Name: Sembrero, Christian Dane D.	**Date Performed: 08/29/25**
Course/Section: CPE 212-CPE31S2	**Date Submitted: 08/29/25**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 1st sem 2025-2026**

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

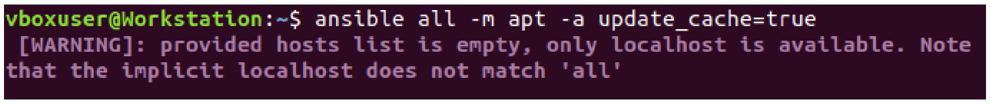
Task 1: Run elevated ad hoc commands

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

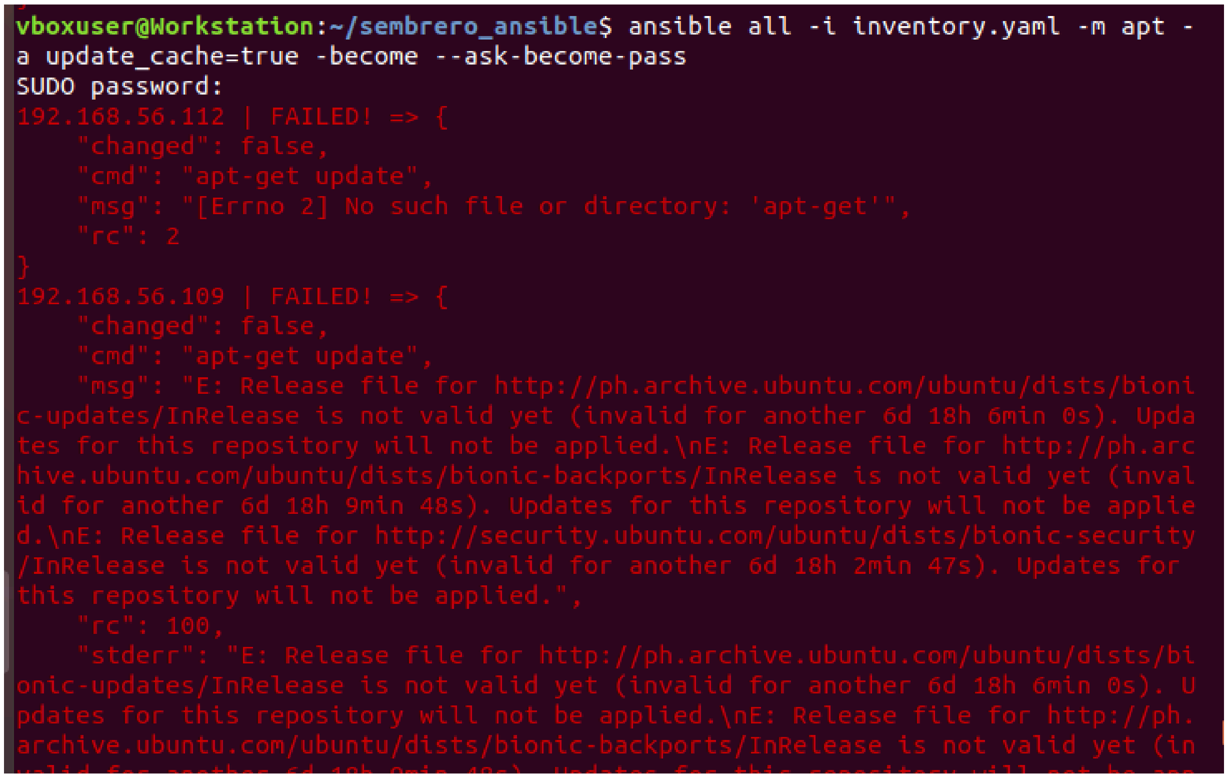
What is the result of the command? Is it successful?



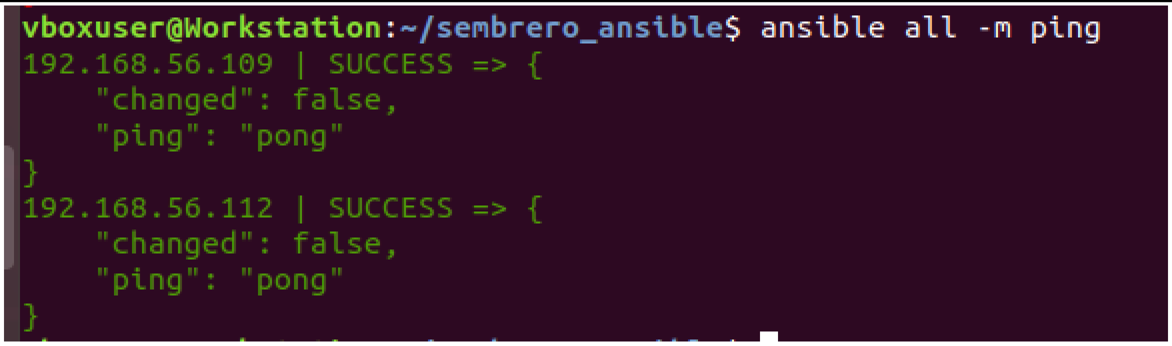
```
vboxuser@Workstation:~$ ansible all -m apt -a update_cache=true
 [WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
```

