Name: Tamayo, Ray Lan A.	**Date Performed: 09/13/2024**
Course/Section: CPE212-CPE31S21	**Date Submitted: 09/13/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY: First 2024-2025**

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

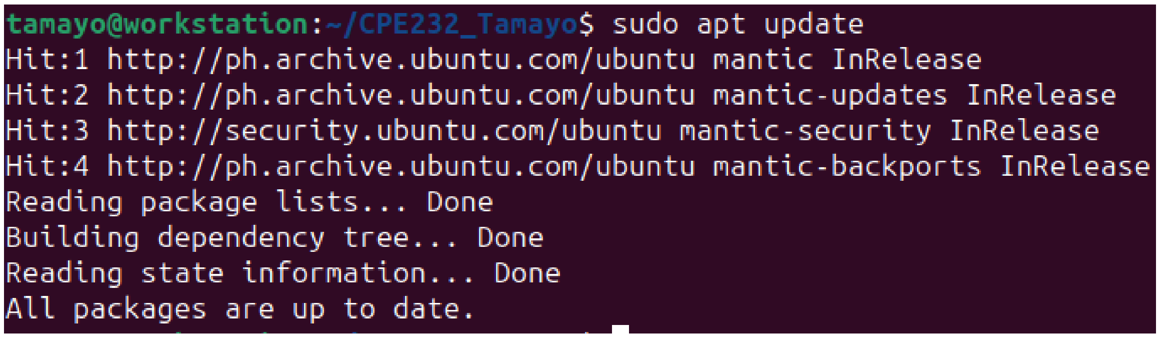
So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages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

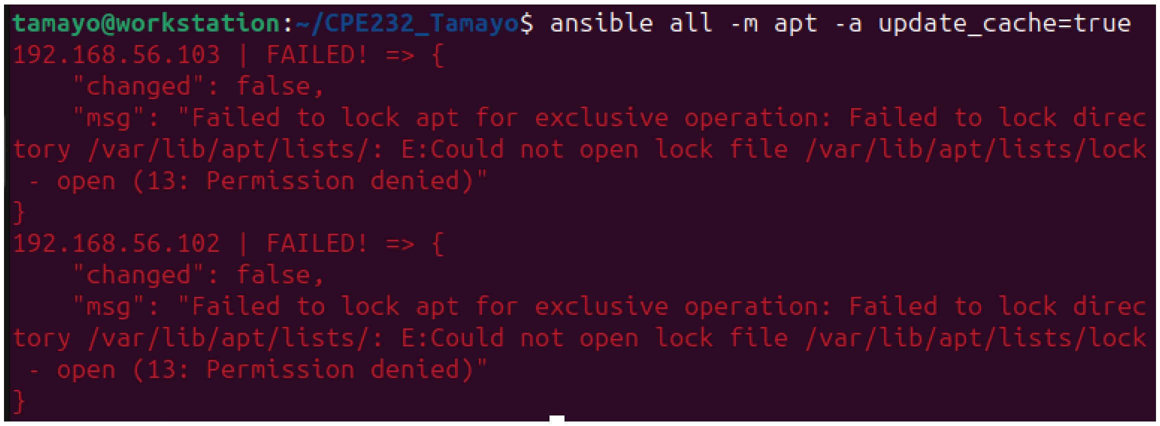
an apt update command in a remote machine. Issue the following command:



```
tamayo@workstation:~/CPE232_Tamayo$ sudo apt update
Hit:1 http://ph.archive.ubuntu.com/ubuntu mantic InRelease
Hit:2 http://ph.archive.ubuntu.com/ubuntu mantic-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu mantic-security InRelease
Hit:4 http://ph.archive.ubuntu.com/ubuntu mantic-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

ansible all -m apt -a update_cache=true

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

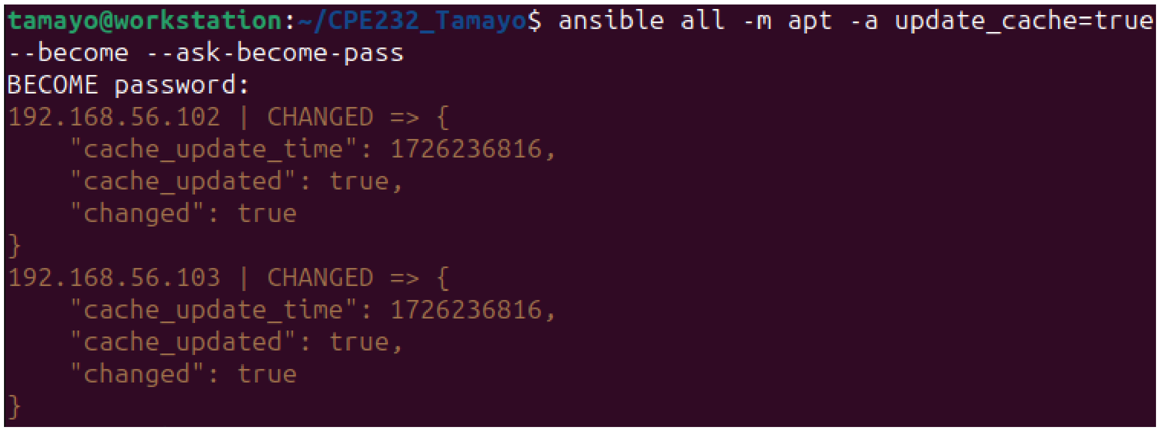


