Name: Karlo Santos	**Date Performed: 09/11/23**
Course/Section: CPE31S5	**Date Submitted: 09/12/23**
Instructor: Engr. Roman Richard	**Semester and SY: 1st sem(2023-2024)**

Activity 4: Running Elevated Ad hoc Commands

1. Objectives:

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. Discussion:

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.

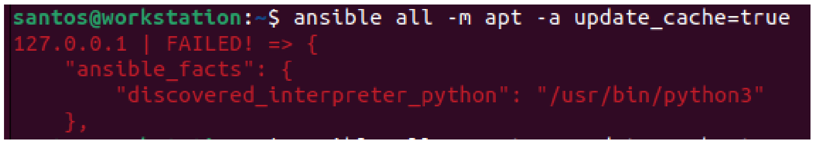
Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, 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:

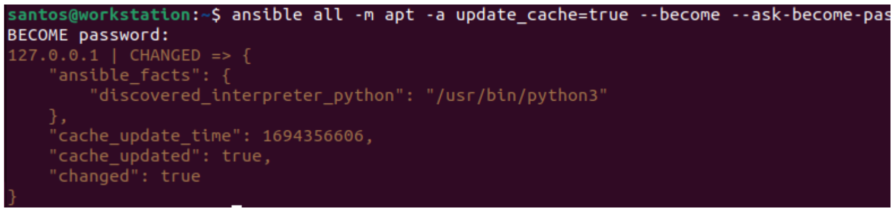
ansible all -m apt -a update_cache=true
What is the result of the command? Is it successful?
The result of the command failed.



```
santos@workstation:~$ ansible all -m apt -a update_cache=true
127.0.0.1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
```

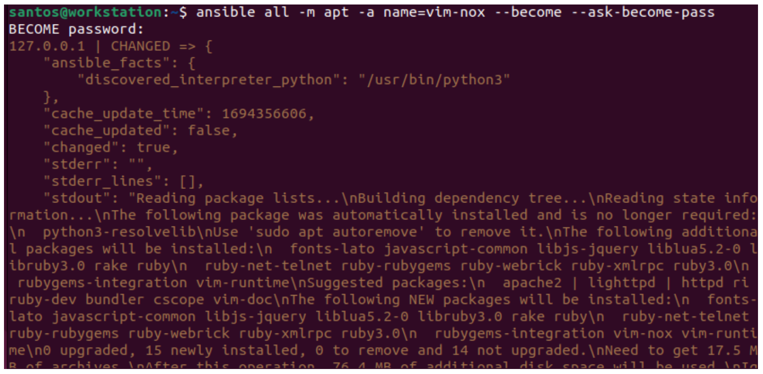
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



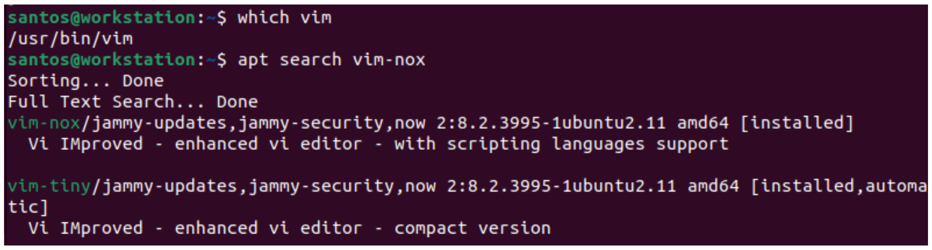
```
santos@workstation:~$ ansible all -m apt -a update_cache=true --become --ask-become-pas
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694356606,
    "cache_updated": true,
    "changed": 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.



