Name: Andreu John L. Salvador	**Date Performed: 11/09/2023**
Course/Section: BSCPE31S5	**Date Submitted: 12/09/2023**
Instructor: Engr. Roman Richard	**Semester and SY: 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.

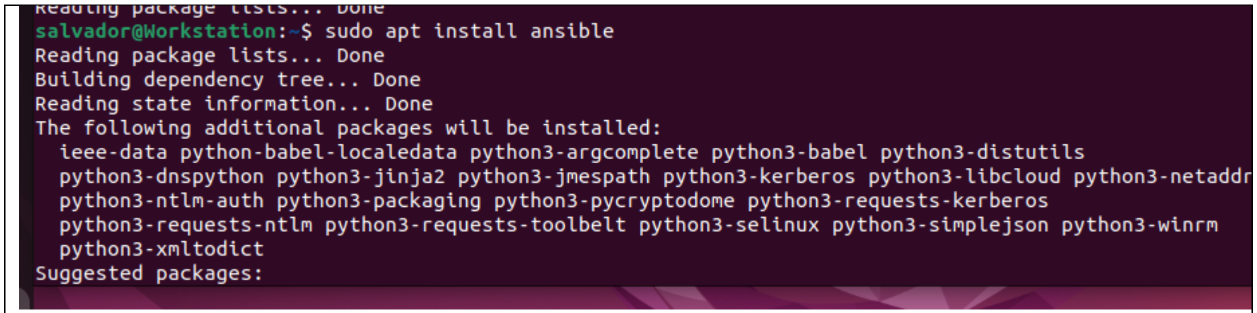
Elevated Ad hoc commands

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:

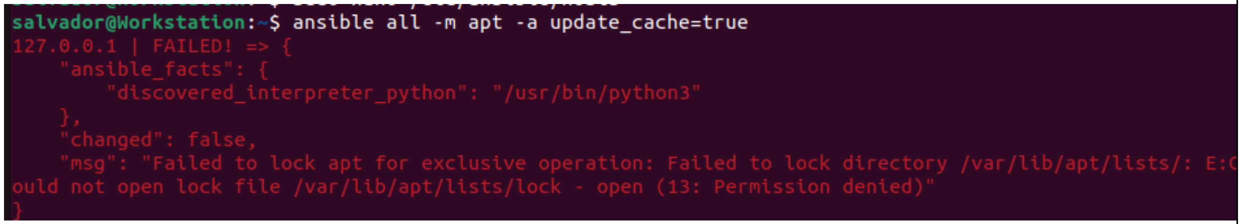


```
Reading package lists... Done
salvador@Workstation:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ieee-data python-babel-localedata python3-argcomplete python3-babel python3-distutils
  python3-dnspython python3-jinja2 python3-jmespath python3-kerberos python3-libcloud python3-netaddr
  python3-ntlm-auth python3-packaging python3-pycryptodome python3-requests-kerberos
  python3-requests-ntlm python3-requests-toolbelt python3-selinux python3-simplejson python3-winrm
  python3-xmltodict
Suggested packages:
```

ansible all -m apt -a update_cache=true

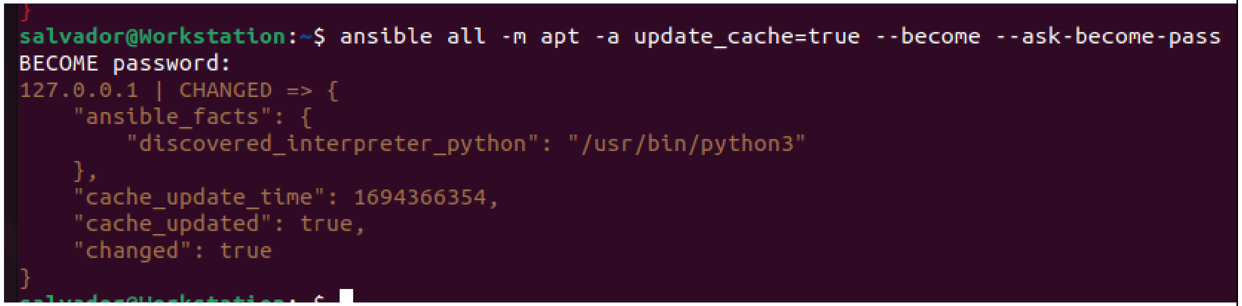
What is the result of the command? Is it successful?

It was unsuccessful



```
salvador@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:C
ould not open lock file /var/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.

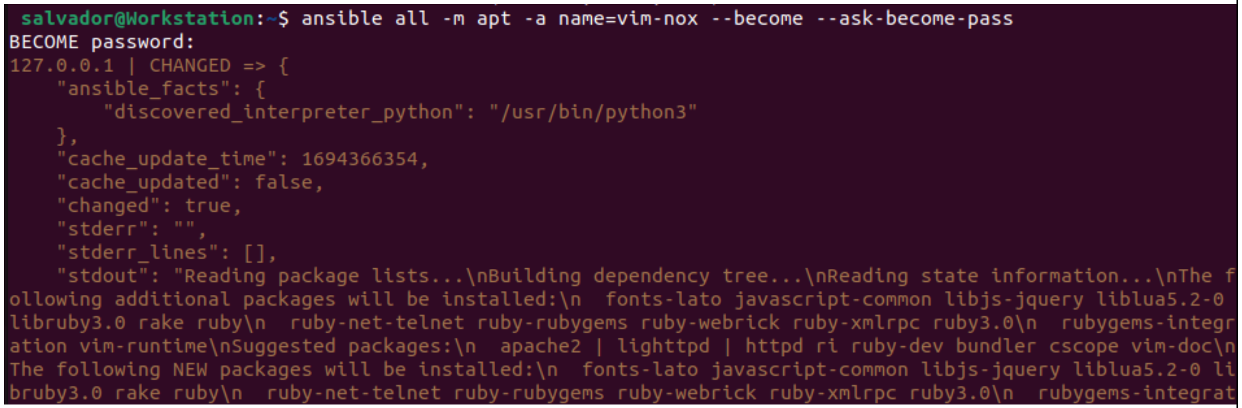


```
}
salvador@Workstation:~$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694366354,
    "cache_updated": true,
    "changed": 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

