Name:Tracey Dee Bringuela	**Date Performed:9/11/24**
Course/Section:CPE31S2	**Date Submitted:9/11/24**
Instructor:	**Semester and SY:**

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

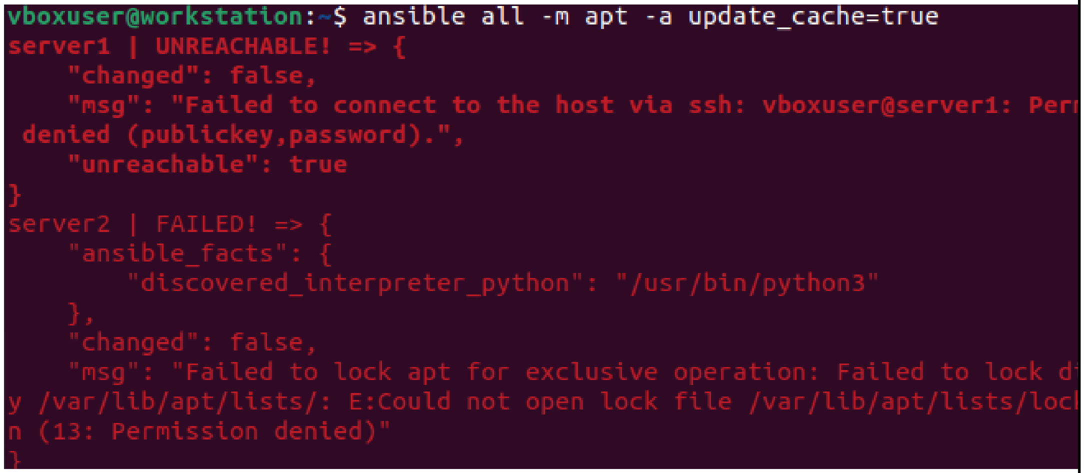
Task 1: Run elevated ad hoc commands

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

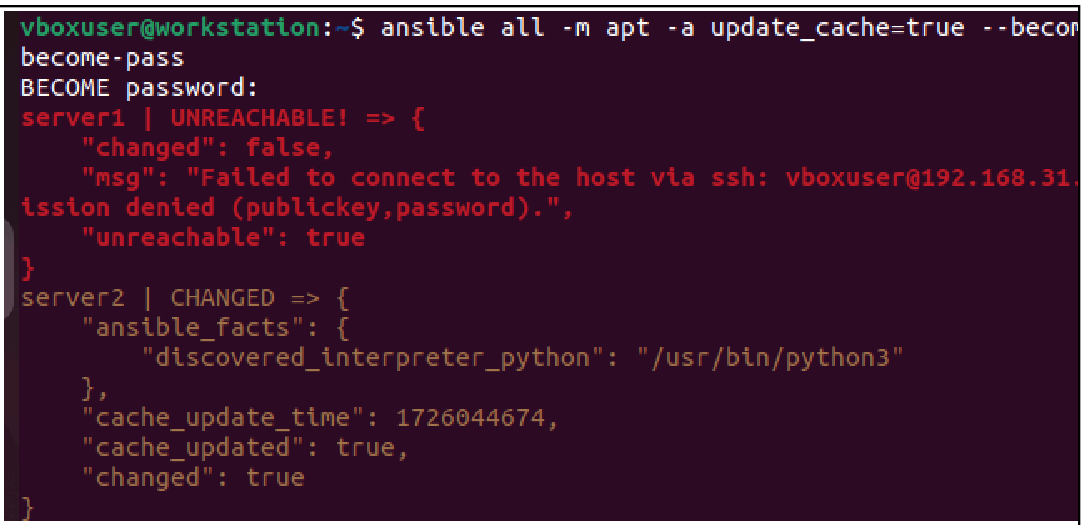
What is the result of the command? Is it successful?



```
vboxuser@workstation:~$ ansible all -m apt -a update_cache=true
server1 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: vboxuser@server1: Per
 denied (publickey,password).",
    "unreachable": true
}
server2 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock di
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/loc
n (13: Permission denied)"
}
```

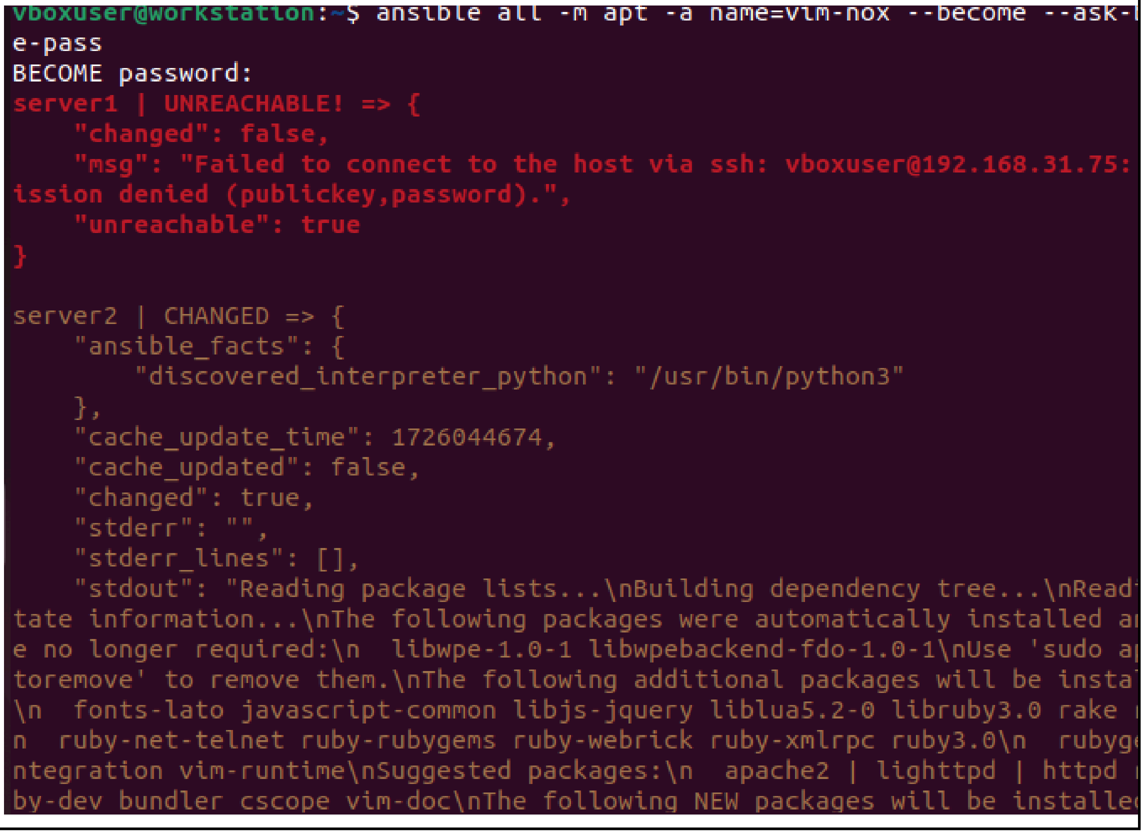
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