```
tamayo@workstation:~/CPE232_Tamayo$ ansible all -m apt -a update_cache=true
192.168.56.103 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock direc
tory /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock
 - open (13: Permission denied)"
}
192.168.56.102 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock direc
tory /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock
 - open (13: Permission denied)"
}
```

The result of the command is failed.

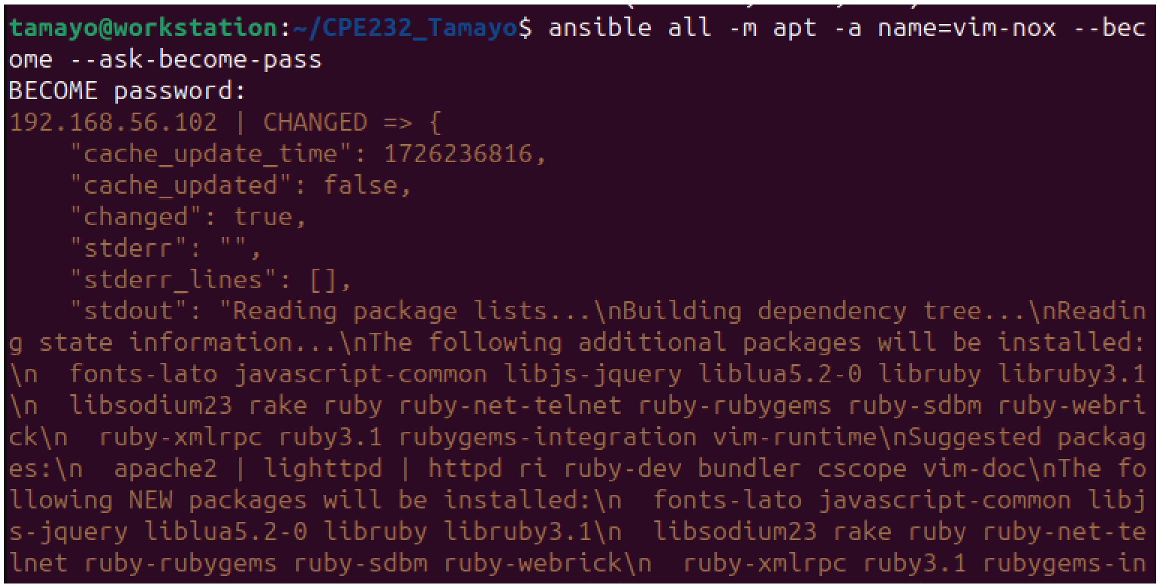
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



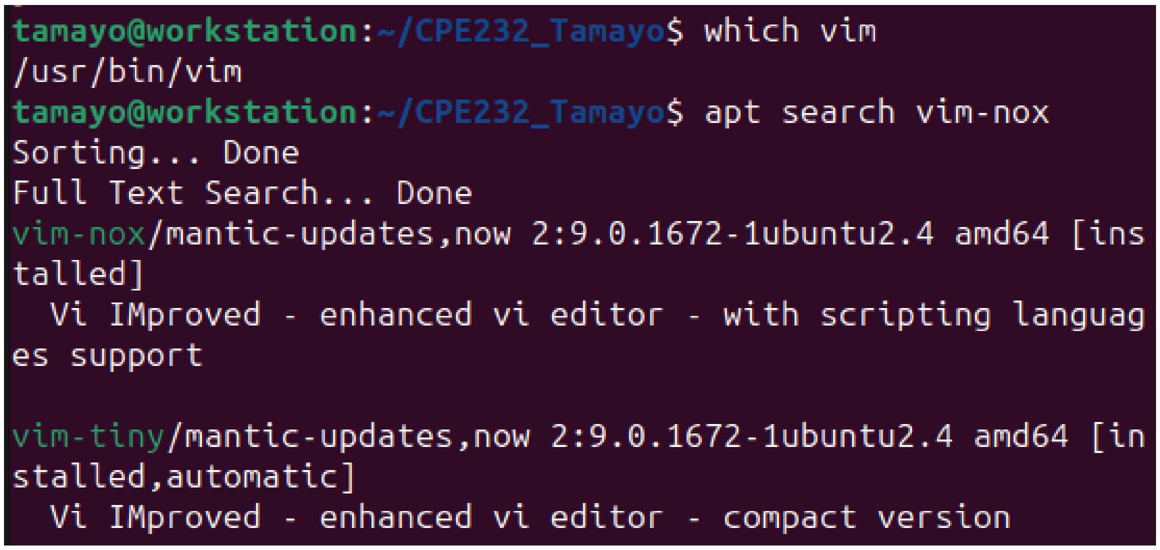
```
tamayo@workstation:~/CPE232_Tamayo$ ansible all -m apt -a update_cache=true
--become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
    "cache_update_time": 1726236816,
    "cache_updated": true,
    "changed": true
}
192.168.56.103 | CHANGED => {
    "cache_update_time": 1726236816,
    "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.



