Name: Pastrana, Mark Laurenz S.	**Date Performed: Sep 12, 2023**
Course/Section: CPE31S5-CPE 232	**Date Submitted: Sep 12, 2023**
Instructor: Engr. Richard Roman	**Semester and SY: 1st, 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

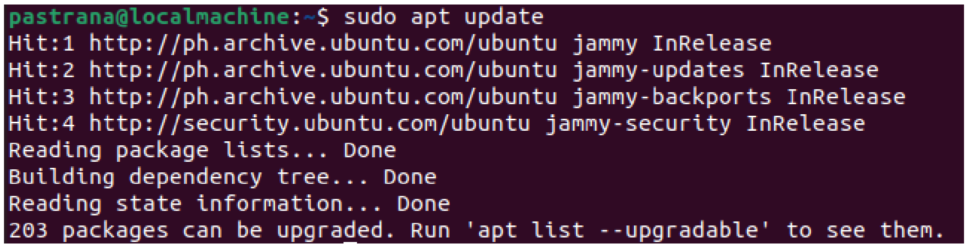
So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages 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 are often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run the 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

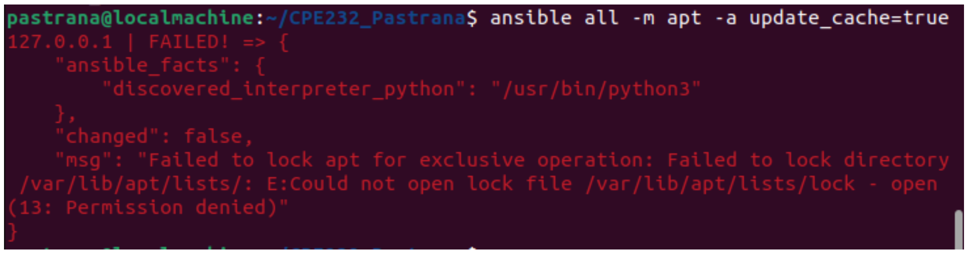
an apt update command in a remote machine. Issue the following command:



```
pastrana@localmachine:~$ sudo apt update
Hit:1 http://ph.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ph.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ph.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
203 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

ansible all -m apt -a update_cache=true

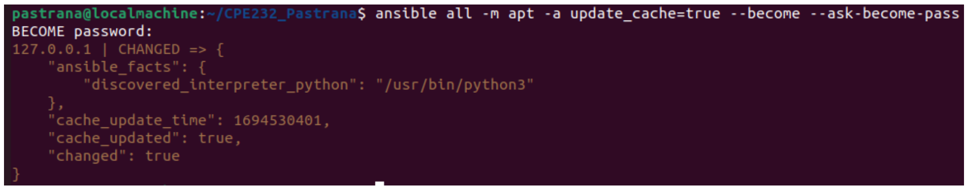
What is the result of the command? Is it successful?



```
pastrana@localmachine:~/CPE232_Pastrana$ ansible all -m apt -a update_cache=true
127.0.0.1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory
 /var/lib/apt/lists/: E:Could not open lock file /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 elevates the privileges and the *--ask-become-pass* asks for the password. For now, even if we only changed the packaged index, we were able to change something on the remote server.

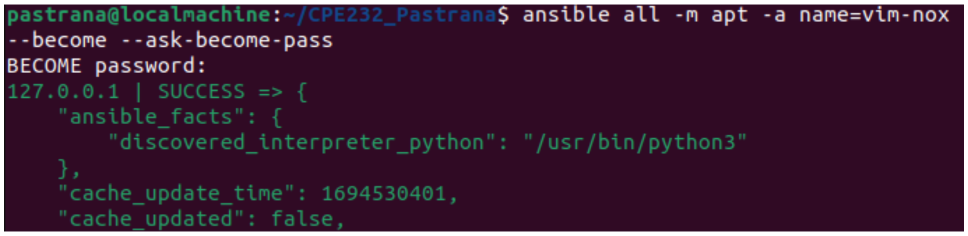
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.



```
pastrana@localmachine:~/CPE232_Pastrana$ 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": 1694530401,
    "cache_updated": true,
    "changed": true
}
```