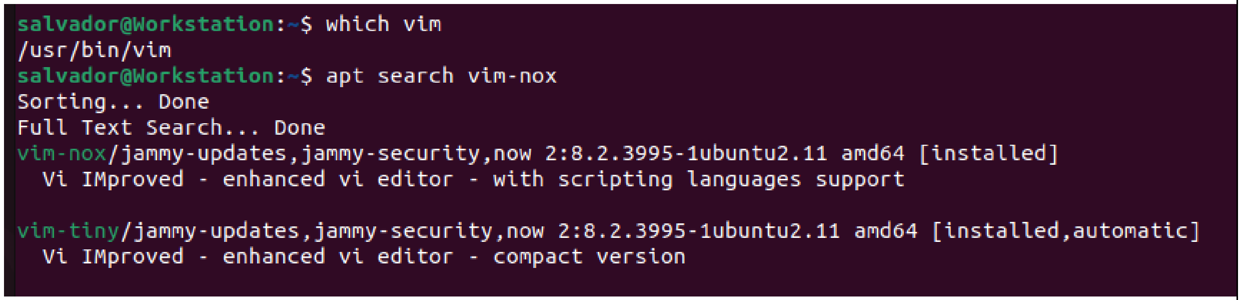
password because the local machine instructed the remote servers to actually install the package.



```
salvador@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": 1694366354,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe f
ollowing additional 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-integr
ation vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd ri ruby-dev bundler cscope vim-doc\n
The following NEW packages will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 li
bruby3.0 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integrat
```

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?

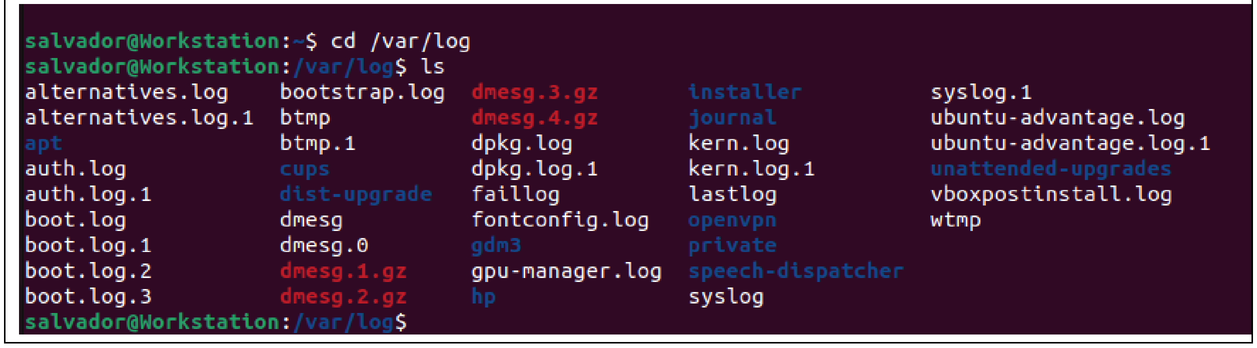
Yes, It was successful



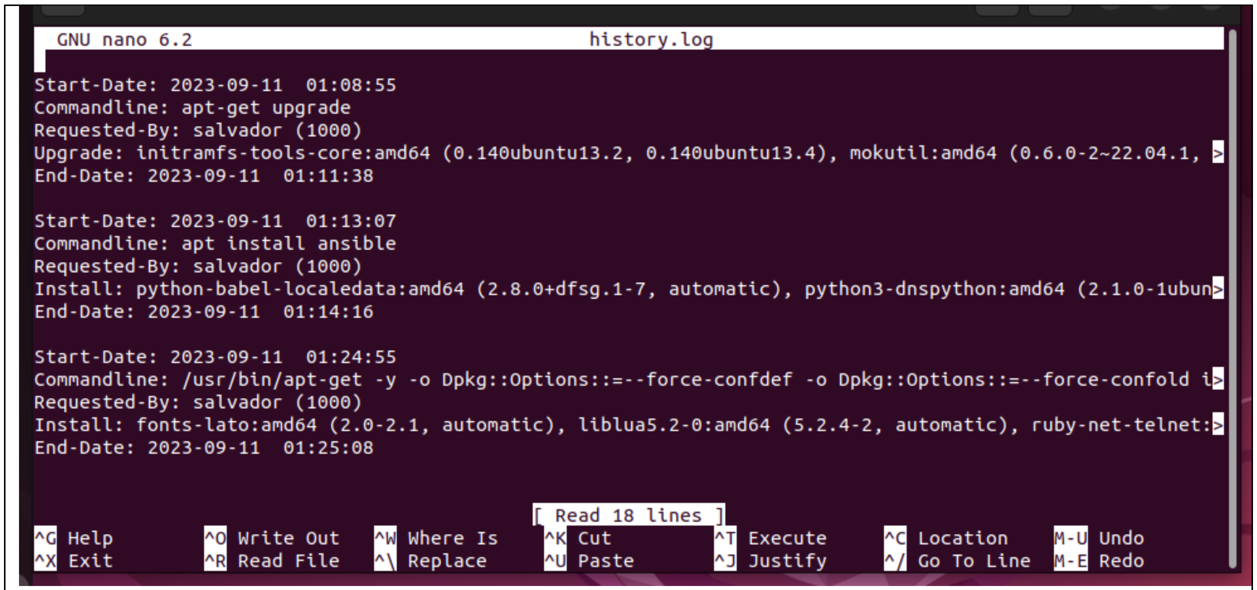
```
salvador@Workstation:~$ which vim
/usr/bin/vim
salvador@Workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,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.

In the Log there are a lot of outcome that came out when issued the command ls. Going to the apt directory and checking the history.log shows the history of what was the early command that I used.



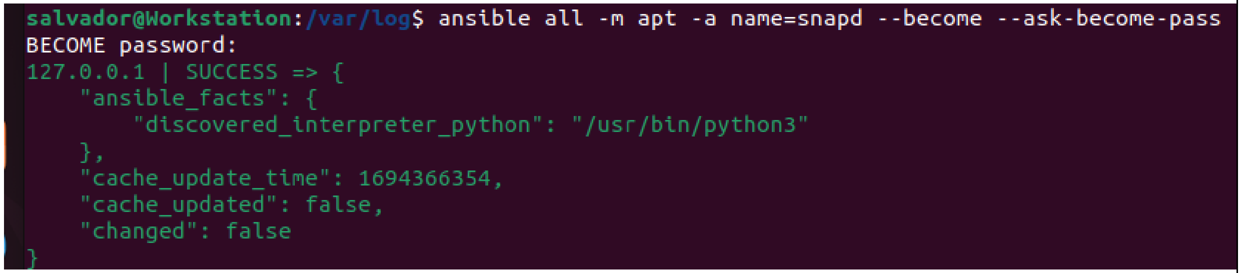
```
salvador@Workstation:~$ cd /var/log
salvador@Workstation:/var/log$ ls
alternatives.log    bootstrap.log  dmesg.3.gz       installer          syslog.1
alternatives.log.1  btmp           dmesg.4.gz       journal            ubuntu-advantage.log
apt                 btmp.1         dpkg.log         kern.log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



```
  GNU nano 6.2                              history.log

Start-Date: 2023-09-11  01:08:55
Commandline: apt-get upgrade
Requested-By: salvador (1000)
Upgrade: initramfs-tools-core:amd64 (0.140ubuntu13.2, 0.140ubuntu13.4), mokutil:amd64 (0.6.0-2~22.04.1, >
End-Date: 2023-09-11  01:11:38

Start-Date: 2023-09-11  01:13:07
Commandline: apt install ansible
Requested-By: salvador (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64 (2.1.0-1ubun>
End-Date: 2023-09-11  01:14:16

Start-Date: 2023-09-11  01:24:55
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold i>
Requested-By: salvador (1000)
Install: fonts-lato:amd64 (2.0-2.1, automatic), liblua5.2-0:amd64 (5.2.4-2, automatic), ruby-net-telnet:>
End-Date: 2023-09-11  01:25:08

                                       [ Read 18 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo
```

