Name: John Erwin S. Ballesteros	**Date Performed:** 9/11/2024
Course/Section: CpE31S2	**Date Submitted:** 9/13/2024
Instructor: Mr. Robin Valenzuela	**Semester and SY:** 1st Sem - 24' - 25'

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:**

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.

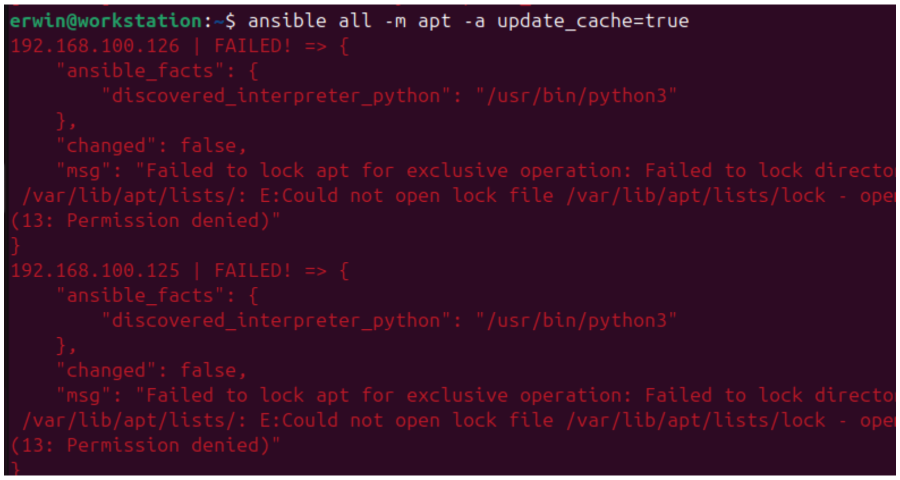
Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals,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

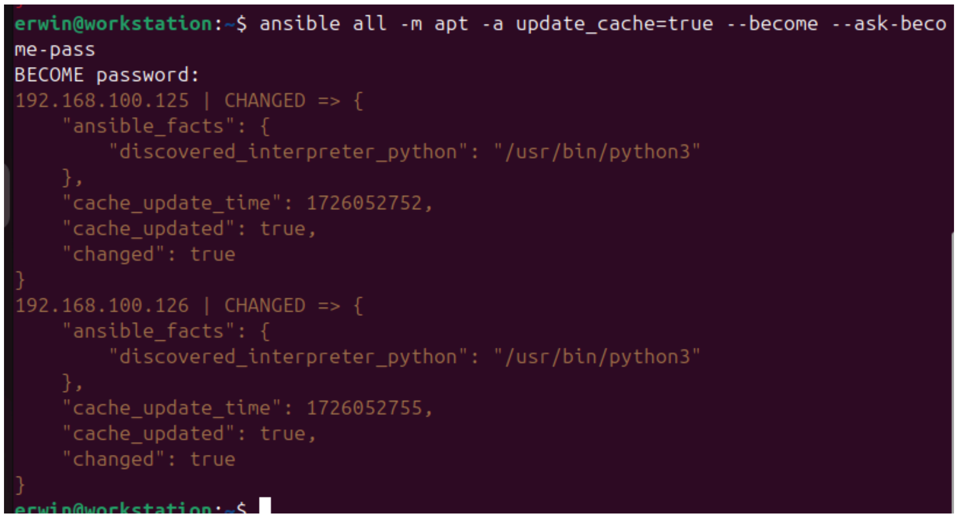
What is the result of the command? Is it successful?



```
erwin@workstation:~$ ansible all -m apt -a update_cache=true
192.168.100.126 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directo
 /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
(13: Permission denied)"
}
192.168.100.125 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directo
 /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
(13: Permission denied)"
}
```

The command was not successful.

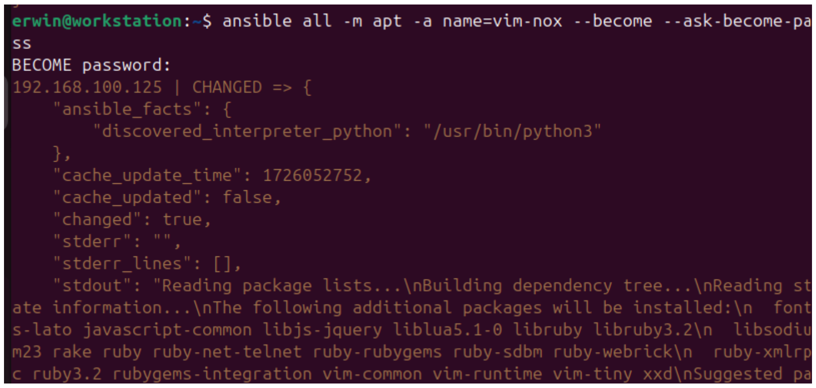
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



```
erwin@workstation:~$ ansible all -m apt -a update_cache=true --become --ask-beco
me-pass
BECOME password:
192.168.100.125 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "cache_updated": true,
    "changed": true
}
192.168.100.126 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052755,
    "cache_updated": true,
    "changed": true
}
erwin@workstation:~$ 
```

