Name: Hazel Aillson T. Manuel	**Date Performed:** 08/28/2025
Course/Section: CPE31S4	**Date Submitted:** 09/12/2025
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1st Sem 2025-2026

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages in the remote machines. We will also create a playbook that will be used for automations.

Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. You can check this documentation if you want to learn more about playbooks. Working with playbooks — Ansible Documentation

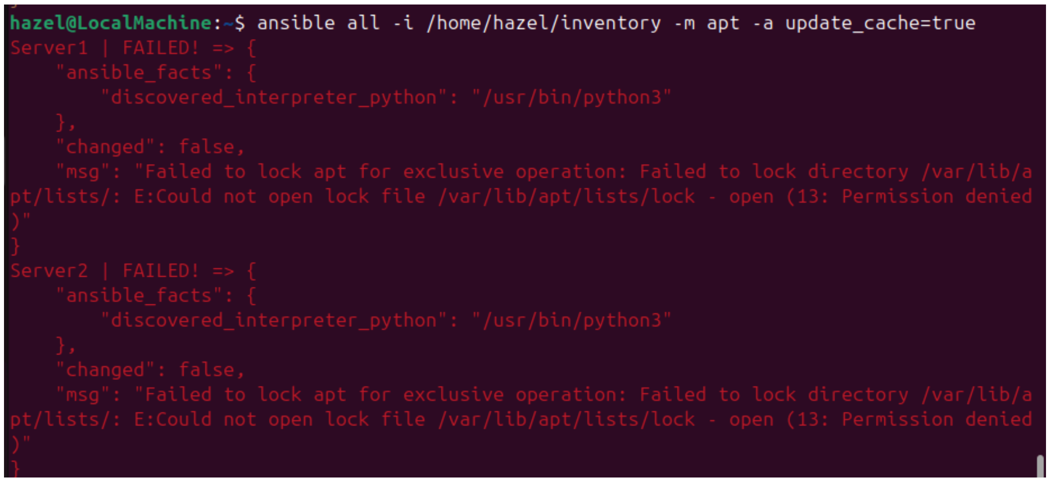
Task 1: Run elevated ad hoc commands

1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package information from the Internet. It is useful to get info on an updated version of packages or their dependencies. We can only run

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

ansible all -m apt -a update_cache=true

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

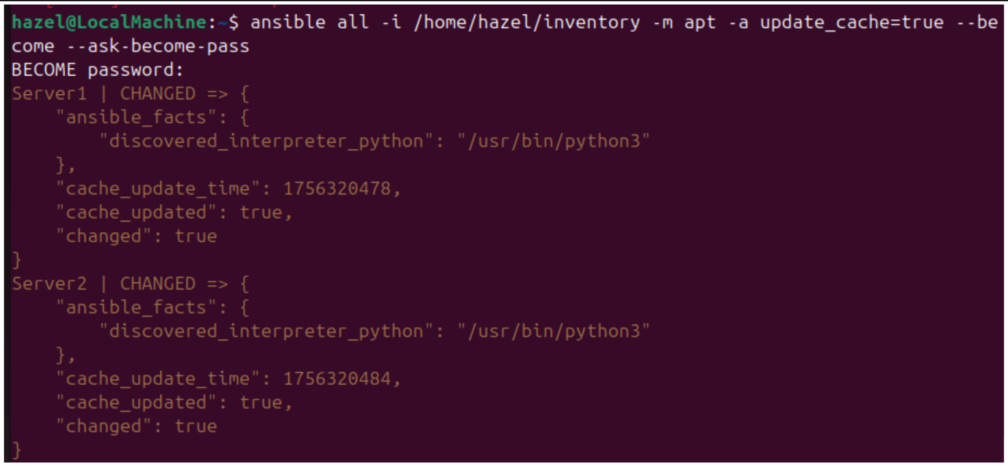


```
hazel@LocalMachine:~$ ansible all -i /home/hazel/inventory -m apt -a update_cache=true
Server1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/a
pt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied
)"
}
Server2 | 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/a
pt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied
)"
}
```

Figure 1: Unsuccessful Result of the command, insufficient privileges.

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.

