Name: Flores, Joshua Mico A.	**Date Performed: 09/11/23**
Course/Section:CPE31S4	**Date Submitted: 09/12/23**
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

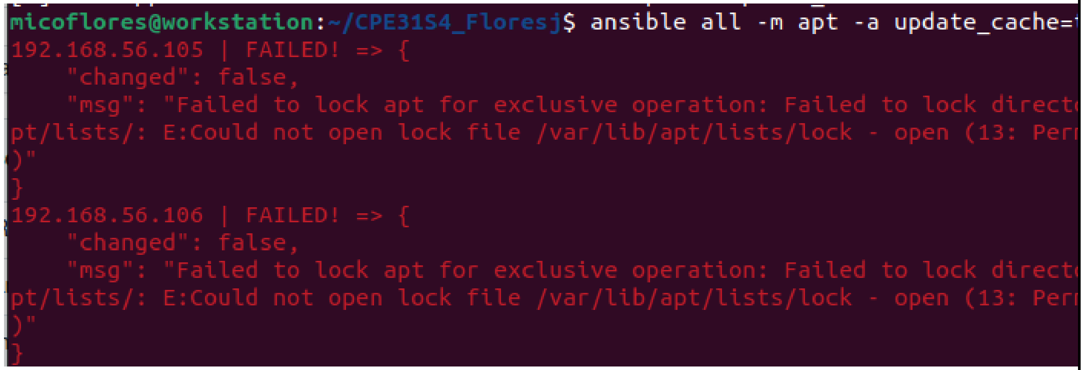
Task 1: Run elevated ad hoc commands

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

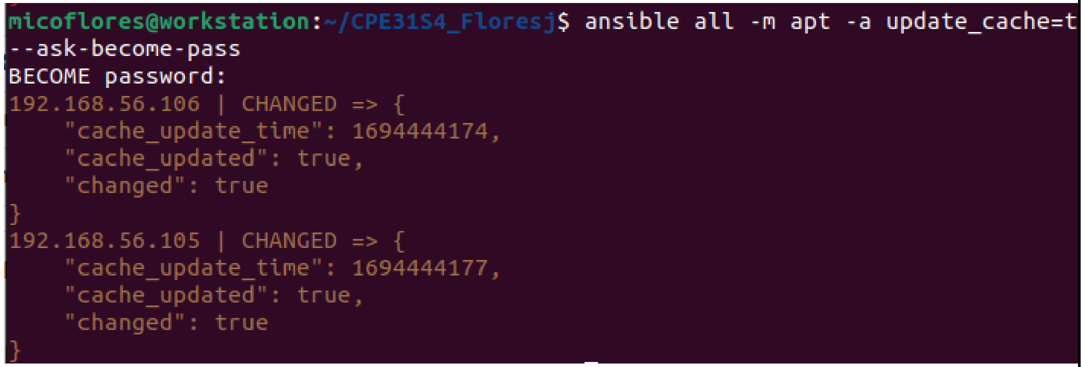
What is the result of the command? Is it successful?



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible all -m apt -a update_cache=
192.168.56.105 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock direct
pt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Per
)"
}
192.168.56.106 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock direct
pt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Per
)"
}
```

The command took some time to initialize but then it failed to execute.

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.

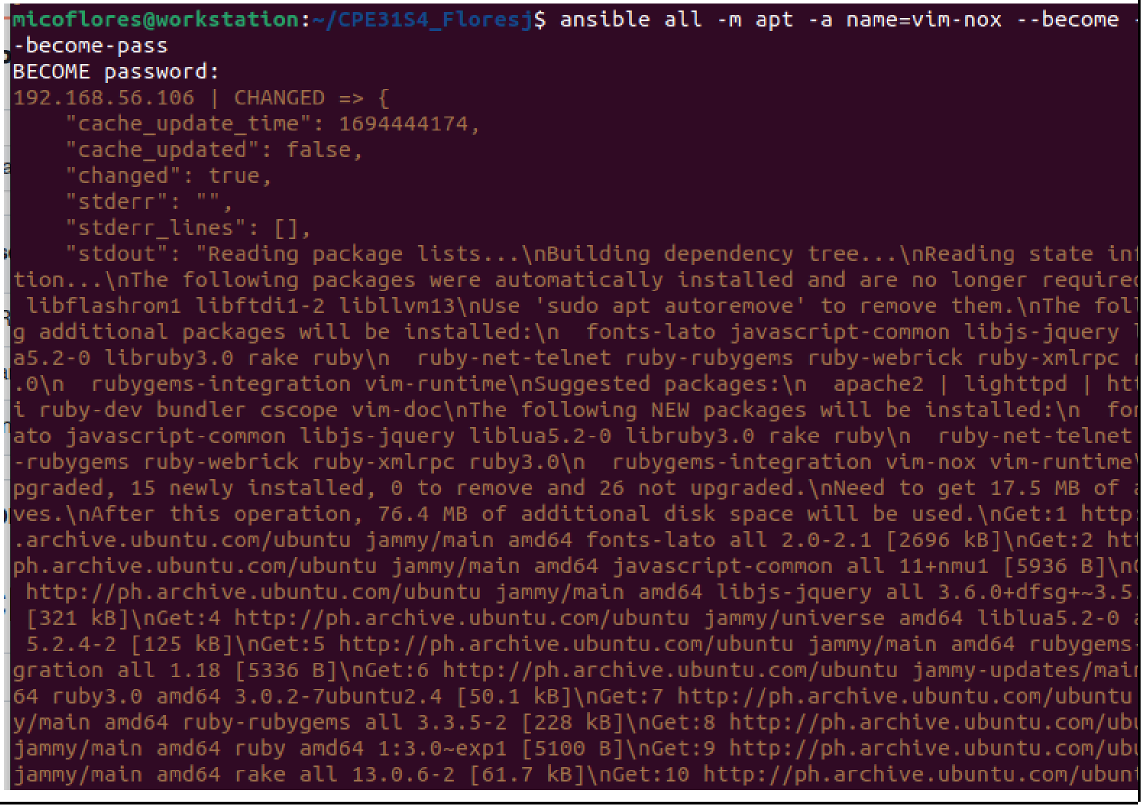