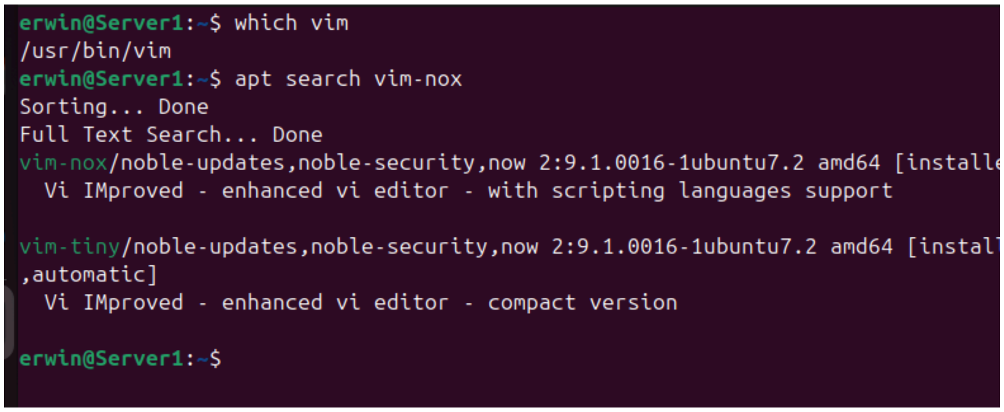
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

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.

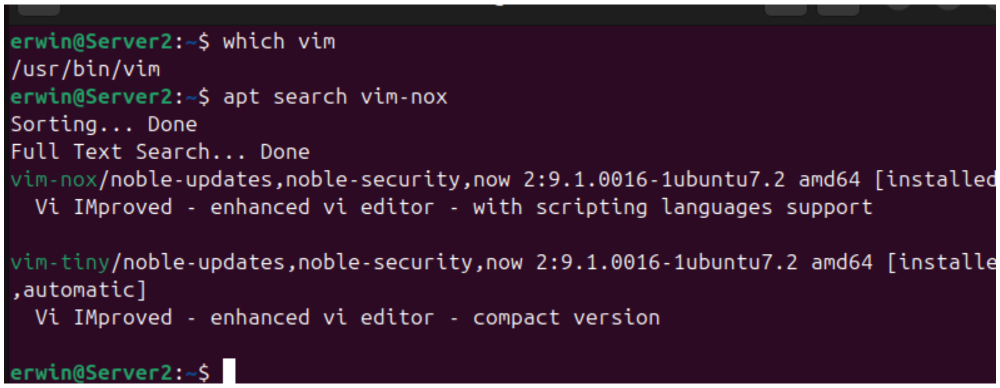


```
erwin@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pa
ss
BECOME password:
192.168.100.125 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nThe following additional packages will be installed:\n  font
s-lato javascript-common libjs-jquery liblua5.1-0 libruby libruby3.2\n  libsodiu
m23 rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-webrick\n  ruby-xmlrp
c ruby3.2 rubygems-integration vim-common vim-runtime vim-tiny xxd\nSuggested pa
```

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?