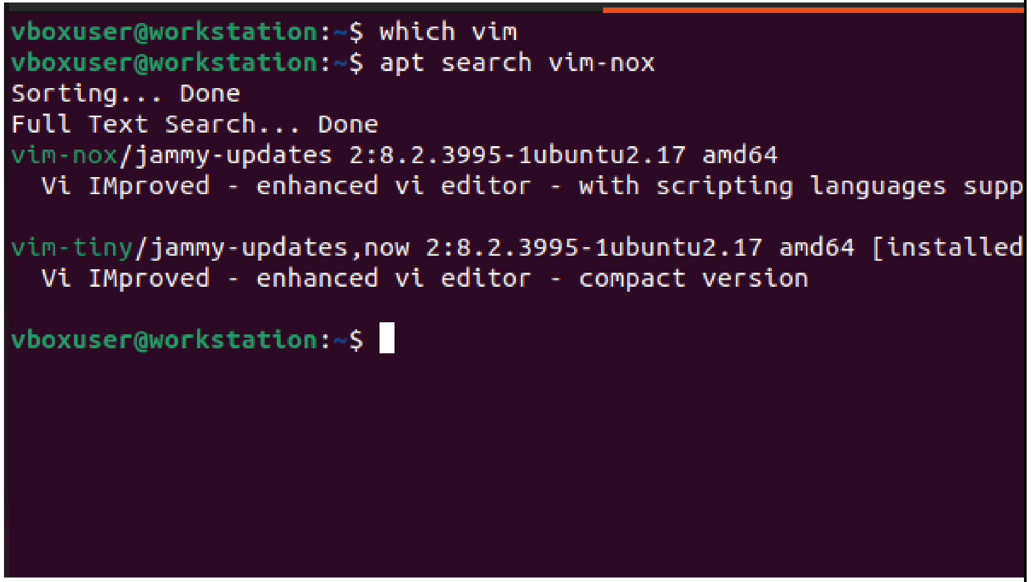
```
vboxuser@workstation:~$ ansible all -m apt -a update_cache=true --becor
become-pass
BECOME password:
server1 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: vboxuser@192.168.31.
ission denied (publickey,password).",
    "unreachable": true
}
server2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726044674,
    "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.



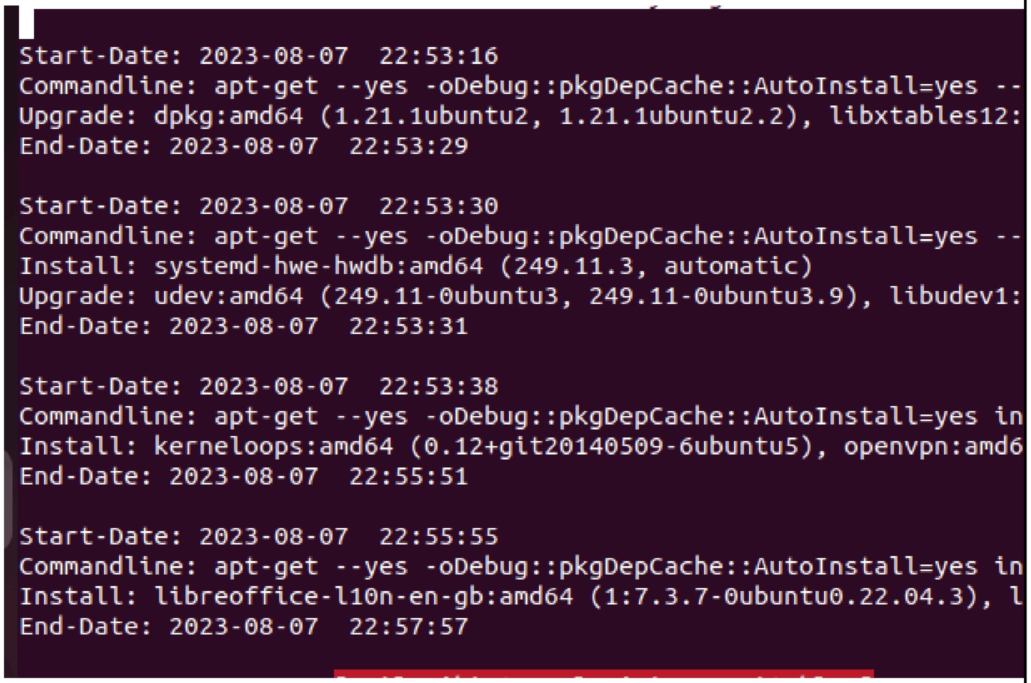
```
vboxuser@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-l
e-pass
BECOME password:
server1 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: vboxuser@192.168.31.75:
ission denied (publickey,password).",
    "unreachable": true
}

server2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726044674,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nRead
tate information...\nThe following packages were automatically installed ar
e no longer required:\n  libwpe-1.0-1 libwpebackend-fdo-1.0-1\nUse 'sudo ap
toremove' to remove them.\nThe following additional packages will be insta
\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake r
n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubyge
ntegration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd r
by-dev bundler cscope vim-doc\nThe following NEW packages will be installe
```

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:~$ which vim
vboxuser@workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates 2:8.2.3995-1ubuntu2.17 amd64
  Vi IMproved - enhanced vi editor - with scripting languages supp

vim-tiny/jammy-updates,now 2:8.2.3995-1ubuntu2.17 amd64 [installed
  Vi IMproved - enhanced vi editor - compact version

vboxuser@workstation:~$ 
```