```
micoflores@workstation:~/CPE31S4_Floresj$ ansible all -m apt -a update_cache=t
--ask-become-pass
BECOME password:
192.168.56.106 | CHANGED => {
    "cache_update_time": 1694444174,
    "cache_updated": true,
    "changed": true
}
192.168.56.105 | CHANGED => {
    "cache_update_time": 1694444177,
    "cache_updated": true,
    "changed": true
}
```

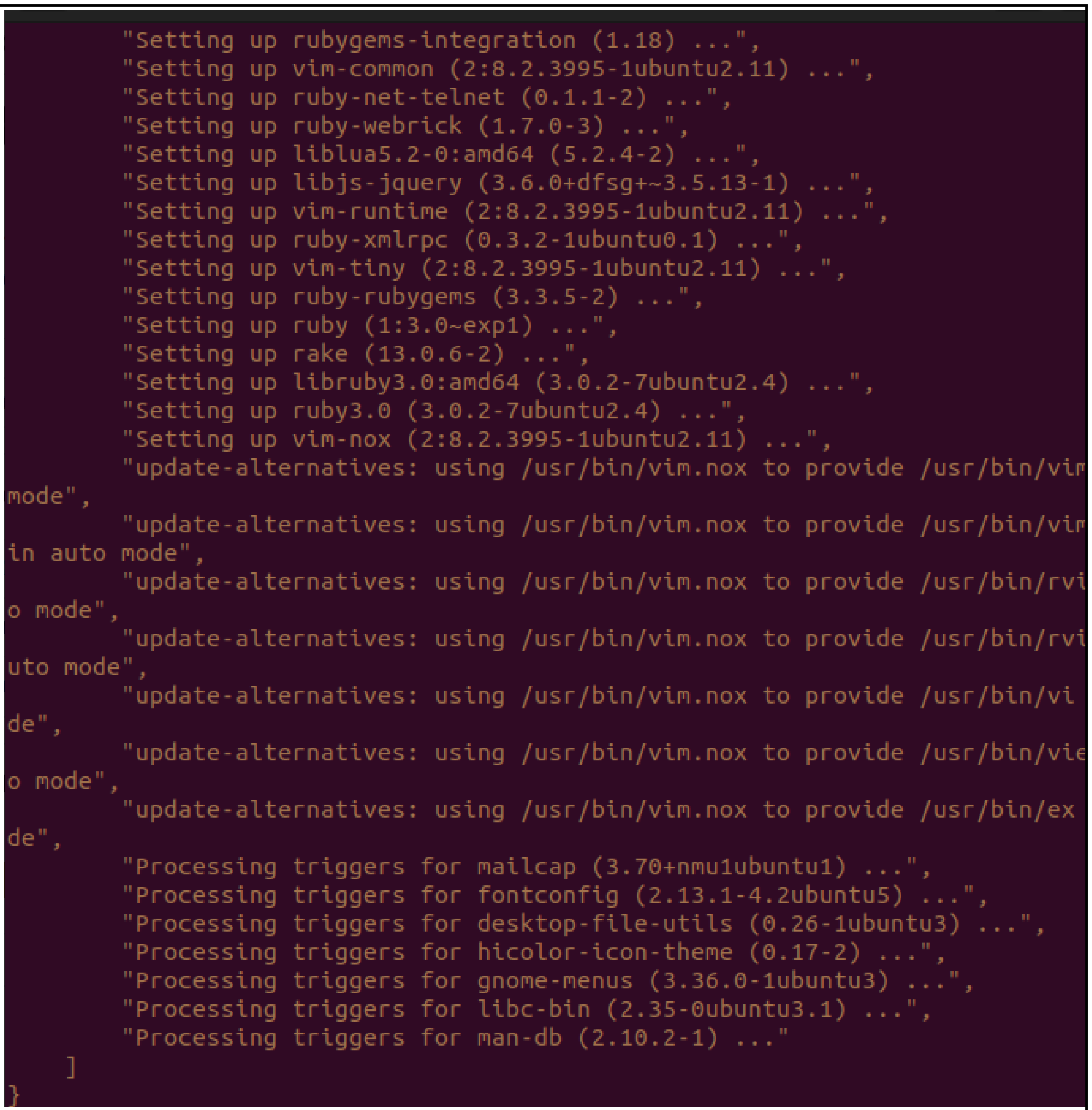
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

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.



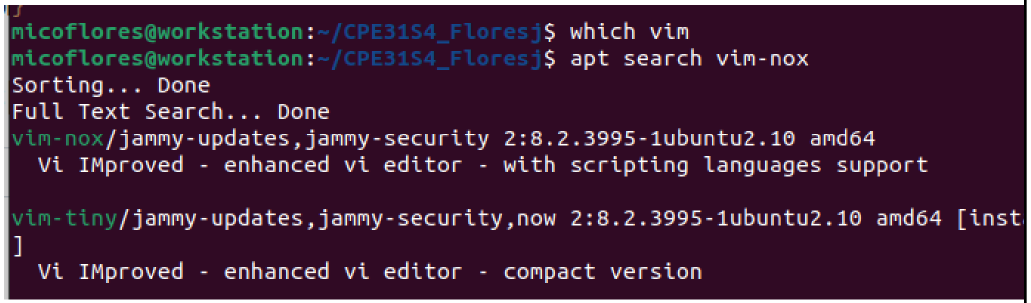
```
micoflores@workstation:~/CPE31S4_Floresj$ ansible all -m apt -a name=vim-nox --become -
-become-pass
BECOME password:
192.168.56.106 | CHANGED => {
    "cache_update_time": 1694444174,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state inf
tion...\nThe following packages were automatically installed and are no longer required
 libflashrom1 libftdi1-2 libllvm13\nUse 'sudo apt autoremove' to remove them.\nThe foll
g additional packages will be installed:\n  fonts-lato javascript-common libjs-jquery l
a5.2-0 libruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc r
.0\n  rubygems-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | htt
i ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n  fon
ato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  ruby-net-telnet
-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration vim-nox vim-runtime\
pgraded, 15 newly installed, 0 to remove and 26 not upgraded.\nNeed to get 17.5 MB of a
ves.\nAfter this operation, 76.4 MB of additional disk space will be used.\nGet:1 http:
.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\nGet:2 htt
ph.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5936 B]\nG
 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 libjs-jquery all 3.6.0+dfsg+~3.5.
 [321 kB]\nGet:4 http://ph.archive.ubuntu.com/ubuntu jammy/universe amd64 liblua5.2-0 a
 5.2.4-2 [125 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-
gration all 1.18 [5336 B]\nGet:6 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main
64 ruby3.0 amd64 3.0.2-7ubuntu2.4 [50.1 kB]\nGet:7 http://ph.archive.ubuntu.com/ubuntu
y/main amd64 ruby-rubygems all 3.3.5-2 [228 kB]\nGet:8 http://ph.archive.ubuntu.com/ubu
jammy/main amd64 ruby amd64 1:3.0~exp1 [5100 B]\nGet:9 http://ph.archive.ubuntu.com/ubu
jammy/main amd64 rake all 13.0.6-2 [61.7 kB]\nGet:10 http://ph.archive.ubuntu.com/ubunt
```



```
        "Setting up rubygems-integration (1.18) ...",
        "Setting up vim-common (2:8.2.3995-1ubuntu2.11) ...",
        "Setting up ruby-net-telnet (0.1.1-2) ...",
        "Setting up ruby-webrick (1.7.0-3) ...",
        "Setting up liblua5.2-0:amd64 (5.2.4-2) ...",
        "Setting up libjs-jquery (3.6.0+dfsg+~3.5.13-1) ...",
        "Setting up vim-runtime (2:8.2.3995-1ubuntu2.11) ...",
        "Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) ...",
        "Setting up vim-tiny (2:8.2.3995-1ubuntu2.11) ...",
        "Setting up ruby-rubygems (3.3.5-2) ...",
        "Setting up ruby (1:3.0~exp1) ...",
        "Setting up rake (13.0.6-2) ...",
        "Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.4) ...",
        "Setting up ruby3.0 (3.0.2-7ubuntu2.4) ...",
        "Setting up vim-nox (2:8.2.3995-1ubuntu2.11) ...",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vim
mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vim
in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/rvi
o mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/rvi
uto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vi
de",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vie
o mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/ex
de",
        "Processing triggers for mailcap (3.70+nmu1ubuntu1) ...",
        "Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...",
        "Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...",
        "Processing triggers for hicolor-icon-theme (0.17-2) ...",
        "Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...",
        "Processing triggers for libc-bin (2.35-0ubuntu3.1) ...",
        "Processing triggers for man-db (2.10.2-1) ..."
    ]
}
```

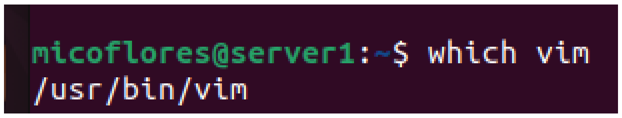
2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?

Workstation

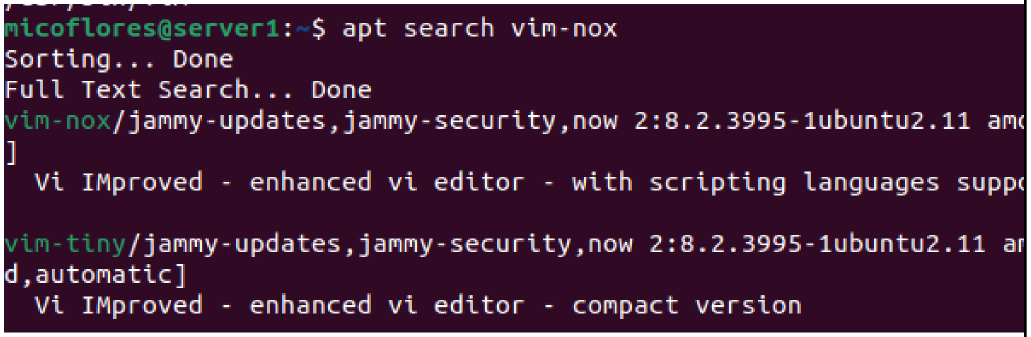


