Name: Aaron Martin P. Caro	**Date Performed: 07/09/2023**
Course/Section:CPE232-CPE31S5	**Date Submitted: 12/09/2023**
Instructor: Prof. 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:

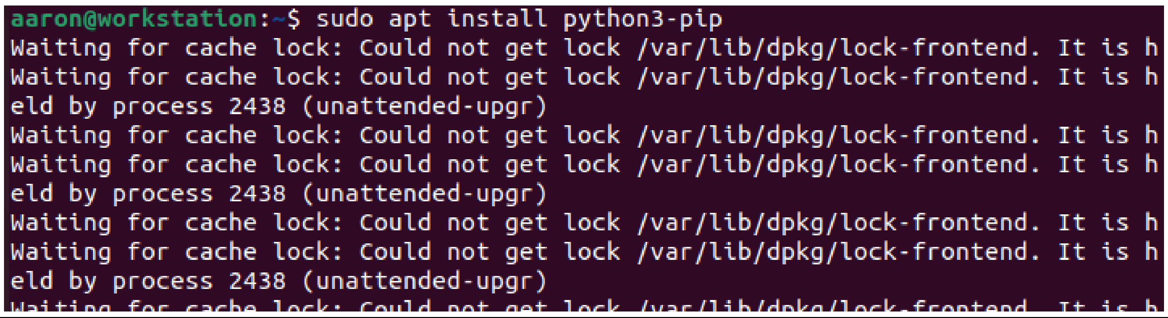
Provide screenshots for each task.

Elevated Ad hoc commands

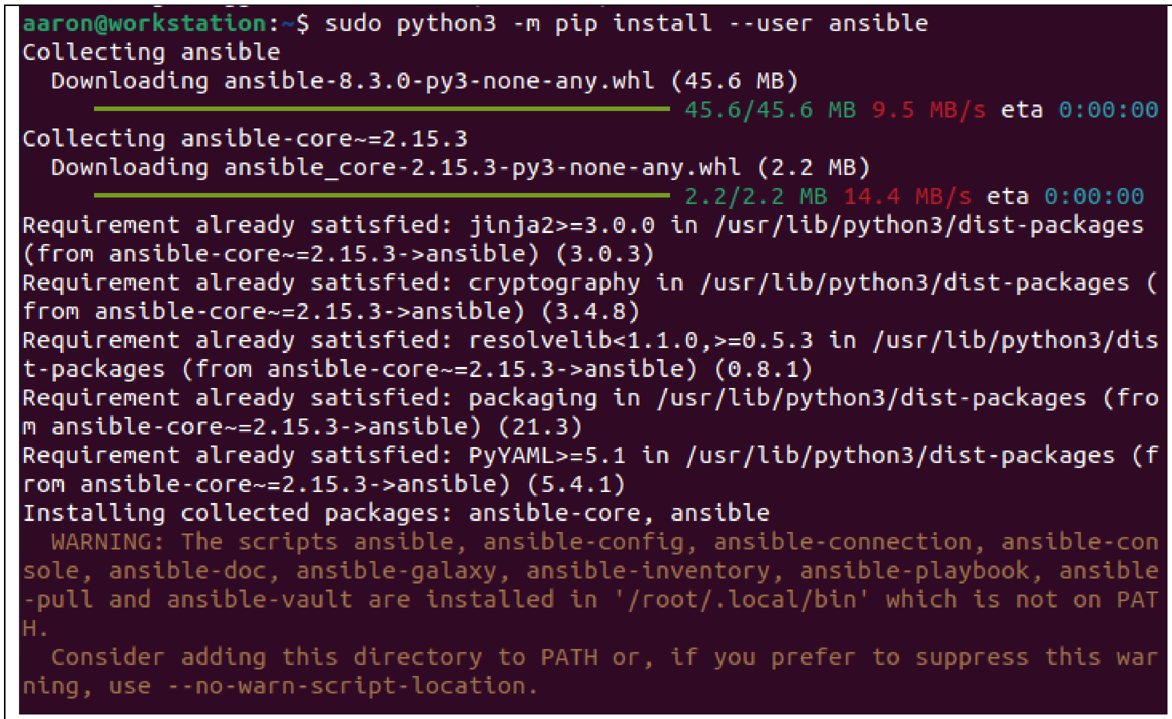
So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages 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