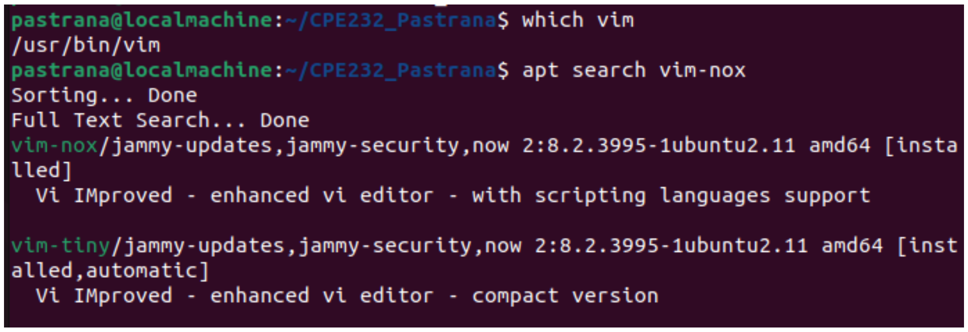
2. Let’s try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just change the module part in 1.1 instruction.

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. `



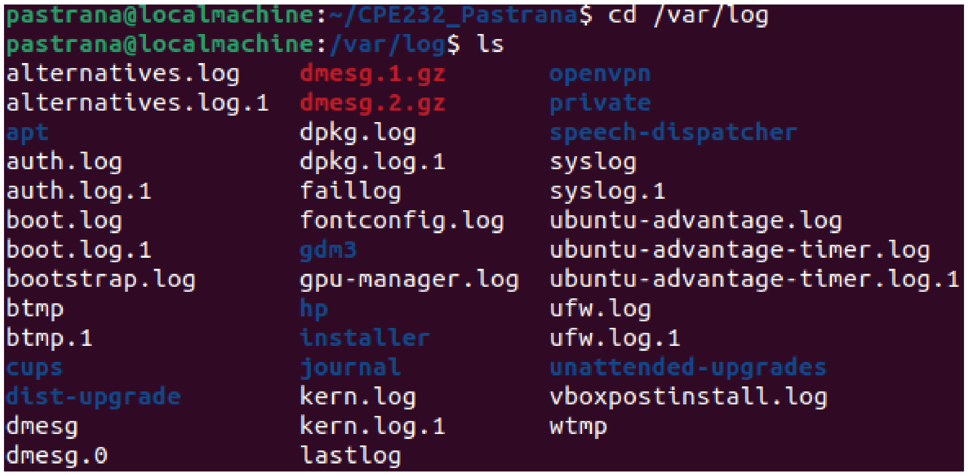
```
pastrana@localmachine:~/CPE232_Pastrana$ ansible all -m apt -a name=vim-nox
--become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694530401,
    "cache_updated": false,
```

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?