```
santos@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": 1694356606,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state info
rmation...\nThe following package was automatically installed and is no longer required:
\n  python3-resolvelib\nUse 'sudo apt autoremove' to remove it.\nThe following additiona
l packages will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 l
ibruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n
 rubygems-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd ri
ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n  fonts-
lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  ruby-net-telnet
ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration vim-nox vim-runti
me\n0 upgraded, 15 newly installed, 0 to remove and 14 not upgraded.\nNeed to get 17.5 M
B of archives.\nAfter this operation, 76.4 MB of additional disk space will be used.\nIg
```

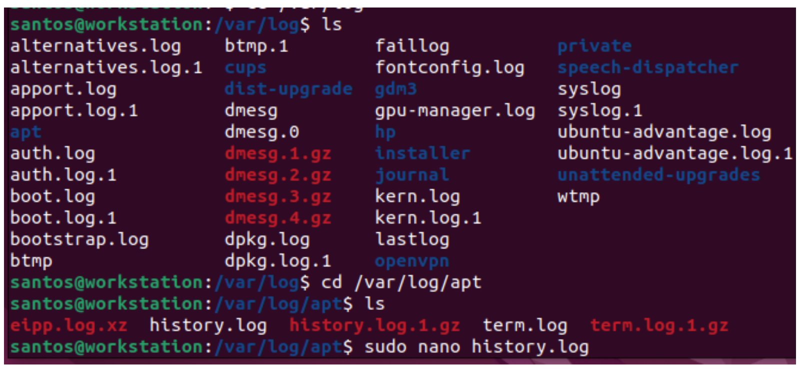
2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?



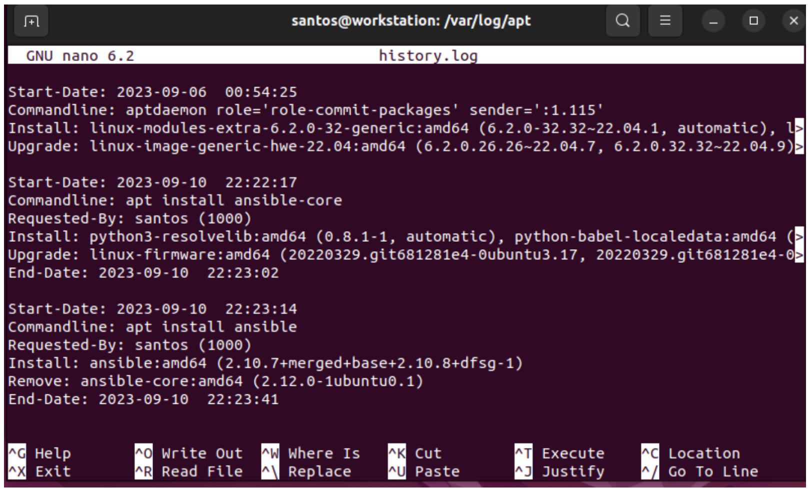
```
santos@workstation:~$ which vim
/usr/bin/vim
santos@workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

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

The command was successful.

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.



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

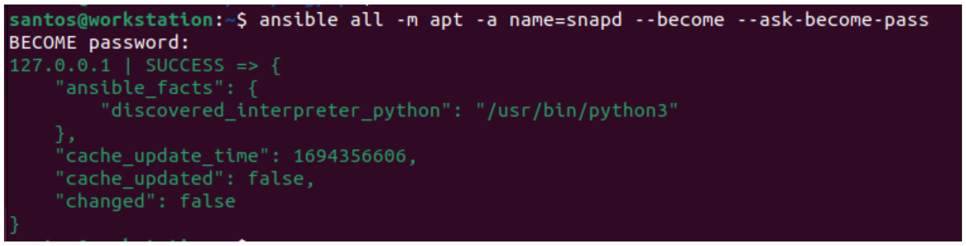


```
santos@workstation: /var/log/apt

  GNU nano 6.2                          history.log

Start-Date: 2023-09-06  00:54:25
Commandline: aptdaemon role='role-commit-packages' sender=':1.115'
Install: linux-modules-extra-6.2.0-32-generic:amd64 (6.2.0-32.32~22.04.1, automatic), l>
Upgrade: linux-image-generic-hwe-22.04:amd64 (6.2.0.26.26~22.04.7, 6.2.0.32.32~22.04.9)>

Start-Date: 2023-09-10  22:22:17
Commandline: apt install ansible-core
Requested-By: santos (1000)
Install: python3-resolvelib:amd64 (0.8.1-1, automatic), python-babel-localedata:amd64 (>
Upgrade: linux-firmware:amd64 (20220329.git681281e4-0ubuntu3.17, 20220329.git681281e4-0>
End-Date: 2023-09-10  22:23:02

Start-Date: 2023-09-10  22:23:14
Commandline: apt install ansible
Requested-By: santos (1000)
Install: ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1)
Remove: ansible-core:amd64 (2.12.0-1ubuntu0.1)
End-Date: 2023-09-10  22:23:41

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

It shows the history of the different installation of packages and also upgrades. As we can see in the screenshot it shows the date, command, and other details about my installation, example of this is the ansible.

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.

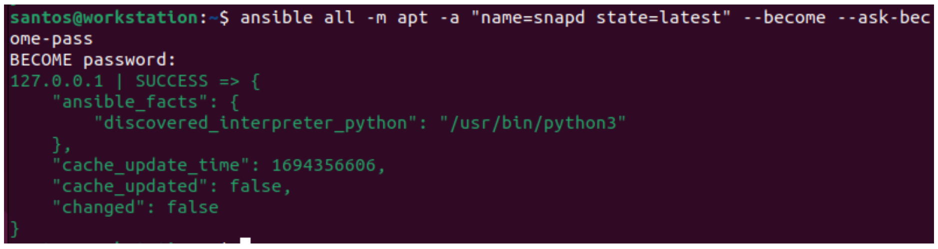
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*
   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?