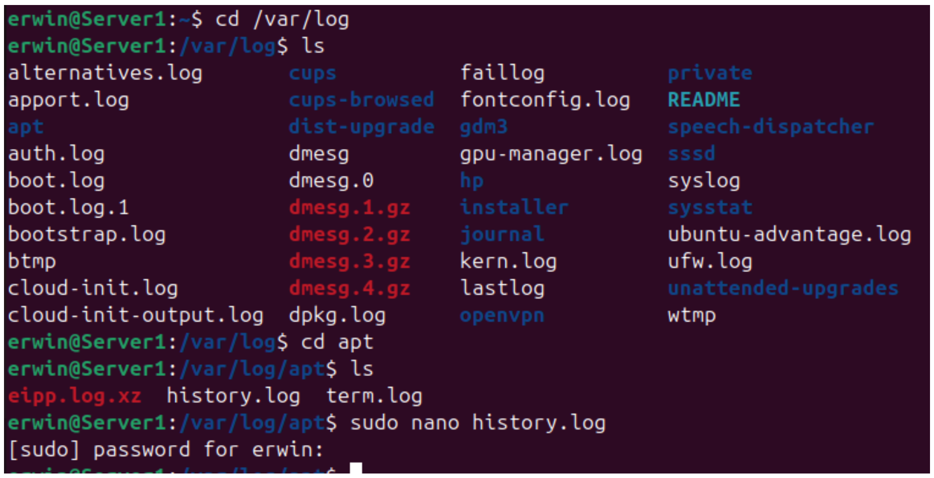


```
erwin@Server1:~$ which vim
/usr/bin/vim
erwin@Server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.2 amd64 [installe
  Vi IMproved - enhanced vi editor - with scripting languages support

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

erwin@Server1:~$
```

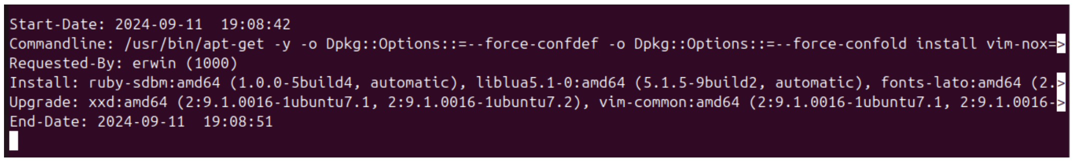


```
erwin@Server2:~$ which vim
/usr/bin/vim
erwin@Server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.2 amd64 [installed
  Vi IMproved - enhanced vi editor - with scripting languages support

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

erwin@Server2:~$
```

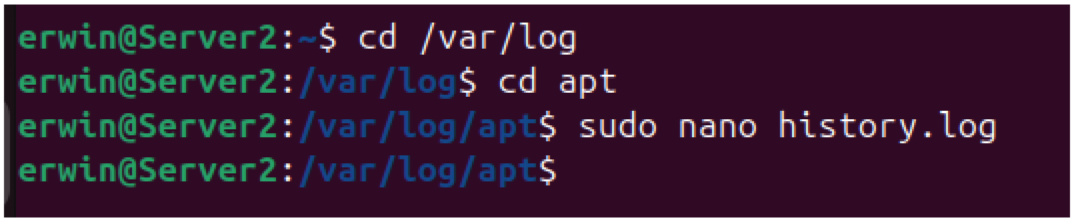
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.



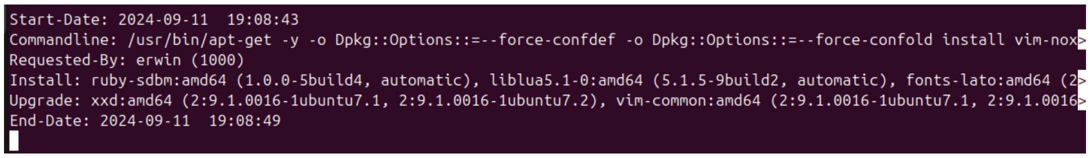
```
erwin@Server1:~$ cd /var/log
erwin@Server1:/var/log$ ls
alternatives.log       cups           faillog            private
apport.log             cups-browsed   fontconfig.log     README
apt                    dist-upgrade   gdm3               speech-dispatcher
auth.log               dmesg          gpu-manager.log    sssd
boot.log               dmesg.0        hp                 syslog
boot.log.1             dmesg.1.gz     installer          sysstat
bootstrap.log          dmesg.2.gz     journal            ubuntu-advantage.log
btmp                   dmesg.3.gz     kern.log           ufw.log
cloud-init.log         dmesg.4.gz     lastlog            unattended-upgrades
cloud-init-output.log  dpkg.log       openvpn            wtmp
erwin@Server1:/var/log$ cd apt
erwin@Server1:/var/log/apt$ ls
eipp.log.xz  history.log  term.log
erwin@Server1:/var/log/apt$ sudo nano history.log
[sudo] password for erwin:
```



```
Start-Date: 2024-09-11  19:08:42
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold install vim-nox=>
Requested-By: erwin (1000)
Install: ruby-sdbm:amd64 (1.0.0-5build4, automatic), liblua5.1-0:amd64 (5.1.5-9build2, automatic), fonts-lato:amd64 (2.>
Upgrade: xxd:amd64 (2:9.1.0016-1ubuntu7.1, 2:9.1.0016-1ubuntu7.2), vim-common:amd64 (2:9.1.0016-1ubuntu7.1, 2:9.1.0016->
End-Date: 2024-09-11  19:08:51
```