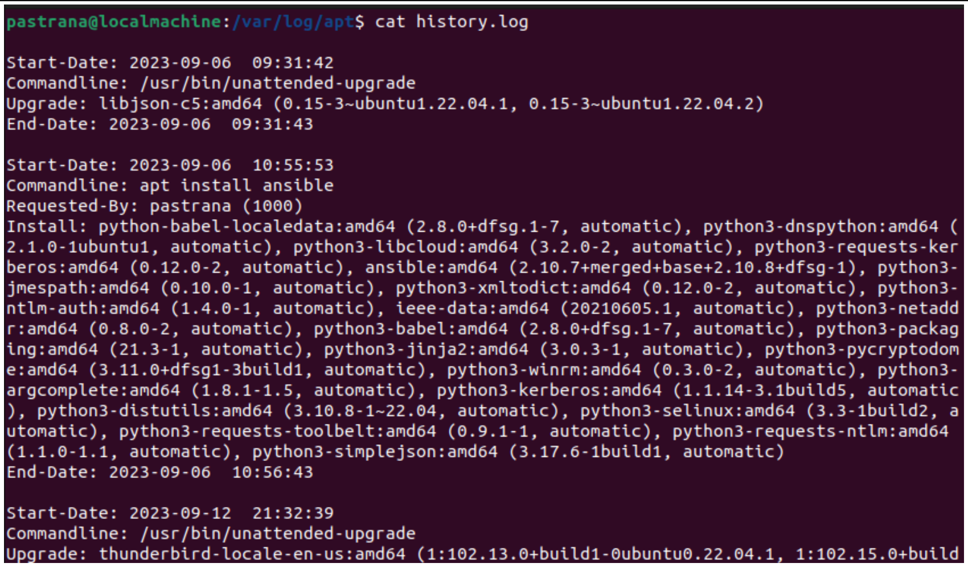


```
pastrana@localmachine:~/CPE232_Pastrana$ which vim
/usr/bin/vim
pastrana@localmachine:~/CPE232_Pastrana$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [insta
lled]
  Vi IMproved - enhanced vi editor - with scripting languages support

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

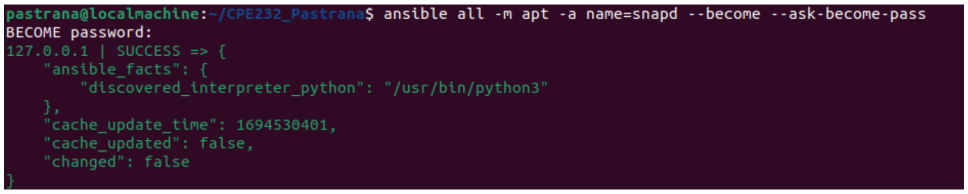


```
pastrana@localmachine:~/CPE232_Pastrana$ cd /var/log
pastrana@localmachine:/var/log$ ls
alternatives.log     dmesg.1.gz       openvpn
alternatives.log.1   dmesg.2.gz       private
apt                  dpkg.log         speech-dispatcher
auth.log             dpkg.log.1       syslog
auth.log.1           faillog          syslog.1
boot.log             fontconfig.log   ubuntu-advantage.log
boot.log.1           gdm3             ubuntu-advantage-timer.log
bootstrap.log        gpu-manager.log  ubuntu-advantage-timer.log.1
btmp                 hp               ufw.log
btmp.1               installer        ufw.log.1
cups                 journal          unattended-upgrades
dist-upgrade         kern.log         vboxpostinstall.log
dmesg                kern.log.1       wtmp
dmesg.0              lastlog
```



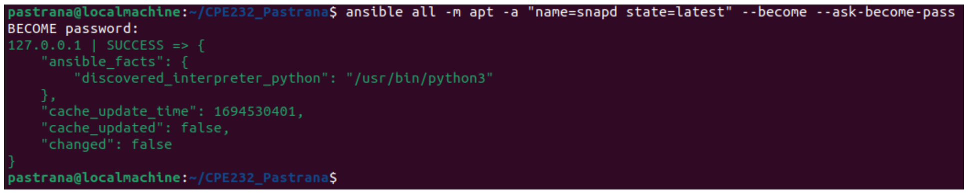
```
pastrana@localmachine:/var/log/apt$ cat history.log

Start-Date: 2023-09-06  09:31:42
Commandline: /usr/bin/unattended-upgrade
Upgrade: libjson-c5:amd64 (0.15-3~ubuntu1.22.04.1, 0.15-3~ubuntu1.22.04.2)
End-Date: 2023-09-06  09:31:43