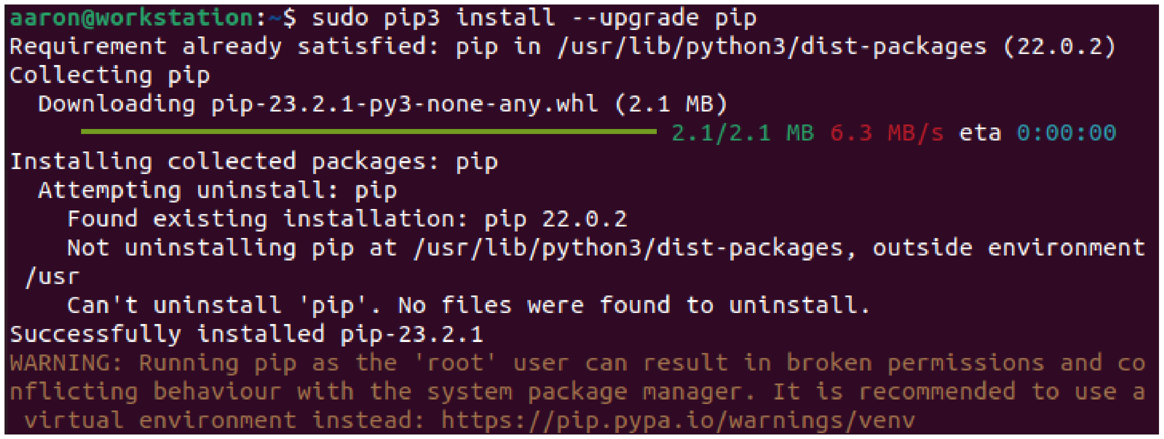
Task 1: Run elevated ad hoc commands



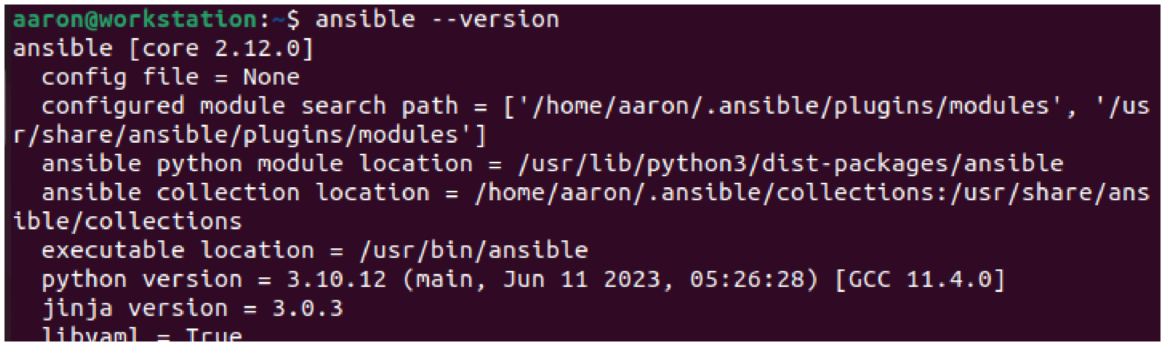
```
aaron@workstation:~$ sudo apt install python3-pip
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
eld by process 2438 (unattended-upgr)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
eld by process 2438 (unattended-upgr)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is h
eld by process 2438 (unattended-upgr)
```



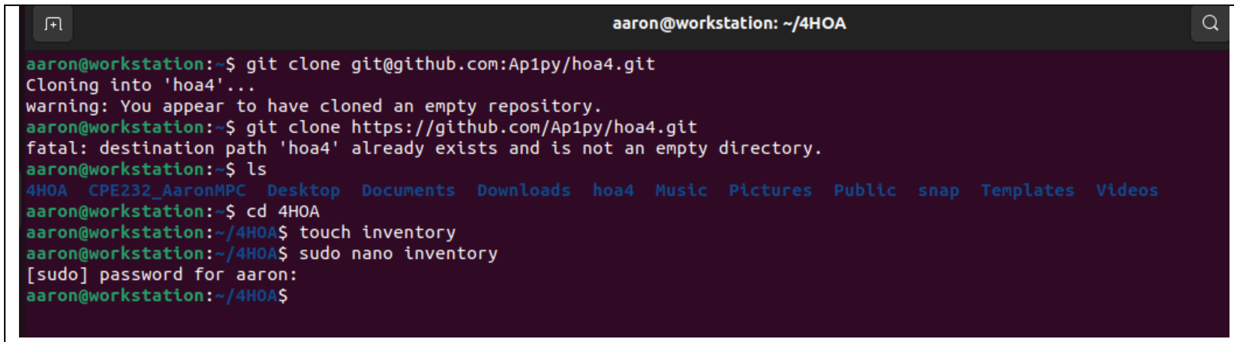
```
aaron@workstation:~$ sudo python3 -m pip install --user ansible
Collecting ansible
  Downloading ansible-8.3.0-py3-none-any.whl (45.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.6/45.6 MB 9.5 MB/s eta 0:00:00
Collecting ansible-core~=2.15.3
  Downloading ansible_core-2.15.3-py3-none-any.whl (2.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 14.4 MB/s eta 0:00:00
Requirement already satisfied: jinja2>=3.0.0 in /usr/lib/python3/dist-packages
(from ansible-core~=2.15.3->ansible) (3.0.3)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (
from ansible-core~=2.15.3->ansible) (3.4.8)
Requirement already satisfied: resolvelib<1.1.0,>=0.5.3 in /usr/lib/python3/dis
t-packages (from ansible-core~=2.15.3->ansible) (0.8.1)
Requirement already satisfied: packaging in /usr/lib/python3/dist-packages (fro
m ansible-core~=2.15.3->ansible) (21.3)
Requirement already satisfied: PyYAML>=5.1 in /usr/lib/python3/dist-packages (f
rom ansible-core~=2.15.3->ansible) (5.4.1)
Installing collected packages: ansible-core, ansible
  WARNING: The scripts ansible, ansible-config, ansible-connection, ansible-con
sole, ansible-doc, ansible-galaxy, ansible-inventory, ansible-playbook, ansible
-pull and ansible-vault are installed in '/root/.local/bin' which is not on PAT
H.
  Consider adding this directory to PATH or, if you prefer to suppress this war
ning, use --no-warn-script-location.
```



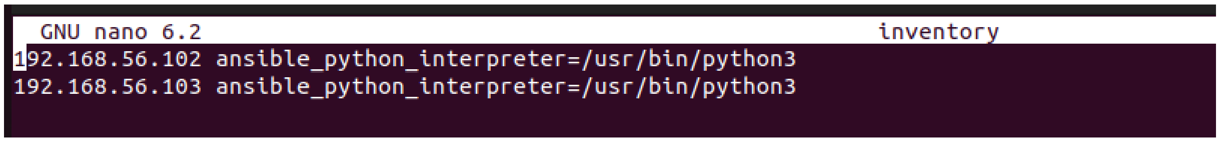
```
aaron@workstation:~$ sudo pip3 install --upgrade pip
Requirement already satisfied: pip in /usr/lib/python3/dist-packages (22.0.2)
Collecting pip
  Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 6.3 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.2
    Not uninstalling pip at /usr/lib/python3/dist-packages, outside environment
 /usr
    Can't uninstall 'pip'. No files were found to uninstall.
Successfully installed pip-23.2.1
WARNING: Running pip as the 'root' user can result in broken permissions and co
nflicting behaviour with the system package manager. It is recommended to use a
 virtual environment instead: https://pip.pypa.io/warnings/venv
```