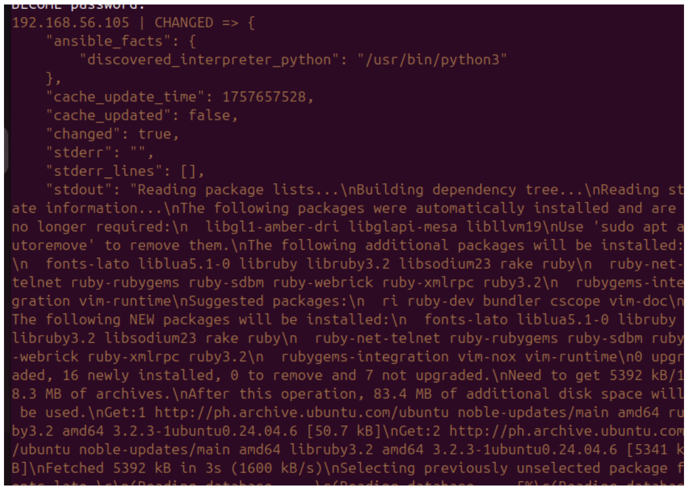
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.



```
hazel@LocalMachine:~$ ansible all -i /home/hazel/inventory -m apt -a update_cache=true --be
come --ask-become-pass
BECOME password:
Server1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1756320478,
    "cache_updated": true,
    "changed": true
}
Server2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1756320484,
    "cache_updated": true,
    "changed": true
}
```

Figure 2: Successful run of the command

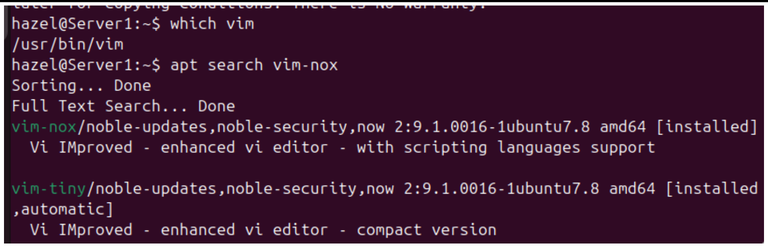
2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction. Here is the command: *ansible all -m apt -a name=vim-nox --become --ask-become-pass.* The command would take some time after typing the password because the local machine instructed the remote servers to actually install the package.



```
192.168.56.105 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757657528,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nThe following packages were automatically installed and are
no longer required:\n  libgl1-amber-dri libglapi-mesa libllvm19\nUse 'sudo apt a
utoremove' to remove them.\nThe following additional packages will be installed:
\n  fonts-lato liblua5.1-0 libruby libruby3.2 libsodium23 rake ruby\n  ruby-net-
telnet ruby-rubygems ruby-sdbm ruby-webrick ruby-xmlrpc ruby3.2\n  rubygems-inte
gration vim-runtime\nSuggested packages:\n  ri ruby-dev bundler cscope vim-doc\n
The following NEW packages will be installed:\n  fonts-lato liblua5.1-0 libruby
libruby3.2 libsodium23 rake ruby\n  ruby-net-telnet ruby-rubygems ruby-sdbm ruby
-webrick ruby-xmlrpc ruby3.2\n  rubygems-integration vim-nox vim-runtime\n0 upgr
aded, 16 newly installed, 0 to remove and 7 not upgraded.\nNeed to get 5392 kB/1
8.3 MB of archives.\nAfter this operation, 83.4 MB of additional disk space will
 be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 ru
by3.2 amd64 3.2.3-1ubuntu0.24.04.6 [50.7 kB]\nGet:2 http://ph.archive.ubuntu.com
/ubuntu noble-updates/main amd64 libruby3.2 amd64 3.2.3-1ubuntu0.24.04.6 [5341 k
B]\nFetched 5392 kB in 3s (1600 kB/s)\nSelecting previously unselected package f
```

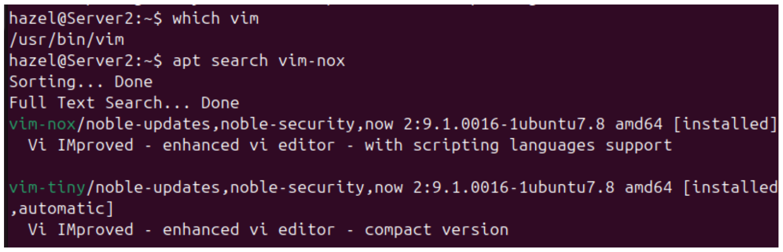
Figure 3: Installing vim-nox to both Manage Nodes through Control Node