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. \

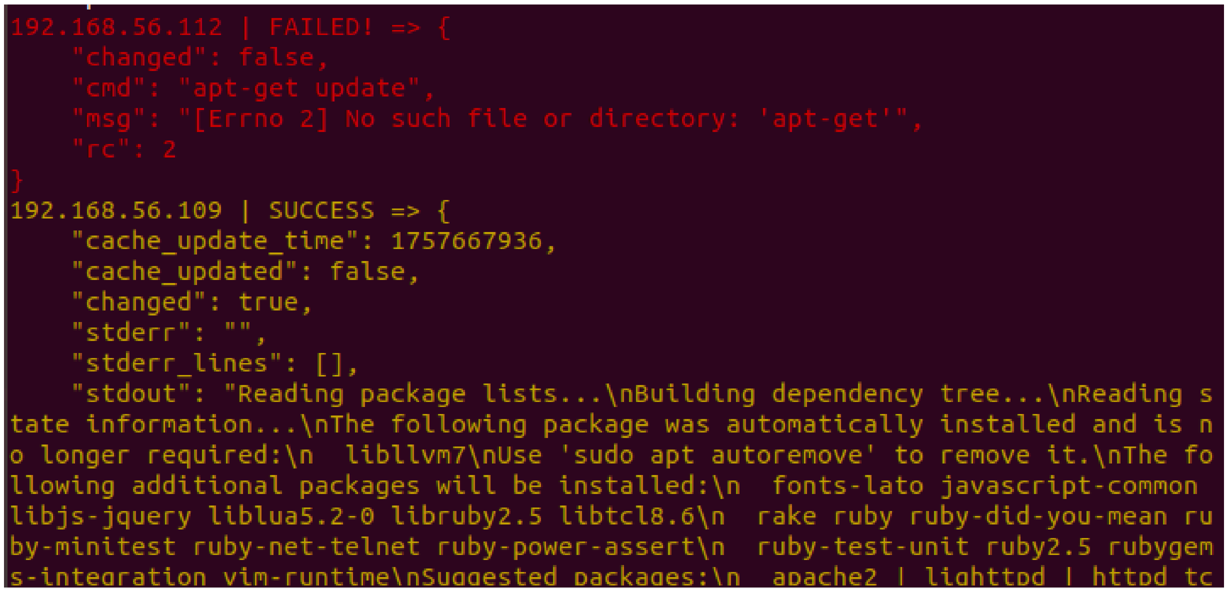


```
vboxuser@Workstation:~/sembrero_ansible$ ansible all -i inventory.yaml -m apt -
a update_cache=true -become --ask-become-pass
SUDO password:
192.168.56.112 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "[Errno 2] No such file or directory: 'apt-get'",
    "rc": 2
}
192.168.56.109 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "E: Release file for http://ph.archive.ubuntu.com/ubuntu/dists/bioni
c-updates/InRelease is not valid yet (invalid for another 6d 18h 6min 0s). Upda
tes for this repository will not be applied.\nE: Release file for http://ph.arc
hive.ubuntu.com/ubuntu/dists/bionic-backports/InRelease is not valid yet (inval
id for another 6d 18h 9min 48s). Updates for this repository will not be applie
d.\nE: Release file for http://security.ubuntu.com/ubuntu/dists/bionic-security
/InRelease is not valid yet (invalid for another 6d 18h 2min 47s). Updates for
this repository will not be applied.",
    "rc": 100,
    "stderr": "E: Release file for http://ph.archive.ubuntu.com/ubuntu/dists/bi
onic-updates/InRelease is not valid yet (invalid for another 6d 18h 6min 0s). U
pdates for this repository will not be applied.\nE: Release file for http://ph.
archive.ubuntu.com/ubuntu/dists/bionic-backports/InRelease is not valid yet (in
```



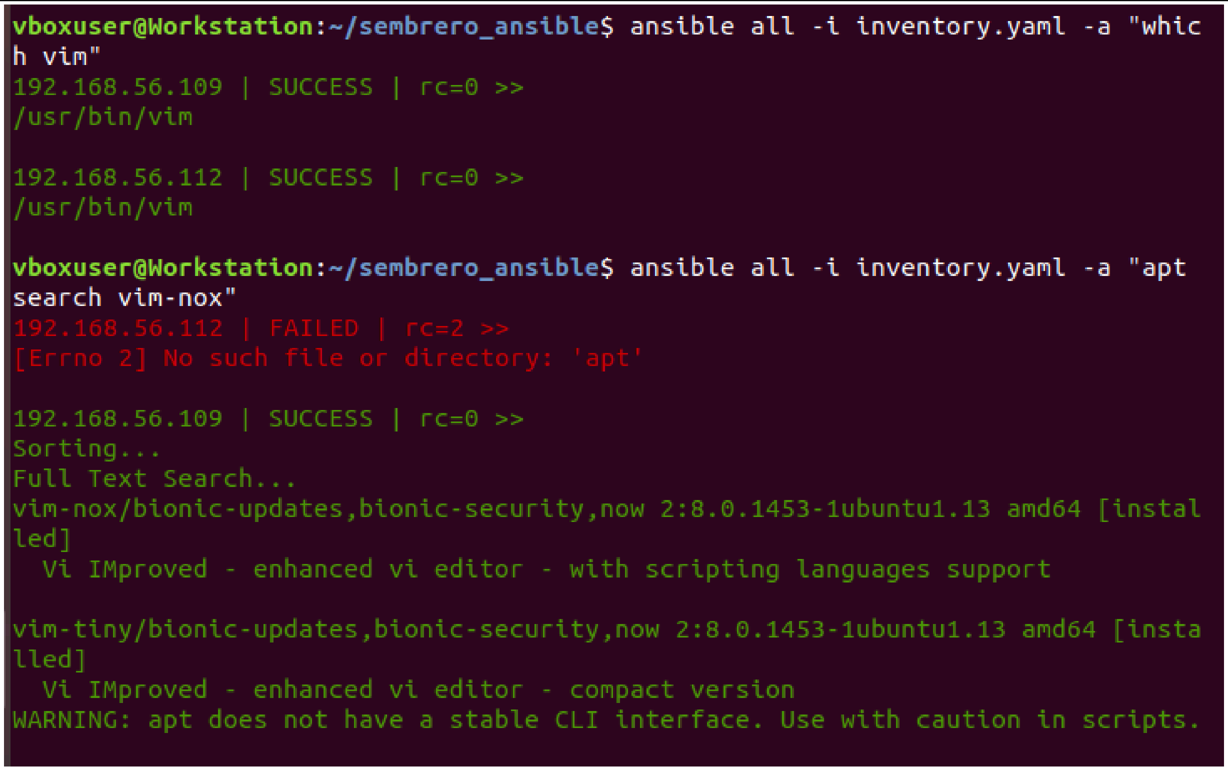
```
vboxuser@Workstation:~/sembrero_ansible$ ansible all -m ping
192.168.56.109 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.56.112 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

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.