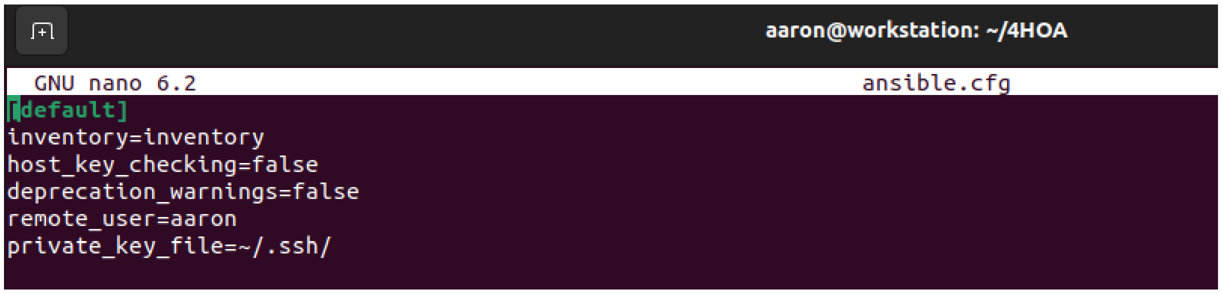
```
aaron@workstation:~$ ansible --version
ansible [core 2.12.0]
  config file = None
  configured module search path = ['/home/aaron/.ansible/plugins/modules', '/us
r/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/aaron/.ansible/collections:/usr/share/ans
ible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]
  jinja version = 3.0.3
  libyaml = True
```



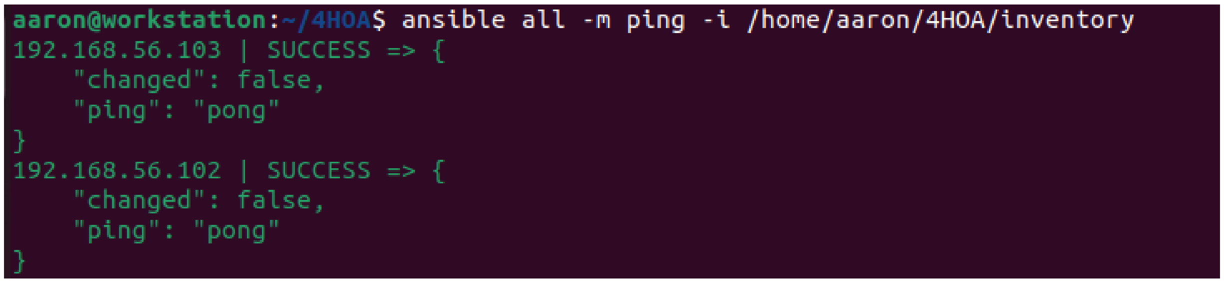
```
aaron@workstation:~$ git clone git@github.com:Ap1py/hoa4.git
Cloning into 'hoa4'...
warning: You appear to have cloned an empty repository.
aaron@workstation:~$ git clone https://github.com/Ap1py/hoa4.git
fatal: destination path 'hoa4' already exists and is not an empty directory.
aaron@workstation:~$ ls
4HOA  CPE232_AaronMPC  Desktop  Documents  Downloads  hoa4  Music  Pictures  Public  snap  Templates  Videos
aaron@workstation:~$ cd 4HOA
aaron@workstation:~/4HOA$ touch inventory
aaron@workstation:~/4HOA$ sudo nano inventory
[sudo] password for aaron:
aaron@workstation:~/4HOA$
```



```
  GNU nano 6.2                                   inventory
192.168.56.102 ansible_python_interpreter=/usr/bin/python3
192.168.56.103 ansible_python_interpreter=/usr/bin/python3
```

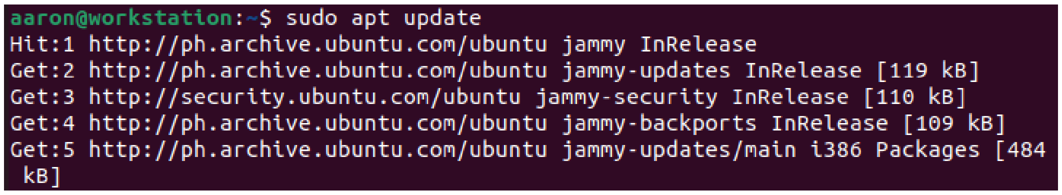


```
  GNU nano 6.2                                   ansible.cfg
[default]
inventory=inventory
host_key_checking=false
deprecation_warnings=false
remote_user=aaron
private_key_file=~/.ssh/
```



```
aaron@workstation:~/4HOA$ ansible all -m ping -i /home/aaron/4HOA/inventory
192.168.56.103 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.56.102 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

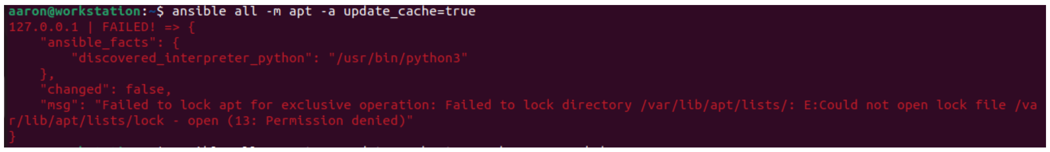
1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package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



```
aaron@workstation:~$ sudo apt update
Hit:1 http://ph.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [484
 kB]
```

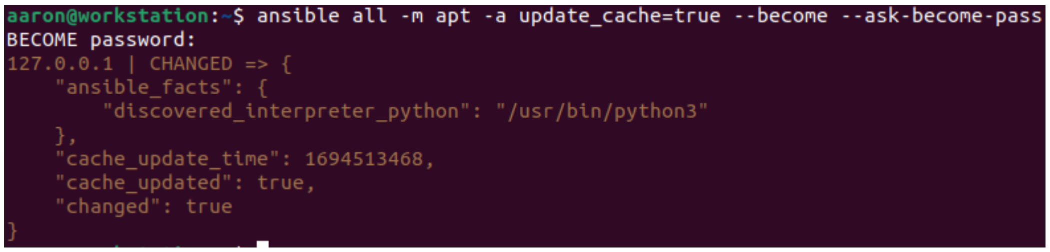
ansible all -m apt -a update_cache=true

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