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***



```
hazel@Server1:~$ which vim
/usr/bin/vim
hazel@Server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.8 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.8 amd64 [installed
,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

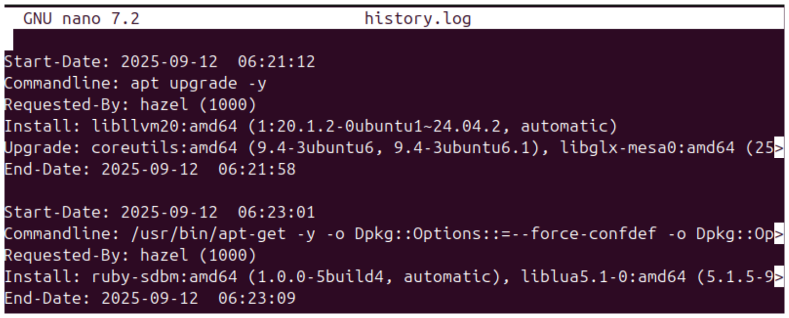
Figure 4: Verifying if vim-nox has been installed in Server1



```
hazel@Server2:~$ which vim
/usr/bin/vim
hazel@Server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.8 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.8 amd64 [installed
,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

Figure 5: Verifying if vim-nox has been installed in Server2

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.



```
  GNU nano 7.2                          history.log

Start-Date: 2025-09-12  06:21:12
Commandline: apt upgrade -y
Requested-By: hazel (1000)
Install: libllvm20:amd64 (1:20.1.2-0ubuntu1~24.04.2, automatic)
Upgrade: coreutils:amd64 (9.4-3ubuntu6, 9.4-3ubuntu6.1), libglx-mesa0:amd64 (25>
End-Date: 2025-09-12  06:21:58

Start-Date: 2025-09-12  06:23:01
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Op>
Requested-By: hazel (1000)
Install: ruby-sdbm:amd64 (1.0.0-5build4, automatic), liblua5.1-0:amd64 (5.1.5-9>
End-Date: 2025-09-12  06:23:09
```

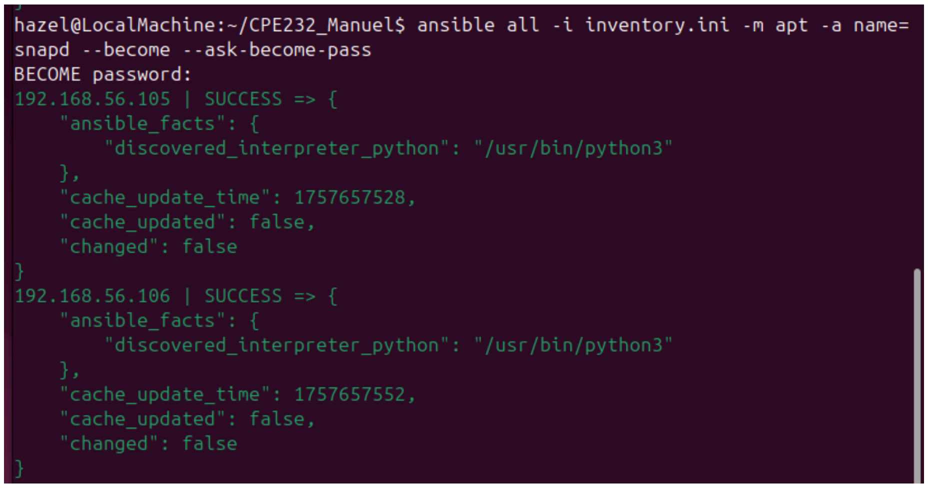
Figure 6: Checking history.log contents

In this log file, it shows that a user "hazel" has done an upgrade and apt-get in the machine. We are able to see that there are two sessions where it involves upgrading. This shows the routine where it updates the machine to improve its performance, security, and compatibility. Overall, this log file shows an audit trail of administrative activity.

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.