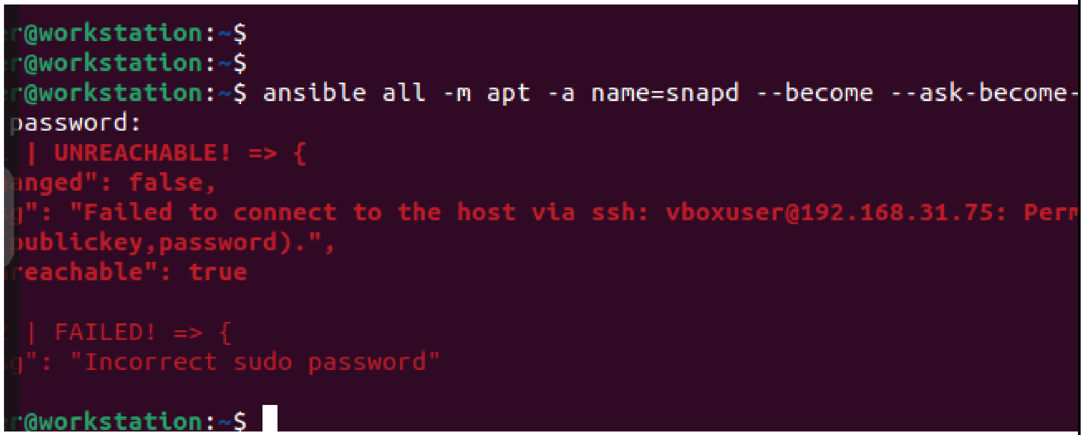
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.



```
Start-Date: 2023-08-07  22:53:16
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --
Upgrade: dpkg:amd64 (1.21.1ubuntu2, 1.21.1ubuntu2.2), libxtables12:
End-Date: 2023-08-07  22:53:29

Start-Date: 2023-08-07  22:53:30
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --
Install: systemd-hwe-hwdb:amd64 (249.11.3, automatic)
Upgrade: udev:amd64 (249.11-0ubuntu3, 249.11-0ubuntu3.9), libudev1:
End-Date: 2023-08-07  22:53:31

Start-Date: 2023-08-07  22:53:38
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes in
Install: kerneloops:amd64 (0.12+git20140509-6ubuntu5), openvpn:amd6
End-Date: 2023-08-07  22:55:51

Start-Date: 2023-08-07  22:55:55
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes in
Install: libreoffice-l10n-en-gb:amd64 (1:7.3.7-0ubuntu0.22.04.3), l
End-Date: 2023-08-07  22:57:57
```

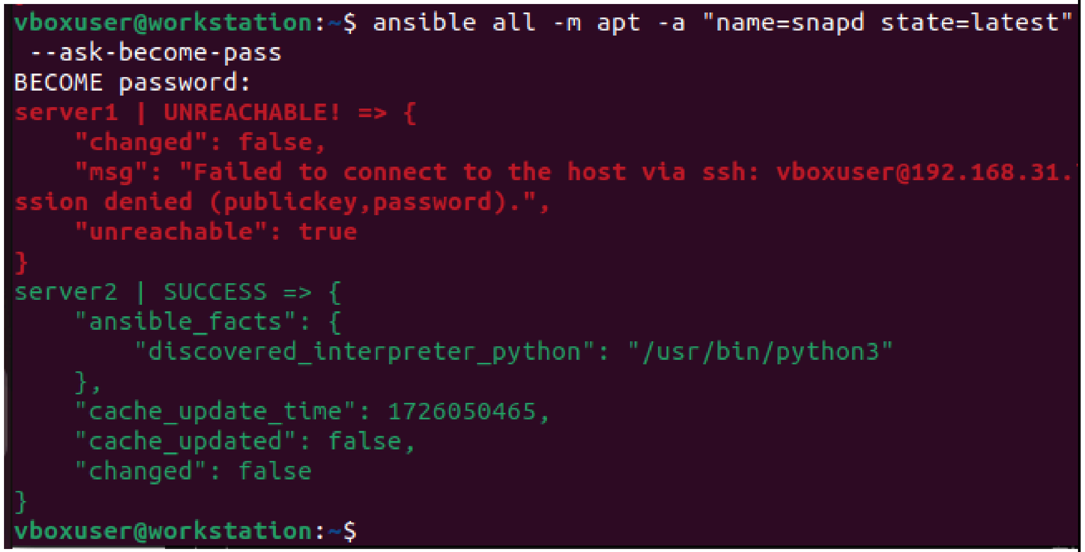
3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

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?



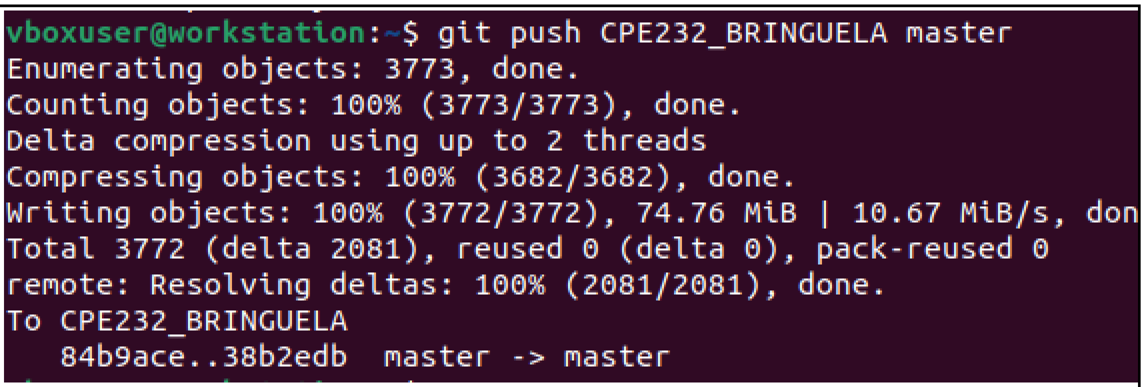
```
r@workstation:~$
r@workstation:~$
r@workstation:~$ ansible all -m apt -a name=snapd --become --ask-become-
password:
 | UNREACHABLE! => {
nged": false,
g": "Failed to connect to the host via ssh: vboxuser@192.168.31.75: Perm
ublickey,password).",
eachable": true

 | FAILED! => {
g": "Incorrect sudo password"

r@workstation:~$
```