```
aaron@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/lists/: E:Could not open lock file /va
r/lib/apt/lists/lock - open (13: Permission denied)"
}
```

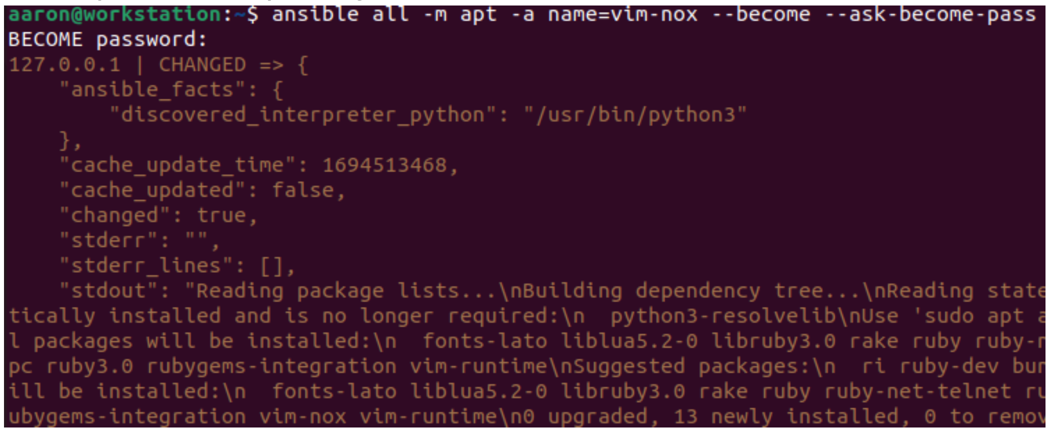
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



```
aaron@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": 1694513468,
    "cache_updated": true,
    "changed": true
}
```

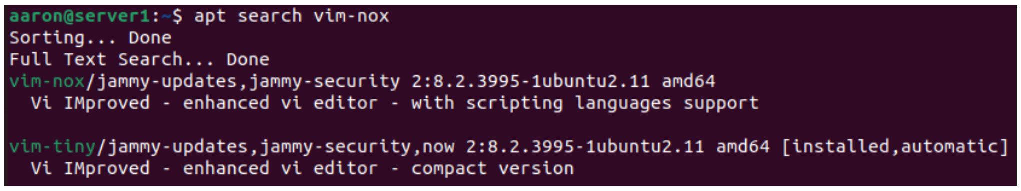
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

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.

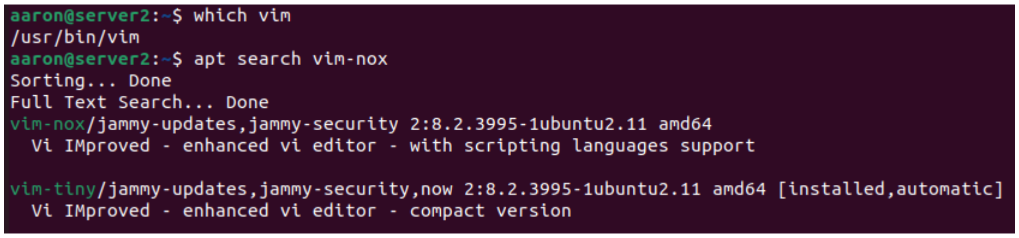


```
aaron@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694513468,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state
tically installed and is no longer required:\n  python3-resolvelib\nUse 'sudo apt a
l packages will be installed:\n  fonts-lato liblua5.2-0 libruby3.0 rake ruby ruby-r
pc ruby3.0 rubygems-integration vim-runtime\nSuggested packages:\n  ri ruby-dev bur
ill be installed:\n  fonts-lato liblua5.2-0 libruby3.0 rake ruby ruby-net-telnet ru
ubygems-integration vim-nox vim-runtime\n0 upgraded, 13 newly installed, 0 to remov
```

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?

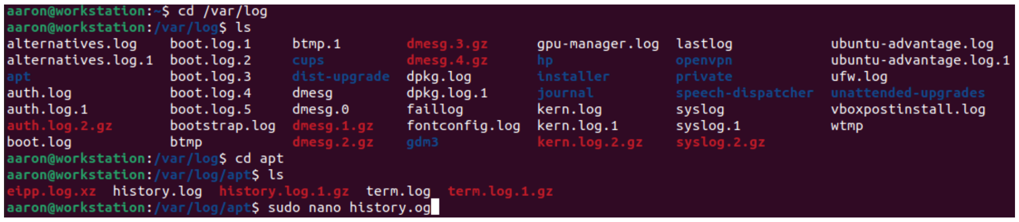


```
aaron@server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

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

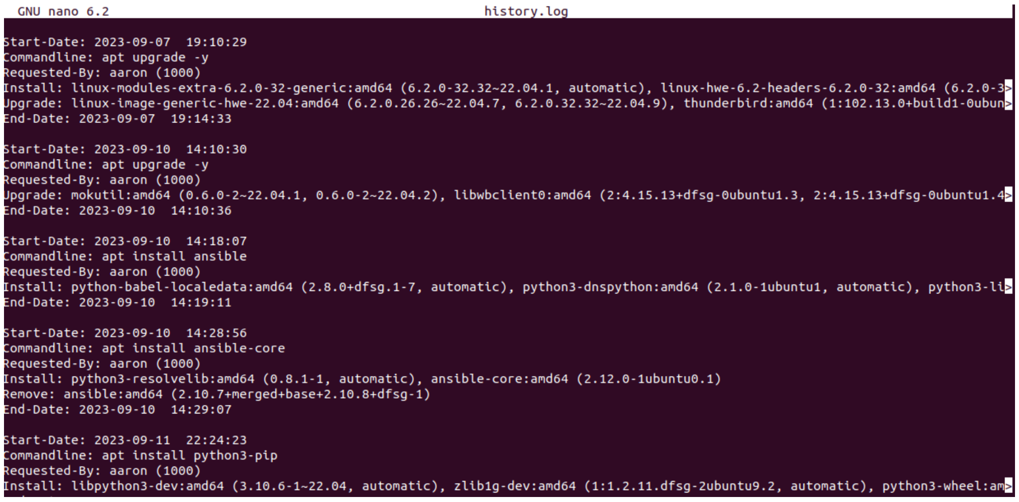


```
aaron@server2:~$ which vim
/usr/bin/vim
aaron@server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

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.



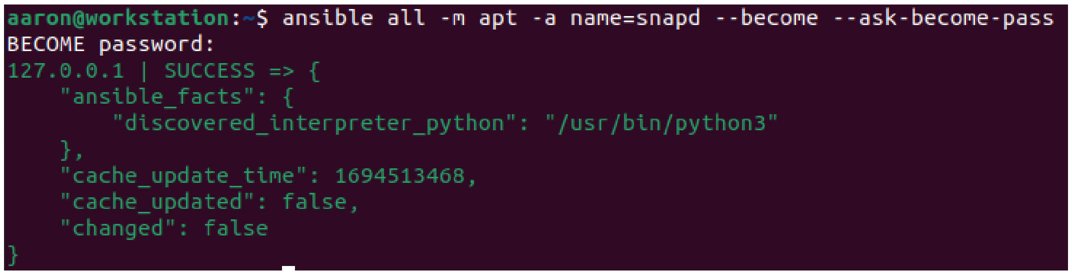
```
aaron@workstation:~$ cd /var/log
aaron@workstation:/var/log$ ls
alternatives.log    boot.log.1      btmp.1        dmesg.3.gz      gpu-manager.log  lastlog            ubuntu-advantage.log
alternatives.log.1  boot.log.2      cups          dmesg.4.gz      hp               openvpn            ubuntu-advantage.log.1
apt                 boot.log.3      dist-upgrade  dpkg.log        installer        private            ufw.log
auth.log            boot.log.4      dmesg         dpkg.log.1      journal          speech-dispatcher  unattended-upgrades
auth.log.1          boot.log.5      dmesg.0       faillog         kern.log         syslog             vboxpostinstall.log
auth.log.2.gz       bootstrap.log   dmesg.1.gz    fontconfig.log  kern.log.1       syslog.1           wtmp
boot.log            btmp            dmesg.2.gz    gdm3            kern.log.2.gz    syslog.2.gz
aaron@workstation:/var/log$ cd apt
aaron@workstation:/var/log/apt$ ls
eipp.log.xz  history.log  history.log.1.gz  term.log  term.log.1.gz
aaron@workstation:/var/log/apt$ sudo nano history.og
```