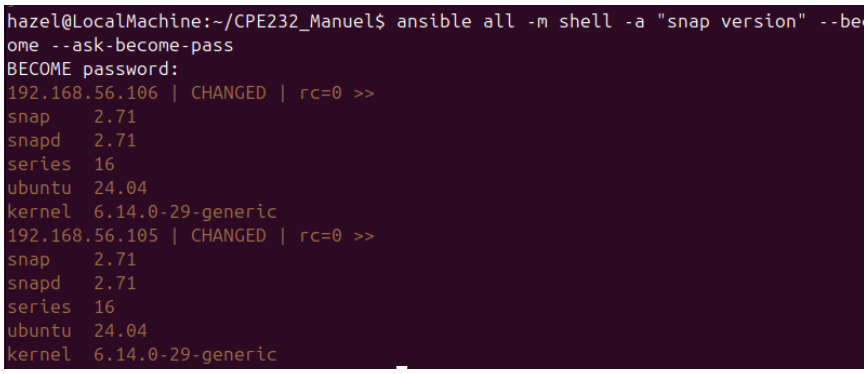
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*

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



```
hazel@LocalMachine:~/CPE232_Manuel$ ansible all -i inventory.ini -m apt -a name=
snapd --become --ask-become-pass
BECOME password:
192.168.56.105 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757657528,
    "cache_updated": false,
    "changed": false
}
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757657552,
    "cache_updated": false,
    "changed": false
}
```

Figure 7: Installing snapd to remote servers



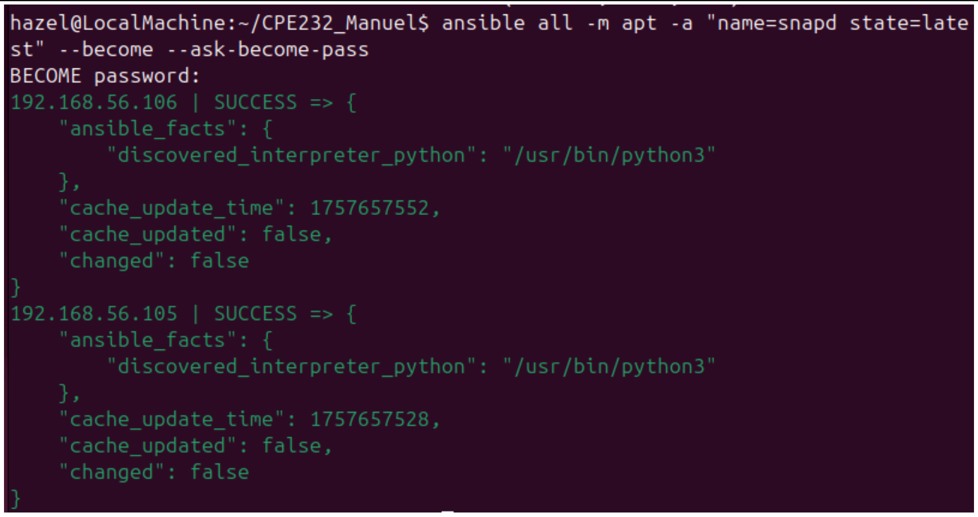
```
hazel@LocalMachine:~/CPE232_Manuel$ ansible all -m shell -a "snap version" --be
ome --ask-become-pass
BECOME password:
192.168.56.106 | CHANGED | rc=0 >>
snap    2.71
snapd   2.71
series  16
ubuntu  24.04
kernel  6.14.0-29-generic
192.168.56.105 | CHANGED | rc=0 >>
snap    2.71
snapd   2.71
series  16
ubuntu  24.04
kernel  6.14.0-29-generic
```

Figure 8: Verifying if snapd is installed in the remote servers

Yes, it was a successful command. Upon running the second command, we are able to confirm that the snapd has been installed to the remote servers. So, yes it did change anything.

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

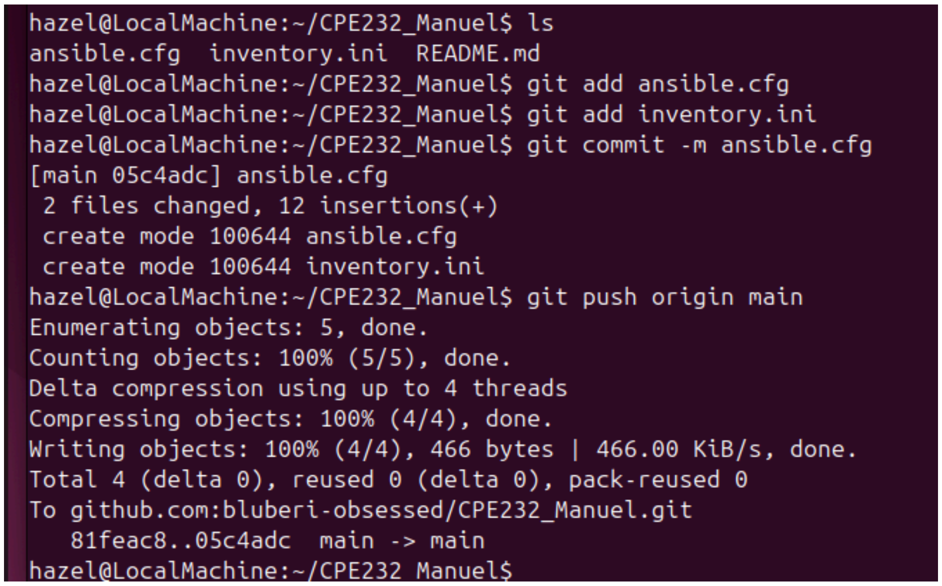
Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.



```
hazel@LocalMachine:~/CPE232_Manuel$ ansible all -m apt -a "name=snapd state=late
st" --become --ask-become-pass
BECOME password:
192.168.56.106 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757657552,
    "cache_updated": false,
    "changed": false
}
192.168.56.105 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757657528,
    "cache_updated": false,
    "changed": false
}
```