```
micoflores@workstation:~/CPE31S4_Floresj$ which vim
micoflores@workstation:~/CPE31S4_Floresj$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.10 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.10 amd64 [inst
]
  Vi IMproved - enhanced vi editor - compact version
```

Server 1

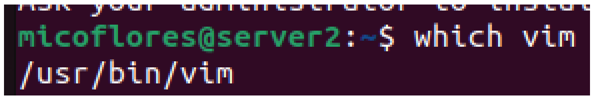


```
micoflores@server1:~$ which vim
/usr/bin/vim
```

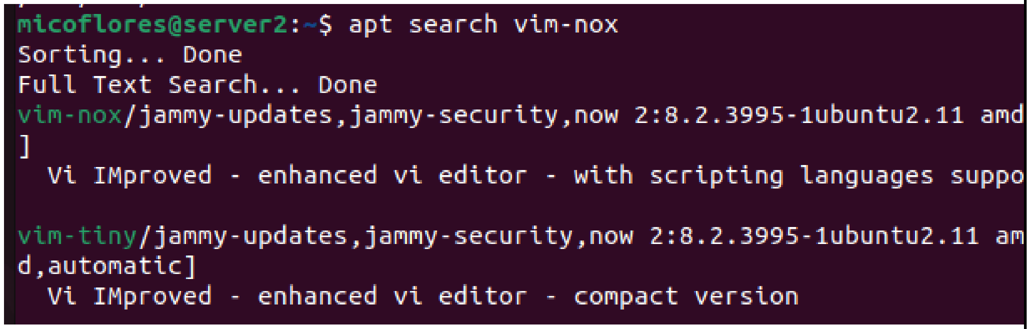


```
micoflores@server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amc
]
  Vi IMproved - enhanced vi editor - with scripting languages suppo

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

Server 2



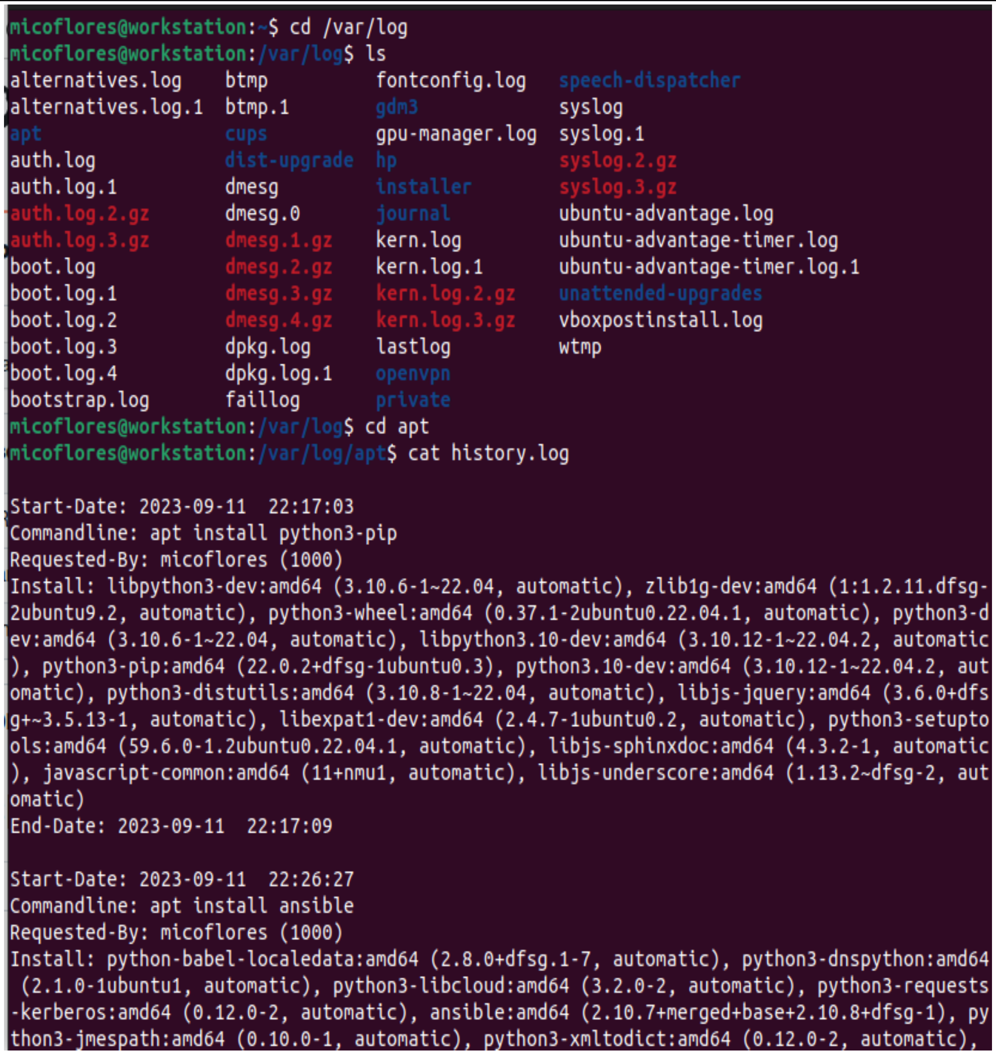
```
micoflores@server2:~$ which vim
/usr/bin/vim
```



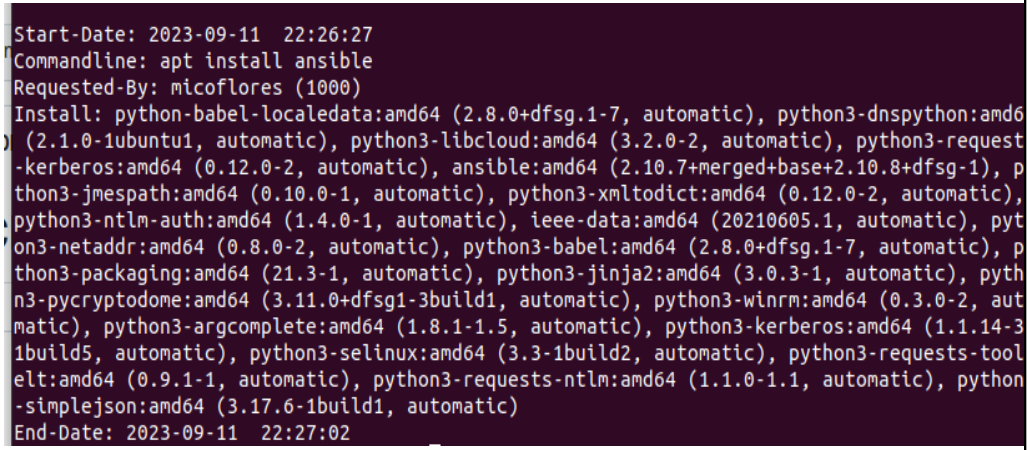
```
micoflores@server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd
]
  Vi IMproved - enhanced vi editor - with scripting languages suppo

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

The package has been installed in both server 1 and server 2 as shown in the images provided.

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.



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

Start-Date: 2023-09-11  22:17:03
Commandline: apt install python3-pip
Requested-By: micoflores (1000)
Install: libpython3-dev:amd64 (3.10.6-1~22.04, automatic), zlib1g-dev:amd64 (1:1.2.11.dfsg-
2ubuntu9.2, automatic), python3-wheel:amd64 (0.37.1-2ubuntu0.22.04.1, automatic), python3-d
ev:amd64 (3.10.6-1~22.04, automatic), libpython3.10-dev:amd64 (3.10.12-1~22.04.2, automatic
), python3-pip:amd64 (22.0.2+dfsg-1ubuntu0.3), python3.10-dev:amd64 (3.10.12-1~22.04.2, aut
omatic), python3-distutils:amd64 (3.10.8-1~22.04, automatic), libjs-jquery:amd64 (3.6.0+dfs
g+~3.5.13-1, automatic), libexpat1-dev:amd64 (2.4.7-1ubuntu0.2, automatic), python3-setupto
ols:amd64 (59.6.0-1.2ubuntu0.22.04.1, automatic), libjs-sphinxdoc:amd64 (4.3.2-1, automatic
), javascript-common:amd64 (11+nmu1, automatic), libjs-underscore:amd64 (1.13.2~dfsg-2, aut
omatic)
End-Date: 2023-09-11  22:17:09

Start-Date: 2023-09-11  22:26:27
Commandline: apt install ansible
Requested-By: micoflores (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64
 (2.1.0-1ubuntu1, automatic), python3-libcloud:amd64 (3.2.0-2, automatic), python3-requests
-kerberos:amd64 (0.12.0-2, automatic), ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1), py
thon3-jmespath:amd64 (0.10.0-1, automatic), python3-xmltodict:amd64 (0.12.0-2, automatic),
```