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*

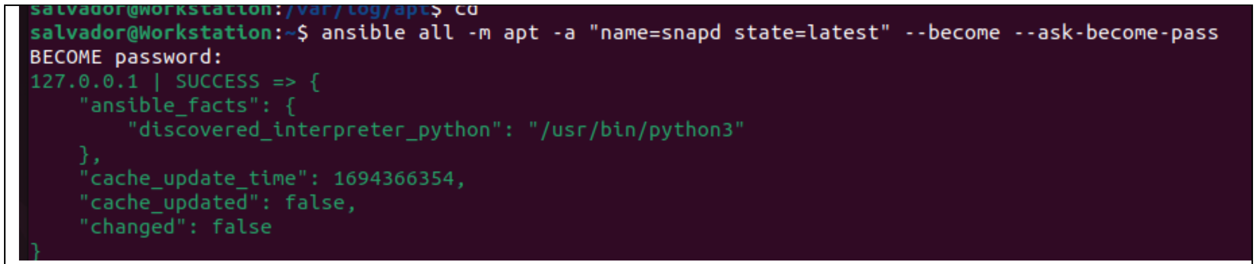


```
salvador@Workstation:/var/log$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694366354,
    "cache_updated": false,
    "changed": false
}
```

   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?

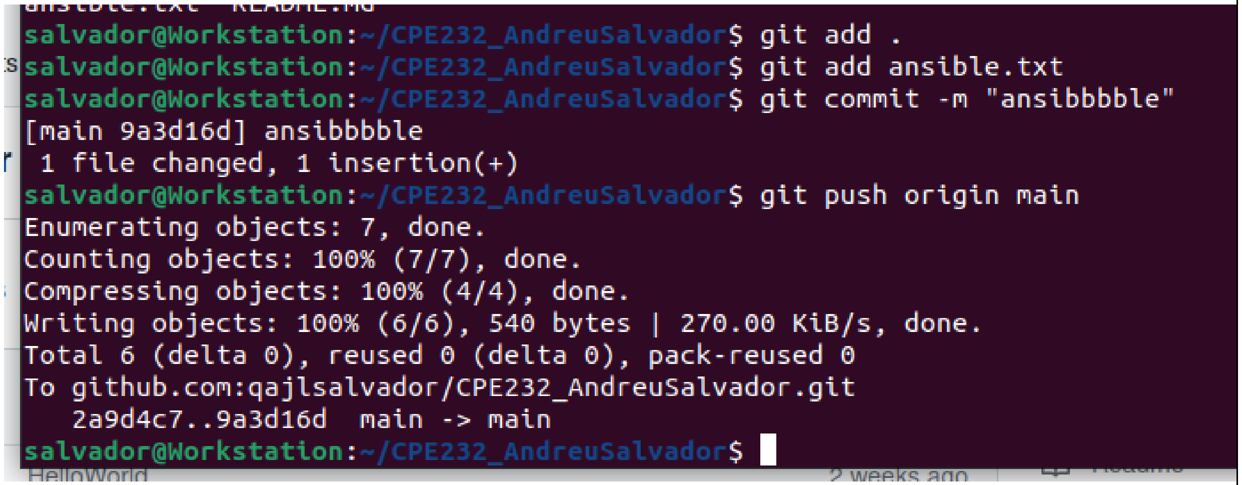
It was a success, Seeing a success in the report and not CHANGED means the package that was supposed to be installed were already installed in the OS therefore checking whether the Package is already installed was a success.

   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.

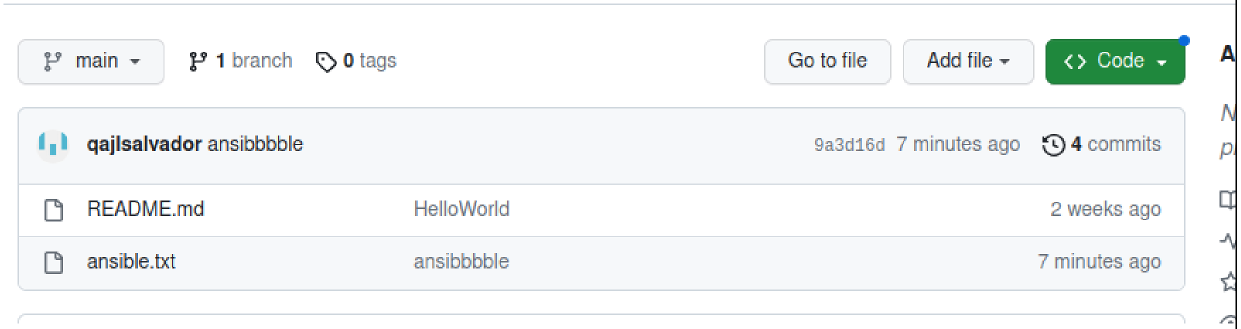


```
salvador@Workstation:/var/log/apt$ cd
salvador@Workstation:~$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694366354,
    "cache_updated": false,
    "changed": false
}
```