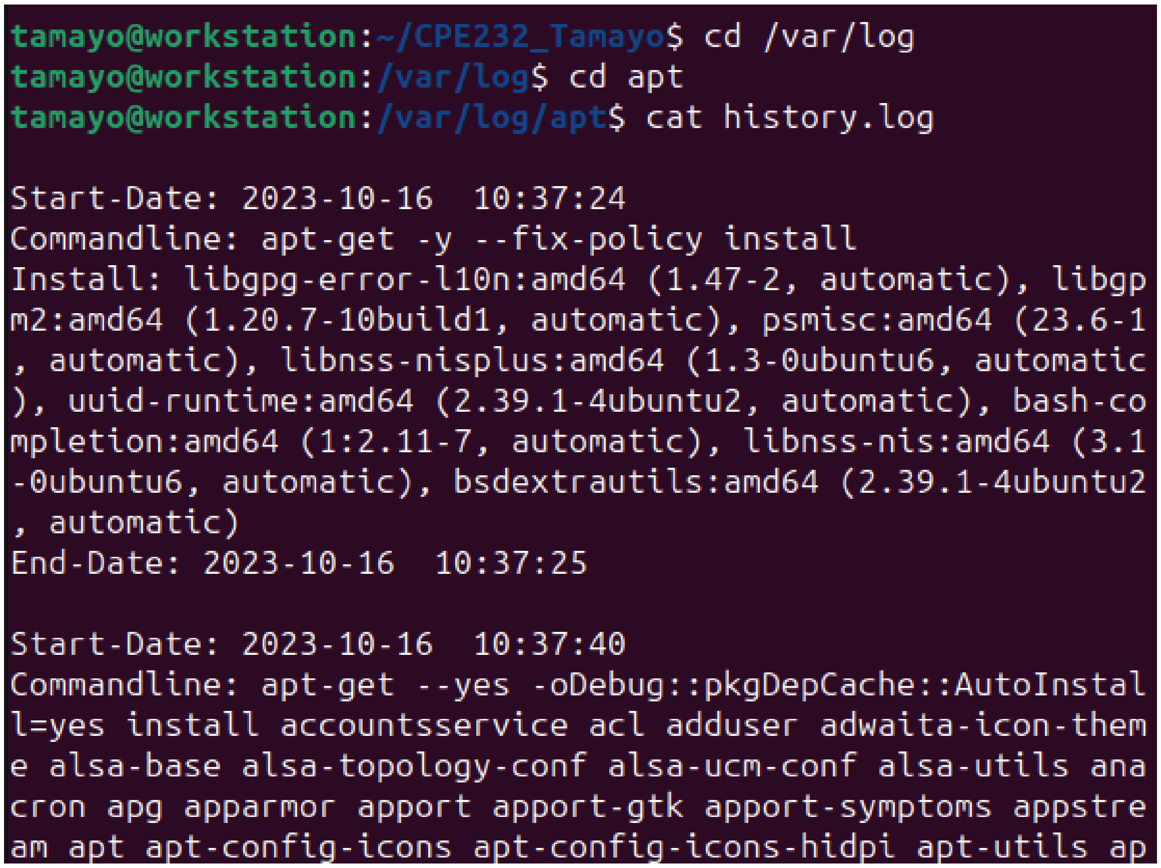
```
tamayo@workstation:~/CPE232_Tamayo$ ansible all -m apt -a name=vim-nox --bec
ome --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
    "cache_update_time": 1726236816,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReadin
g state information...\nThe following additional packages will be installed:
\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby libruby3.1
\n  libsodium23 rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-webri
ck\n  ruby-xmlrpc ruby3.1 rubygems-integration vim-runtime\nSuggested packag
es:\n  apache2 | lighttpd | httpd ri ruby-dev bundler cscope vim-doc\nThe fo
llowing NEW packages will be installed:\n  fonts-lato javascript-common libj
s-jquery liblua5.2-0 libruby libruby3.1\n  libsodium23 rake ruby ruby-net-te
lnet ruby-rubygems ruby-sdbm ruby-webrick\n  ruby-xmlrpc ruby3.1 rubygems-in
```

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?



```
tamayo@workstation:~/CPE232_Tamayo$ which vim
/usr/bin/vim
tamayo@workstation:~/CPE232_Tamayo$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/mantic-updates,now 2:9.0.1672-1ubuntu2.4 amd64 [ins
talled]
  Vi IMproved - enhanced vi editor - with scripting languag
es support

vim-tiny/mantic-updates,now 2:9.0.1672-1ubuntu2.4 amd64 [in
stalled,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.

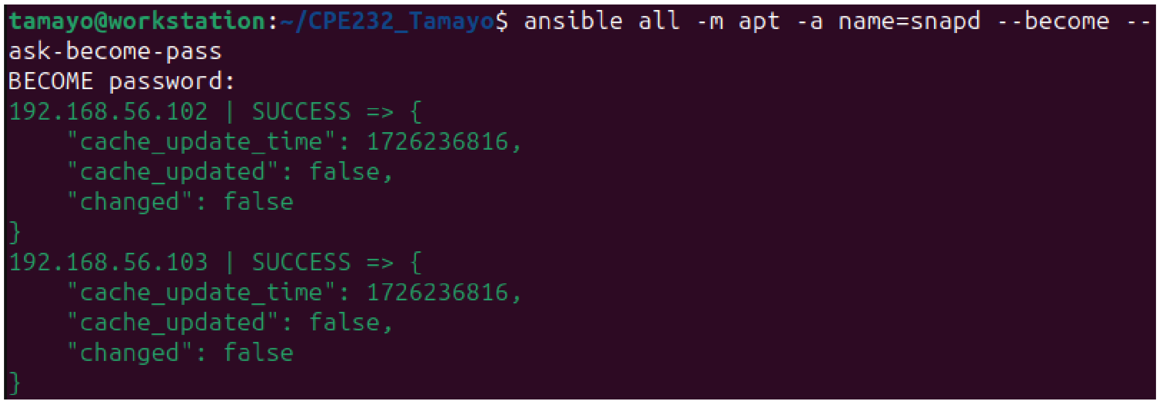


```
tamayo@workstation:~/CPE232_Tamayo$ cd /var/log
tamayo@workstation:/var/log$ cd apt
tamayo@workstation:/var/log/apt$ cat history.log