```
192.168.56.112 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "[Errno 2] No such file or directory: 'apt-get'",
    "rc": 2
}
192.168.56.109 | SUCCESS => {
    "cache_update_time": 1757667936,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe fo
llowing additional packages will be installed:\n  fonts-lato javascript-common
libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ru
by-minitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygem
s-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd tc
```

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?

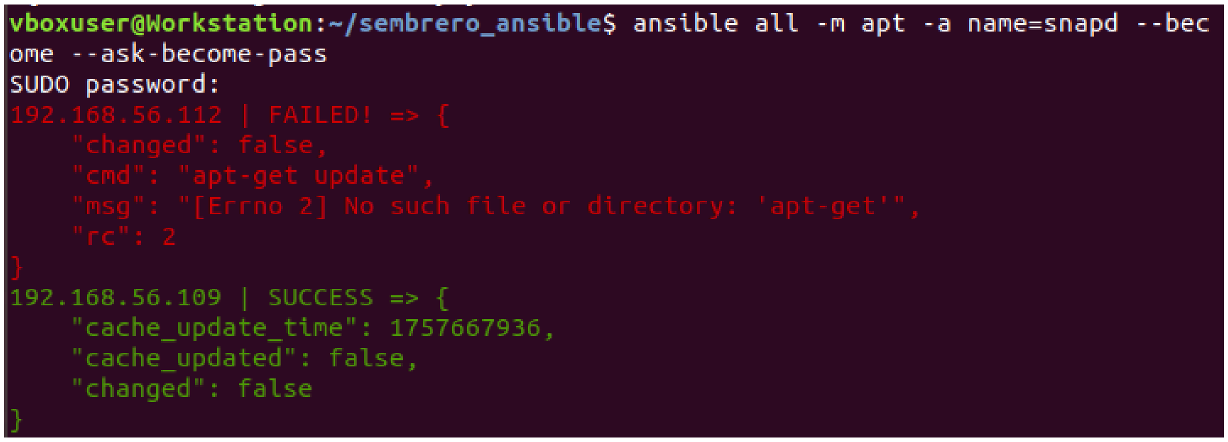


```
vboxuser@Workstation:~/sembrero_ansible$ ansible all -i inventory.yaml -a "whic
h vim"
192.168.56.109 | SUCCESS | rc=0 >>
/usr/bin/vim

192.168.56.112 | SUCCESS | rc=0 >>
/usr/bin/vim

vboxuser@Workstation:~/sembrero_ansible$ ansible all -i inventory.yaml -a "apt
search vim-nox"
192.168.56.112 | FAILED | rc=2 >>
[Errno 2] No such file or directory: 'apt'

192.168.56.109 | SUCCESS | rc=0 >>
Sorting...
Full Text Search...
vim-nox/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled]
  Vi IMproved - enhanced vi editor - compact version
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

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.

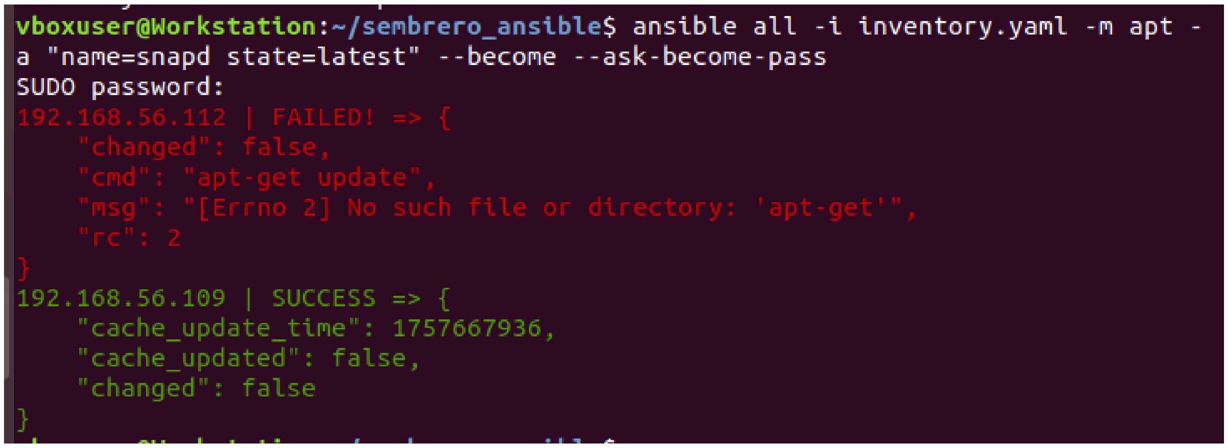
3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