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:~$ ansible all -m apt -a "name=snapd state=latest"
 --ask-become-pass
BECOME password:
server1 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: vboxuser@192.168.31.
ssion denied (publickey,password).",
    "unreachable": true
}
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726050465,
    "cache_updated": false,
    "changed": false
}
vboxuser@workstation:~$
```

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

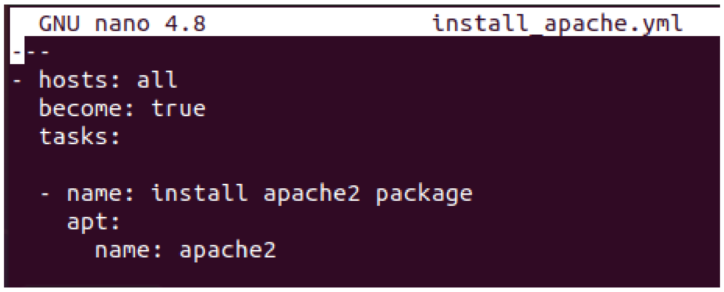


```
vboxuser@workstation:~$ git push CPE232_BRINGUELA master
Enumerating objects: 3773, done.
Counting objects: 100% (3773/3773), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3682/3682), done.
Writing objects: 100% (3772/3772), 74.76 MiB | 10.67 MiB/s, don
Total 3772 (delta 2081), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2081/2081), done.
To CPE232_BRINGUELA
   84b9ace..38b2edb  master -> master
```

Task 2: Writing our First Playbook

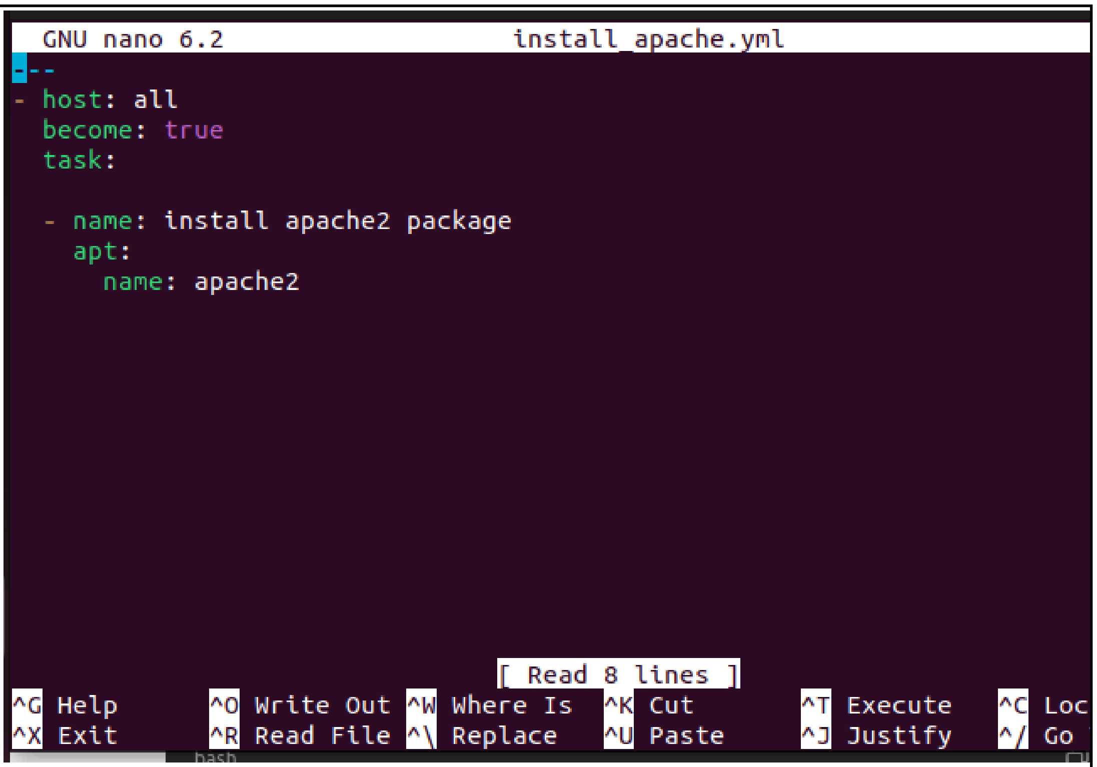
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and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

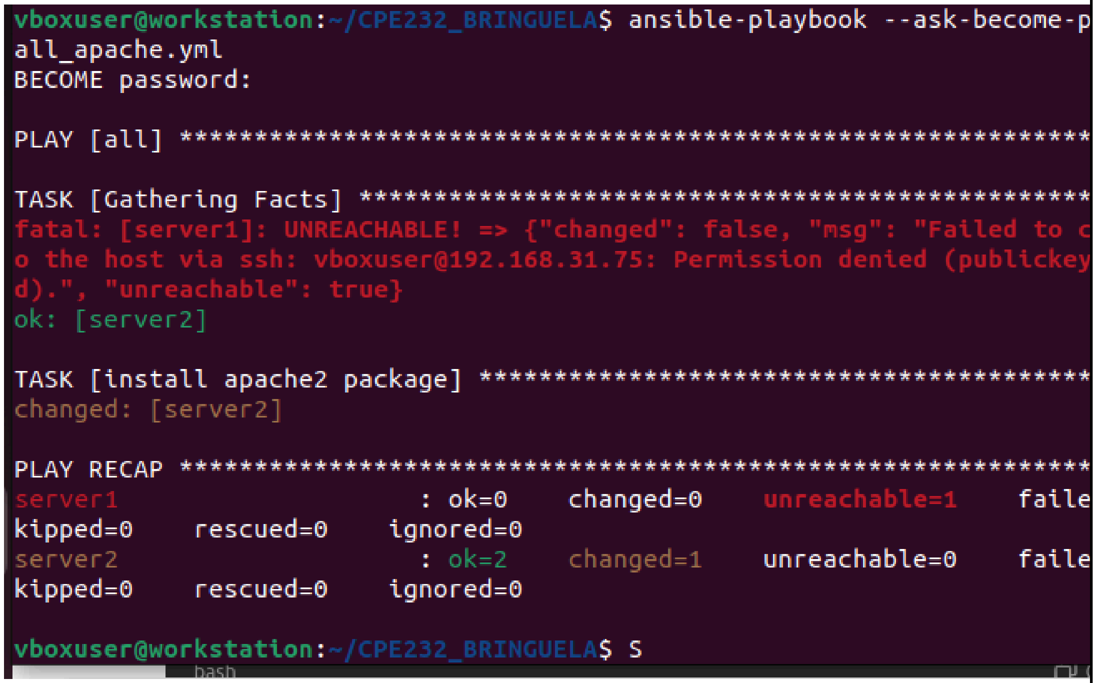
   /Make sure to save the file. Take note also of the alignments of the texts.