Start-Date: 2023-10-16  10:37:24
Commandline: apt-get -y --fix-policy install
Install: libgpg-error-l10n:amd64 (1.47-2, automatic), libgp
m2:amd64 (1.20.7-10build1, automatic), psmisc:amd64 (23.6-1
, automatic), libnss-nisplus:amd64 (1.3-0ubuntu6, automatic
), uuid-runtime:amd64 (2.39.1-4ubuntu2, automatic), bash-co
mpletion:amd64 (1:2.11-7, automatic), libnss-nis:amd64 (3.1
-0ubuntu6, automatic), bsdextrautils:amd64 (2.39.1-4ubuntu2
, automatic)
End-Date: 2023-10-16  10:37:25

Start-Date: 2023-10-16  10:37:40
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstal
l=yes install accountsservice acl adduser adwaita-icon-them
e alsa-base alsa-topology-conf alsa-ucm-conf alsa-utils ana
cron apg apparmor apport apport-gtk apport-symptoms appstre
am apt apt-config-icons apt-config-icons-hidpi apt-utils ap
```

I was able to see the date as well as the version of the package.

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.

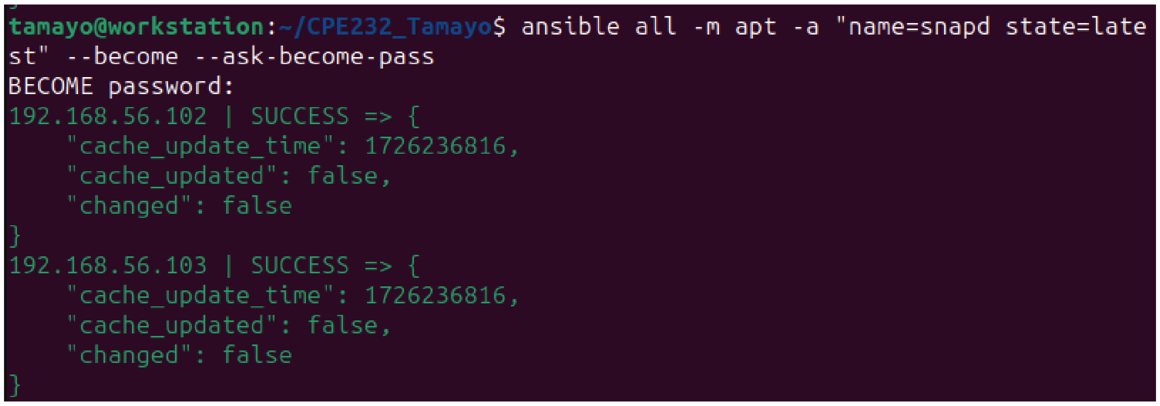
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*
   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?



```
tamayo@workstation:~/CPE232_Tamayo$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1726236816,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1726236816,
    "cache_updated": false,
    "changed": false
}
```

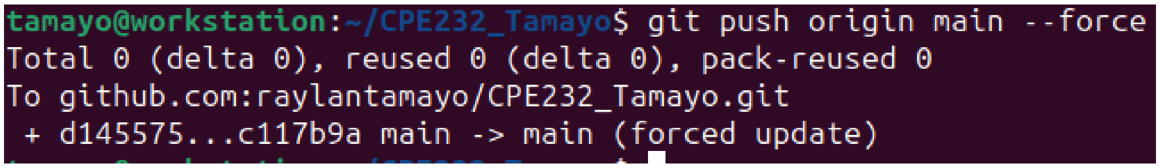
It is success and does not change anything.

   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.



```
tamayo@workstation:~/CPE232_Tamayo$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1726236816,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1726236816,
    "cache_updated": false,
    "changed": false
}
```

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

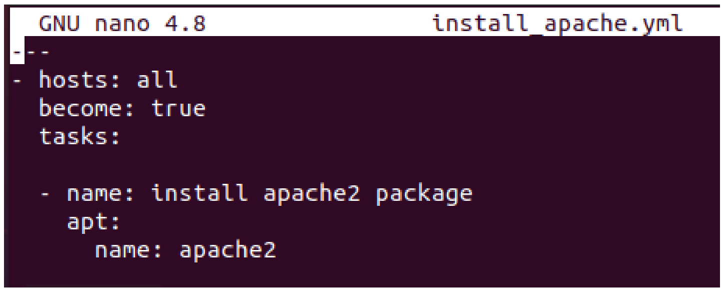