Figure 8: Installing snapd to remote servers again but with quotations and ensures it's the latest version

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



```
hazel@LocalMachine:~/CPE232_Manuel$ ls
ansible.cfg  inventory.ini  README.md
hazel@LocalMachine:~/CPE232_Manuel$ git add ansible.cfg
hazel@LocalMachine:~/CPE232_Manuel$ git add inventory.ini
hazel@LocalMachine:~/CPE232_Manuel$ git commit -m ansible.cfg
[main 05c4adc] ansible.cfg
 2 files changed, 12 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 inventory.ini
hazel@LocalMachine:~/CPE232_Manuel$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 466 bytes | 466.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:bluberi-obsessed/CPE232_Manuel.git
   81feac8..05c4adc  main -> main
hazel@LocalMachine:~/CPE232_Manuel$
```

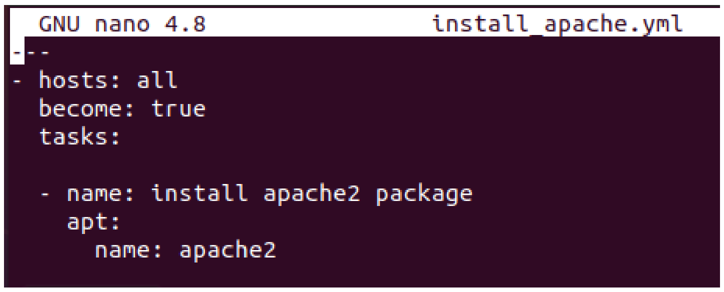
Figure 9: Committing all changes to CPE232_Manuel Repository in git

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

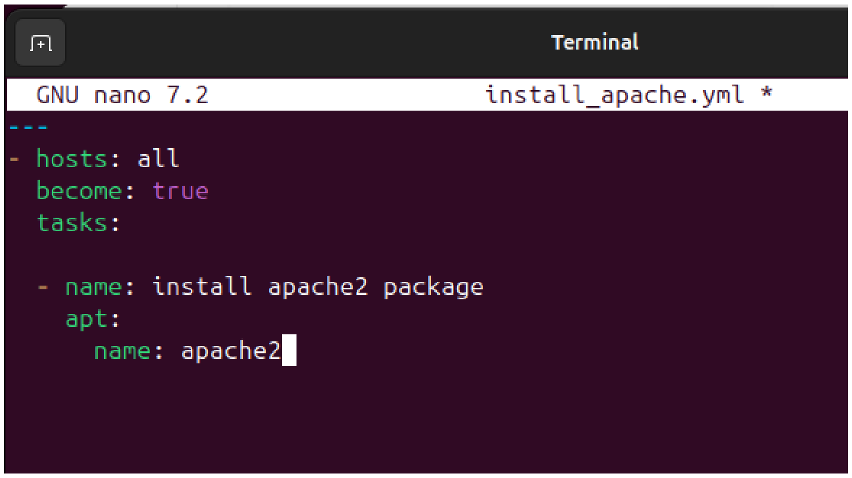
previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. This will create a playbook file called *install_apache.yml*. The .yml is the basic standard extension for playbook files.