Start-Date: 2023-09-06  10:55:53
Commandline: apt install ansible
Requested-By: pastrana (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64 (
2.1.0-1ubuntu1, automatic), python3-libcloud:amd64 (3.2.0-2, automatic), python3-requests-ker
beros:amd64 (0.12.0-2, automatic), ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1), python3-
jmespath:amd64 (0.10.0-1, automatic), python3-xmltodict:amd64 (0.12.0-2, automatic), python3-
ntlm-auth:amd64 (1.4.0-1, automatic), ieee-data:amd64 (20210605.1, automatic), python3-netadd
r:amd64 (0.8.0-2, automatic), python3-babel:amd64 (2.8.0+dfsg.1-7, automatic), python3-packag
ing:amd64 (21.3-1, automatic), python3-jinja2:amd64 (3.0.3-1, automatic), python3-pycryptodom
e:amd64 (3.11.0+dfsg1-3build1, automatic), python3-winrm:amd64 (0.3.0-2, automatic), python3-
argcomplete:amd64 (1.8.1-1.5, automatic), python3-kerberos:amd64 (1.1.14-3.1build5, automatic
), python3-distutils:amd64 (3.10.8-1~22.04, automatic), python3-selinux:amd64 (3.3-1build2, a
utomatic), python3-requests-toolbelt:amd64 (0.9.1-1, automatic), python3-requests-ntlm:amd64
(1.1.0-1.1, automatic), python3-simplejson:amd64 (3.17.6-1build1, automatic)
End-Date: 2023-09-06  10:56:43

Start-Date: 2023-09-12  21:32:39
Commandline: /usr/bin/unattended-upgrade
Upgrade: thunderbird-locale-en-us:amd64 (1:102.13.0+build1-0ubuntu0.22.04.1, 1:102.15.0+build
```

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?



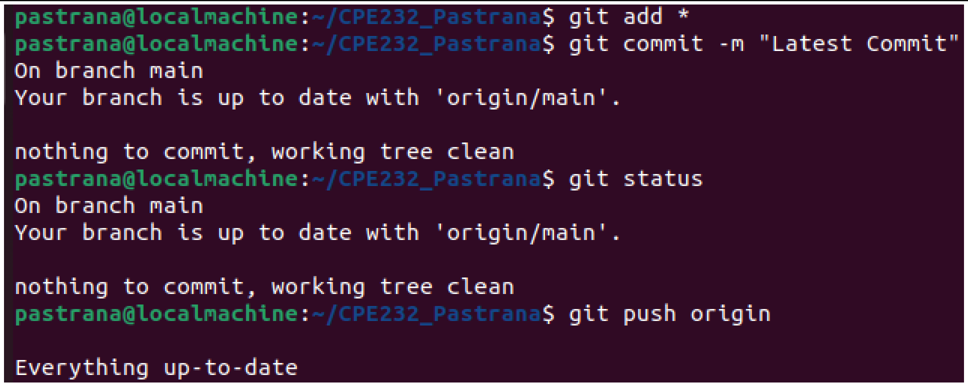
```
pastrana@localmachine:~/CPE232_Pastrana$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694530401,
    "cache_updated": false,
    "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.



```
pastrana@localmachine:~/CPE232_Pastrana$ 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": 1694530401,
    "cache_updated": false,
    "changed": false
}
pastrana@localmachine:~/CPE232_Pastrana$
```

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

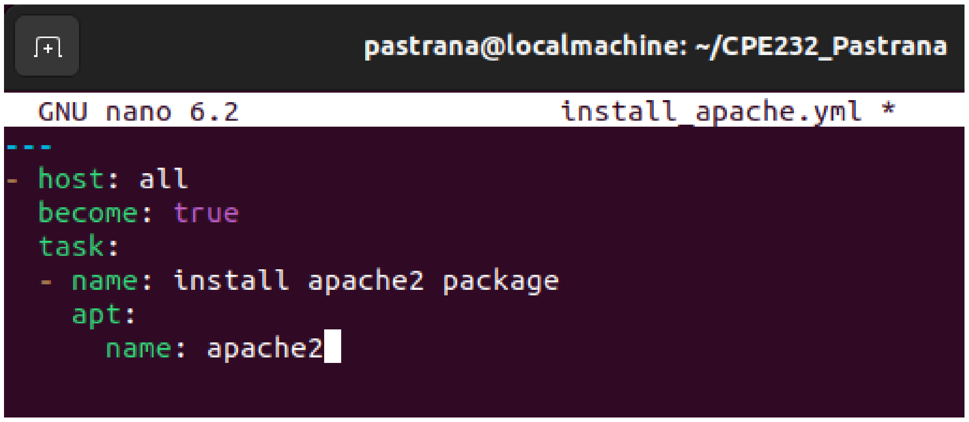


```
pastrana@localmachine:~/CPE232_Pastrana$ git add *
pastrana@localmachine:~/CPE232_Pastrana$ git commit -m "Latest Commit"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
pastrana@localmachine:~/CPE232_Pastrana$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
pastrana@localmachine:~/CPE232_Pastrana$ git push origin

Everything up-to-date
```