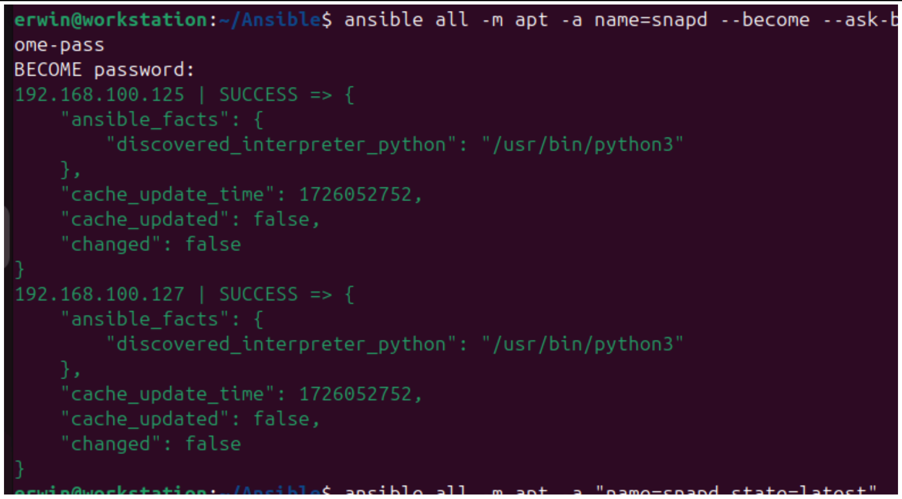
```
erwin@Server2:~$ cd /var/log
erwin@Server2:/var/log$ cd apt
erwin@Server2:/var/log/apt$ sudo nano history.log
erwin@Server2:/var/log/apt$
```



```
Start-Date: 2024-09-11  19:08:43
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold install vim-nox>
Requested-By: erwin (1000)
Install: ruby-sdbm:amd64 (1.0.0-5build4, automatic), liblua5.1-0:amd64 (5.1.5-9build2, automatic), fonts-lato:amd64 (2>
Upgrade: xxd:amd64 (2:9.1.0016-1ubuntu7.1, 2:9.1.0016-1ubuntu7.2), vim-common:amd64 (2:9.1.0016-1ubuntu7.1, 2:9.1.0016>
End-Date: 2024-09-11  19:08:49
```

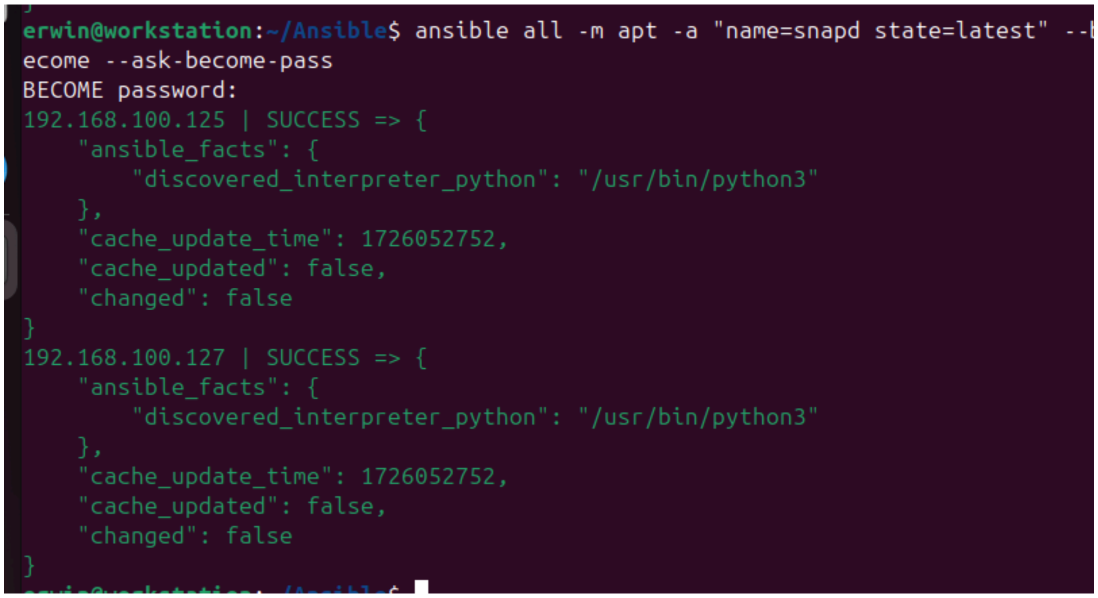
3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

   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?



```
erwin@workstation:~/Ansible$ ansible all -m apt -a name=snapd --become --ask-b
ome-pass
BECOME password:
192.168.100.125 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "cache_updated": false,
    "changed": false
}
192.168.100.127 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "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.