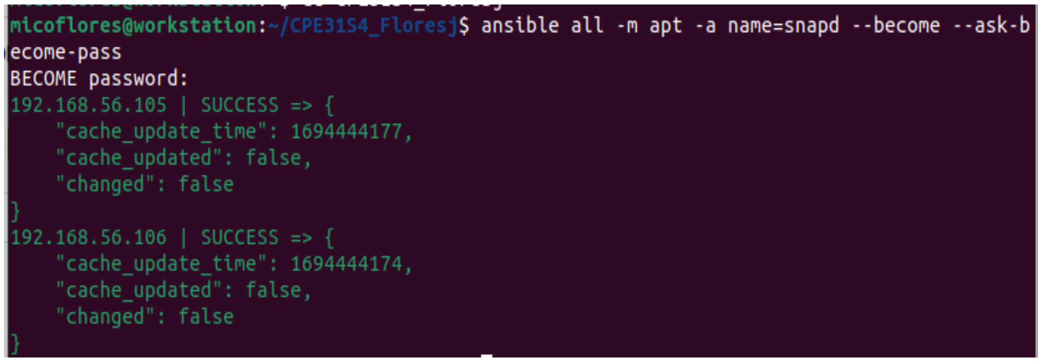


```
Start-Date: 2023-09-11  22:26:27
Commandline: apt install ansible
Requested-By: micoflores (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd6
 (2.1.0-1ubuntu1, automatic), python3-libcloud:amd64 (3.2.0-2, automatic), python3-request
-kerberos:amd64 (0.12.0-2, automatic), ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1), p
thon3-jmespath:amd64 (0.10.0-1, automatic), python3-xmltodict:amd64 (0.12.0-2, automatic),
python3-ntlm-auth:amd64 (1.4.0-1, automatic), ieee-data:amd64 (20210605.1, automatic), pyt
on3-netaddr:amd64 (0.8.0-2, automatic), python3-babel:amd64 (2.8.0+dfsg.1-7, automatic), p
thon3-packaging:amd64 (21.3-1, automatic), python3-jinja2:amd64 (3.0.3-1, automatic), pyth
n3-pycryptodome:amd64 (3.11.0+dfsg1-3build1, automatic), python3-winrm:amd64 (0.3.0-2, aut
matic), python3-argcomplete:amd64 (1.8.1-1.5, automatic), python3-kerberos:amd64 (1.1.14-3
1build5, automatic), python3-selinux:amd64 (3.3-1build2, automatic), python3-requests-tool
elt:amd64 (0.9.1-1, automatic), python3-requests-ntlm:amd64 (1.1.0-1.1, automatic), python
-simplejson:amd64 (3.17.6-1build1, automatic)
End-Date: 2023-09-11  22:27:02
```

It shows the start time and end time when the servers were accessed.

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.

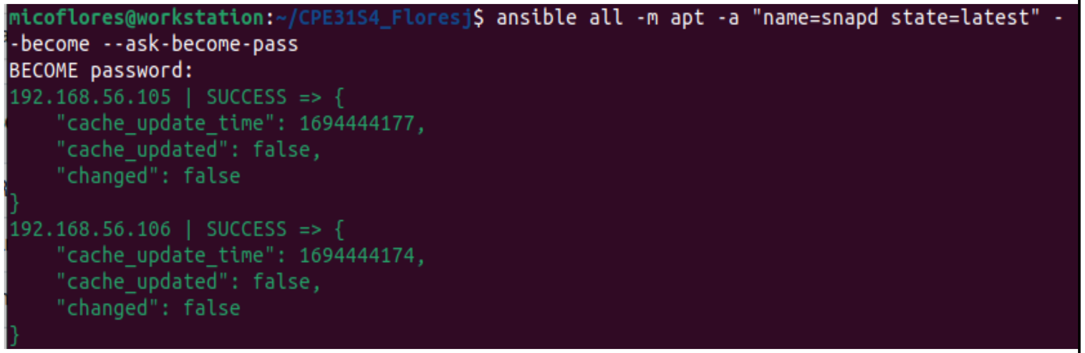
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*
   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible all -m apt -a name=snapd --become --ask-b
ecome-pass
BECOME password:
192.168.56.105 | SUCCESS => {
    "cache_update_time": 1694444177,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "cache_update_time": 1694444174,
    "cache_updated": false,
    "changed": false
}
```

   The servers have installed a package called snapd.

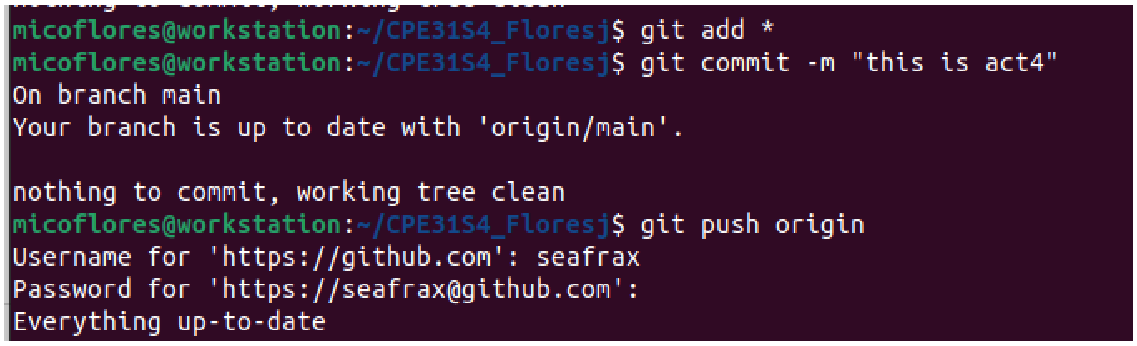
   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*
   Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible all -m apt -a "name=snapd state=latest" -
-become --ask-become-pass
BECOME password:
192.168.56.105 | SUCCESS => {
    "cache_update_time": 1694444177,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "cache_update_time": 1694444174,
    "cache_updated": false,
    "changed": false
}
```

   This command ensures that the package “snapd” is the latest version.

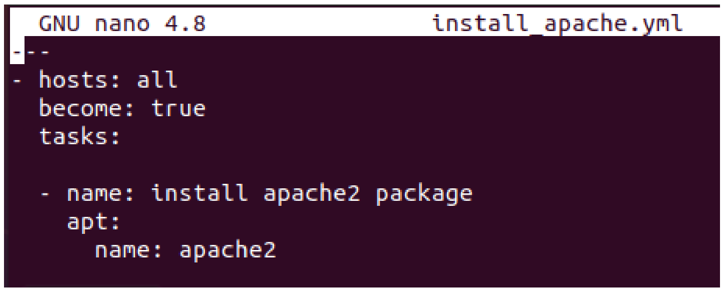
4. At this point, make sure to commit all changes to GitHub.



```
micoflores@workstation:~/CPE31S4_Floresj$ git add *
micoflores@workstation:~/CPE31S4_Floresj$ git commit -m "this is act4"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
micoflores@workstation:~/CPE31S4_Floresj$ git push origin
Username for 'https://github.com': seafrax
Password for 'https://seafrax@github.com':
Everything up-to-date
```

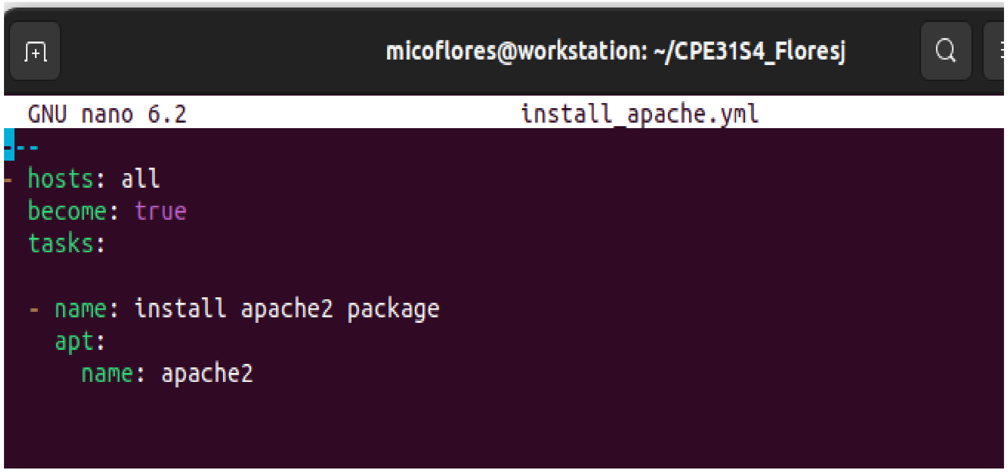
Task 2: Writing our First Playbook

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:



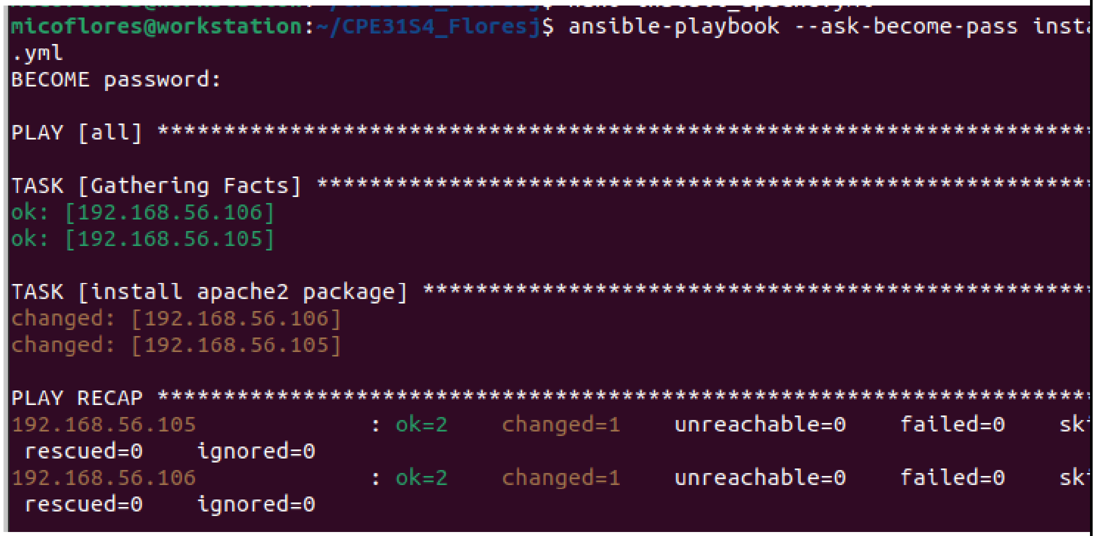
```
  GNU nano 4.8                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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



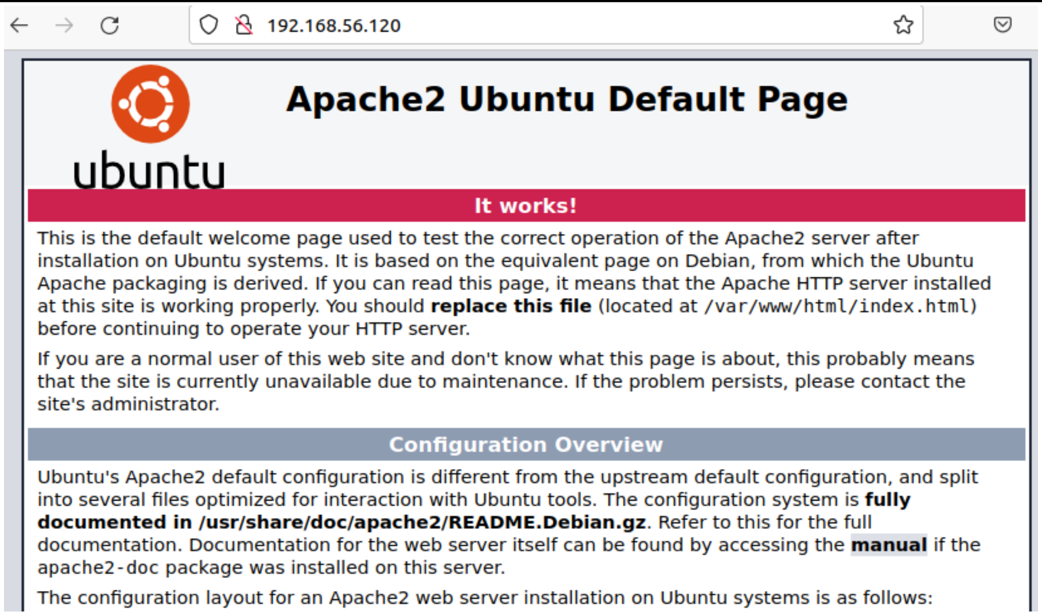
```
GNU nano 6.2                                   install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible-playbook --ask-become-pass insta
.yml
BECOME password:

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

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

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

PLAY RECAP *********************************************************************
192.168.56.105             : ok=2    changed=1    unreachable=0    failed=0    sk
 rescued=0    ignored=0
192.168.56.106             : ok=2    changed=1    unreachable=0    failed=0    sk
 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.


192.168.56.120

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.

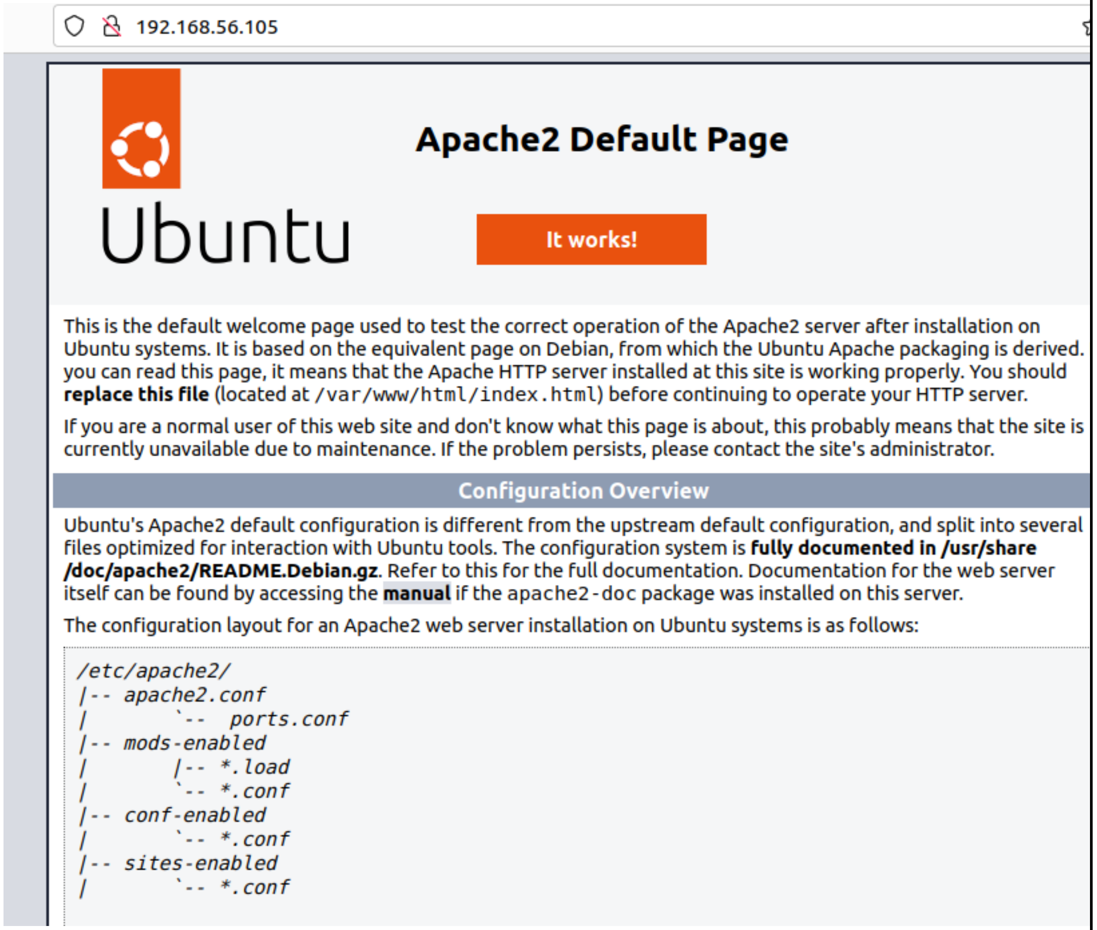
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

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:

Server 1


192.168.56.105

Apache2 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. 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.