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
```

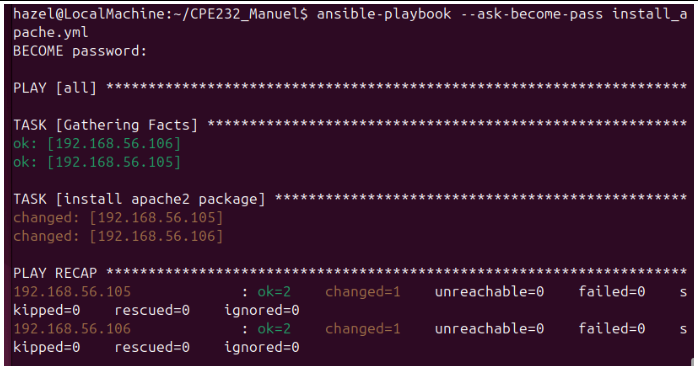
Make sure to save the file. Take note also of the alignments of the texts.



```
GNU nano 7.2                        install_apache.yml *
---
- hosts: all
  become: true
  tasks:

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

Figure 10: Creating a new notebook where it will install apache

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

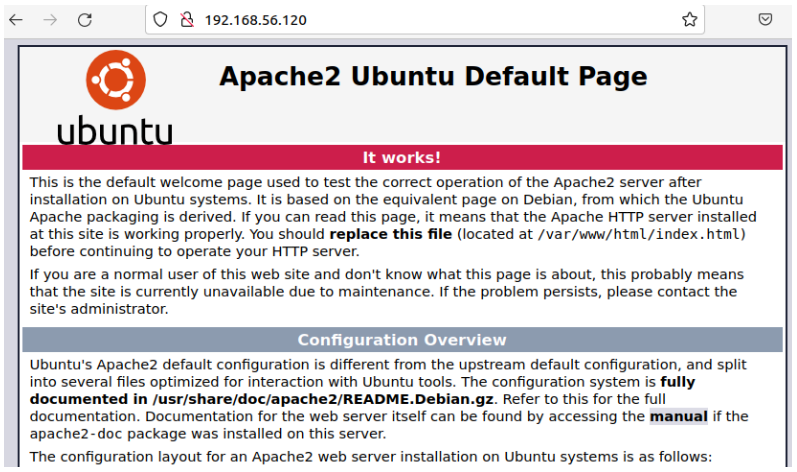


```
hazel@LocalMachine:~/CPE232_Manuel$ ansible-playbook --ask-become-pass install_a
pache.yml
BECOME password:

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

TASK [Gathering Facts] **********************************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

TASK [install apache2 package] **************************************************
changed: [192.168.56.105]
changed: [192.168.56.106]

PLAY RECAP **********************************************************************
192.168.56.105             : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.106             : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figure 11: Running the playbook file

The result shows that it is able to connect properly to the remote servers. After that, it proceeds to install apache2 where the "CHANGED" confirms that the playbook works. This "CHANGED" means that there is changes that happened in the remote servers, in this case, an apache was installed

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!

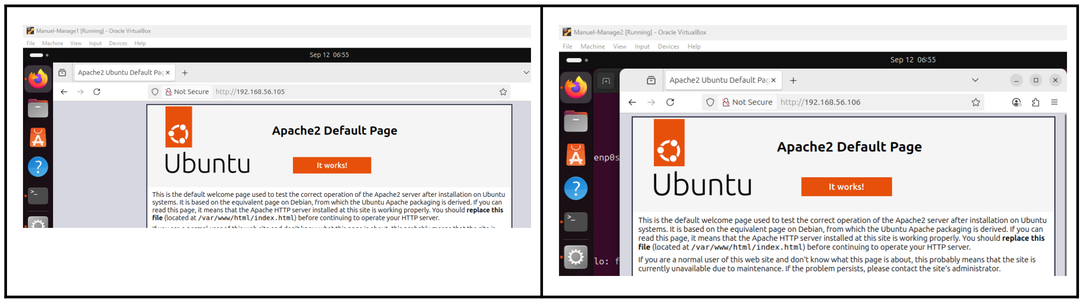
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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



Manuel-Manage1 [Running] - Oracle VirtualBox

Sep 12 06:55

Apache2 Ubuntu Default Pa

Not Secure http://192.168.56.105

Apache2 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.

Manuel-Manage2 [Running] - Oracle VirtualBox

Sep 12 06:55

Apache2 Ubuntu Default Pa

Not Secure http://192.168.56.106

Apache2 Default Page

Ubuntu

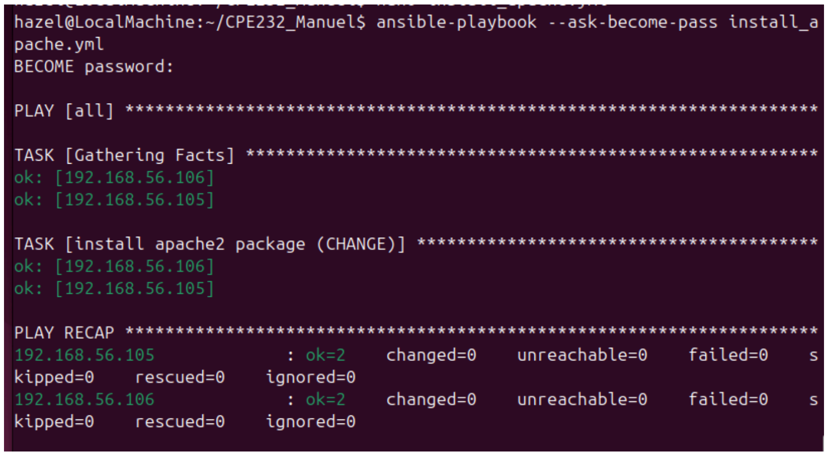
It works!

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.

Figure 12: Verifying if apache is installed in the remote servers

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?

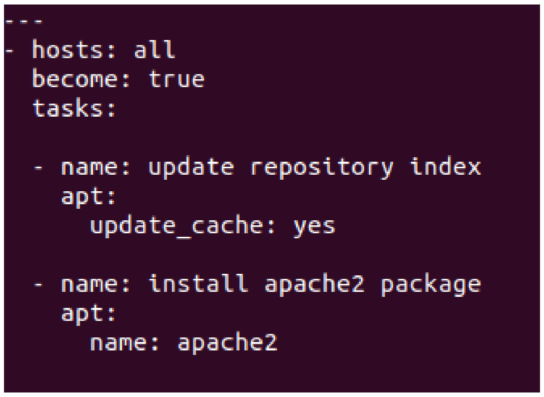