```
tamayo@workstation:~/CPE232_Tamayo$ git push origin main --force
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:raylantamayo/CPE232_Tamayo.git
 + d145575...c117b9a main -> main (forced update)
```

Task 2: Writing our First Playbook

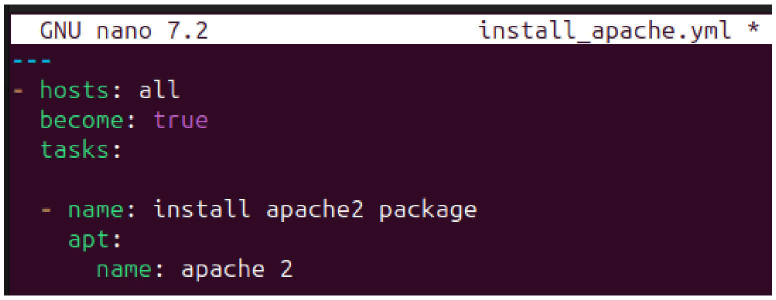
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and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

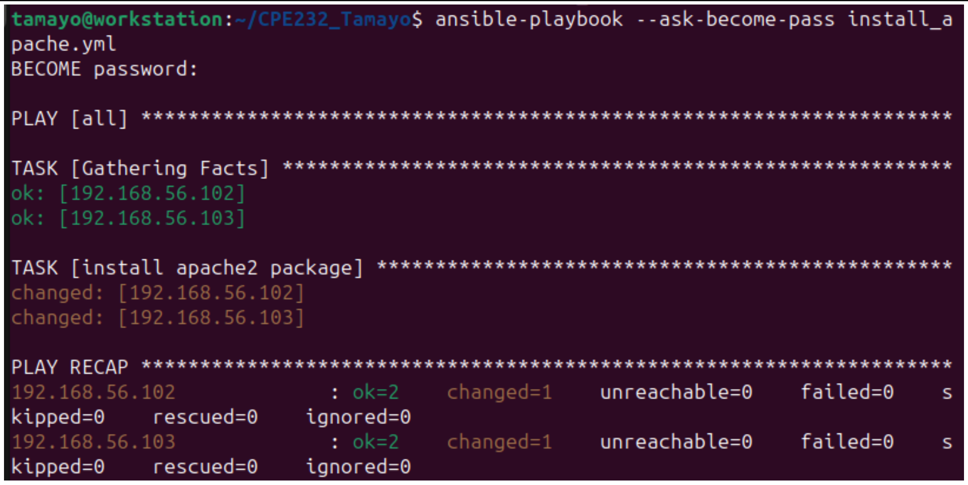
```
  GNU nano 4.8                      install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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

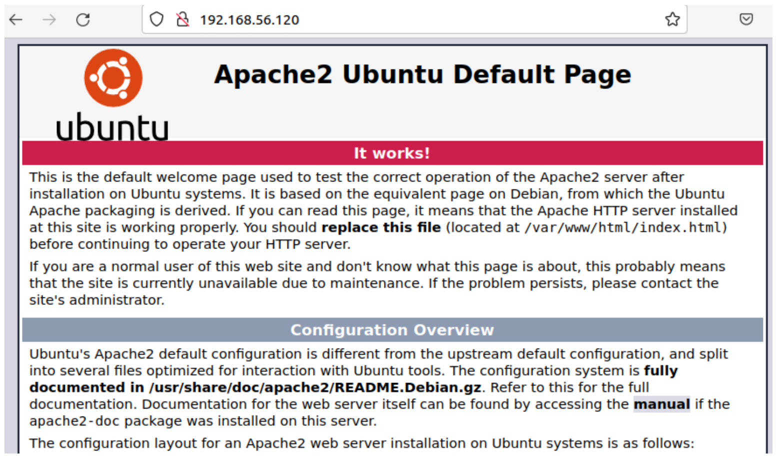
```
  GNU nano 7.2                          install_apache.yml *
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache 2
```

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



```
tamayo@workstation:~/CPE232_Tamayo$ ansible-playbook --ask-become-pass install_a
pache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [install apache2 package] *************************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

PLAY RECAP *********************************************************************
192.168.56.102             : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.103             : ok=2    changed=1    unreachable=0    failed=0    s
kipped=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.

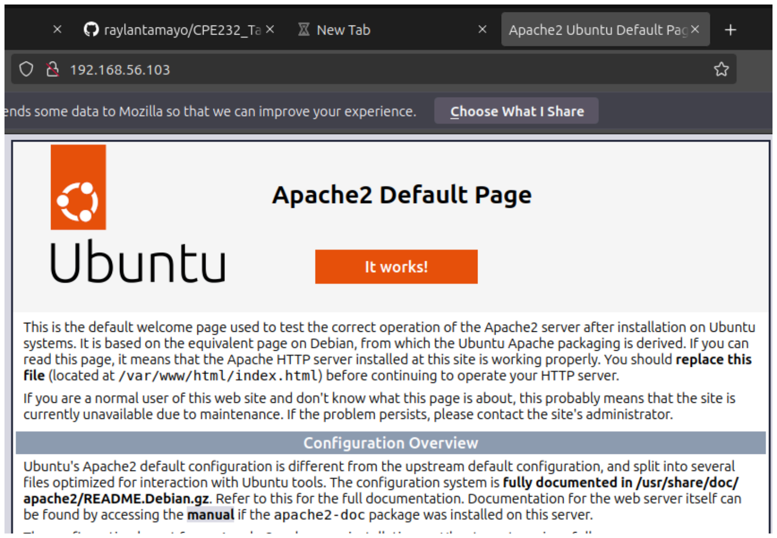
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

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:

SERVER 1



raylantamayo/CPE232_Ta
New Tab
Apache2 Ubuntu Default Pa
192.168.56.103
ends some data to Mozilla so that we can improve your experience.
Choose What I Share

Apache2 Default Page

Ubuntu

It works!

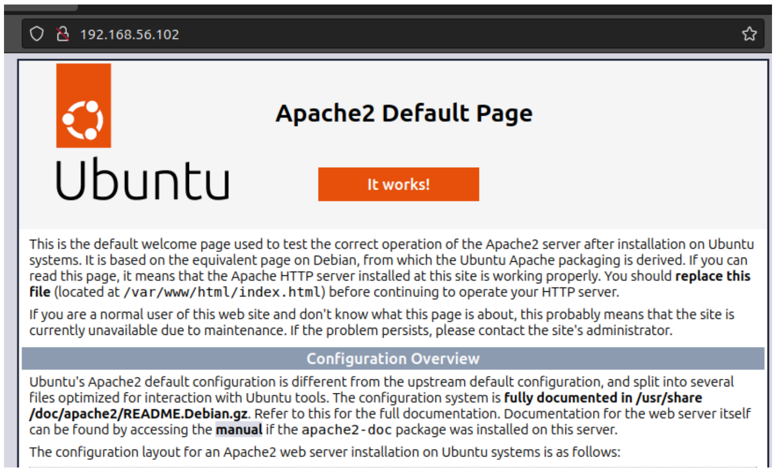
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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

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.

SERVER 2