   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?



```
vboxuser@Workstation:~/sembrero_ansible$ ansible all -m apt -a name=snapd --bec
ome --ask-become-pass
SUDO password:
192.168.56.112 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "[Errno 2] No such file or directory: 'apt-get'",
    "rc": 2
}
192.168.56.109 | SUCCESS => {
    "cache_update_time": 1757667936,
    "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.



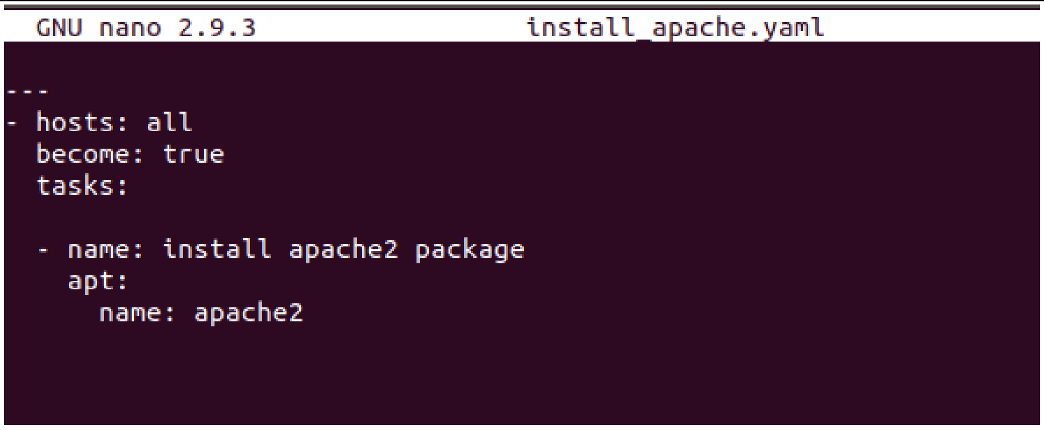
```
vboxuser@Workstation:~/sembrero_ansible$ ansible all -i inventory.yaml -m apt -
a "name=snapd state=latest" --become --ask-become-pass
SUDO password:
192.168.56.112 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "[Errno 2] No such file or directory: 'apt-get'",
    "rc": 2
}
192.168.56.109 | SUCCESS => {
    "cache_update_time": 1757667936,
    "cache_updated": false,
    "changed": false
}
```

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

Task 2: Writing our First Playbook

1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and the place or commands that ansible will carry out to bring to that state. You can use an editor to create a playbook. Before we proceed, make sure that you are in the directory of the repository that we use in the previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. This will create a playbook file called *install_apache.yml*. The .yml is the basic standard extension for playbook files.

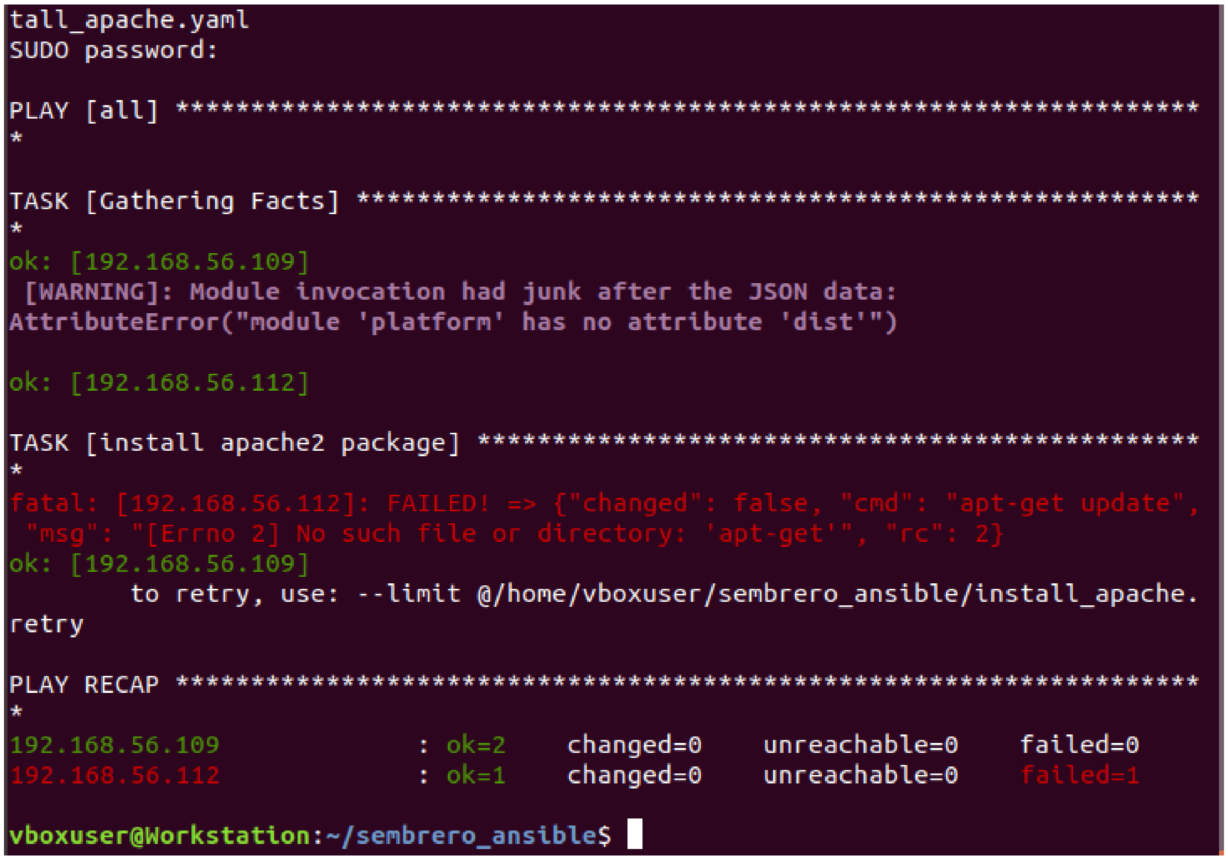
   When the editor appears, type the following:



```
  GNU nano 2.9.3                     install_apache.yaml

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

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

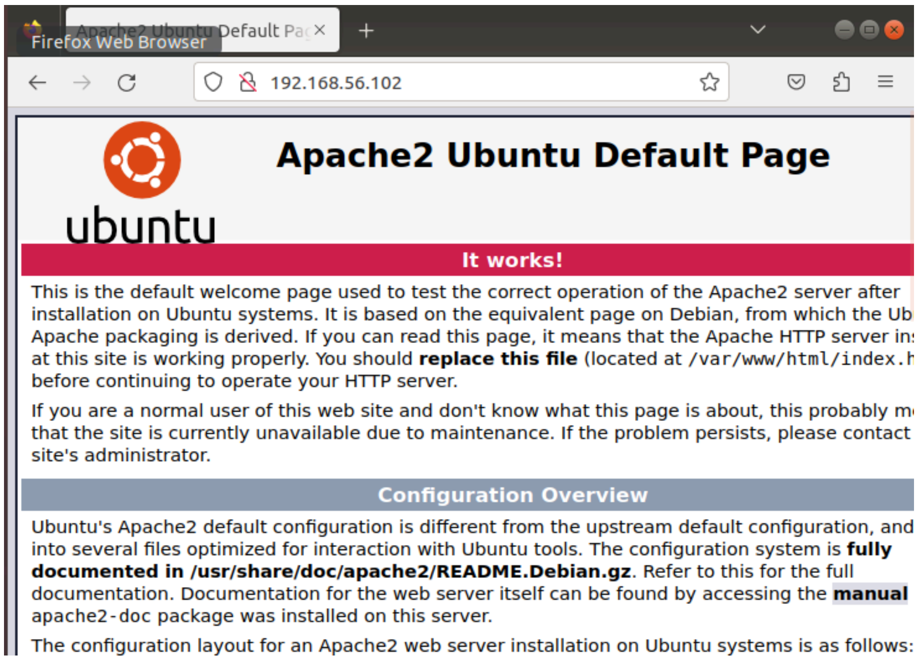
Make sure to save the file. Take note also of the alignments of the texts.

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



```
tall_apache.yaml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.109]
 [WARNING]: Module invocation had junk after the JSON data:
AttributeError("module 'platform' has no attribute 'dist'")

ok: [192.168.56.112]

TASK [install apache2 package] *************************************************
*
fatal: [192.168.56.112]: FAILED! => {"changed": false, "cmd": "apt-get update",
 "msg": "[Errno 2] No such file or directory: 'apt-get'", "rc": 2}
ok: [192.168.56.109]
        to retry, use: --limit @/home/vboxuser/sembrero_ansible/install_apache.
retry

PLAY RECAP *********************************************************************
*
192.168.56.109             : ok=2    changed=0    unreachable=0    failed=0
192.168.56.112             : ok=1    changed=0    unreachable=0    failed=1

vboxuser@Workstation:~/sembrero_ansible$ 
```

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 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 Ub Apache packaging is derived. If you can read this page, it means that the Apache HTTP server in at this site is working properly. You should **replace this file** (located at `/var/www/html/index.h` before continuing to operate your HTTP server.

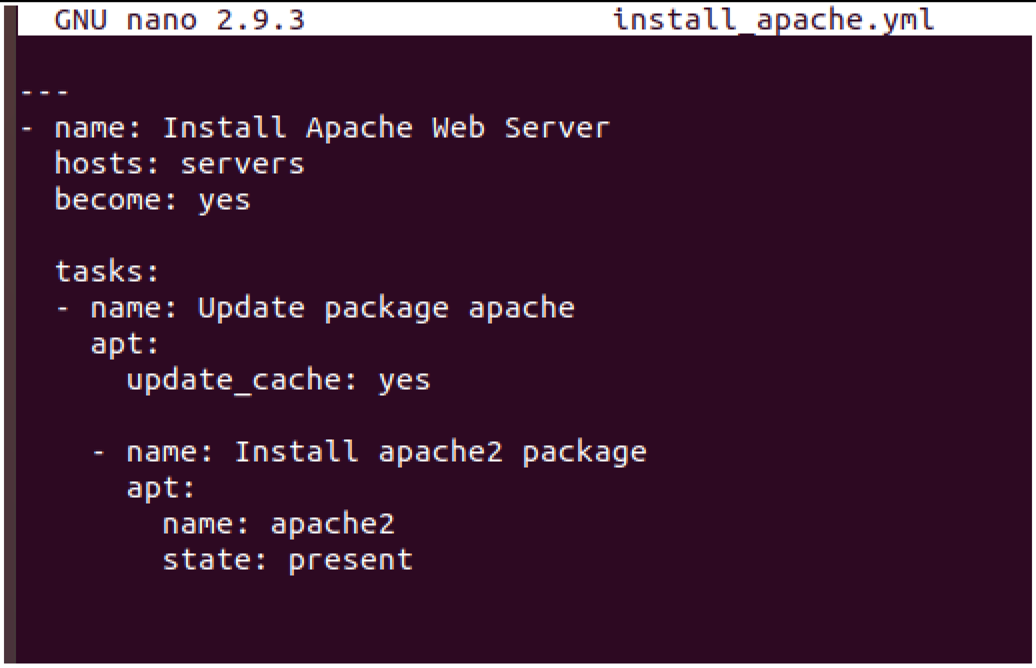
If you are a normal user of this web site and don't know what this page is about, this probably m that the site is currently unavailable due to maintenance. If the problem persists, please contact site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and 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** apache2-doc package was installed on this server.