```
santos@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": 1694356606,
    "cache_updated": false,
    "changed": false
}
```

   Based on the result, it shows that it was successful. Also, it shows that it didn't change anything in the remote server since it shows false in the "changed".

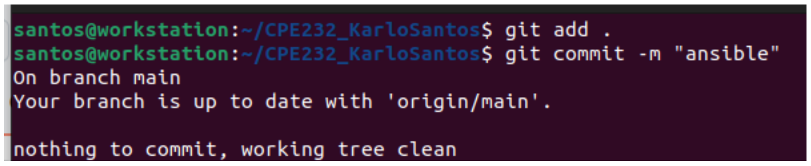
   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*



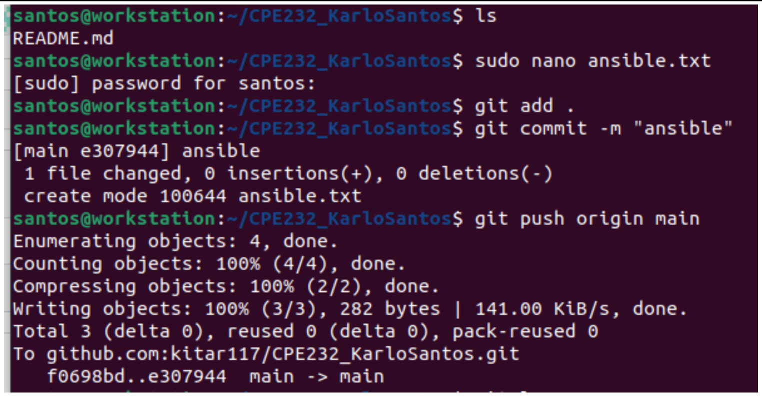
```
santos@workstation:~$ ansible all -m apt -a "name=snapd state=latest" --become --ask-bec
ome-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694356606,
    "cache_updated": false,
    "changed": false
}
```

   Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.
   It contains the same output after we execute the command even though we add a state=latest inside of the double quotation.

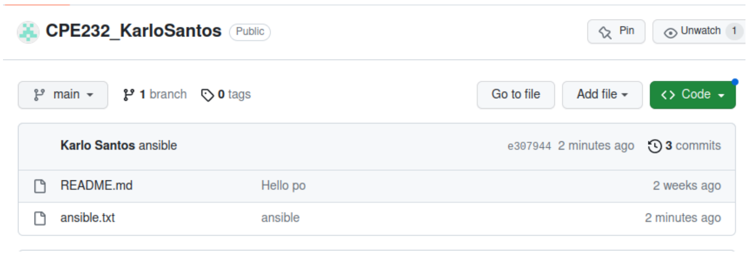
4. At this point, make sure to commit all changes to GitHub.



```
santos@workstation:~/CPE232_KarloSantos$ git add .
santos@workstation:~/CPE232_KarloSantos$ git commit -m "ansible"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



```
santos@workstation:~/CPE232_KarloSantos$ ls
README.md
santos@workstation:~/CPE232_KarloSantos$ sudo nano ansible.txt
[sudo] password for santos:
santos@workstation:~/CPE232_KarloSantos$ git add .
santos@workstation:~/CPE232_KarloSantos$ git commit -m "ansible"
[main e307944] ansible
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ansible.txt
santos@workstation:~/CPE232_KarloSantos$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 141.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kitar117/CPE232_KarloSantos.git
   f0698bd..e307944  main -> main
```