```
192.168.56.102
```

Ubuntu

Apache2 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.

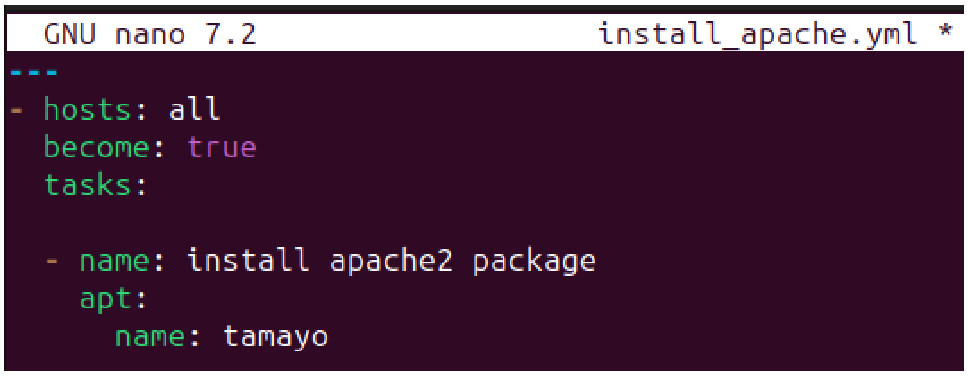
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

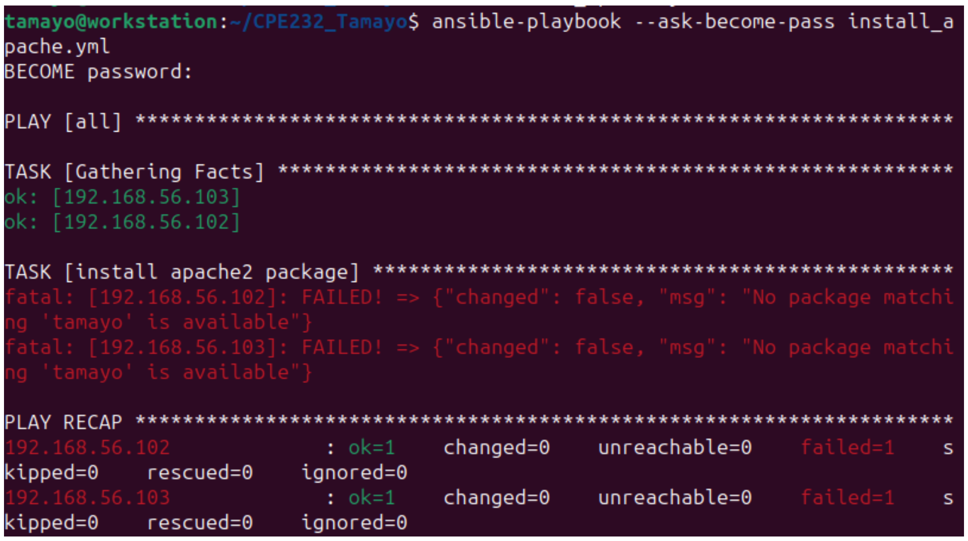
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share /doc/apache2/README.Debian.gz**. Refer to this for the full documentation.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



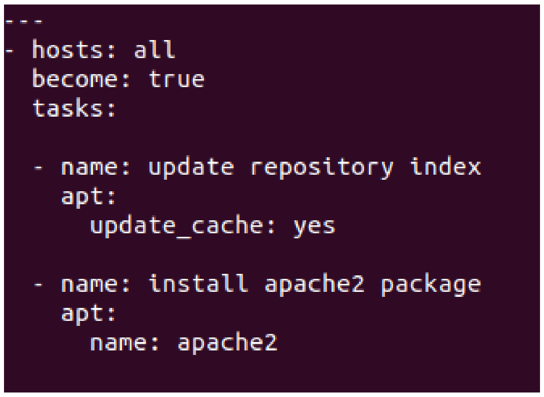
```
  GNU nano 7.2                    install_apache.yml *
---
- hosts: all
  become: true
  tasks:

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



```
tamayo@workstation:~/CPE232_Tamayo$ ansible-playbook --ask-become-pass install_a
pache.yml
BECOME password:

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

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [install apache2 package] *************************************************
fatal: [192.168.56.102]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'tamayo' is available"}
fatal: [192.168.56.103]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'tamayo' is available"}

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

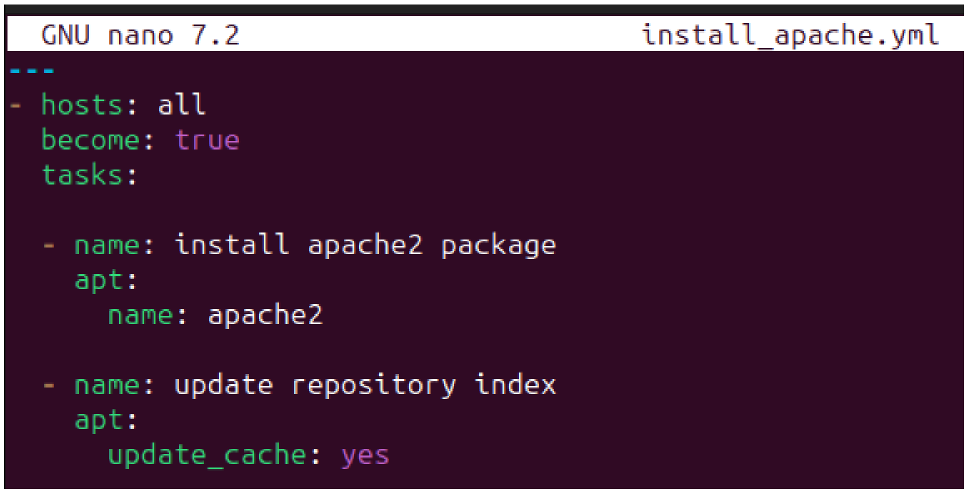
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional

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.



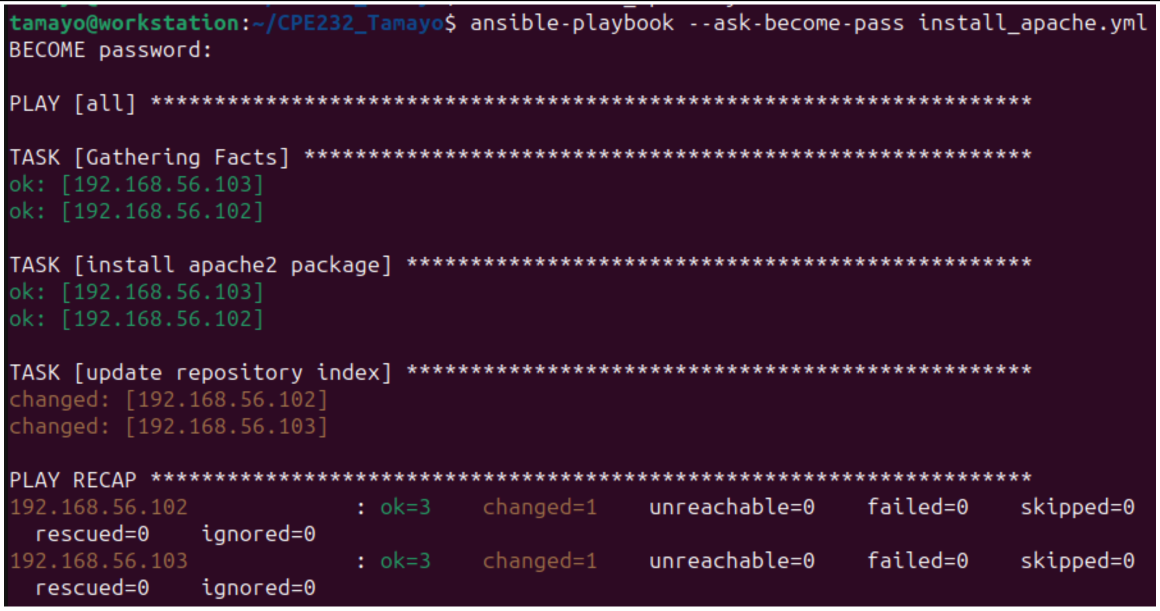
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

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

Save the changes to this file and exit.



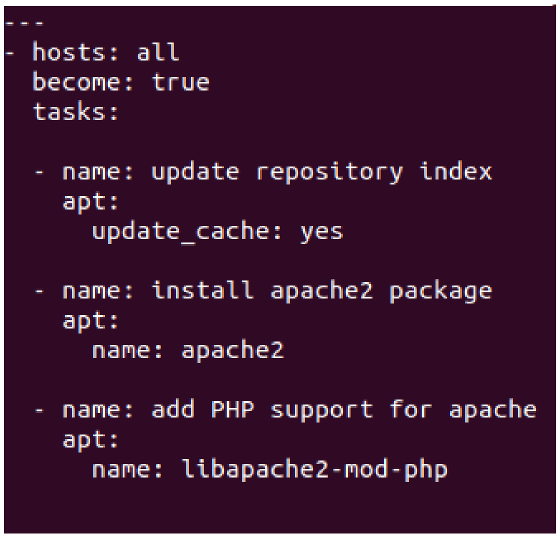
```
  GNU nano 7.2                                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2

  - name: update repository index
    apt:
      update_cache: yes
```

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



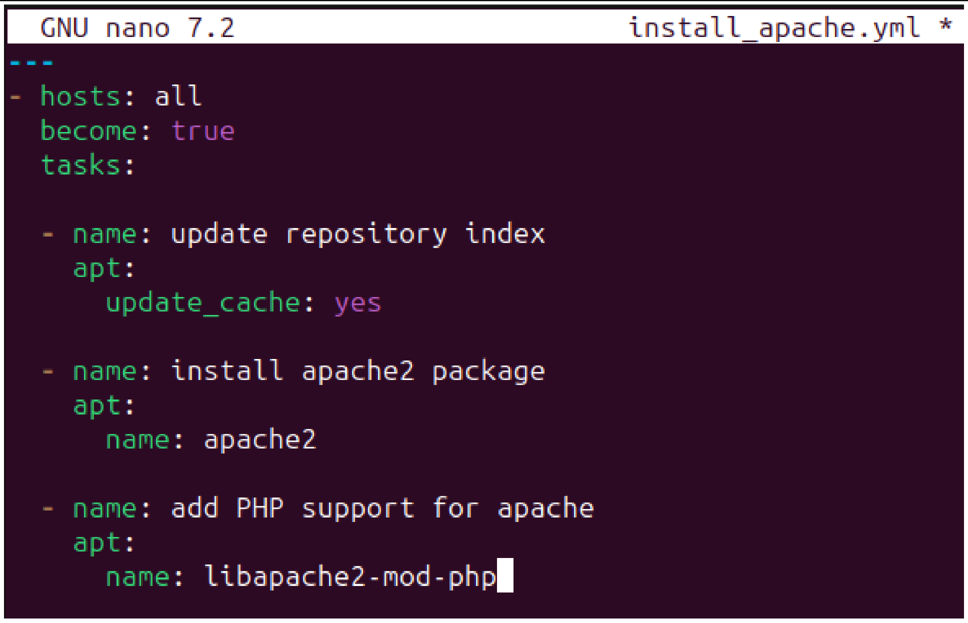
```
tamayo@workstation:~/CPE232_Tamayo$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] *******************************************************
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [install apache2 package] ***********************************************
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [update repository index] ***********************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