```
hazel@LocalMachine:~/CPE232_Manuel$ ansible-playbook --ask-become-pass install_a
pache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

TASK [install apache2 package (CHANGE)] ****************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

PLAY RECAP *********************************************************************
192.168.56.105             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.106             : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figure 12: Running playbook with new task name

Upon changing the task name, the output shows that the task name of installing apache have also been changed.

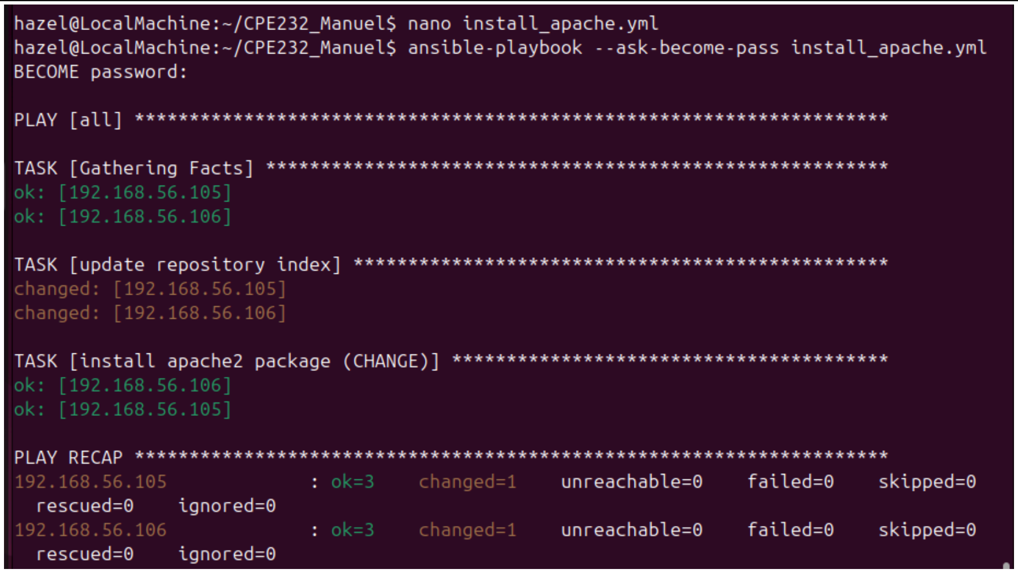
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities)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