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

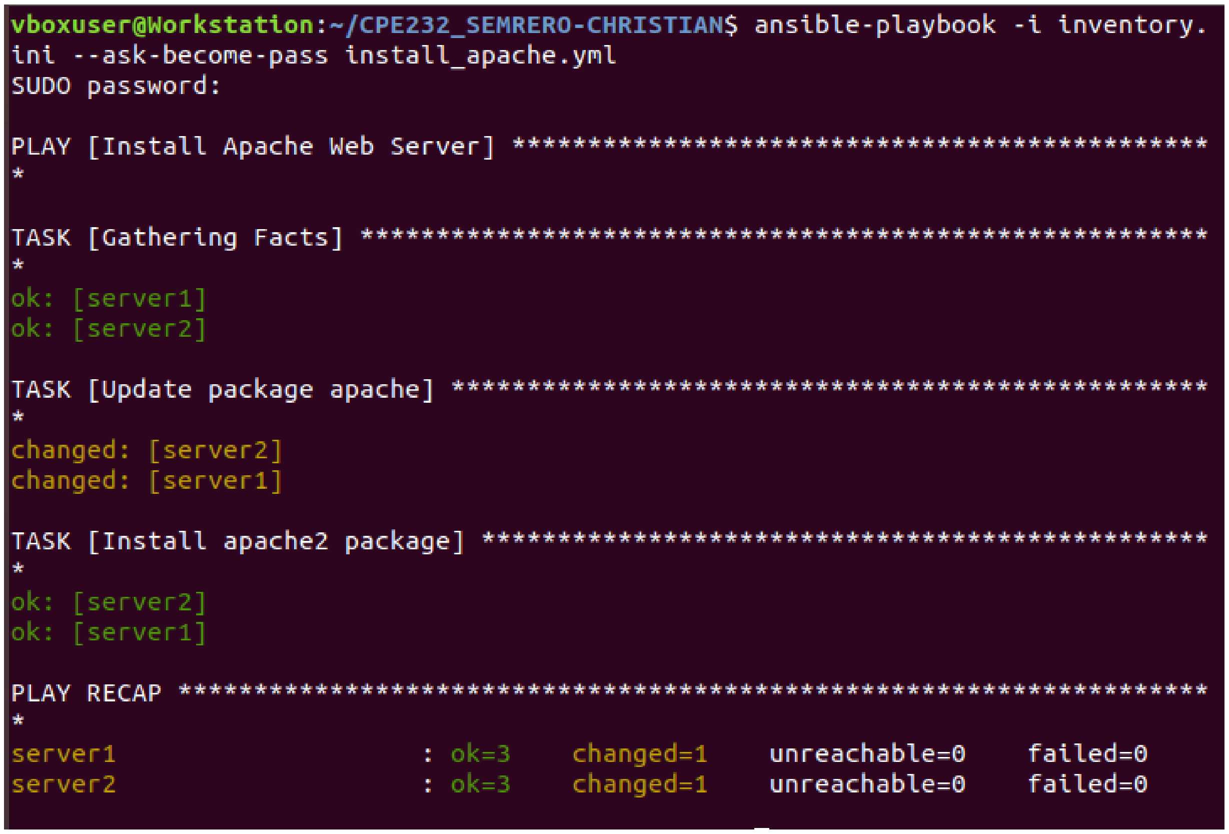
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache.* This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.



```
  GNU nano 2.9.3                    install_apache.yml

---
- name: Install Apache Web Server
  hosts: servers
  become: yes

  tasks:
  - name: Update package apache
    apt:
      update_cache: yes

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

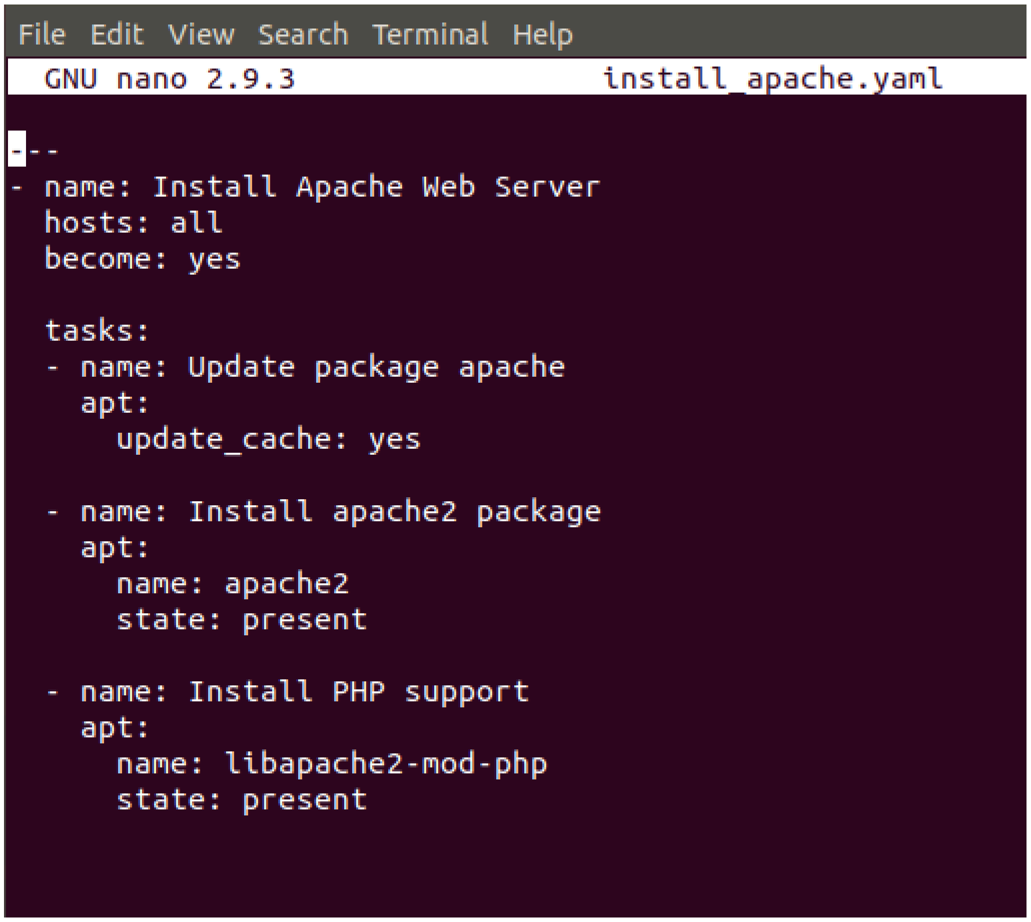
Save the changes to this file and exit.

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