```
  GNU nano 6.2                         install_apache.yml
---
- host: all
  become: true
  task:

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

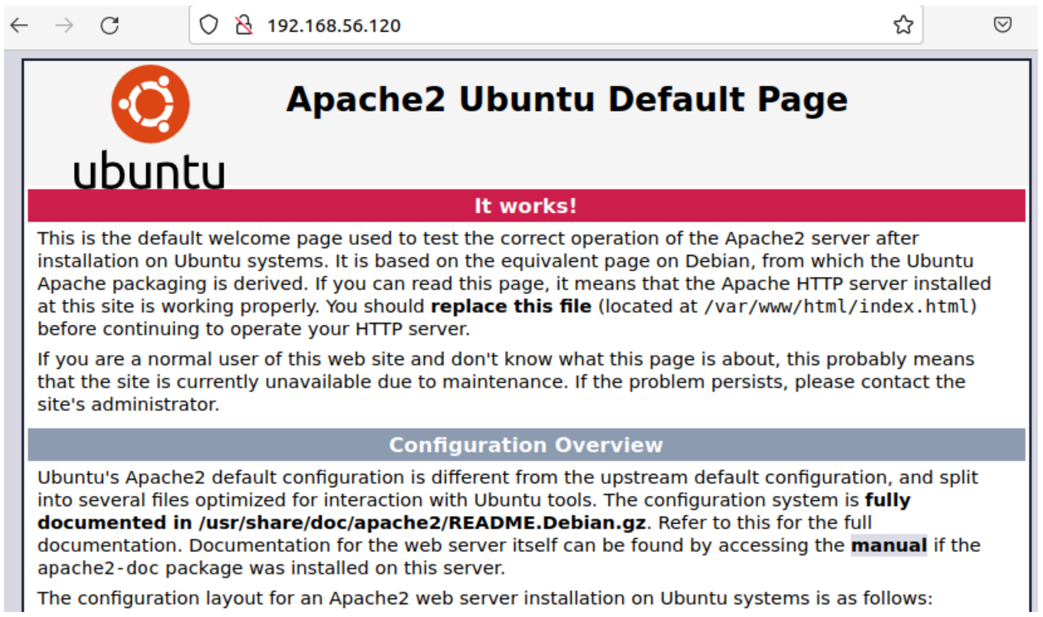
```
                                [ Read 8 lines ]
^G Help         ^O Write Out ^W Where Is  ^K Cut          ^T Execute    ^C Loc
^X Exit         ^R Read File ^\ Replace   ^U Paste        ^J Justify    ^/ Go
```

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



```
vboxuser@workstation:~/CPE232_BRINGUELA$ ansible-playbook --ask-become-p
all_apache.yml
BECOME password:

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

TASK [Gathering Facts] ****************************************************
fatal: [server1]: UNREACHABLE! => {"changed": false, "msg": "Failed to c
o the host via ssh: vboxuser@192.168.31.75: Permission denied (publickey
d).", "unreachable": true}
ok: [server2]

TASK [install apache2 package] ********************************************
changed: [server2]

PLAY RECAP ****************************************************************
server1                    : ok=0    changed=0    unreachable=1    faile
kipped=0    rescued=0    ignored=0
server2                    : ok=2    changed=1    unreachable=0    faile
kipped=0    rescued=0    ignored=0

vboxuser@workstation:~/CPE232_BRINGUELA$ S
```

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!

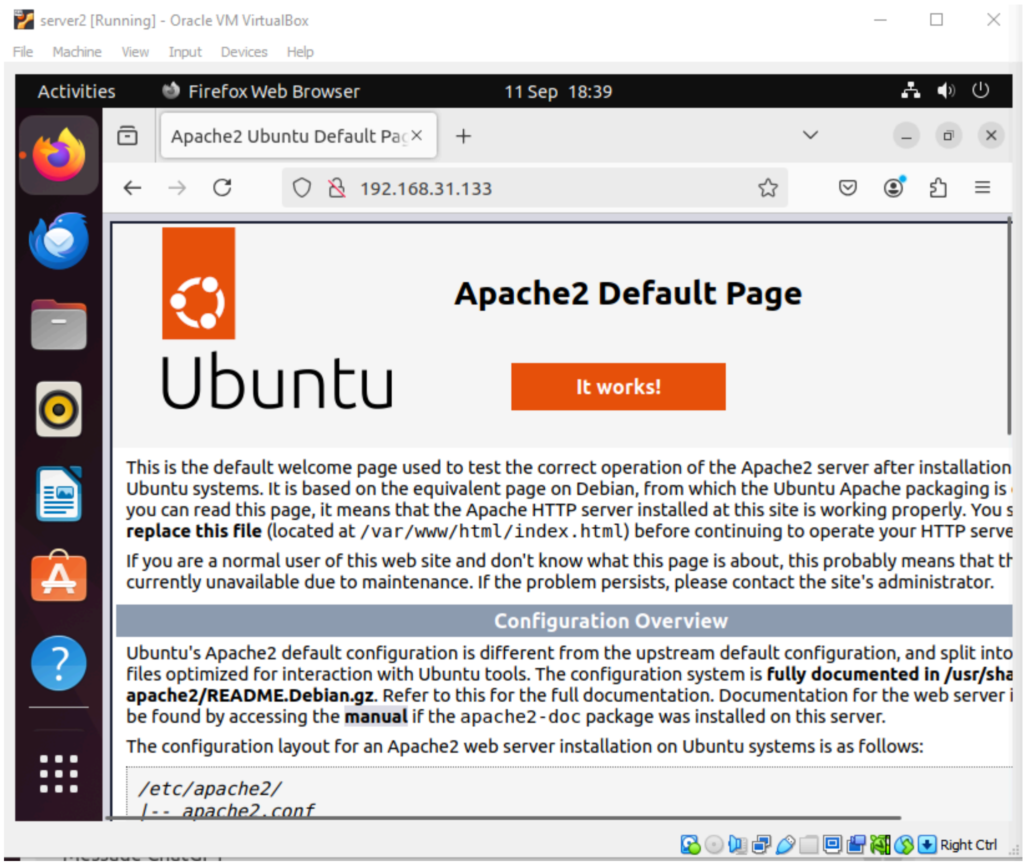
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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



server2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
11 Sep 18:39
Apache2 Ubuntu Default Pa
Right Ctrl

192.168.31.133

Apache2 Default Page

It works!