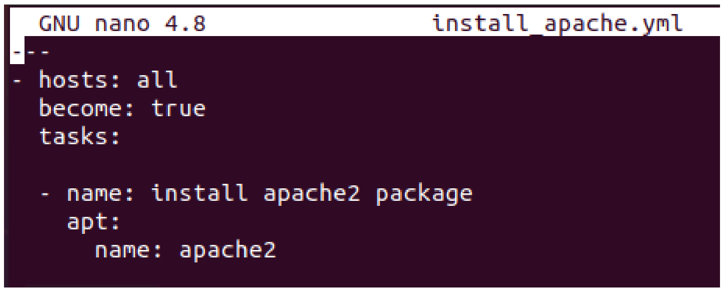
CPE232_KarloSantos Public

```
Karlo Santos ansible          e307944  2 minutes ago   3 commits
README.md        Hello po                    2 weeks ago
ansible.txt      ansible                     2 minutes ago
```

Task 2: Writing our First Playbook

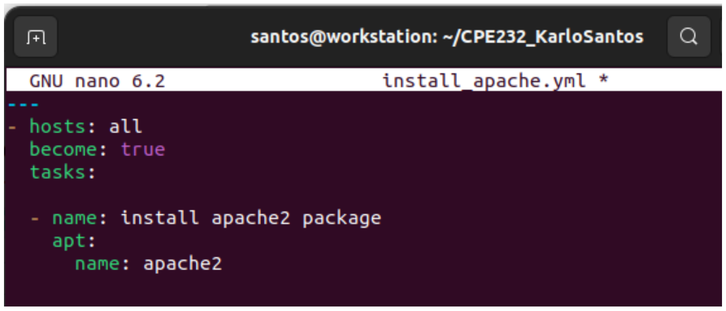
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and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



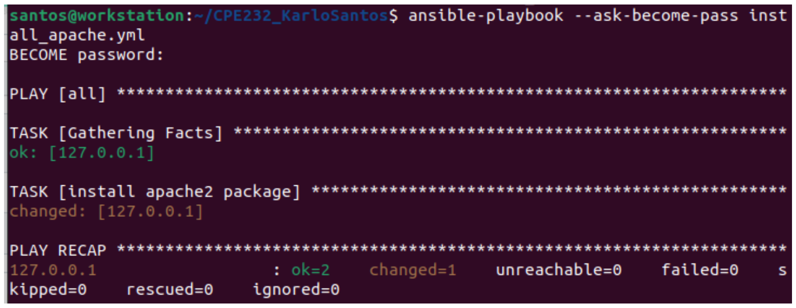
```
 GNU nano 4.8                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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



```
santos@workstation: ~/CPE232_KarloSantos
  GNU nano 6.2                  install_apache.yml *
---
- hosts: all
  become: true
  tasks:

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

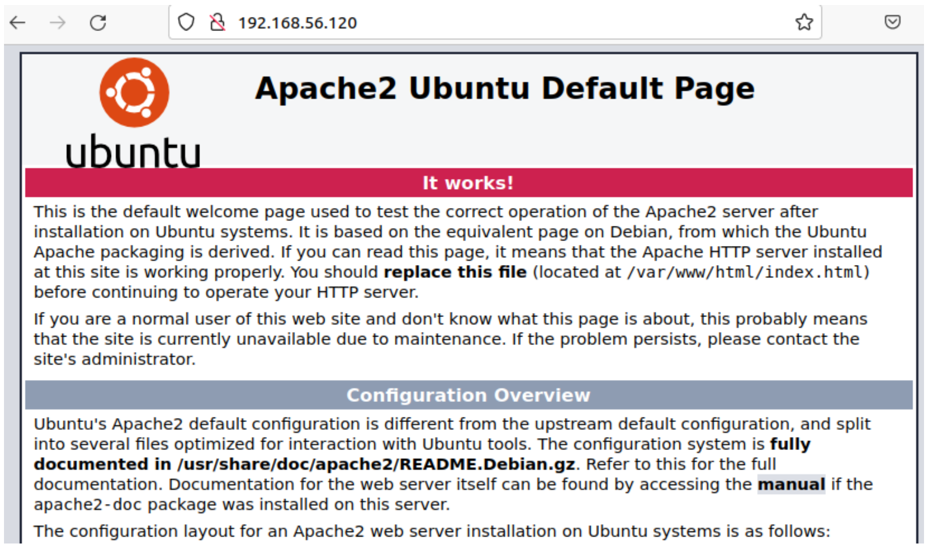
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.



```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass inst
all_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    s
kipped=0    rescued=0    ignored=0
```

The result of this command is shown in the first part is "Gathering Facts", it is about gathering or getting different host information and it shows it is successful. Next is about installation of apache2 base on the task name, it use the apt base on the given command.