```
vboxuser@Workstation:~/CPE232_SEMRERO-CHRISTIAN$ ansible-playbook -i inventory.
ini --ask-become-pass install_apache.yml
SUDO password:

PLAY [Install Apache Web Server] **********************************************
*

TASK [Gathering Facts] ********************************************************
*
ok: [server1]
ok: [server2]

TASK [Update package apache] **************************************************
*
changed: [server2]
changed: [server1]

TASK [Install apache2 package] ************************************************
*
ok: [server2]
ok: [server1]

PLAY RECAP ********************************************************************
*
server1                    : ok=3    changed=1    unreachable=0    failed=0
server2                    : ok=3    changed=1    unreachable=0    failed=0
```

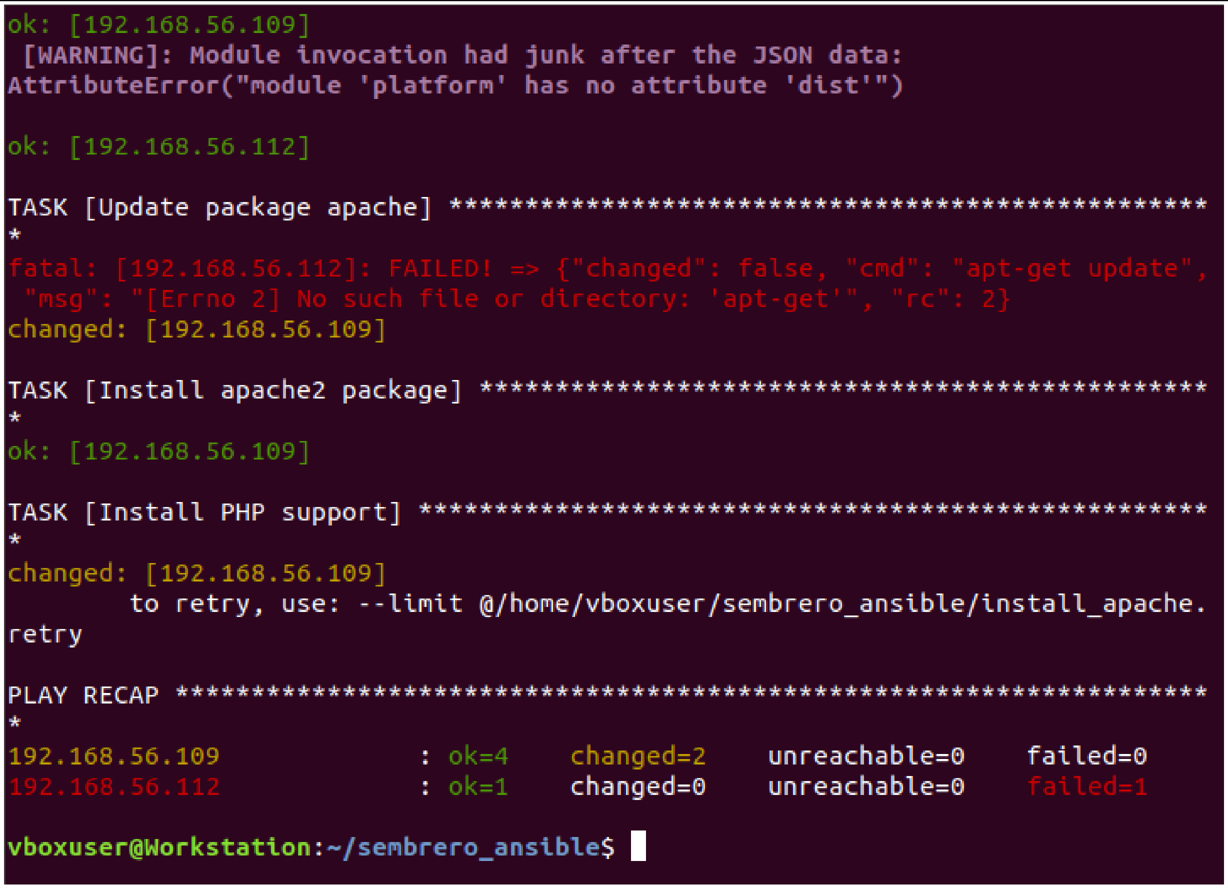
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
---
- name: Install Apache Web Server
  hosts: all
  become: yes

  tasks:
  - name: Update package apache
    apt:
      update_cache: yes

  - name: Install apache2 package
    apt:
      name: apache2
      state: present

  - name: Install PHP support
    apt:
      name: libapache2-mod-php
      state: present
```