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

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

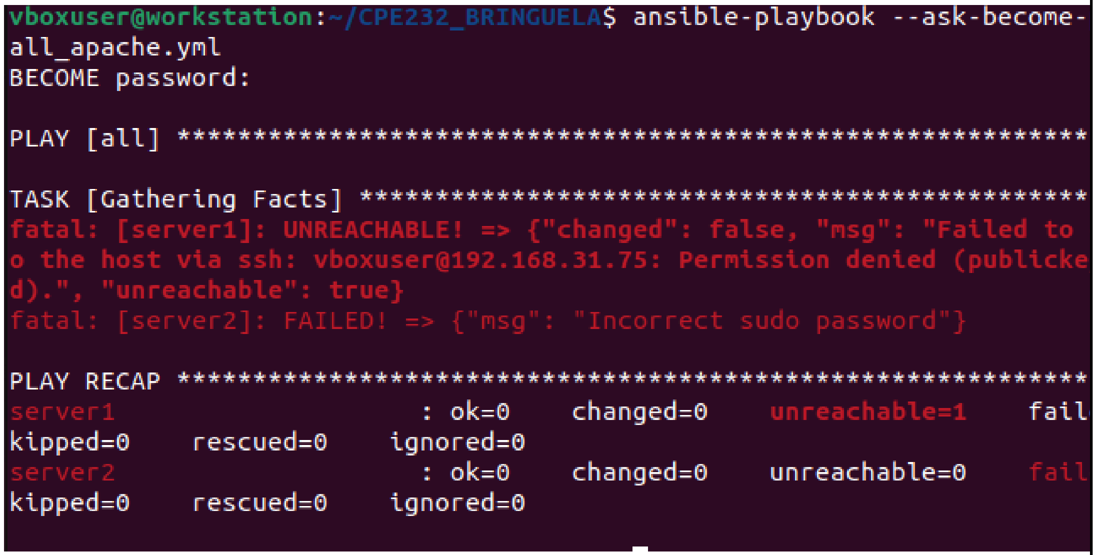
Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/sha apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server i 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
```

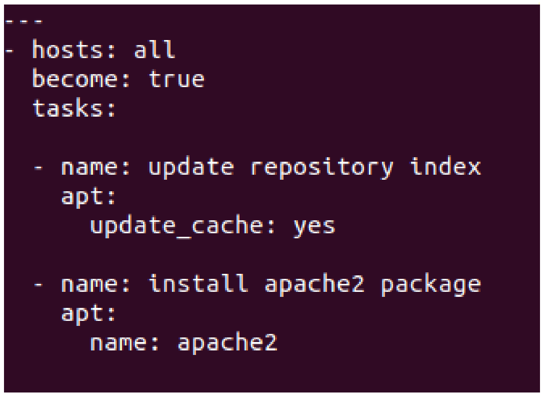
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?



```
vboxuser@workstation:~/CPE232_BRINGUELA$ ansible-playbook --ask-become-
all_apache.yml
BECOME password:

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

TASK [Gathering Facts] *************************************************
fatal: [server1]: UNREACHABLE! => {"changed": false, "msg": "Failed to
o the host via ssh: vboxuser@192.168.31.75: Permission denied (publicke
d).", "unreachable": true}
fatal: [server2]: FAILED! => {"msg": "Incorrect sudo password"}

PLAY RECAP *************************************************************
server1                    : ok=0    changed=0    unreachable=1    fail
kipped=0    rescued=0    ignored=0
server2                    : ok=0    changed=0    unreachable=0    fail
kipped=0    rescued=0    ignored=0
```

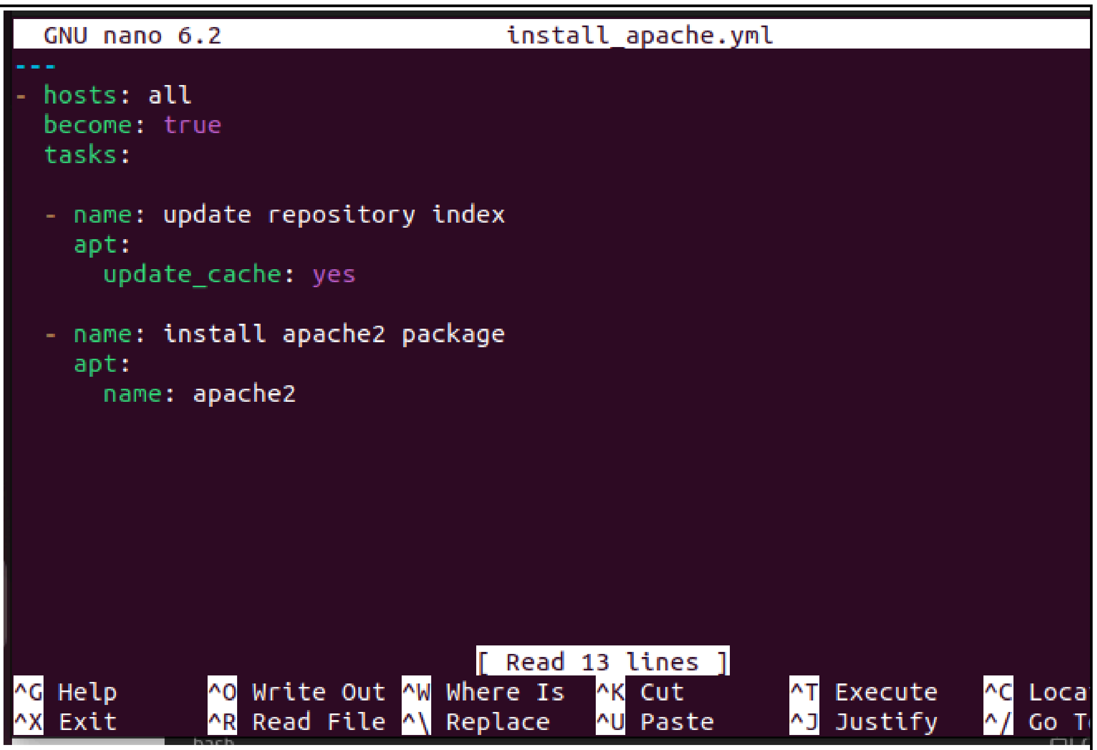
5.

6. 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
```