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 used 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.

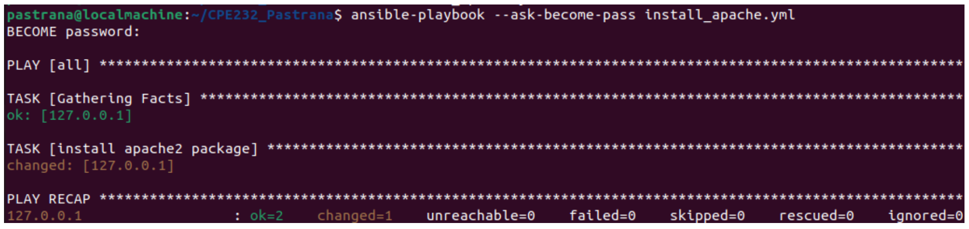
   When the editor appears, type the following:



```
pastrana@localmachine: ~/CPE232_Pastrana
  GNU nano 6.2                        install_apache.yml *
---
- host: all
  become: true
  task:
  - name: install apache2 package
    apt:
      name: apache2
```

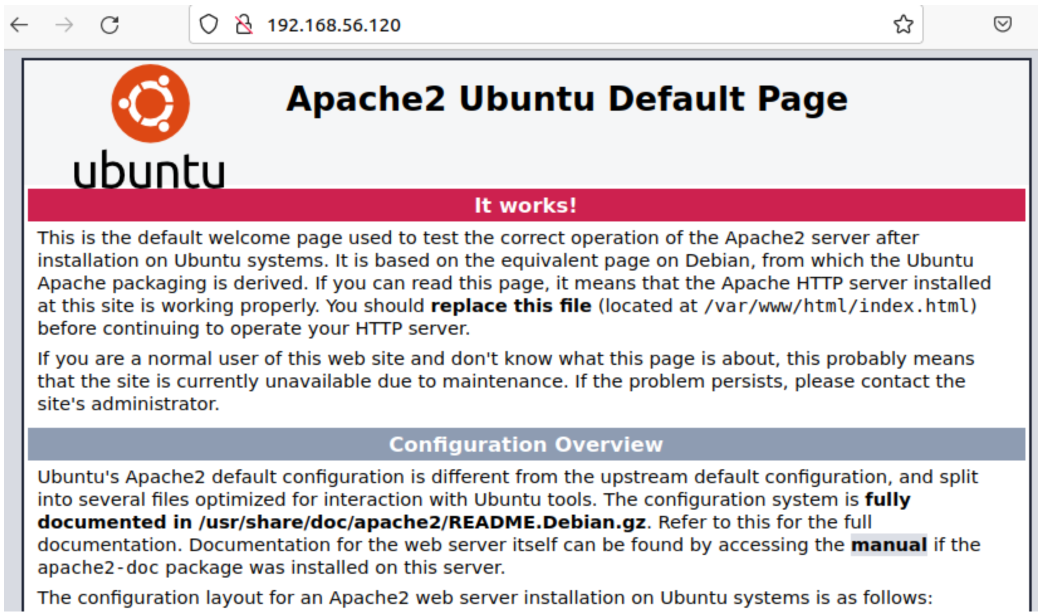
   Make sure to save the file. Take note also of the alignments of the texts.