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

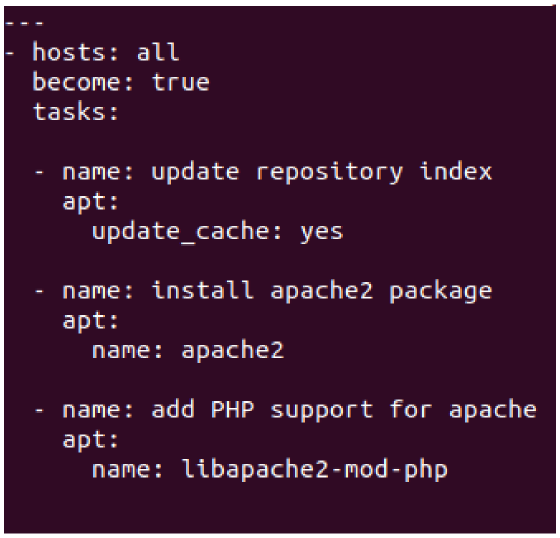


```
hazel@LocalMachine:~/CPE232_Manuel$ nano install_apache.yml
hazel@LocalMachine:~/CPE232_Manuel$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [update repository index] *************************************************
changed: [192.168.56.105]
changed: [192.168.56.106]

TASK [install apache2 package (CHANGE)] ****************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

PLAY RECAP *********************************************************************
192.168.56.105             : ok=3    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
192.168.56.106             : ok=3    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
```

Figure 13: Running playbook with update_cache

Yes, there have been changes to the remote server. With the first task, it commands the remote server to first run the command "apt update" before continuing to the next task.

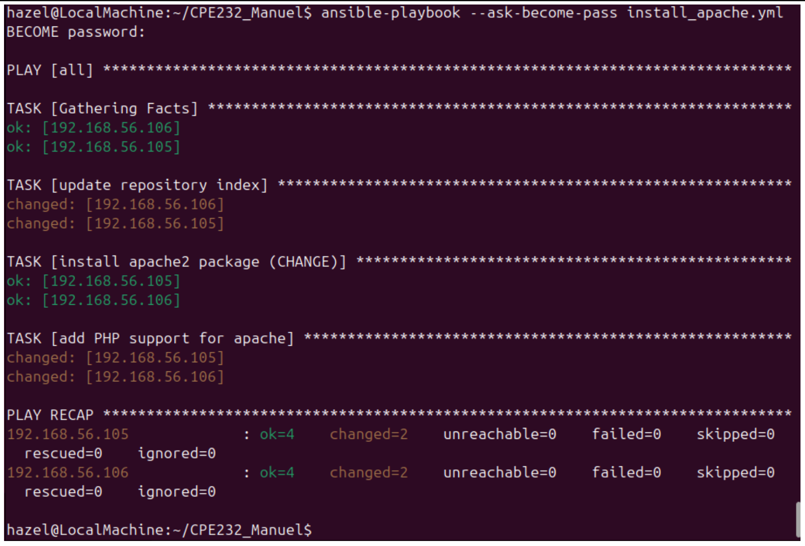
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
---
- 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.

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



```
hazel@LocalMachine:~/CPE232_Manuel$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *****************************************************************
ok: [192.168.56.106]
ok: [192.168.56.105]

TASK [update repository index] *********************************************************
changed: [192.168.56.106]
changed: [192.168.56.105]

TASK [install apache2 package (CHANGE)] ************************************************
ok: [192.168.56.105]
ok: [192.168.56.106]

TASK [add PHP support for apache] ******************************************************
changed: [192.168.56.105]
changed: [192.168.56.106]

PLAY RECAP *****************************************************************************
192.168.56.105             : ok=4    changed=2    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
192.168.56.106             : ok=4    changed=2    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0

hazel@LocalMachine:~/CPE232_Manuel$
```

Figure 13: Running playbook with new installation of PHP

Yes, there have been changes to the remote server. This time, we are able to install PHP which is useful for supporting apache.

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

   https://github.com/bluberi-obsessed/CPE232_Manuel.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   Playbooks are important because they let you automate tasks on remote machines, like installing or updating software. Instead of typing commands one by one, a playbook runs everything in order and makes sure each machine ends up in the same state. It saves time, avoids mistakes, and helps you manage systems more easily.
2. Summarize what we have done on this activity.
   We learned how to use Ansible to run commands with admin access, like updating packages and installing tools such as vim-nox and snapd. Then, we created a playbook to install Apache2 and added more tasks like updating package lists and enabling PHP. We tested each step, checked the results, and committed everything to GitHub.