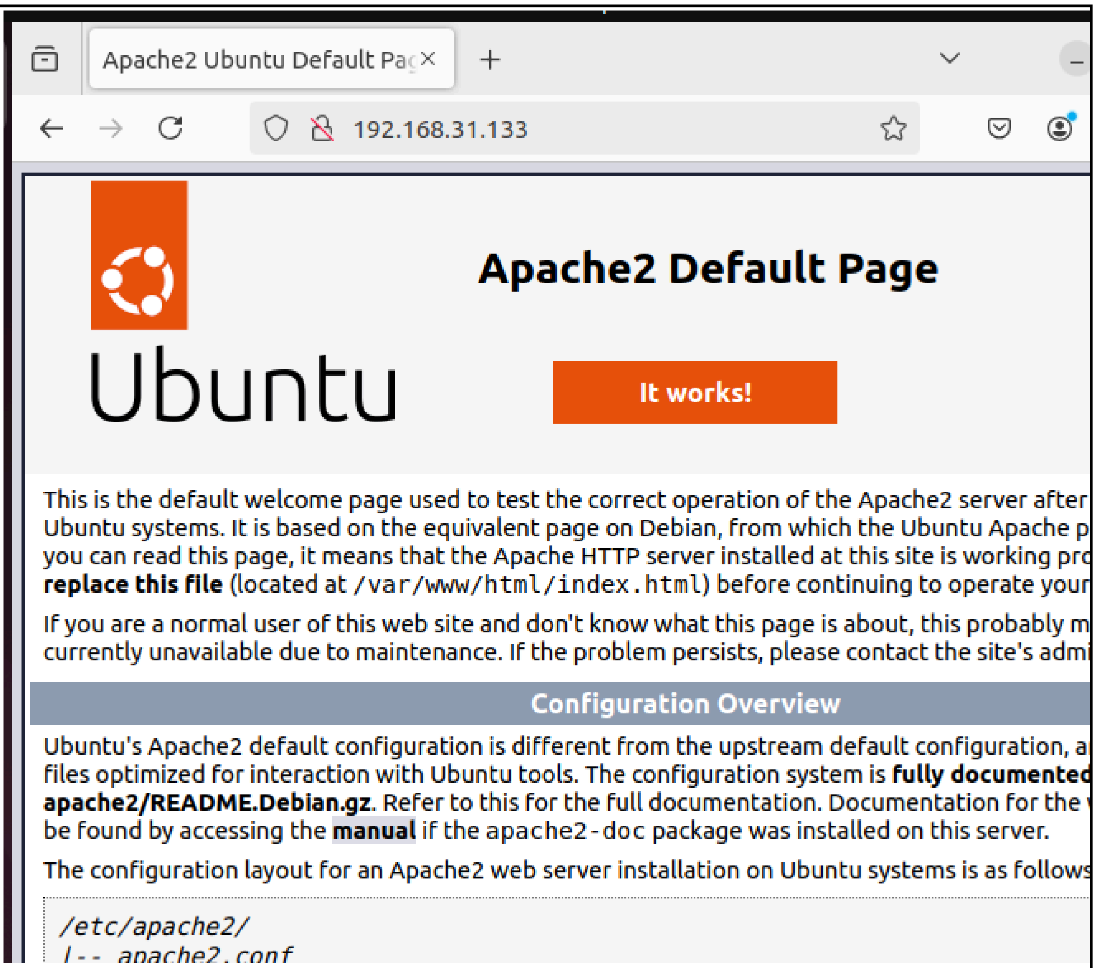
Save the changes to this file and exit.



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

  - name: update repository index
    apt:
      update_cache: yes

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

```
                               [ Read 13 lines ]
^G Help         ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Loca
^X Exit         ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To
```

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



Apache2 Ubuntu Default Pa

192.168.31.133

Apache2 Default Page

It works!

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

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

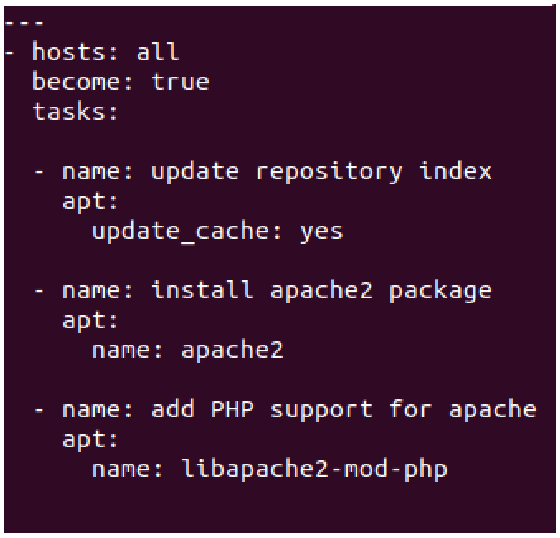
Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, a
files optimized for interaction with Ubuntu tools. The configuration system is **fully documented**
apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the
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