```
erwin@workstation:~/Ansible$ ansible all -m apt -a "name=snapd state=latest" --b
ecome --ask-become-pass
BECOME password:
192.168.100.125 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "cache_updated": false,
    "changed": false
}
192.168.100.127 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1726052752,
    "cache_updated": false,
    "changed": false
}
```

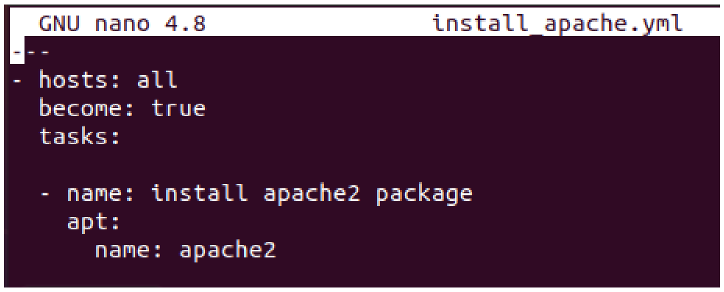
4. At this point, make sure to commit all changes to GitHub.

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.

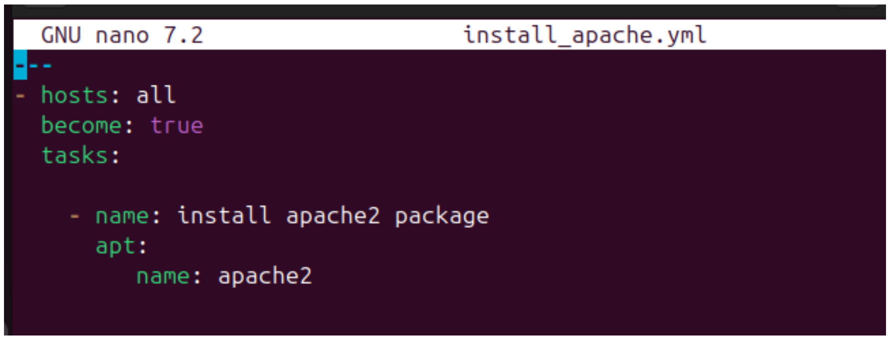
When the editor appears, type the following:



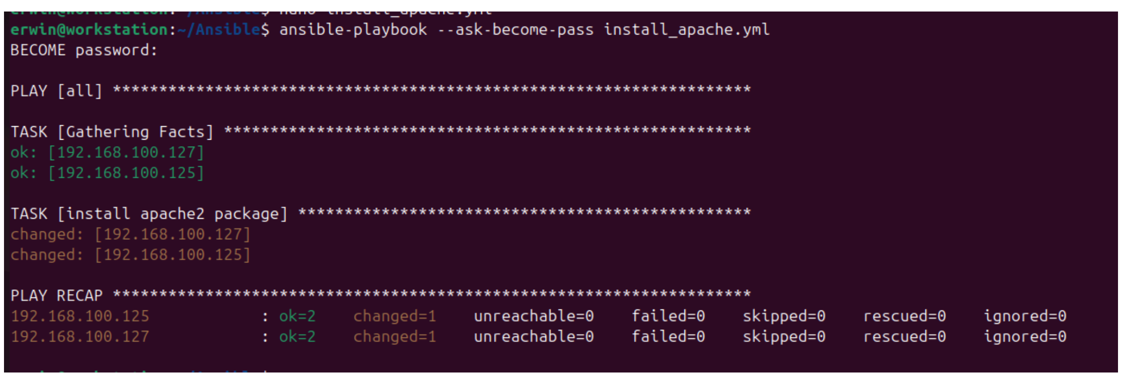
```
 GNU nano 4.8                      install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