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



```
salvador@Workstation:~/CPE232_AndreuSalvador$ git add .
salvador@Workstation:~/CPE232_AndreuSalvador$ git add ansible.txt
salvador@Workstation:~/CPE232_AndreuSalvador$ git commit -m "ansibbbble"
[main 9a3d16d] ansibbbble
 1 file changed, 1 insertion(+)
salvador@Workstation:~/CPE232_AndreuSalvador$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 540 bytes | 270.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qajlsalvador/CPE232_AndreuSalvador.git
   2a9d4c7..9a3d16d  main -> main
salvador@Workstation:~/CPE232_AndreuSalvador$
```



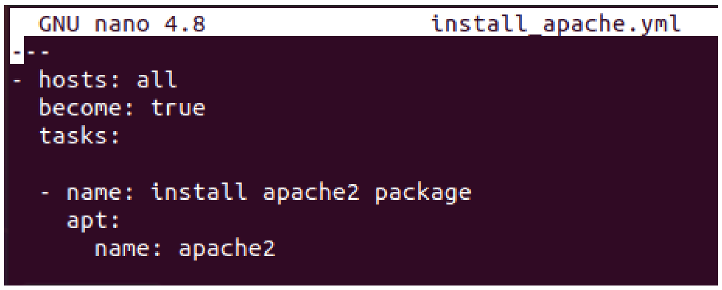
main · 1 branch · 0 tags · Go to file · Add file · Code

qajlsalvador ansibbbble		9a3d16d 7 minutes ago · 4 commits
README.md	HelloWorld	2 weeks ago
ansible.txt	ansibbbble	7 minutes ago

Task 2: Writing our First Playbook

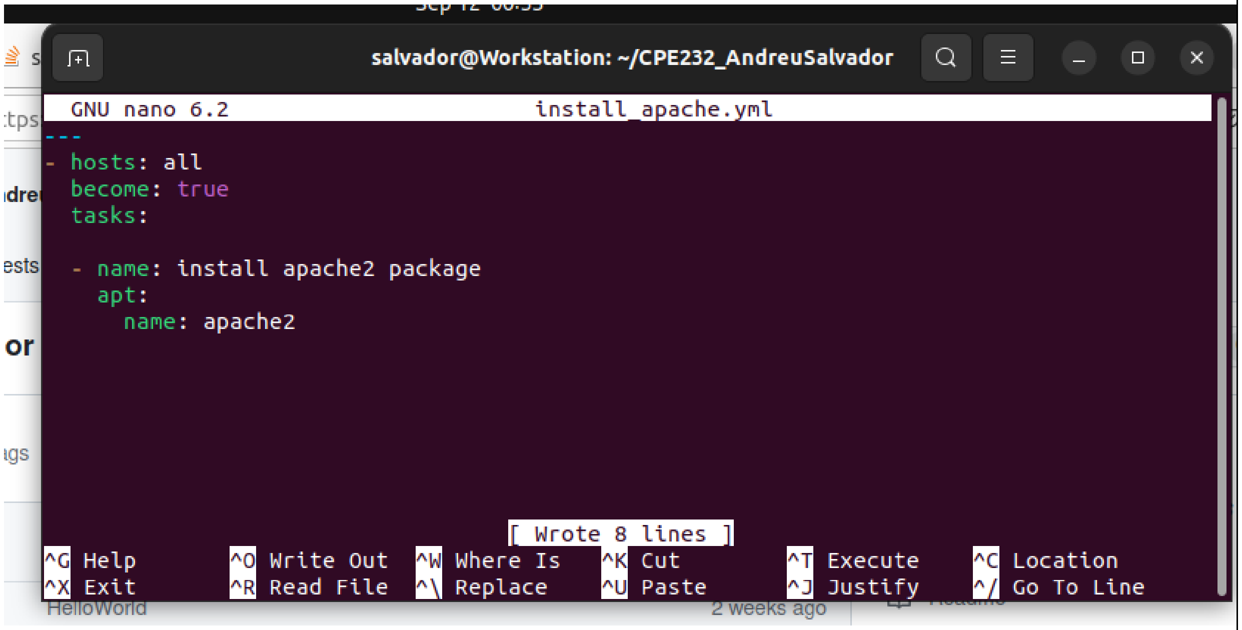
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and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

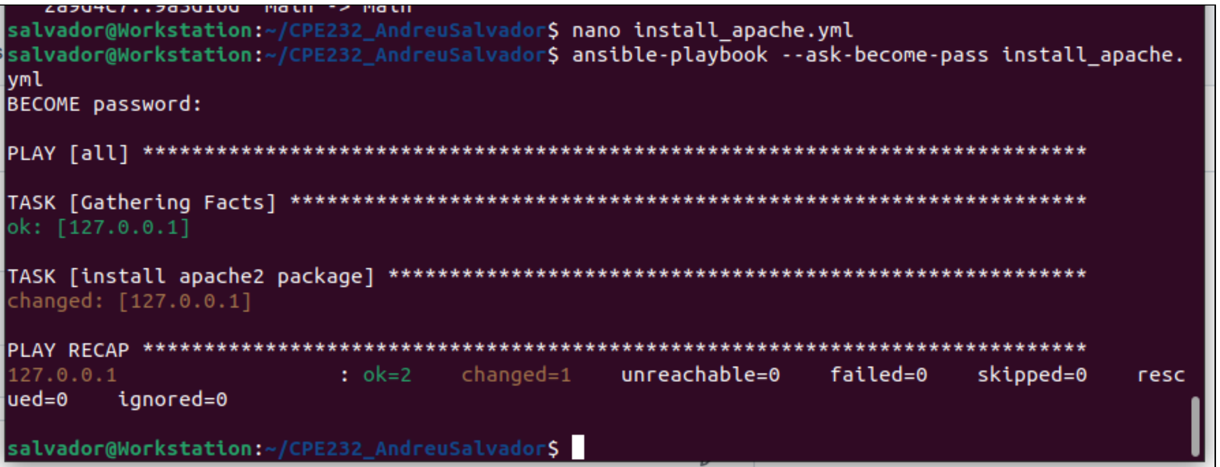
Make sure to save the file. Take note also of the alignments of the texts.