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!

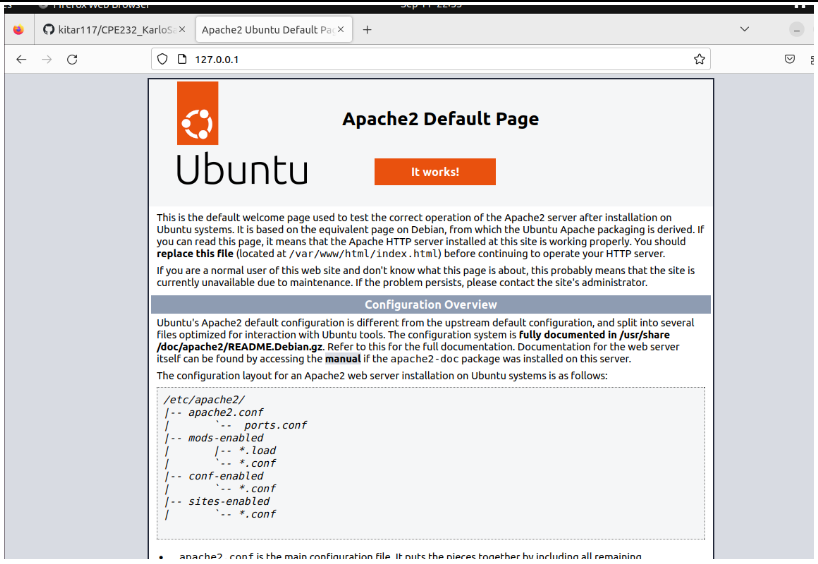
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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



Apache2 Default Page

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

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

Configuration Overview

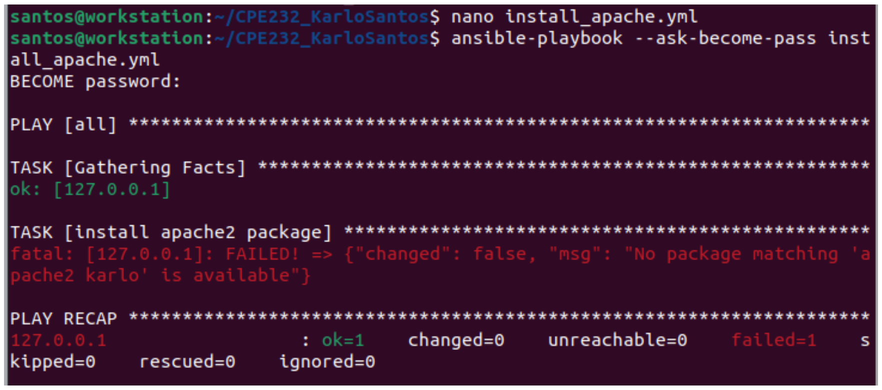
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share /doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

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

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

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

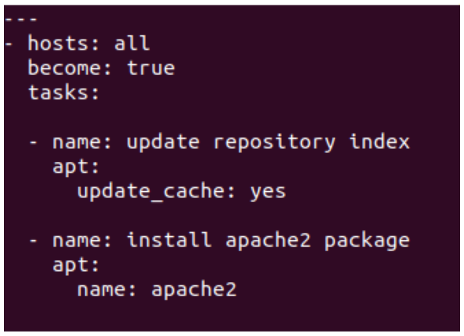
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?



```
santos@workstation:~/CPE232_KarloSantos$ nano install_apache.yml
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass inst
all_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 'a
pache2 karlo' is available"}

PLAY RECAP **********************************************************************
127.0.0.1                  : ok=1    changed=0    unreachable=0    failed=1    s
kipped=0    rescued=0    ignored=0
```