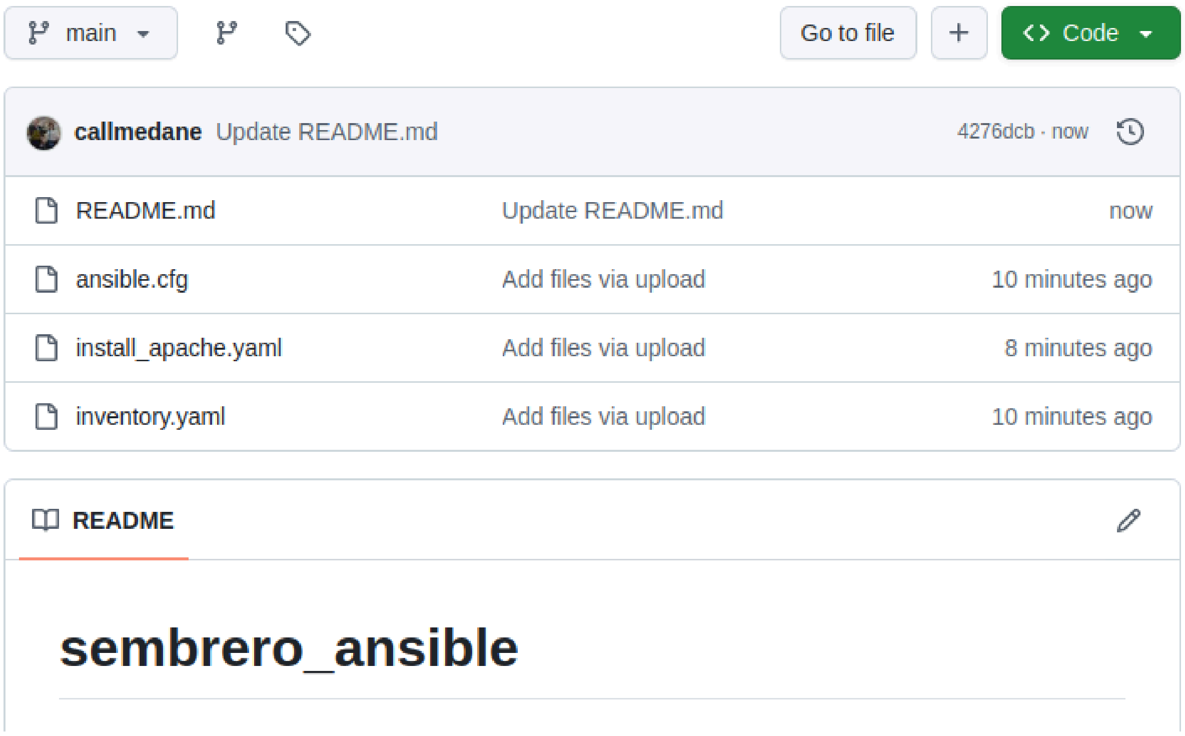
Save the changes to this file and exit.

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



```
ok: [192.168.56.109]
 [WARNING]: Module invocation had junk after the JSON data:
AttributeError("module 'platform' has no attribute 'dist'")

ok: [192.168.56.112]

TASK [Update package apache] ***************************************************
*
fatal: [192.168.56.112]: FAILED! => {"changed": false, "cmd": "apt-get update",
 "msg": "[Errno 2] No such file or directory: 'apt-get'", "rc": 2}
changed: [192.168.56.109]

TASK [Install apache2 package] *************************************************
*
ok: [192.168.56.109]

TASK [Install PHP support] *****************************************************
*
changed: [192.168.56.109]
        to retry, use: --limit @/home/vboxuser/sembrero_ansible/install_apache.
retry

PLAY RECAP *********************************************************************
*
192.168.56.109             : ok=4    changed=2    unreachable=0    failed=0
192.168.56.112             : ok=1    changed=0    unreachable=0    failed=1

vboxuser@Workstation:~/sembrero_ansible$
```

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



callmedane Update README.md 4276dcb · now

README.md	Update README.md	now
ansible.cfg	Add files via upload	10 minutes ago
install_apache.yaml	Add files via upload	8 minutes ago
inventory.yaml	Add files via upload	10 minutes ago

sembrero_ansible

Reflections:

Answer the following:

1. What is the importance of using a playbook?

I found it much easier to organize the task because I was able to use a playbook. The playbook saved time, and it made it easier to repeat the project. Instead of typing out all of those long commands one by one, I just had to run the playbook, and Ansible did all the work on both servers. To me it felt less stressful because I organized everything together. It was less work to duplicate the setup in case I have to do it again.

2. Summarize what we have done on this activity.

I learned to build my first playbook to install Apache, and then I was able to add update cache and even PHP support. I ran into some initial errors based on how I had my hosts set up, but I figured out I just had to pay closer attention to the inventory hosts in the playbook. Once I got that taken care of, it was awesome to see my Apache and PHP was set up properly. Overall it was a good experience of hands on practice to see how automation can help.