salvador@Workstation: ~/CPE232_AndreuSalvador

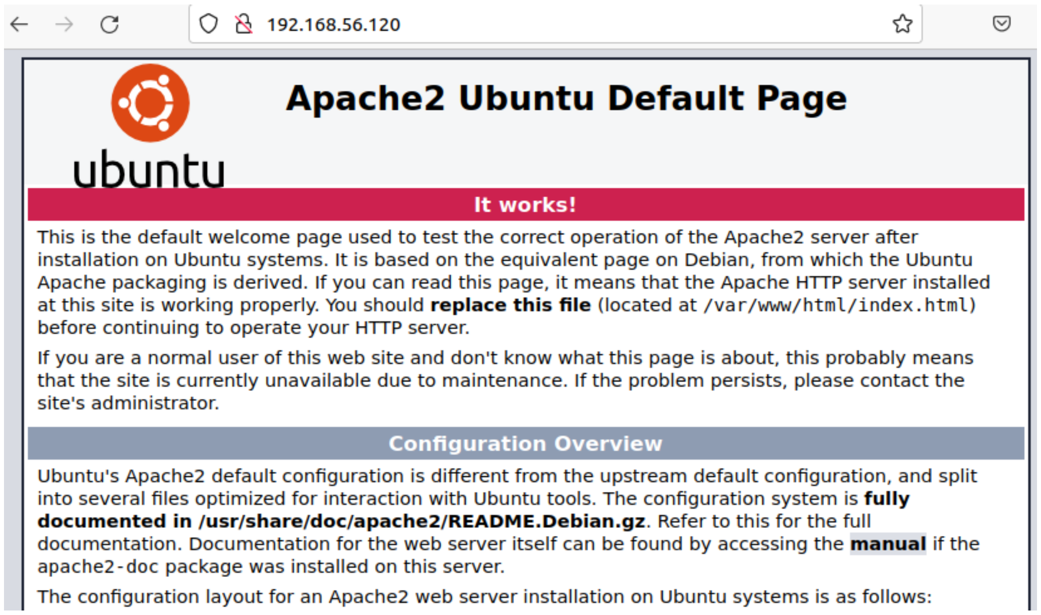
```
  GNU nano 6.2                            install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2







                                [ Wrote 8 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
```

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



```
salvador@Workstation:~/CPE232_AndreuSalvador$ nano install_apache.yml
salvador@Workstation:~/CPE232_AndreuSalvador$ 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    resc
ued=0    ignored=0

salvador@Workstation:~/CPE232_AndreuSalvador$
```

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!

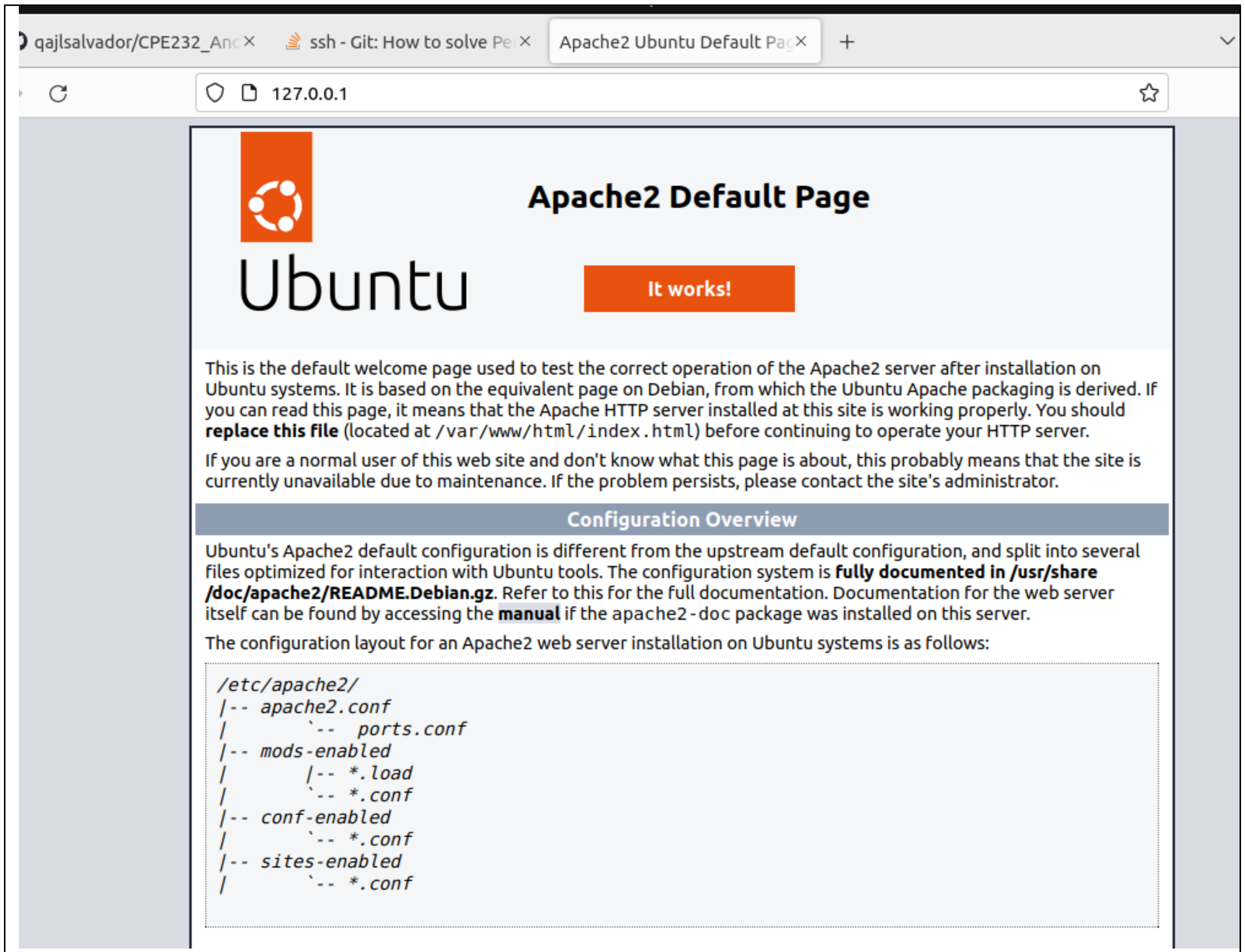
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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