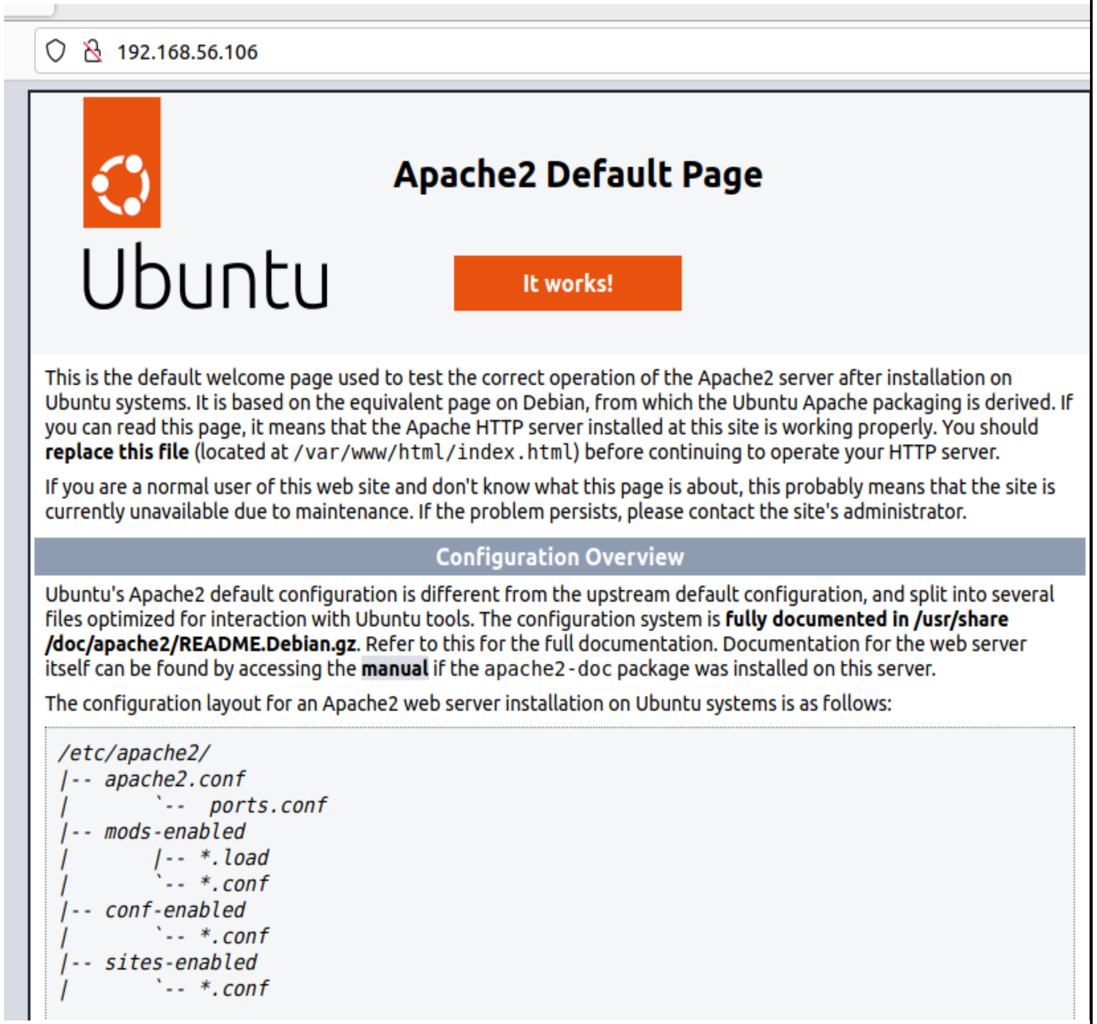
Configuration Overview

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
```

Server 2


192.168.56.106

Apache2 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.

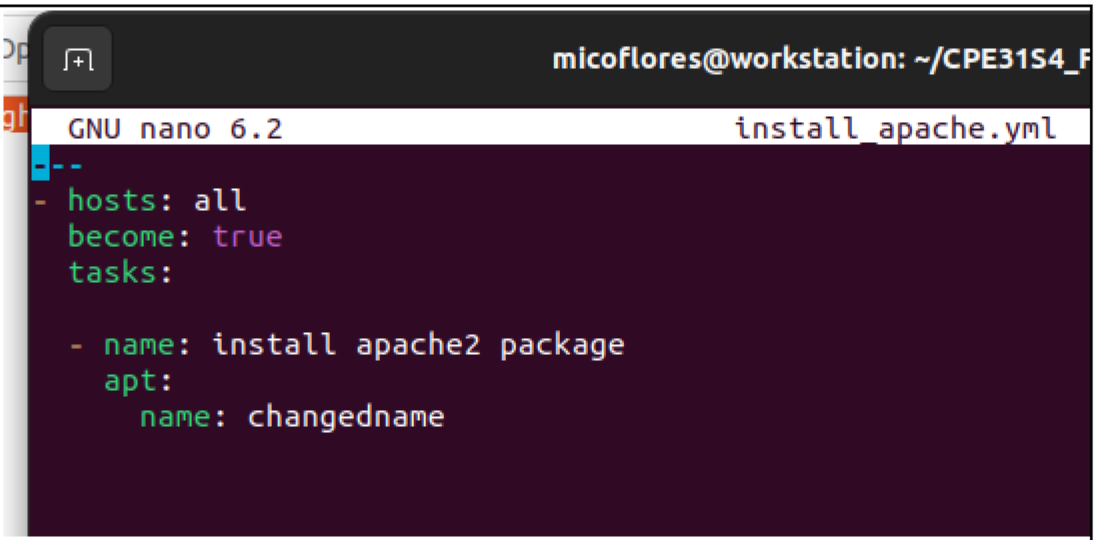
Configuration Overview

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.

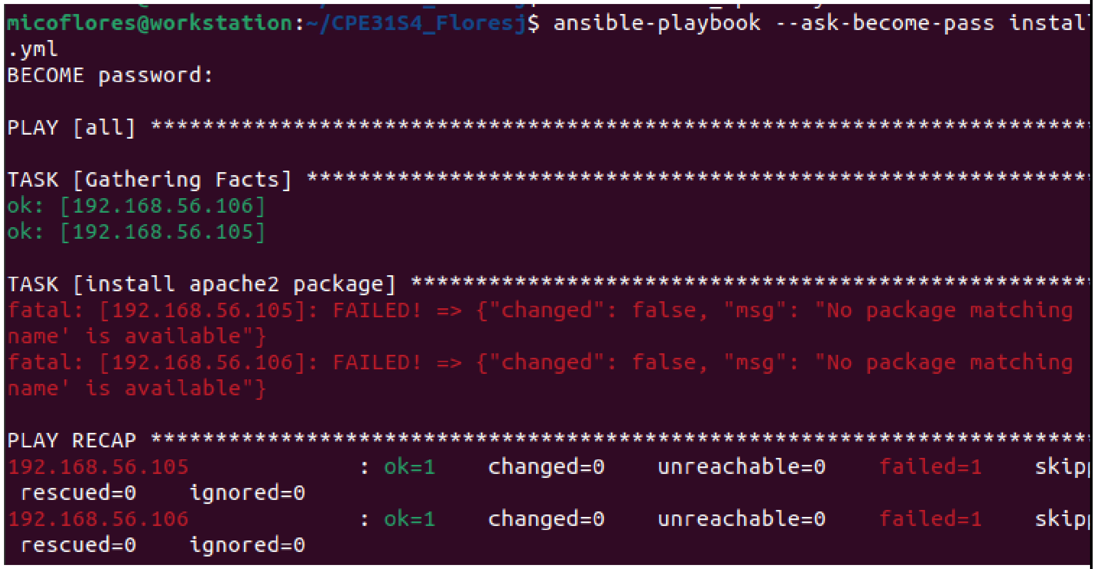
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

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

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?



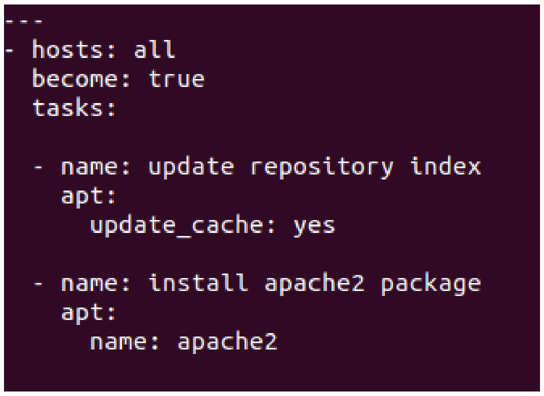
```
micoflores@workstation: ~/CPE31S4_F
GNU nano 6.2                                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