```
  GNU nano 6.2                                    history.log

Start-Date: 2023-09-07  19:10:29
Commandline: apt upgrade -y
Requested-By: aaron (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 (6.2.0-3>
Upgrade: linux-image-generic-hwe-22.04:amd64 (6.2.0.26.26~22.04.7, 6.2.0.32.32~22.04.9), thunderbird:amd64 (1:102.13.0+build1-0ubun>
End-Date: 2023-09-07  19:14:33

Start-Date: 2023-09-10  14:10:30
Commandline: apt upgrade -y
Requested-By: aaron (1000)
Upgrade: mokutil:amd64 (0.6.0-2~22.04.1, 0.6.0-2~22.04.2), libwbclient0:amd64 (2:4.15.13+dfsg-0ubuntu1.3, 2:4.15.13+dfsg-0ubuntu1.4>
End-Date: 2023-09-10  14:10:36

Start-Date: 2023-09-10  14:18:07
Commandline: apt install ansible
Requested-By: aaron (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64 (2.1.0-1ubuntu1, automatic), python3-li>
End-Date: 2023-09-10  14:19:11

Start-Date: 2023-09-10  14:28:56
Commandline: apt install ansible-core
Requested-By: aaron (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: 2023-09-10  14:29:07

Start-Date: 2023-09-11  22:24:23
Commandline: apt install python3-pip
Requested-By: aaron (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:am>
```

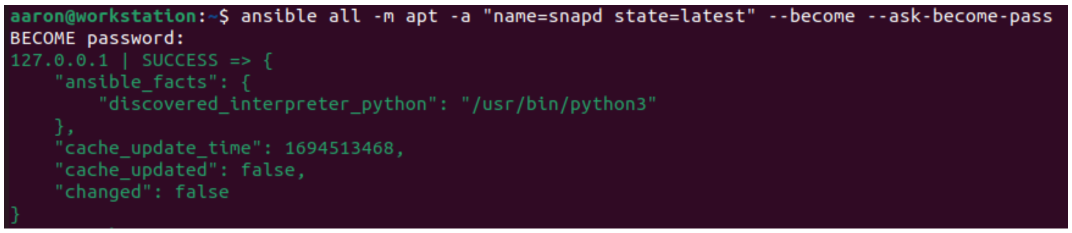
3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

   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?



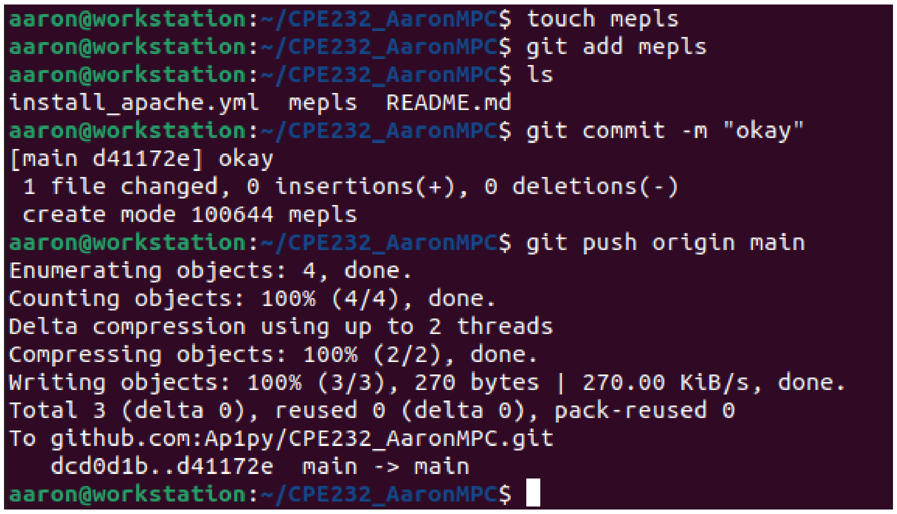
```
aaron@workstation:~$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694513468,
    "cache_updated": false,
    "changed": false
}
```

   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.

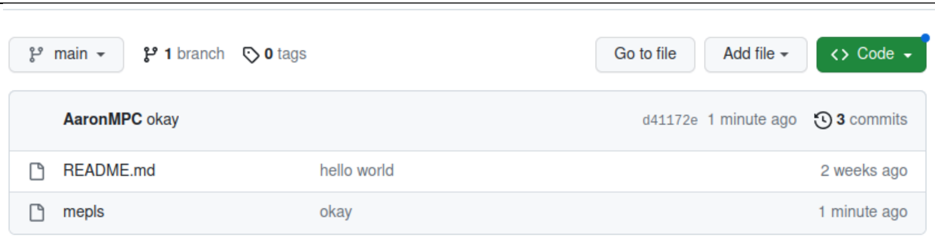