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

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

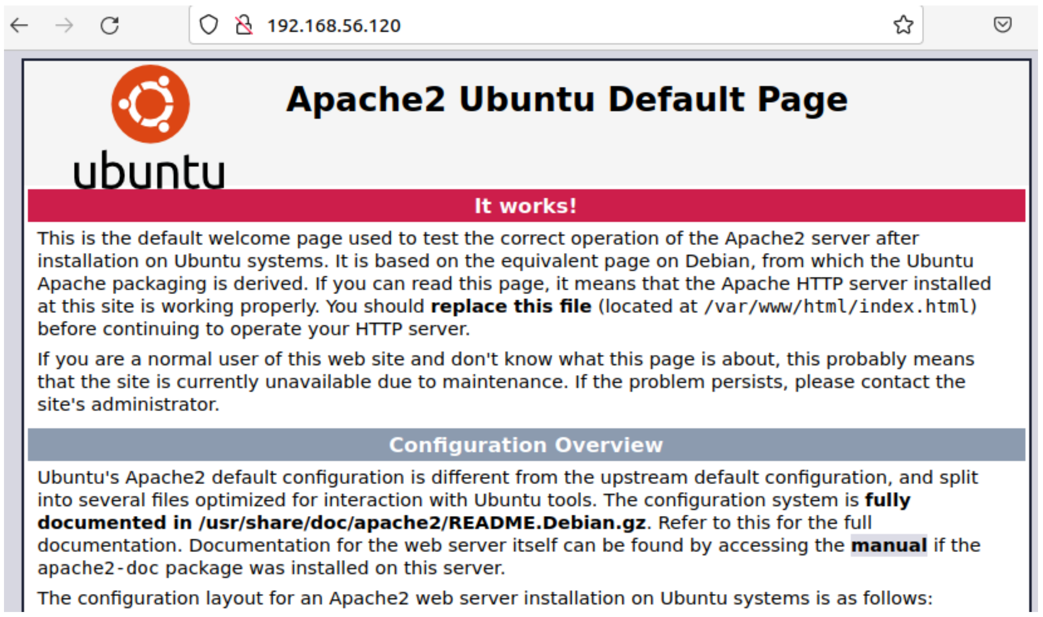


```
 GNU nano 7.2                      install_apache.yml
---
- hosts: all
  become: true
  tasks:

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



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

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

TASK [Gathering Facts] **********************************************************
ok: [192.168.100.127]
ok: [192.168.100.125]

TASK [install apache2 package] **************************************************
changed: [192.168.100.127]
changed: [192.168.100.125]

PLAY RECAP **********************************************************************
192.168.100.125            : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.100.127            : 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.

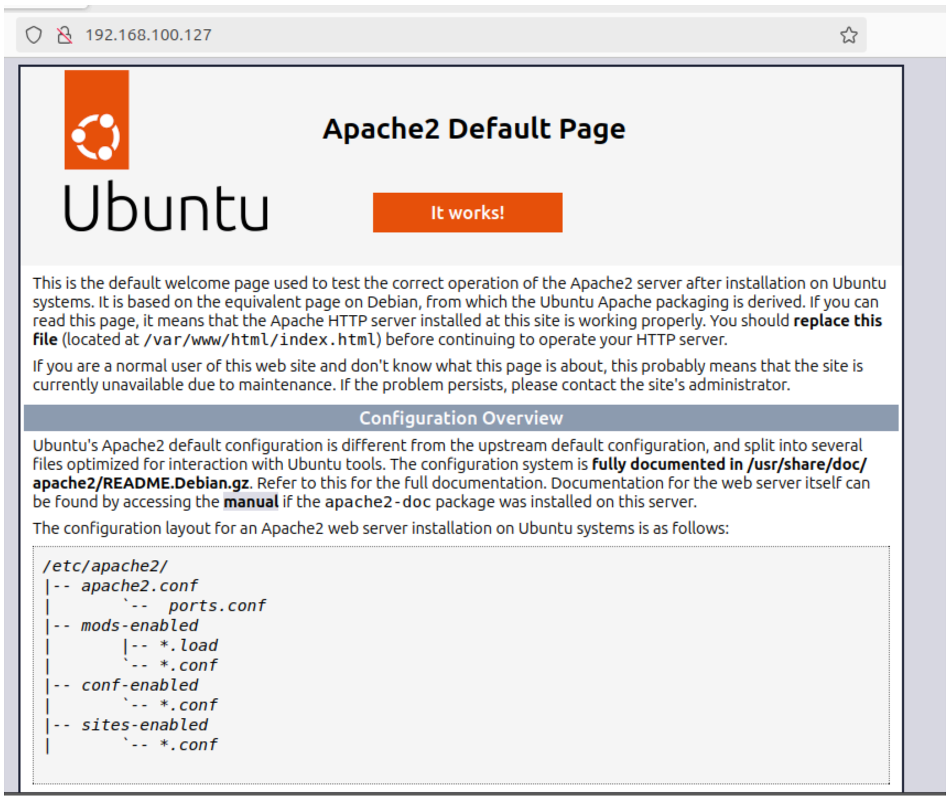
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