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



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible-playbook --ask-become-pass instal
.yml
BECOME password:

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

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

TASK [install apache2 package] *****************************************************
fatal: [192.168.56.105]: FAILED! => {"changed": false, "msg": "No package matching
name' is available"}
fatal: [192.168.56.106]: FAILED! => {"changed": false, "msg": "No package matching
name' is available"}

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

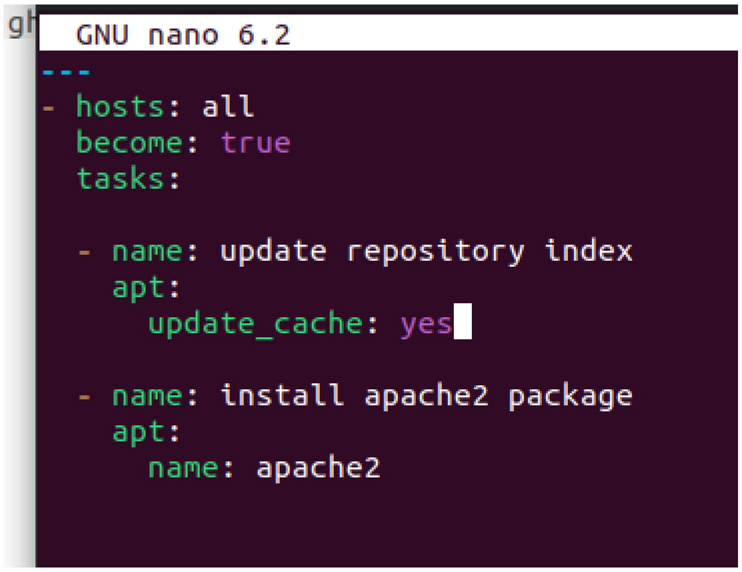
The name should be an appropriate and valid name to execute the playbook properly, otherwise, an error will occur.

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.



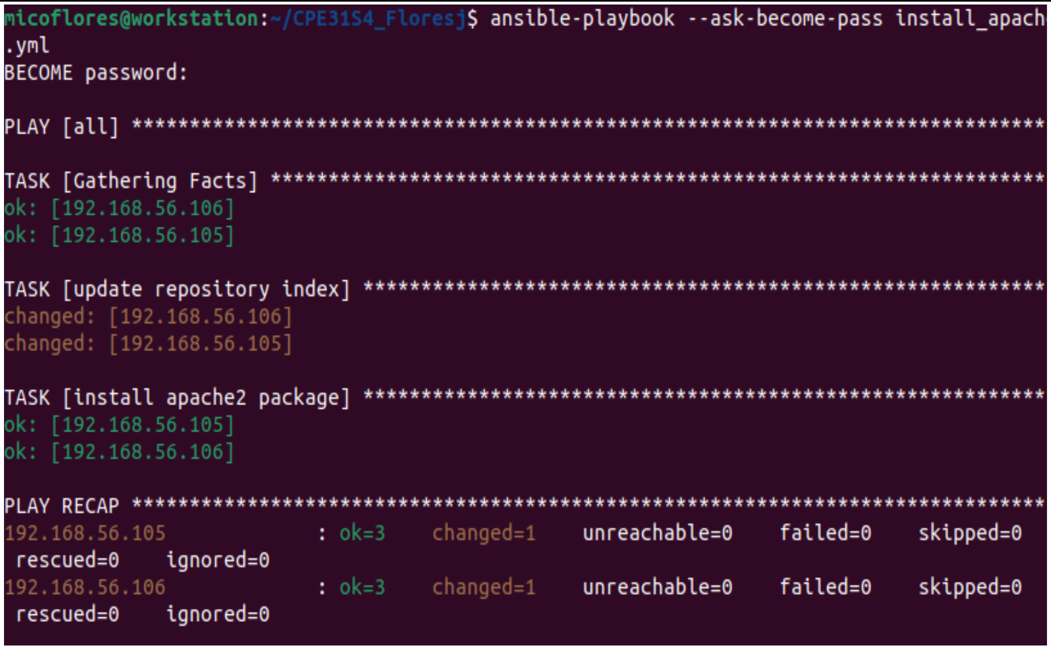
```
---
- hosts: all
  become: true
  tasks:

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

  - name: update repository index
    apt:
      update_cache: yes

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

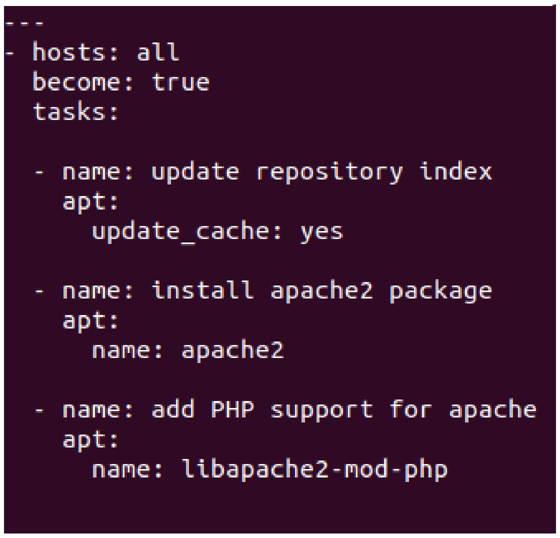
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible-playbook --ask-become-pass install_apach
.yml
BECOME password:

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

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

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

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

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
```

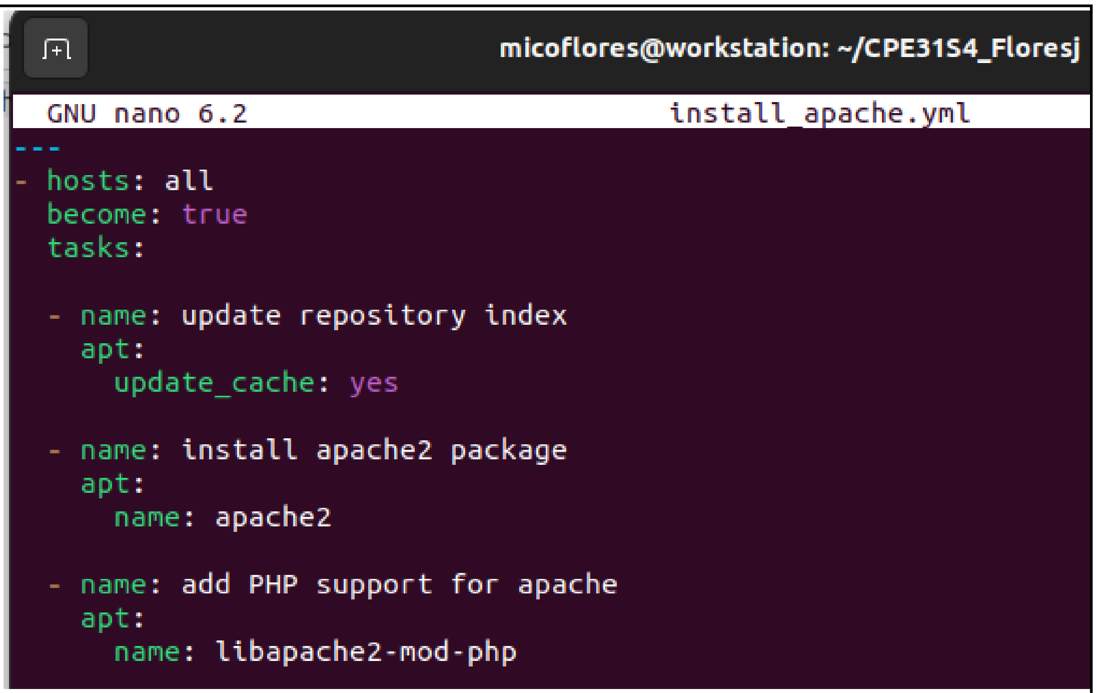
Updating the repository index and installing apache 2 package to both servers are successful.