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



```
pastrana@localmachine:~/CPE232_Pastrana$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

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

3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.



192.168.56.120

Apache2 Ubuntu Default Page

It works!

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

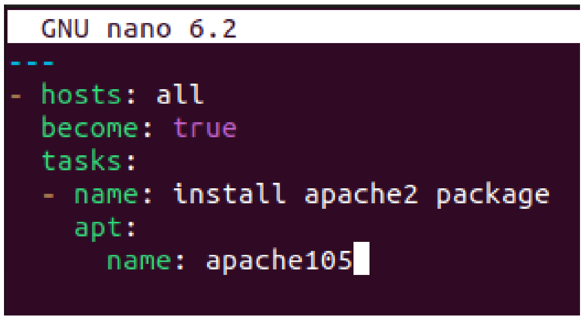
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

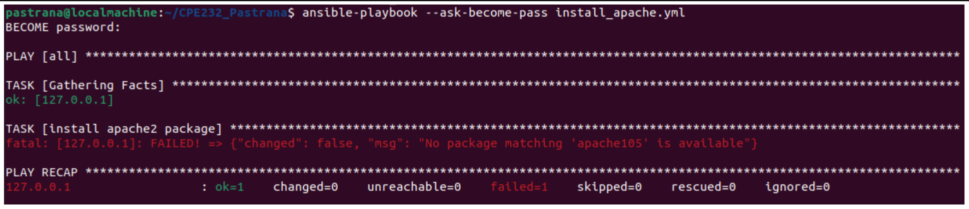
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation.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

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

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

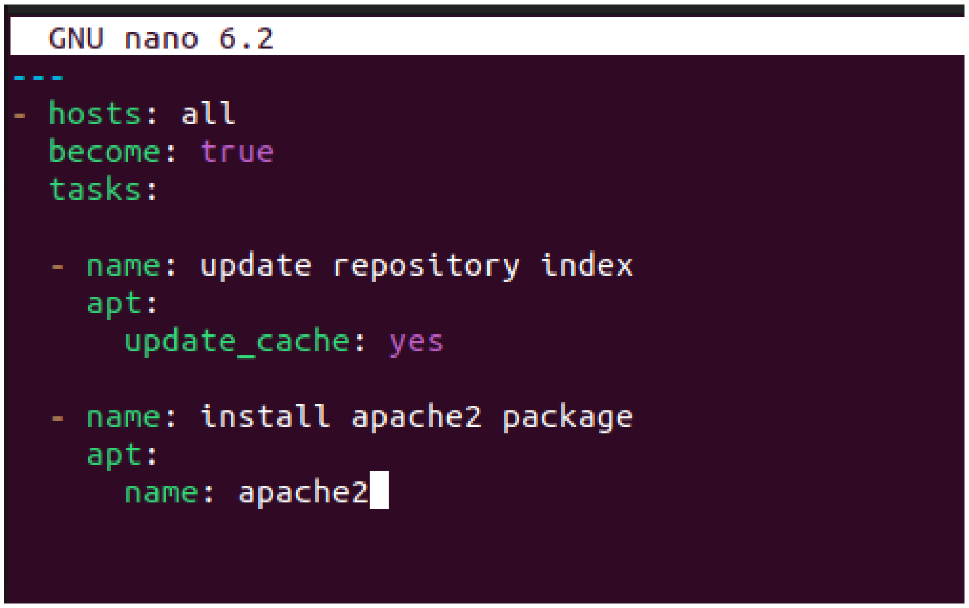


```
 GNU nano 6.2
---
- hosts: all
  become: true
  tasks:
  - name: install apache2 package
    apt:
      name: apache105
```



```
pastrana@localmachine:~/CPE232_Pastrana$ 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 'apache105' is available"}