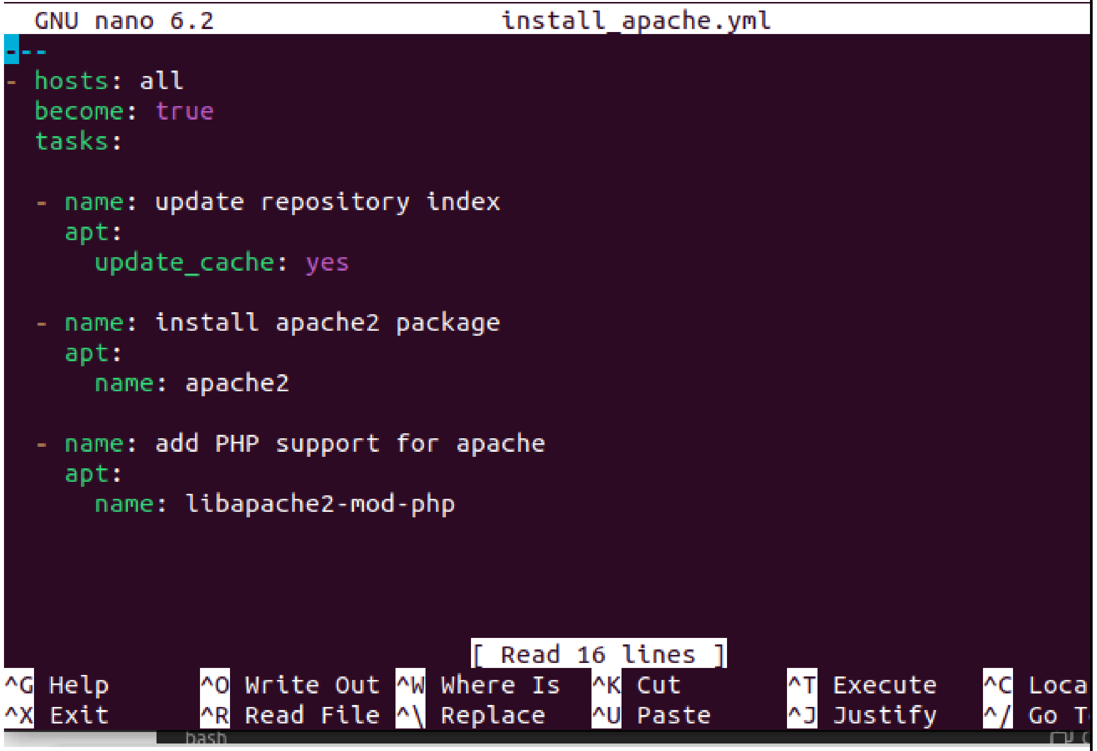
```
/etc/apache2/
|-- apache2.conf
```

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



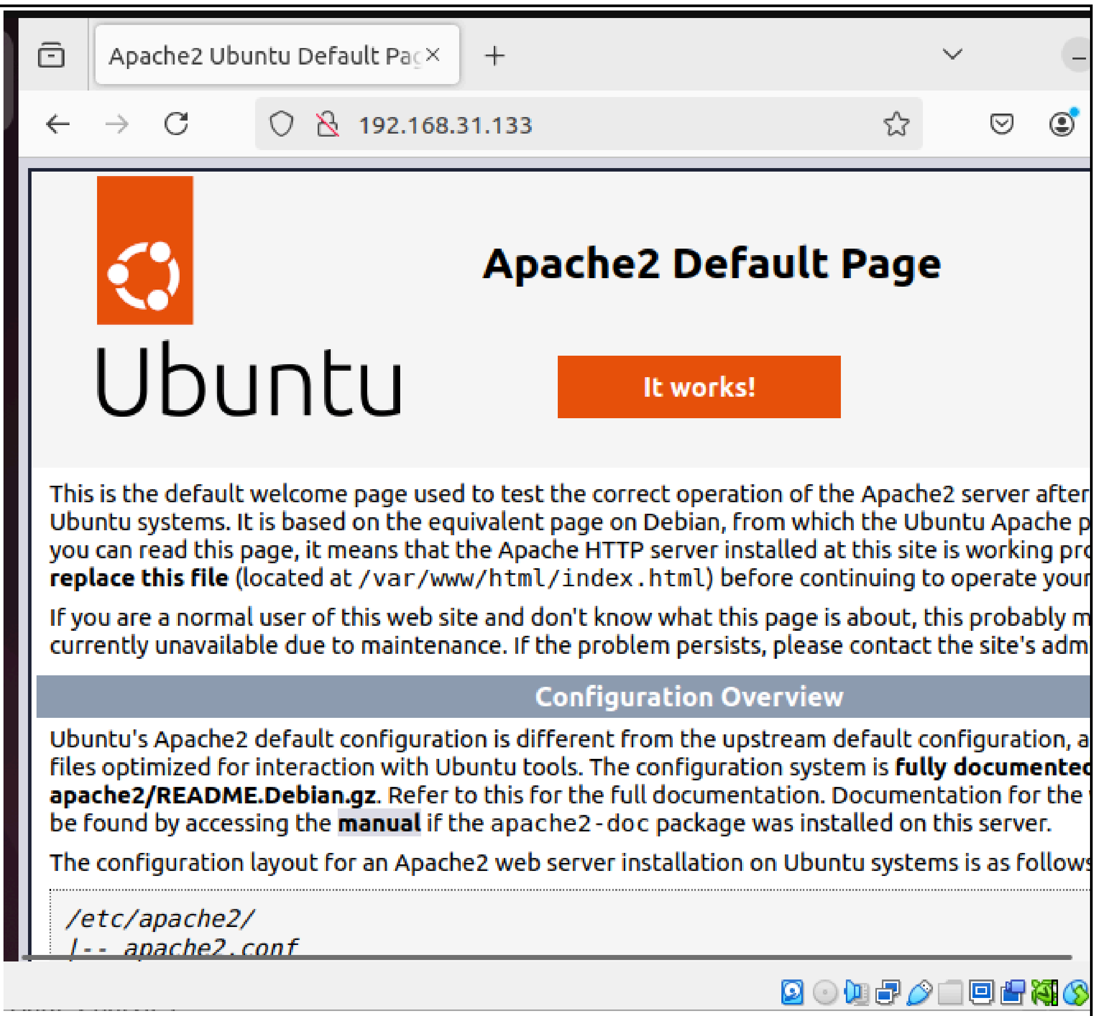
```
---
- hosts: all
  become: true
  tasks:

  - 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



                               [ Read 16 lines ]
^G Help       ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Loca
^X Exit       ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go T
```

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



Apache2 Default Page

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

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

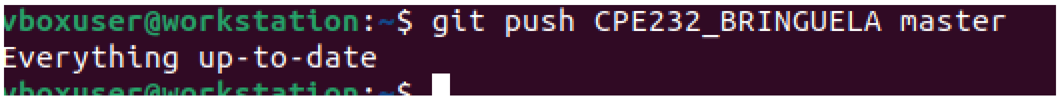
Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, a
files optimized for interaction with Ubuntu tools. The configuration system is **fully documented**
apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the
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
```

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



```
vboxuser@workstation:~$ git push CPE232_BRINGUELA master
Everything up-to-date
vboxuser@workstation:~$
```

Reflections:

Answer the following:

1. What is the importance of using a playbook?

   Playbooks ensure that configurations and tasks are applied uniformly across all target systems, maintaining consistency and reducing discrepancies. They automate repetitive and complex tasks, saving time and minimizing human error, which leads to more efficient system management. Additionally, playbooks act as a living document of applied configurations and changes, offering clear records that aid in troubleshooting, auditing, and future maintenance.

2. Summarize what we have done on this activity.

   We executed an Ansible command to update the package cache on remote servers, which initially failed due to insufficient privileges but succeeded after using the --become option to elevate permissions. We then installed the vim-nox package on these servers, confirmed its installation, and reviewed the APT logs for details on the process. Additionally, we used Ansible to install the snapd package and explored the effects of different state parameters. Finally, we committed all changes to GitHub, ensuring that modifications were properly tracked and versioned.