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
```

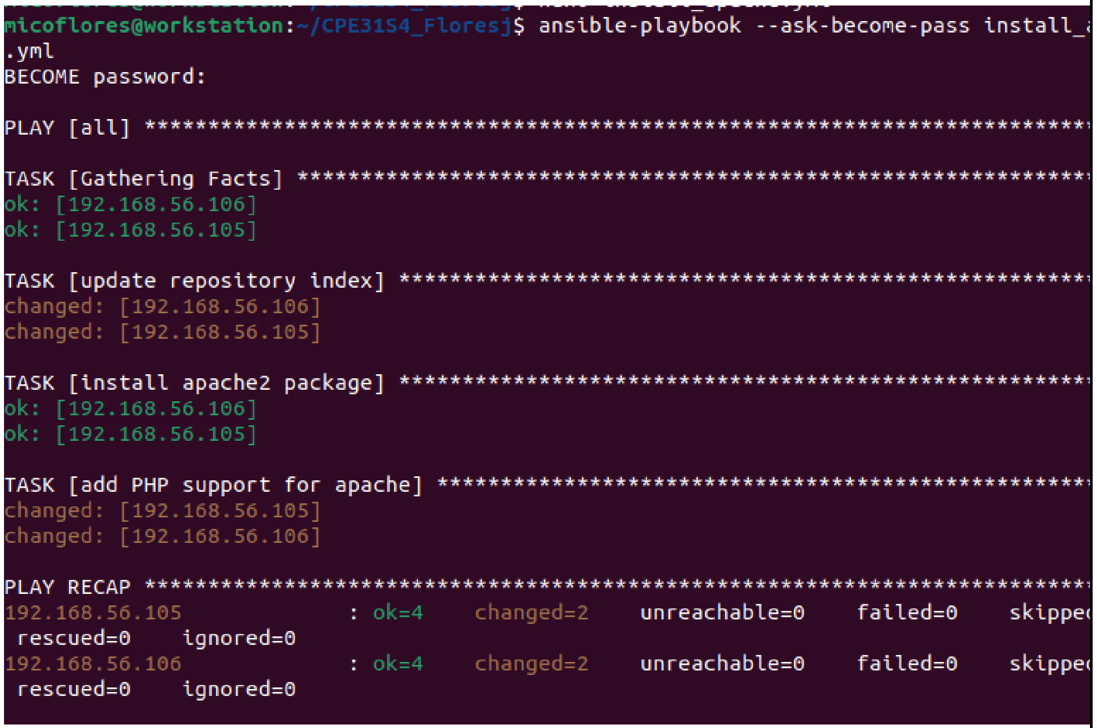
Save the changes to this file and exit.



micoflores@workstation: ~/CPE31S4_Floresj

```
  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
```

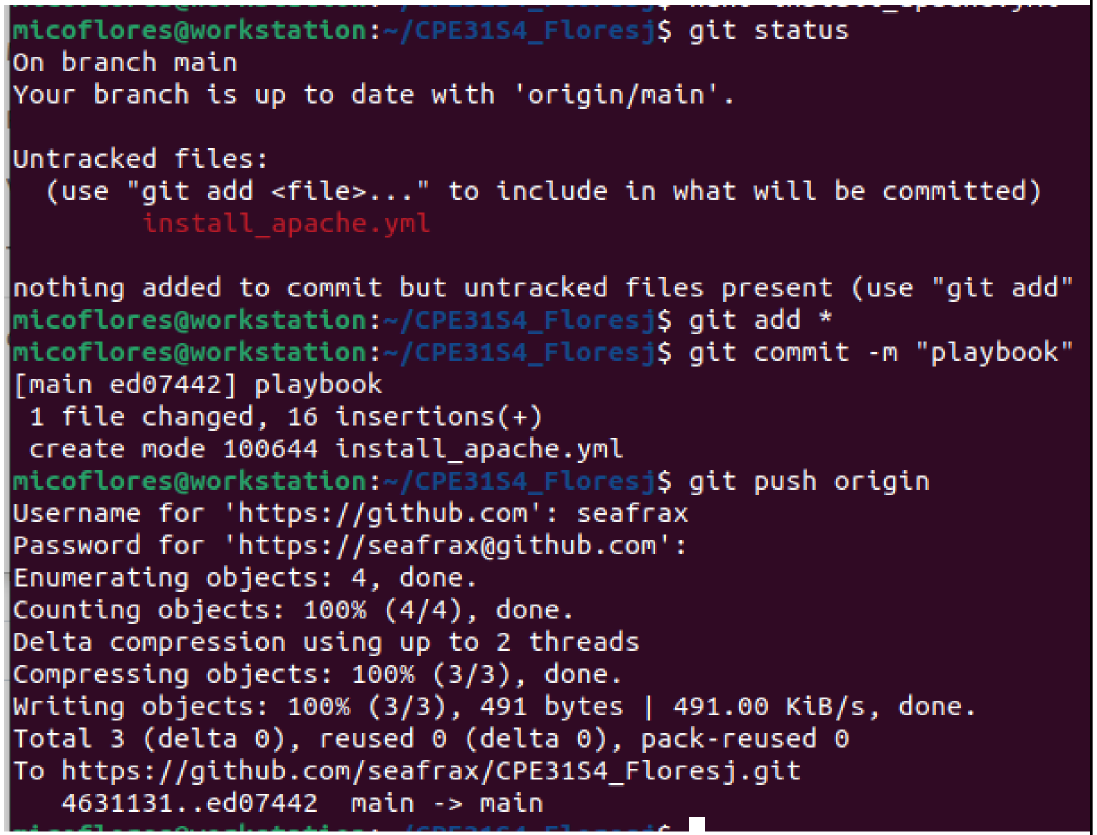
8. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
micoflores@workstation:~/CPE31S4_Floresj$ ansible-playbook --ask-become-pass install_a
.yml
BECOME password:

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

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

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

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

TASK [add PHP support for apache] ************************************************
changed: [192.168.56.105]
changed: [192.168.56.106]

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

The three tasks were able to execute properly and now the 2 servers are now updated because of the playbook.

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



```
micoflores@workstation:~/CPE31S4_Floresj$ 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)
        install_apache.yml

nothing added to commit but untracked files present (use "git add"
micoflores@workstation:~/CPE31S4_Floresj$ git add *
micoflores@workstation:~/CPE31S4_Floresj$ git commit -m "playbook"
[main ed07442] playbook
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
micoflores@workstation:~/CPE31S4_Floresj$ git push origin
Username for 'https://github.com': seafrax
Password for 'https://seafrax@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 491 bytes | 491.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/seafrax/CPE31S4_Floresj.git
   4631131..ed07442  main -> main
```

Link: https://github.com/seafrax/CPE31S4_Floresj

Reflections:
Answer the following:

1. What is the importance of using a playbook?

   - It is to simplify a series of tasks to multiple servers by just running the playbook. By carefully inputting the necessary details like host, name, and tasks and also consider the indentation lines in editing the .yml file, it will be convenient for the user to just automatically run tasks to multiple nodes or servers in the local host (manageNode).

2. Summarize what we have done on this activity.

   - I have implemented elevated ad hoc commands, and created my first playbook. Using the ansible, I was able to automate a series of tasks such as updating, installing apache, and packages. All of these can be executed in the local machine so it doesn't need to be done manually. All of the changes that happened in this activity were pushed to the github repository I created so my repository is updated.

Conclusion:

- In this activity I have learned on how to use the ansible in ubuntu linux together with elevated ad hoc commands, on how to configure the ansible.cfg and the inventory so I can connect to my existing servers. I was able to do multiple tasks that were executed in my local machine to make changes in my servers which was successful. In addition to this I was able to create a playbook which is very convenient for the user because it automates tasks and this results in being time efficient and effective administrator.