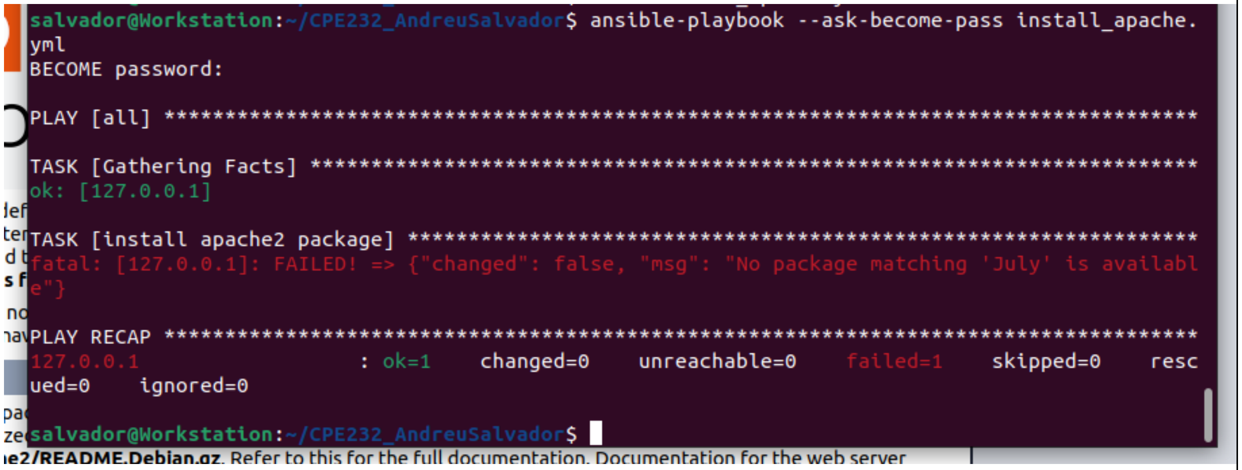
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share /doc/apache2/README.Debian.gz**. Refer to this for the full documentation.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

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?

As expected, the file read wrong because of wrong indication of the name therefore the action with become false or will deem to fail.



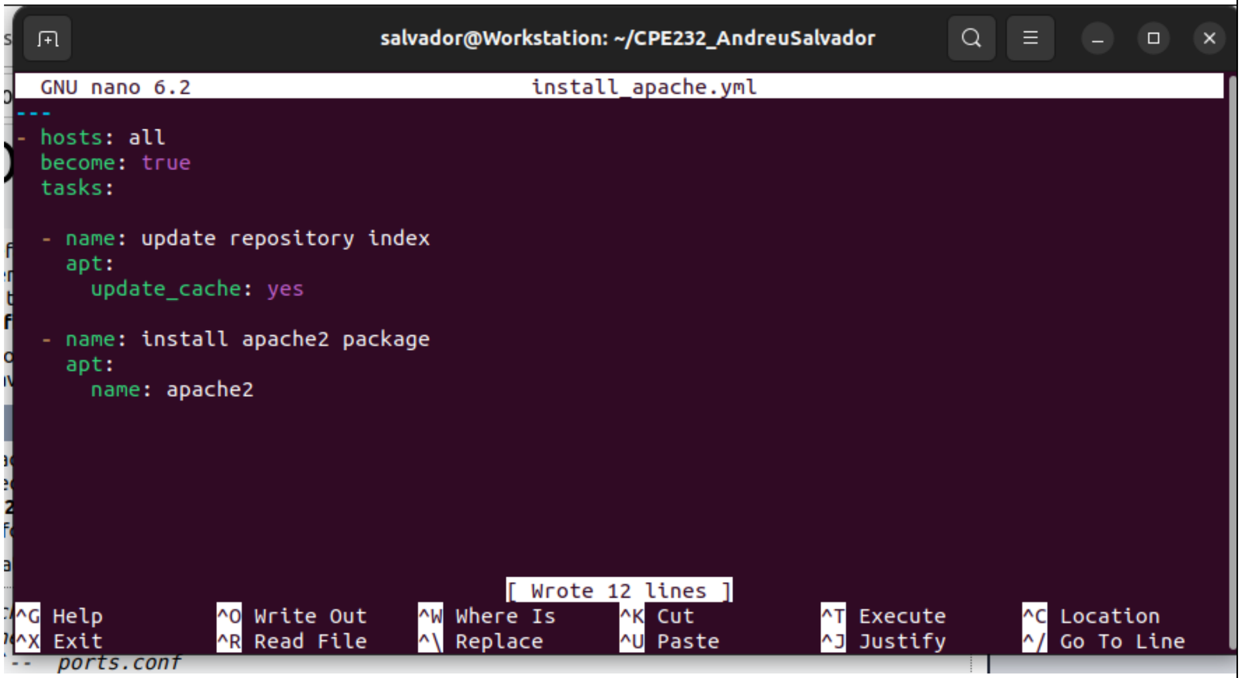
```
salvador@Workstation:~/CPE232_AndreuSalvador$ ansible-playbook --ask-become-pass install_apache.
yml
BECOME password:

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 'July' is availabl
e"}

PLAY RECAP *********************************************************************************
127.0.0.1                  : ok=1    changed=0    unreachable=0    failed=1    skipped=0    resc
ued=0    ignored=0

salvador@Workstation:~/CPE232_AndreuSalvador$
```