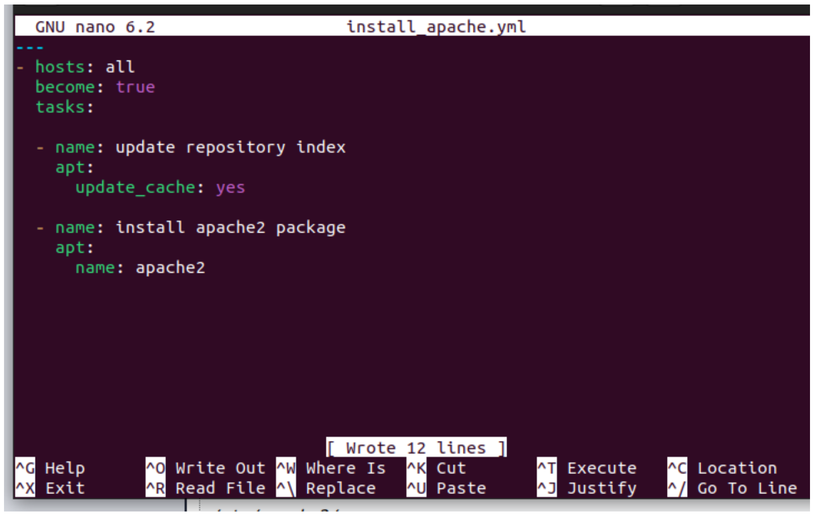
As we can see after I change the name of the package, the output shows an error or “FAILED” since no package matches the changes in it.

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.



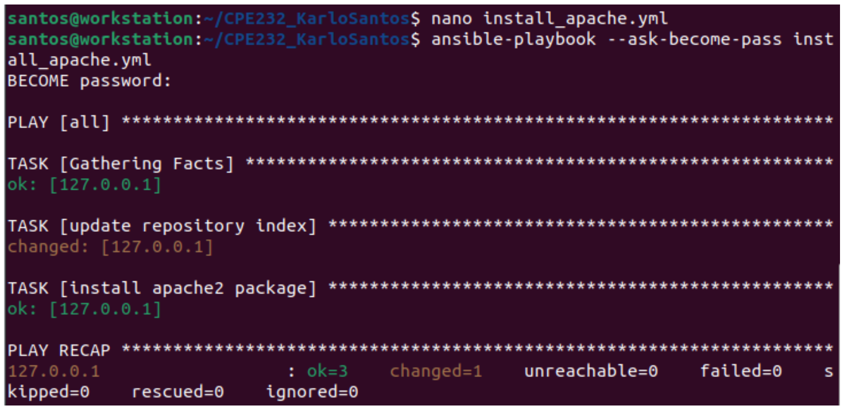
```
---
- hosts: all
  become: true
  tasks:

  - 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                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

                               [ Wrote 12 lines ]
^G Help       ^O Write Out ^W Where Is  ^K Cut       ^T Execute    ^C Location
^X Exit       ^R Read File ^\ Replace   ^U Paste     ^J Justify    ^/ Go To Line
```

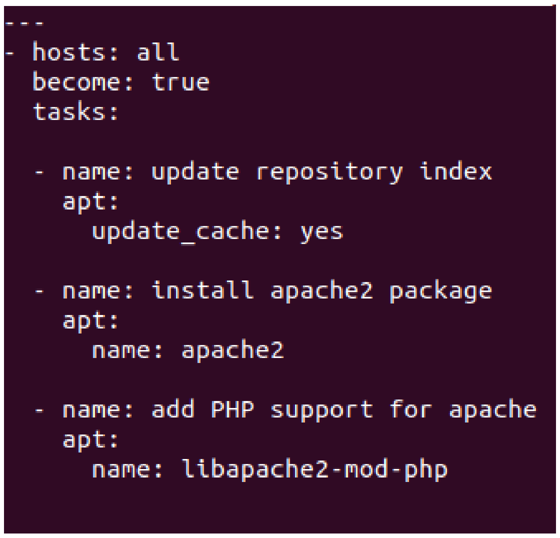
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
santos@workstation:~/CPE232_KarloSantos$ nano install_apache.yml
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass inst
all_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]

PLAY RECAP *********************************************************************
127.0.0.1                  : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

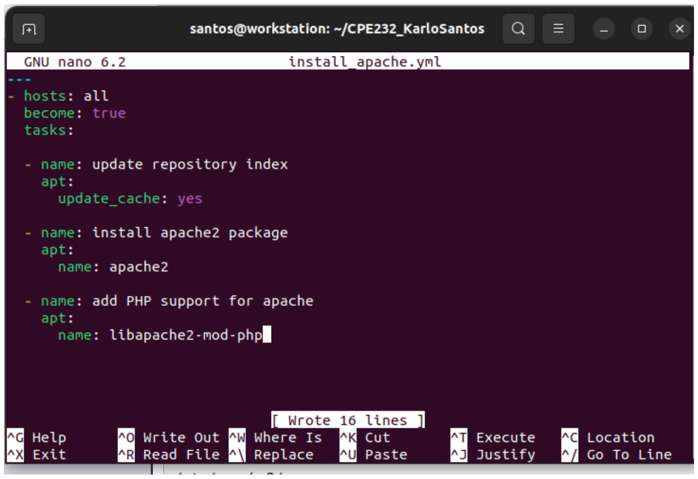
Based on the output, yes the new command changes something in the remote server. We can see that it update the repository and it shows “changes” in the status output.