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:

Verification



192.168.100.127

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.

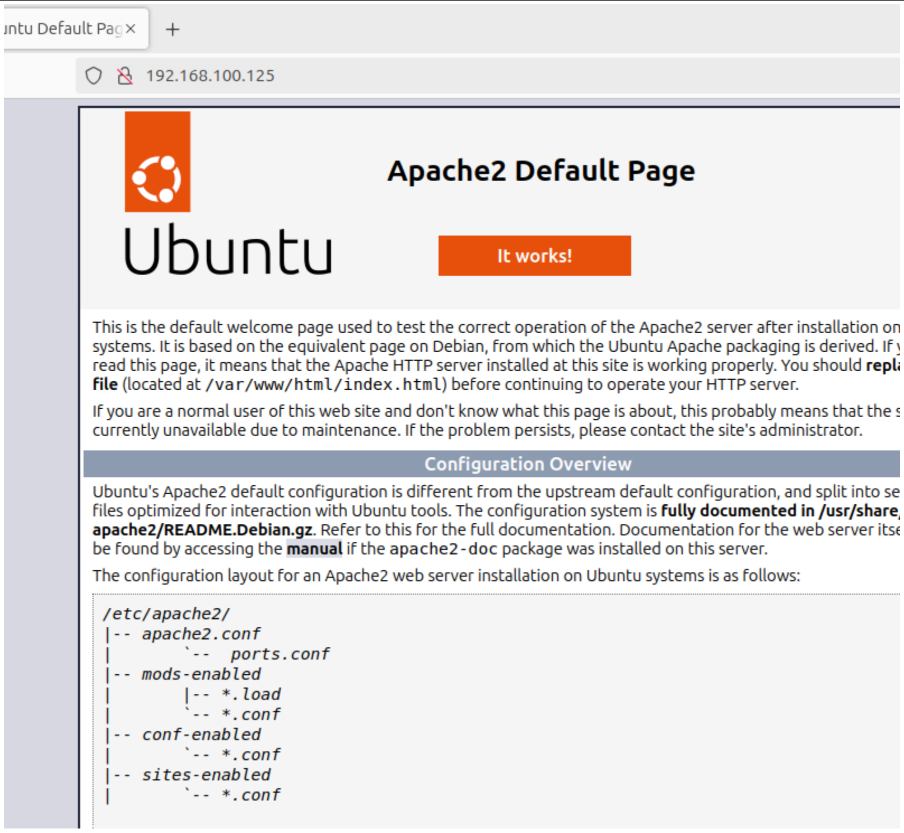
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

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:

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



Apache2 Default Page

This is the default welcome page used to test the correct operation of the Apache2 server after installation on
systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If y
read this page, it means that the Apache HTTP server installed at this site is working properly. You should **repla**
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 s
currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

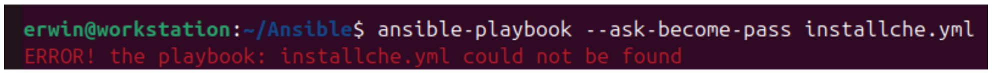
Configuration Overview

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

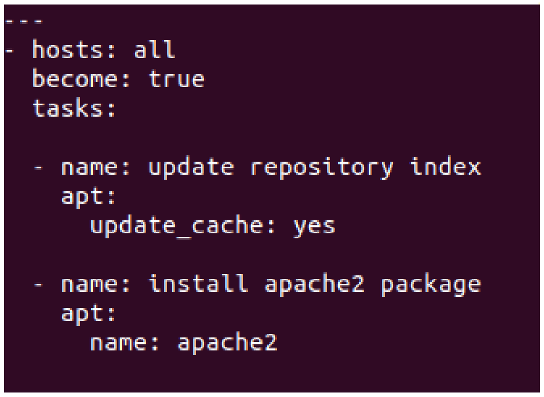
```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
|
```

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?