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

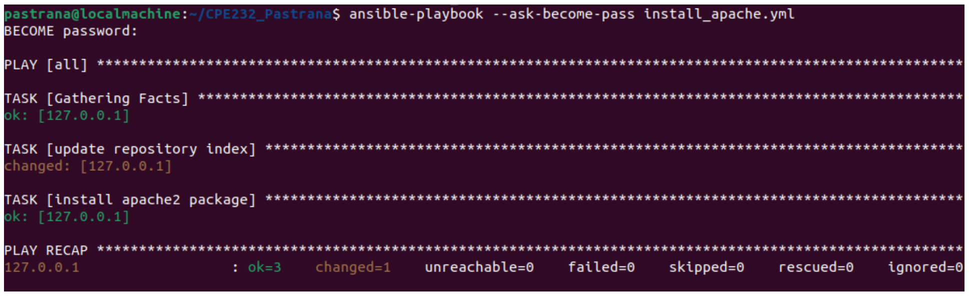
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.



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

  - name: update repository index
    apt:
      update_cache: yes

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

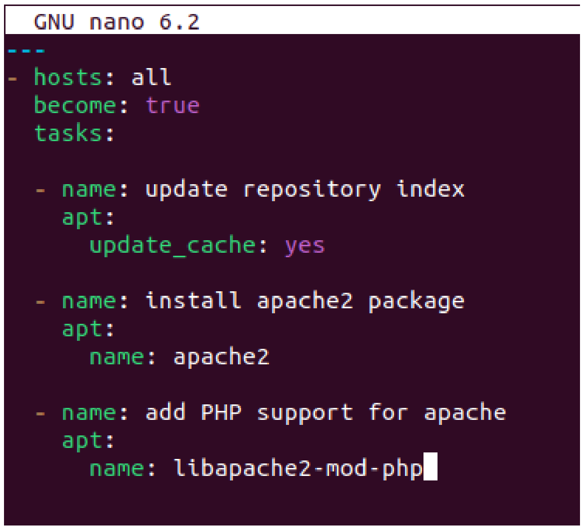
Save the changes to this file and exit.

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



```
pastrana@localmachine:~/CPE232_Pastrana$ 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    rescued=0    ignored=0
```