```
aaron@workstation:~$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694513468,
    "cache_updated": false,
    "changed": false
}
```

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



```
aaron@workstation:~/CPE232_AaronMPC$ touch mepls
aaron@workstation:~/CPE232_AaronMPC$ git add mepls
aaron@workstation:~/CPE232_AaronMPC$ ls
install_apache.yml  mepls  README.md
aaron@workstation:~/CPE232_AaronMPC$ git commit -m "okay"
[main d41172e] okay
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 mepls
aaron@workstation:~/CPE232_AaronMPC$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes | 270.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ap1py/CPE232_AaronMPC.git
   dcd0d1b..d41172e  main -> main
aaron@workstation:~/CPE232_AaronMPC$
```

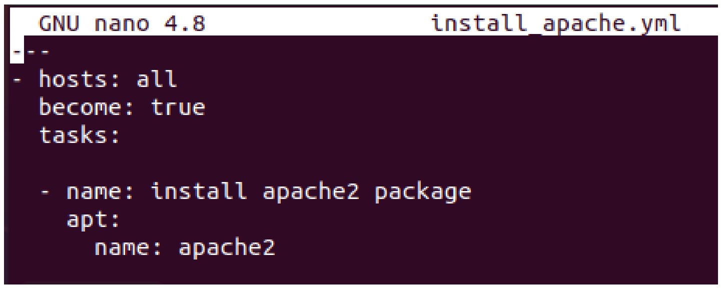


```
AaronMPC okay                         d41172e  1 minute ago   3 commits
README.md        hello world                                  2 weeks ago
mepls            okay                                         1 minute ago
```

Task 2: Writing our First Playbook

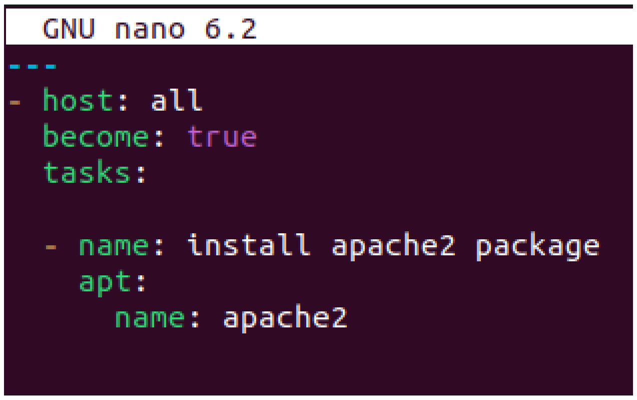
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and 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:



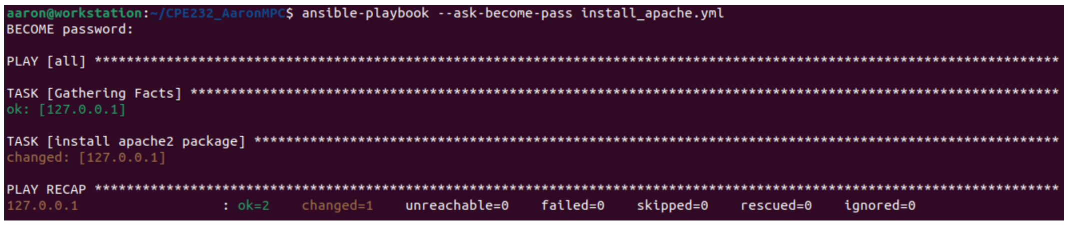
```
  GNU nano 4.8                        install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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



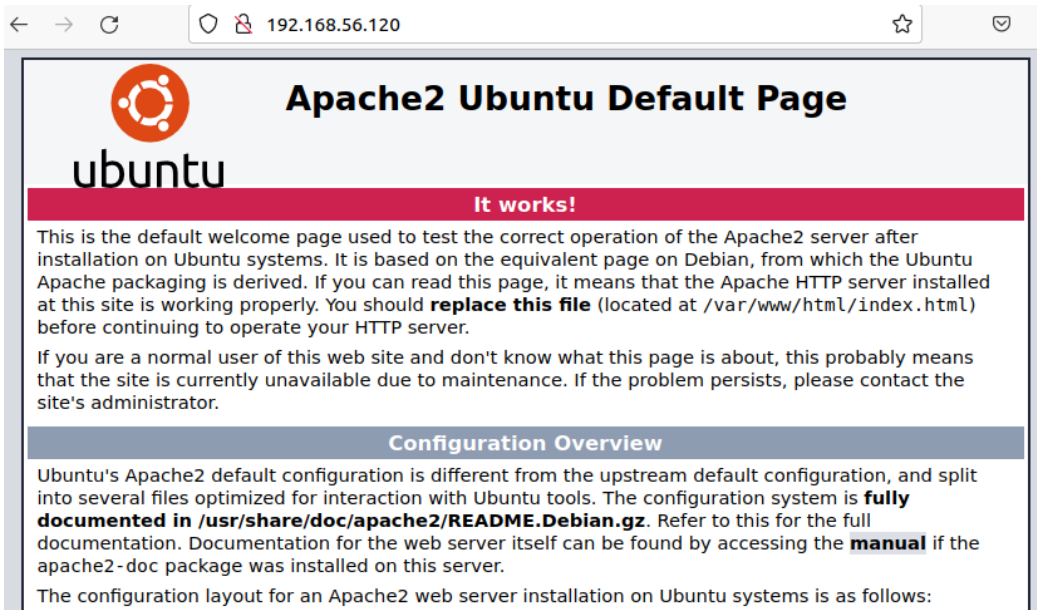
```
  GNU nano 6.2
---
- host: all
  become: true
  tasks:

  - 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.



```
aaron@workstation:~/CPE232_AaronMPC$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

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

ubuntu

It works!