PLAY RECAP *******************************************************************
192.168.56.102             : ok=3    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
192.168.56.103             : 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.



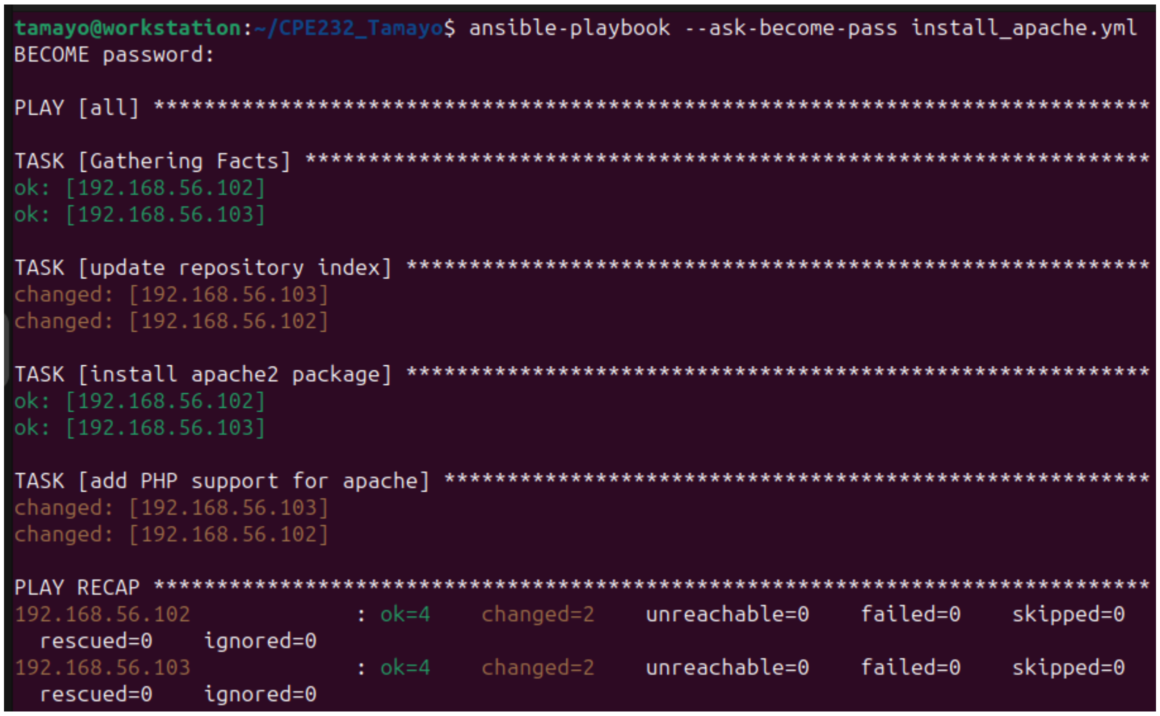
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

Save the changes to this file and exit.



```
  GNU nano 7.2                              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
```

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



```
tamayo@workstation:~/CPE232_Tamayo$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ******************************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update repository index] **********************************************************
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] **********************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [add PHP support for apache] *******************************************************
changed: [192.168.56.103]
changed: [192.168.56.102]

PLAY RECAP ******************************************************************************
192.168.56.102             : ok=4    changed=2    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
192.168.56.103             : ok=4    changed=2    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0
```

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

https://github.com/raylantamayo/CPE232_Tamayo

Reflections:

Answer the following:

1. **What is the importance of using a playbook?**

   Using a playbook makes system configuration and administration simpler is crucial in Ubuntu. Playbooks automate repeated tasks to guarantee consistency and reduce human error. They are often connected to programs such as Ansible. They facilitate mistake recovery and easy infrastructure scaling. Playbooks enhance teamwork by providing clear, repetitive instructions. Ultimately, they increase system reliability, save time, and free up administrators to focus on more critical tasks, all of which increase system efficiency and stability.

2. **Summarize what we have done on this activity.**

   I was able to understand how Ansible functions with this exercise. Here, we created an inventory and ansible.cfg file. We were able to link the workstation to servers 1 and 2 in this manner. We also created a yaml version of the playbook. Through inputting the IP addresses of servers 1 and 2, I successfully installed Apache, package, the repository index, and PHP assistance.

CONCLUSION

To sum up, automating Ansible tasks using playbooks and utilizing commands for remote machine management are crucial steps toward simplifying system administration. By doing this, human error rates are reduced and efficiency is increased, resulting in a more reliable IT system. We can more easily coordinate complex operations by utilizing Ansible's capabilities, which guarantees a more responsive and reliable network environment for today's computing demands.