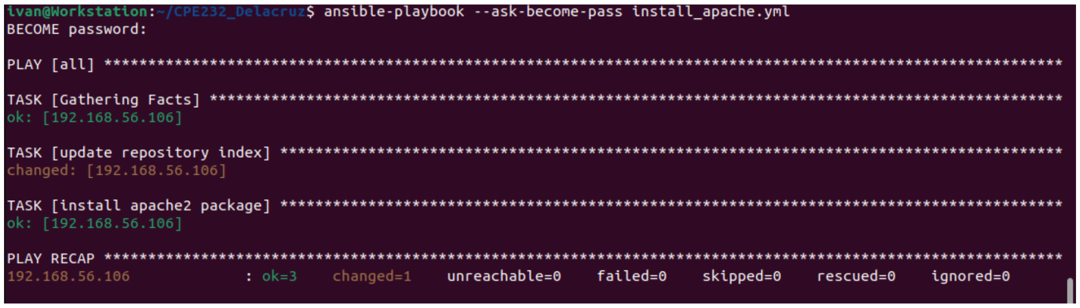
```
  GNU nano 6.2                                           install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

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

Save the changes to this file and exit.



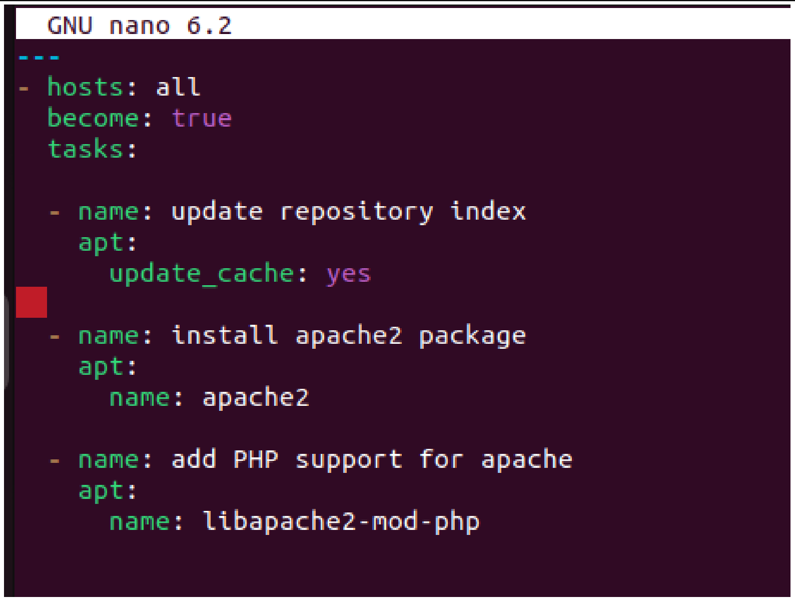
```
ivan@Workstation:~/CPE232_Delacruz$ nano install_apache.yml
ivan@Workstation:~/CPE232_Delacruz$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [install apache2 package] *************************************************************************
fatal: [192.168.56.106]: FAILED! => {"changed": false, "msg": "No package matching 'YESyesYo' is available"}

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

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



```
ivan@Workstation:~/CPE232_Delacruz$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

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

PLAY RECAP *********************************************************************************************
192.168.56.106             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

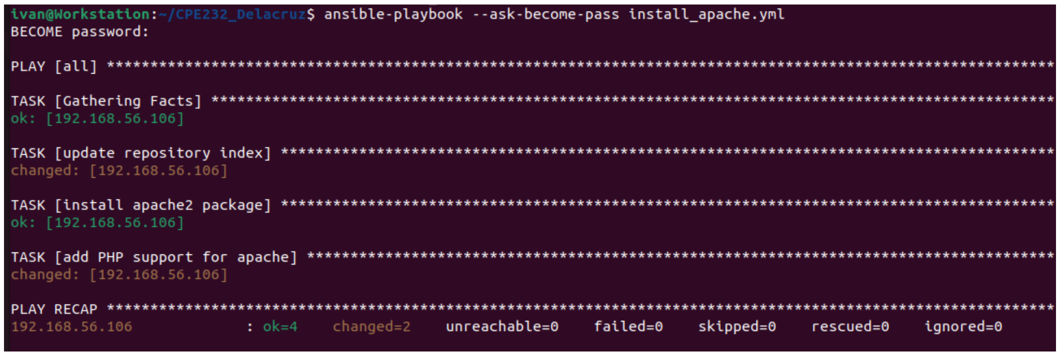
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
  GNU nano 6.2
---
- 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
```

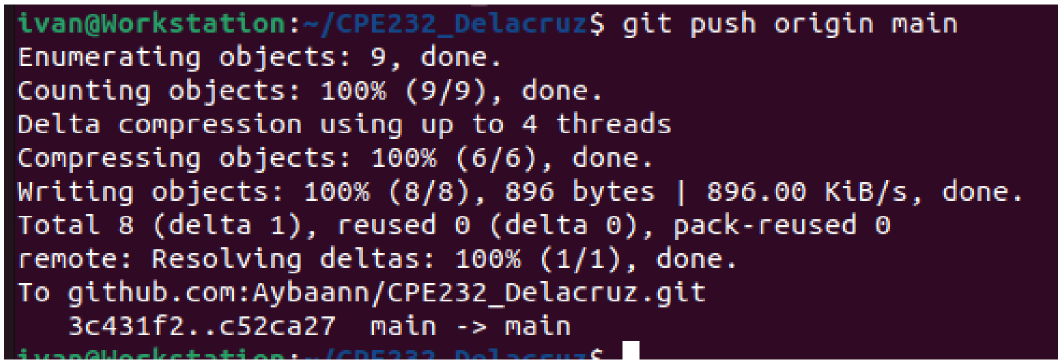
Save the changes to this file and exit.

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



```
ivan@Workstation:~/CPE232_Delacruz$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

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

TASK [add PHP support for apache] **************************************************************************************
changed: [192.168.56.106]

PLAY RECAP *************************************************************************************************************
192.168.56.106             : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

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



```
ivan@Workstation:~/CPE232_Delacruz$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 896 bytes | 896.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:Aybaann/CPE232_Delacruz.git
   3c431f2..c52ca27  main -> main
ivan@Workstation:~/CPE232_Delacruz$
```

Github link:

https://github.com/Aybaann/CPE232_Delacruz.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
    - The importance of the playbooks can define the state we desire for all the remote servers that we manage. Also, The condition that we added in the playbook can be saved, and it means that the play can be shared and used again.

2. Summarize what we have done on this activity.
    - In this activity, we created a .cfg and file for the ansible and put a script inside of it. We also used various commands that made changes to the remote machine. Throughout the activity, I was able to learn ad hoc commands that would install,update and upgrade packages in the remote machine and also created playbooks that record and execute ansible's configuration.