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
```

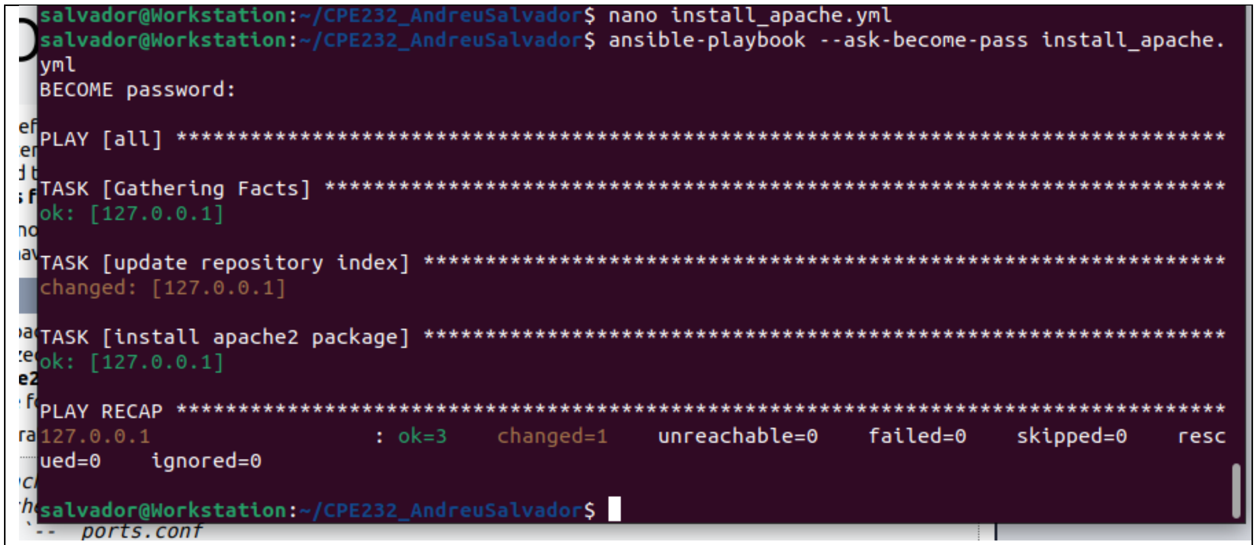
Save the changes to this file and exit.



```
salvador@Workstation: ~/CPE232_AndreuSalvador
 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
```

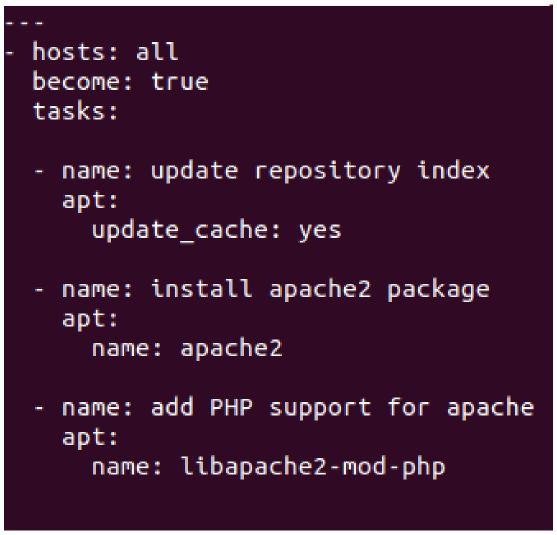
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?

Yes, there is changed=1 in the report after running the ansible again.



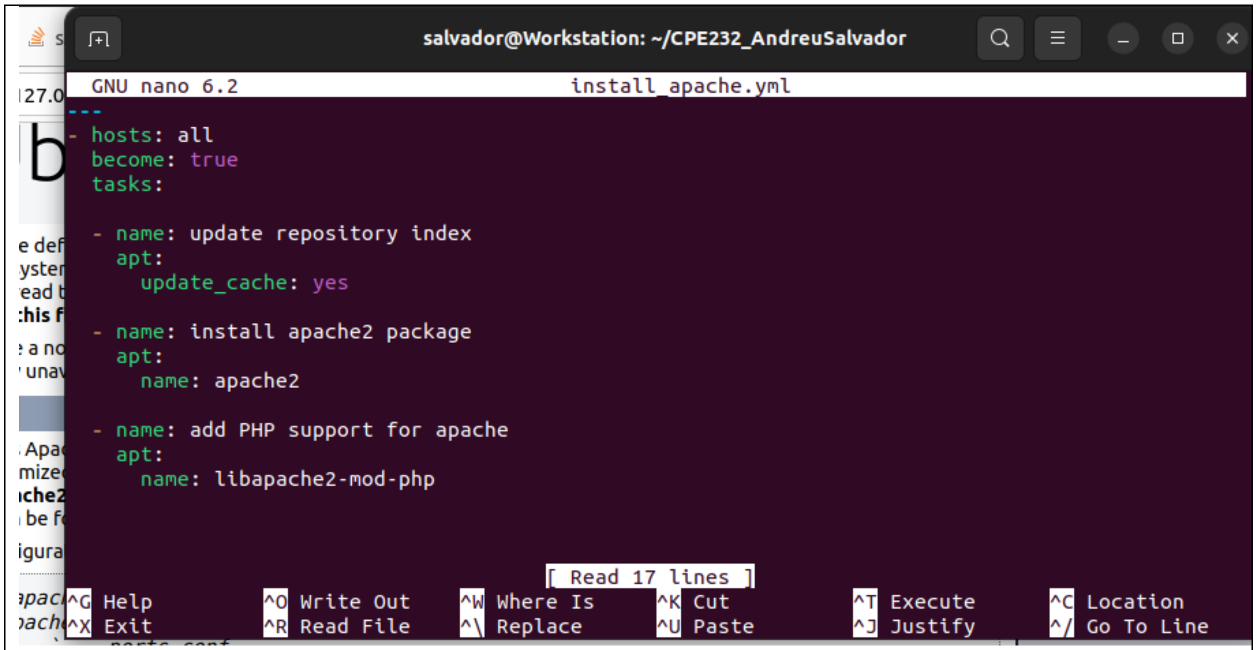
```
salvador@Workstation:~/CPE232_AndreuSalvador$ nano install_apache.yml
salvador@Workstation:~/CPE232_AndreuSalvador$ ansible-playbook --ask-become-pass install_apache.
yml
BECOME password:

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]

PLAY RECAP ******************************************************************************
127.0.0.1                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0