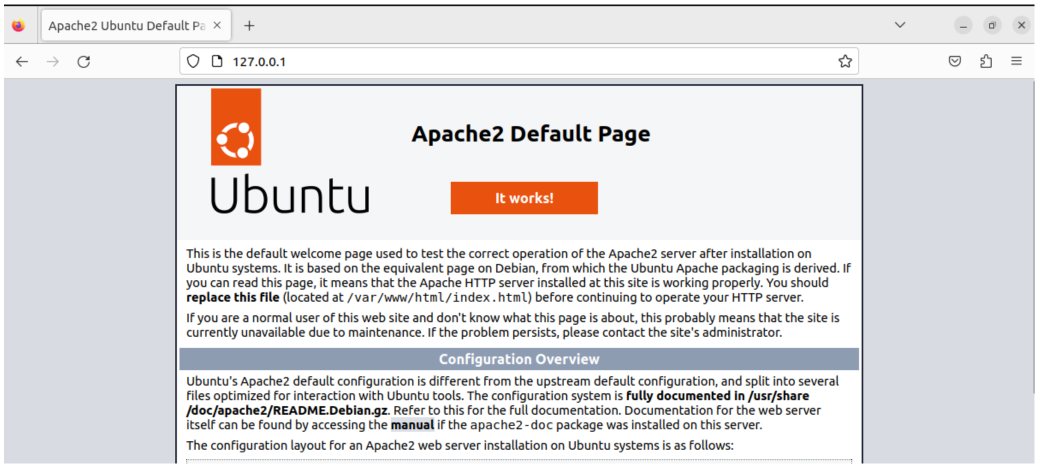
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at 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:



Apache2 Ubuntu Default Pa ×

127.0.0.1

Apache2 Default Page

Ubuntu

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.

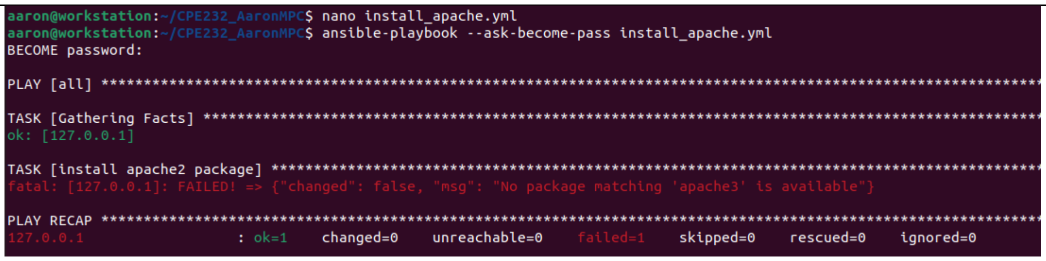
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

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.

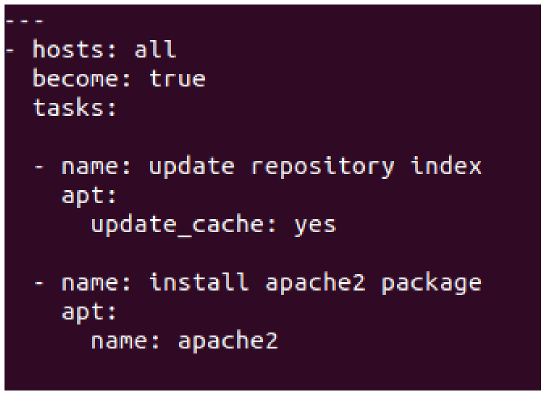
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

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?



```
aaron@workstation:~/CPE232_AaronMPC$ nano install_apache.yml
aaron@workstation:~/CPE232_AaronMPC$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [install apache2 package] ******************************************************************************************
fatal: [127.0.0.1]: FAILED! => {"changed": false, "msg": "No package matching 'apache3' is available"}

PLAY RECAP **************************************************************************************************************
127.0.0.1                  : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=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.

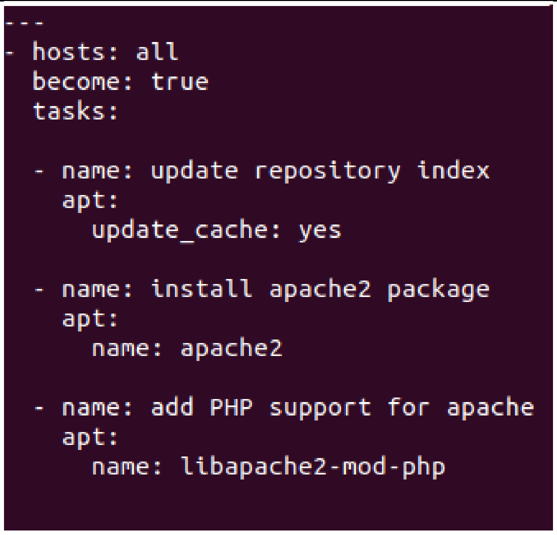


```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

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

Save the changes to this file and exit.

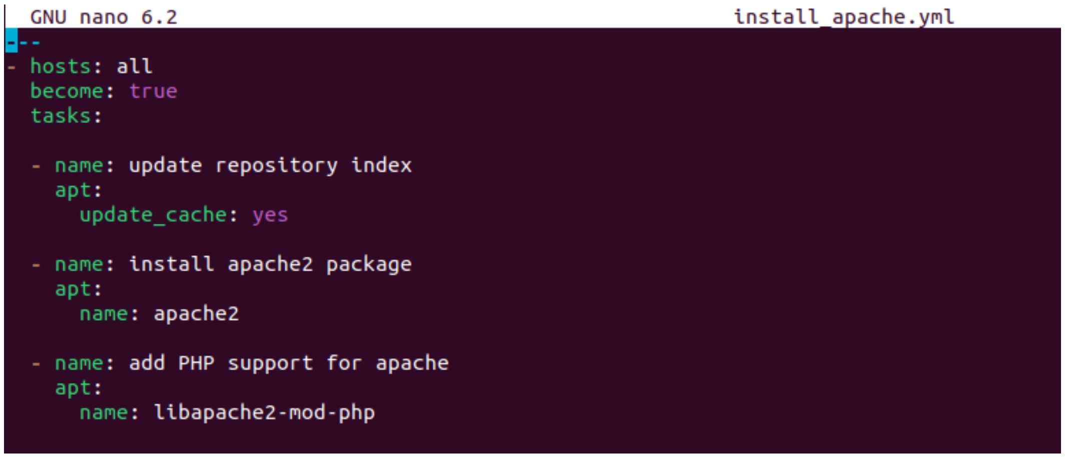
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?