```
erwin@workstation:~/Ansible$ ansible-playbook --ask-become-pass installche.yml
ERROR! the playbook: installche.yml could not be found
```

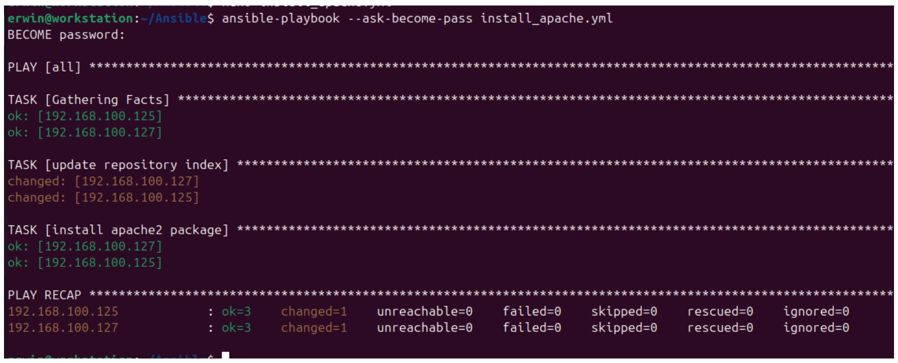
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache.* This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities)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

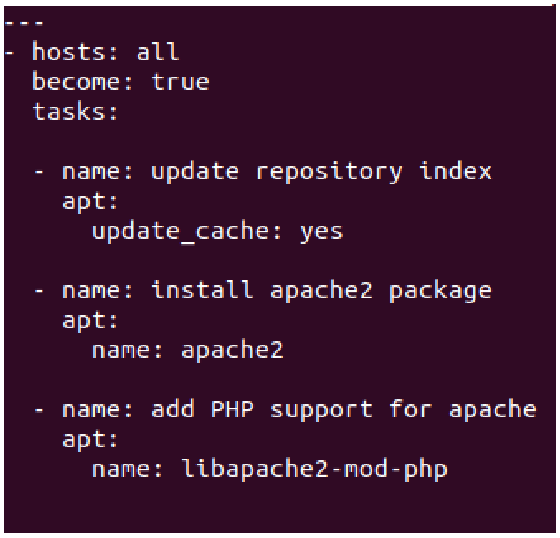
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?



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

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

TASK [Gathering Facts] *********************************************************************************
ok: [192.168.100.125]
ok: [192.168.100.127]

TASK [update repository index] *************************************************************************
changed: [192.168.100.127]
changed: [192.168.100.125]

TASK [install apache2 package] *************************************************************************
ok: [192.168.100.127]
ok: [192.168.100.125]

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

The added repository index allows the module to refresh the cache of the package in the remote system. Simply, it performs the apt update for us.

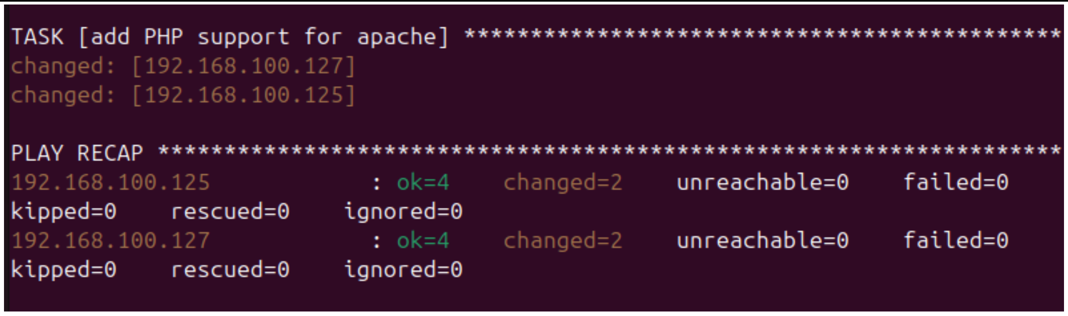
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
---
- 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?



```
TASK [add PHP support for apache] ***********************************************
changed: [192.168.100.127]
changed: [192.168.100.125]

PLAY RECAP **********************************************************************
192.168.100.125            : ok=4    changed=2    unreachable=0    failed=0
kipped=0    rescued=0    ignored=0
192.168.100.127            : ok=4    changed=2    unreachable=0    failed=0
kipped=0    rescued=0    ignored=0
```

The changes that it does that when utilizing apache it will now be able to recognize PHP files also.

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

   [Moznaim/Ansible-HOA-4 (github.com)]

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - A playbook in Ansible is essential because it automates tasks across multiple remote machines, ensuring consistency and efficiency. It allows users to define a series of tasks that can be executed repeatedly without manual intervention, reducing errors.
2. Summarize what we have done on this activity.
   - In this activity, we ran elevated ad hoc commands to make changes on remote machines and used an Ansible playbook to automate those tasks. This ensures that actions are carried out consistently across systems without manual repetition. The playbook streamlines and simplifies the process of managing multiple machines efficiently.