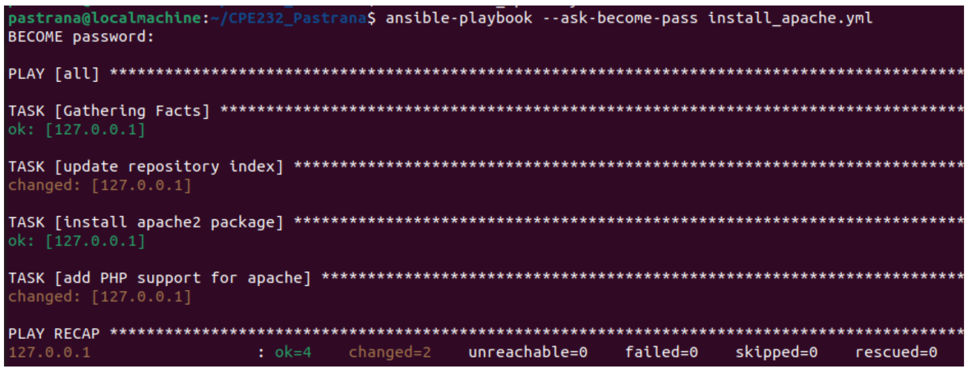
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
  GNU nano 6.2
---
- 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
```

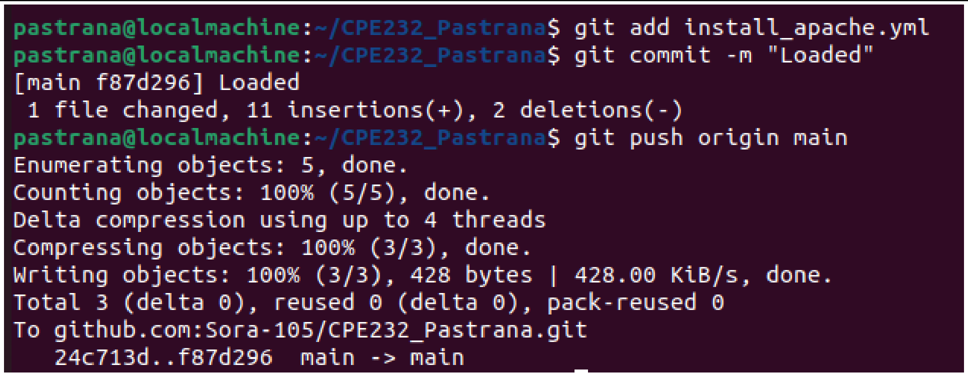
Save the changes to this file and exit.

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

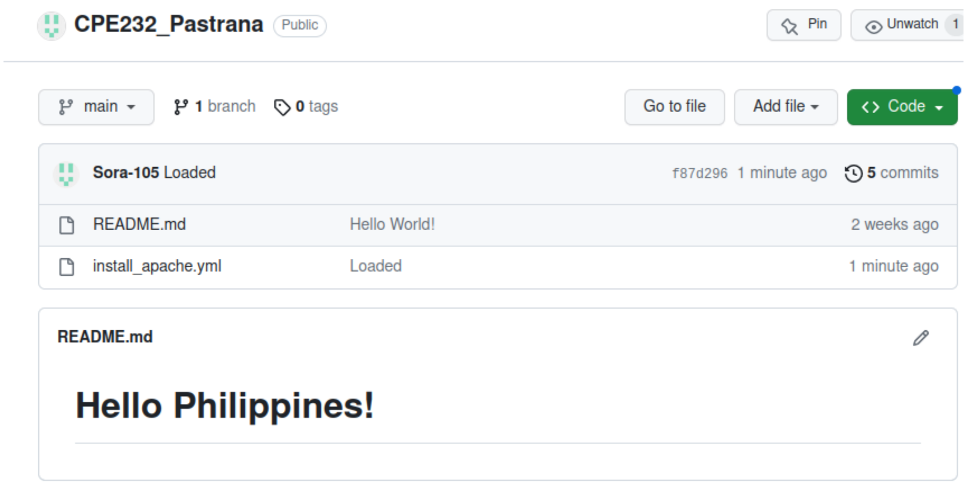


```
pastrana@localmachine:~/CPE232_Pastrana$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

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

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

TASK [add PHP support for apache] **********************************************************
changed: [127.0.0.1]

PLAY RECAP *********************************************************************************
127.0.0.1                  : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
```

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



```
pastrana@localmachine:~/CPE232_Pastrana$ git add install_apache.yml
pastrana@localmachine:~/CPE232_Pastrana$ git commit -m "Loaded"
[main f87d296] Loaded
 1 file changed, 11 insertions(+), 2 deletions(-)
pastrana@localmachine:~/CPE232_Pastrana$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 428 bytes | 428.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Sora-105/CPE232_Pastrana.git
   24c713d..f87d296  main -> main
```

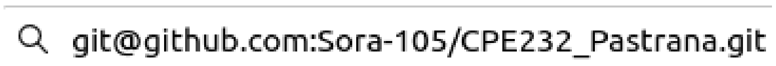


CPE232_Pastrana Public

Sora-105 Loaded		f87d296 1 minute ago 5 commits
README.md	Hello World!	2 weeks ago
install_apache.yml	Loaded	1 minute ago

README.md

Hello Philippines!



git@github.com:Sora-105/CPE232_Pastrana.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - Using a playbook in Ansible is essential for efficiently and consistently automating infrastructure management activities. It saves time, decreases errors, and provides flexibility, making them a crucial tool for infrastructure management and maintenance.

2. Summarize what we have done on this activity.
   - We kept track of everything we did in a repository. The servers are then configured so that we can utilize ansible commands on the distant servers. Following that, we installed Apache 2 and tested it in the browser. We finished constructing the playbook using them.