salvador@Workstation:~/CPE232_AndreuSalvador$ 
```

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



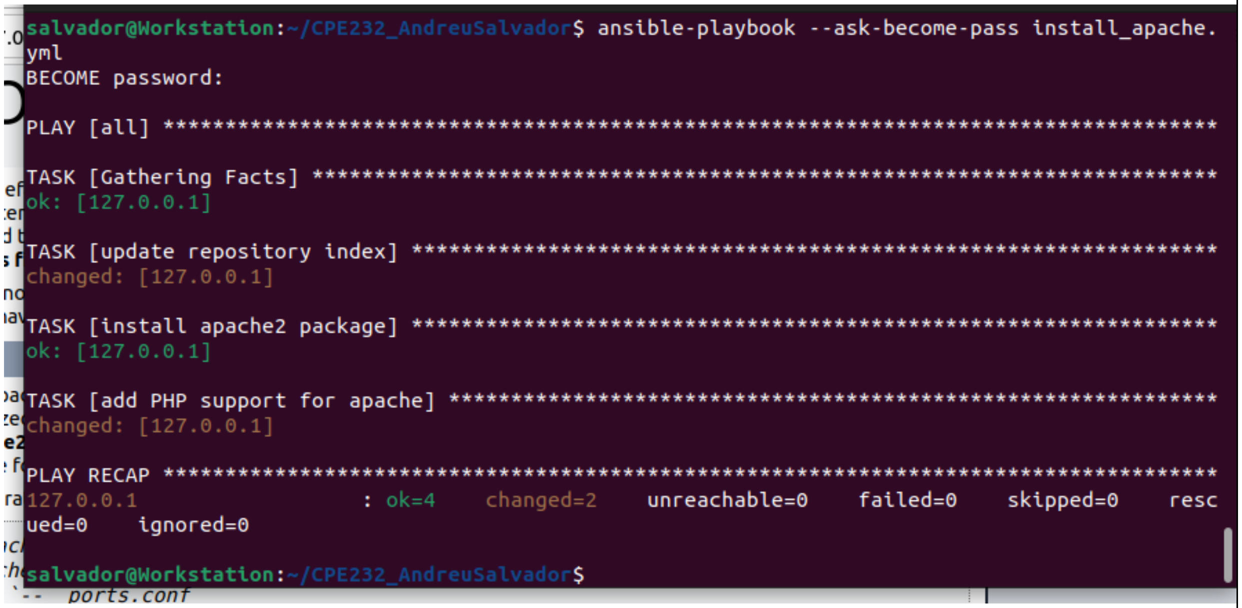
```
---
- hosts: all
  become: true
  tasks:

  - 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.



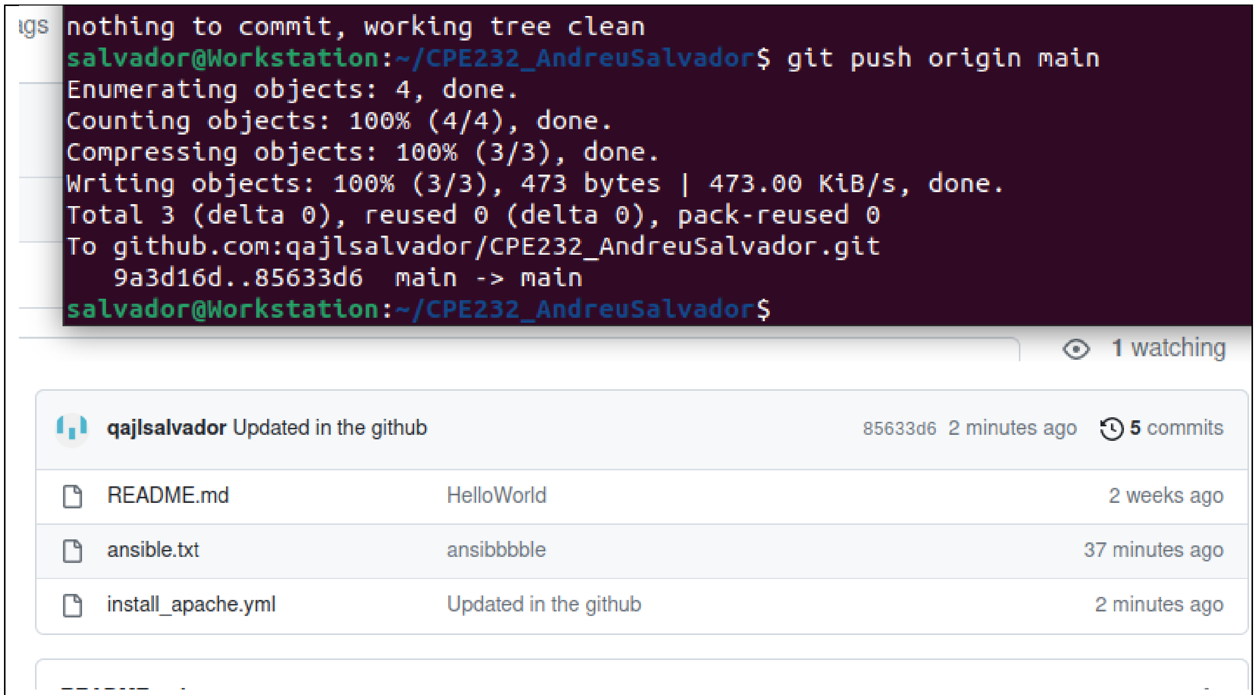
```
salvador@Workstation: ~/CPE232_AndreuSalvador
 GNU nano 6.2                          install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - 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 17 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?



```
salvador@Workstation:~/CPE232_AndreuSalvador$ 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    resc
ued=0    ignored=0

salvador@Workstation:~/CPE232_AndreuSalvador$
```

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



```
nothing to commit, working tree clean
salvador@Workstation:~/CPE232_AndreuSalvador$ 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), 473 bytes | 473.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:qajlsalvador/CPE232_AndreuSalvador.git
   9a3d16d..85633d6  main -> main
salvador@Workstation:~/CPE232_AndreuSalvador$
```

qajlsalvador Updated in the github 85633d6 2 minutes ago 5 commits

README.md	HelloWorld	2 weeks ago
ansible.txt	ansibbbble	37 minutes ago
install_apache.yml	Updated in the github	2 minutes ago

https://github.com/qajlsalvador/CPE232_AndreuSalvador/blob/main/install_apache.yml

Reflections:

Answer the following:

1. What is the importance of using a playbook?

Ansible uses playbook in order to automate things, meaning using a playbook you can automatically run a multiple code processes with a single code that you can call. Inside the playbook you can store a lot of codes to run, whether to install something or update something. With this your programming and work will be lessen because of the automation that the playbook offers. You can also use this in order to check or run a code that you will use often, that's why it's called an automation it automatically run the code that you store inside the playbook and with less time to consume.

2. Summarize what we have done on this activity.

At the beginning of the activity, we were tasked to install ansible inside the Workstation of Ubuntu. After that we've done the implementation of the ansible by creating a playbook that will produce our automation, but before that we've created a github repository in order to check and monitor the changes that we will be doing by the use of the playbook. Lastly we have demonstrated the creation of the playbook and showed how the automation occur by adding codes inside the playbook. Run after run, we inserted new codes inside of it for us to determine whether we really did the proper procedure in making the playbook.