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



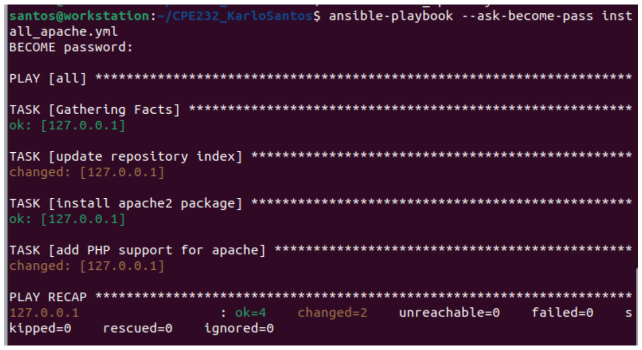
```
---
- hosts: all
  become: true
  tasks:

  - 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.



```
santos@workstation: ~/CPE232_KarloSantos
 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

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

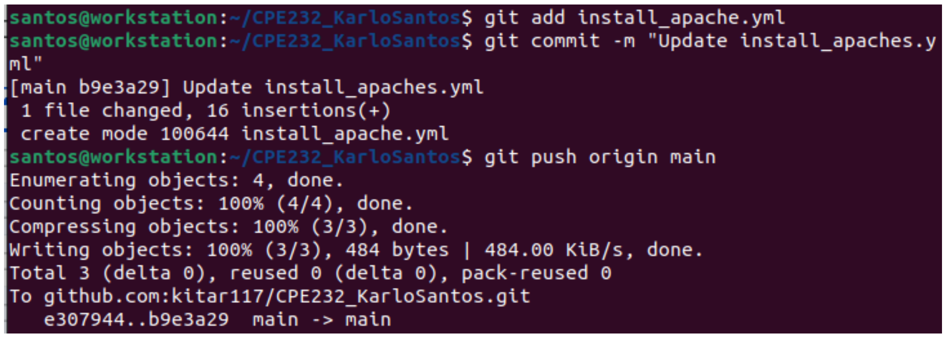
8. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
santos@workstation:~/CPE232_KarloSantos$ ansible-playbook --ask-become-pass inst
all_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    s
kipped=0    rescued=0    ignored=0
```

Yes, the new command changes something in the remote servers. As we can see in the last part it added PHP support in the apache and it shows “changed” in the status output.

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



```
santos@workstation:~/CPE232_KarloSantos$ git add install_apache.yml
santos@workstation:~/CPE232_KarloSantos$ git commit -m "Update install_apaches.y
ml"
[main b9e3a29] Update install_apaches.yml
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
santos@workstation:~/CPE232_KarloSantos$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 484 bytes | 484.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kitar117/CPE232_KarloSantos.git
   e307944..b9e3a29  main -> main
```

https://github.com/kitar117/CPE232_KarloSantos/blob/main/install_apache.yml

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - **It is used in automation of different tasks instead of doing it manually. Playbooks used also in ansible, it is the one that tells what is the task and where it will be executed. It helps in making sure that a certain task will be going to execute only on that specific device. The playbook can also be used not just in automation and configuration, it can be used to program applications and also servers. Like in this activity, it also helps in updating and installing software packages, and also manages the remote server.**

2. Summarize what we have done on this activity.
   - **In this activity, I am able to learn different things like how to use ansible in the system. I learned the introduction about the topic with the help of the discussion where it contains the things about playbook and ansible. Next, with the help of different tasks in this activity, I will be able to learn how to install, update and also upgrade different packages that will help in managing different remote servers. I learn how to use and execute different commands to manage and also make changes in the remote**

machine. After that, it helped me to understand more about playbooks and Ansible and how to use it in automation. Like in this activity, the task is making a playbook that will automate ansible commands. It can be used in making changes like updating and also giving support to the remote machine. Lastly, after doing the different task in this activity, I sync all the changes in the GitHub repository.