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



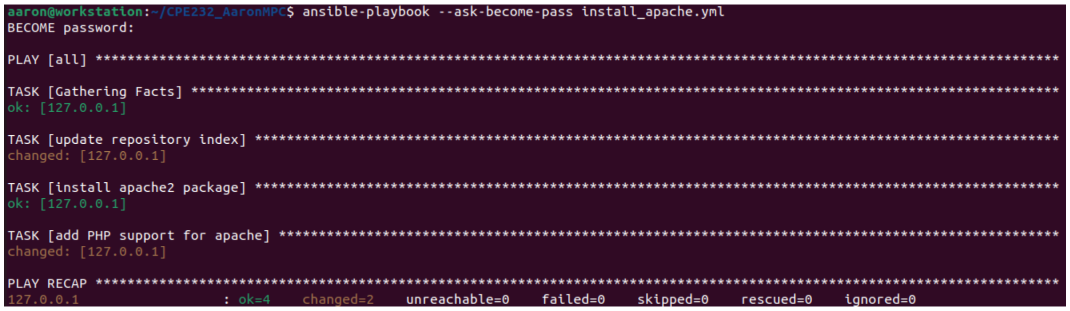
```
---
- hosts: all
  become: true
  tasks:

  - 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.



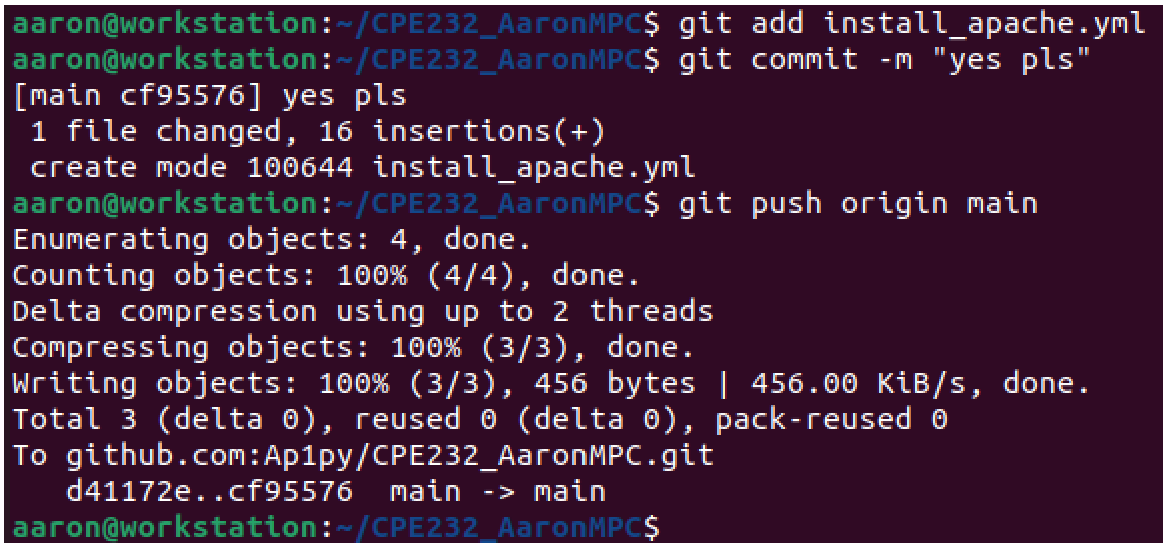
```
 GNU nano 6.2                                              install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - 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?

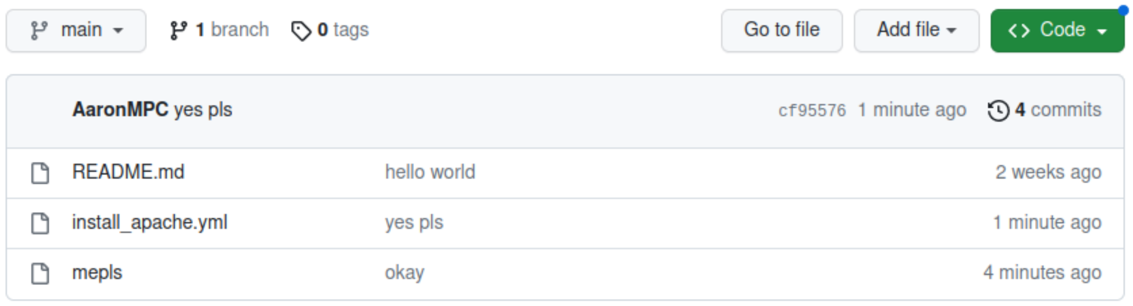


```
aaron@workstation:~/CPE232_AaronMPC$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

TASK [install 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    rescued=0    ignored=0
```

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



```
aaron@workstation:~/CPE232_AaronMPC$ git add install_apache.yml
aaron@workstation:~/CPE232_AaronMPC$ git commit -m "yes pls"
[main cf95576] yes pls
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
aaron@workstation:~/CPE232_AaronMPC$ git push origin main
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), 456 bytes | 456.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ap1py/CPE232_AaronMPC.git
   d41172e..cf95576  main -> main
aaron@workstation:~/CPE232_AaronMPC$
```



AaronMPC yes pls		cf95576 1 minute ago **4** commits
README.md	hello world	2 weeks ago
install_apache.yml	yes pls	1 minute ago
mepls	okay	4 minutes ago

Reflections:

Answer the following:

1. **What is the importance of using a playbook?**
   Because it specifies a set of tasks and configurations that must be executed consistently across multiple systems, a playbook is essential in Ubuntu or any other environment. This computerization guarantees normalization, diminishes human mistakes, and paces up framework organization assignments, making it basic for overseeing the foundation proficiently and keeping up with framework unwavering quality. Playbooks are a fundamental part of infrastructure-as-code. They make it possible to deploy, configure, and maintain Ubuntu servers and services in a way that is both predictable and scalable.

2. **Summarize what we have done on this activity.**
   On this activity a playbook on automating ansible commands ansible's playbooks are essential for automating tasks and commands on remote computers. They promote consistency and efficiency in configuration management and administration tasks by enabling system administrators to define a series of actions to be carried out on multiple remote systems. Playbooks influence Ansible's revelatory linguistic structure, empowering clients to indicate wanted states instead of manual bit-by-bit directions, lessening the gamble of human blunder. Because they are written in YAML, these playbooks are simple to read, write, and keep track of. At last, playbooks act as a foundation for the computerization of Ansible orders, smoothing out complex tasks and guaranteeing consistency across disseminated frameworks.