Name: Andaya,Lyka C.	Date Performed: September 11, 2023
Course/Section: CPE 31S4	Date Submitted: September 12, 2023
Instructor: Dr. Taylar	Semester and SY: 1st Sem 2023-2024

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

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

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

Task 1: Run elevated ad hoc commands

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

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

ansible all -m apt -a update_cache=true

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

```
andayalyka@managenode:~/ansibles4_andaya$ ansible all -m apt -a update_cache=tru
e
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
192.168.56.102 | FAILED! => {
    "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)"
}
192.168.56.103 | FAILED! => {
    "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 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.

```
andayalyka@managenode:~/ansibles4_andaya$ ansible all -m apt -a update_cache=tru
e --become --ask-become-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:
192.168.56.102 | CHANGED => {
    "cache_update_time": 1694509790,
    "cache_updated": true,
    "changed": true
}
192.168.56.103 | CHANGED => {
    "cache_update_time": 1694509790,
    "cache_updated": true,
    "changed": true
}
```

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

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

```
andayalyka@managenode:~/ansibles4_andaya$ ansible all -m apt -a name=vim-nox --b
ecome --ask-become-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694509790,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694509790,
    "cache_updated": false,
    "changed": 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?

```
andayalyka@managenode:~/ansibles4_andaya$ which vim
andayalyka@managenode:~/ansibles4_andaya$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security 2:8.0.1453-1ubuntu1.13 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - compact version

andayalyka@managenode:~/ansibles4_andaya$ which vim apt search vim-nox
/usr/bin/apt
```

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.

```
andayalyka@managenode:/var/log/apt$ cat history.log

Start-Date: 2023-09-05  17:18:26
Commandline: apt install python3-pip
Requested-By: andayalyka (1000)
Install: libgcc-7-dev:amd64 (7.5.0-3ubuntu1~18.04, automatic), libmpx2:amd64 (8.
4.0-1ubuntu1~18.04, automatic), python3-dev:amd64 (3.6.7-1~18.04, automatic), py
thon3-distutils:amd64 (3.6.9-1~18.04, automatic), linux-libc-dev:amd64 (4.15.0-2
13.224, automatic), libfakeroot:amd64 (1.22-2ubuntu1, automatic), libc6-dev:amd6
4 (2.27-3ubuntu1.6, automatic), libpython3.6-dev:amd64 (3.6.9-1~18.04ubuntu1.12,
 automatic), libexpat1-dev:amd64 (2.2.5-3ubuntu0.9, automatic), libalgorithm-dif
f-perl:amd64 (1.19.03-1, automatic), libalgorithm-merge-perl:amd64 (0.08-3, auto
matic), libitm1:amd64 (8.4.0-1ubuntu1~18.04, automatic), g++:amd64 (4:7.4.0-1ubu
ntu2.3, automatic), python3-pip:amd64 (9.0.1-2.3~ubuntu1.18.04.8), python3-wheel
:amd64 (0.30.0-0.2ubuntu0.1, automatic), gcc:amd64 (4:7.4.0-1ubuntu2.3, automati
c), libcilkrts5:amd64 (7.5.0-3ubuntu1~18.04, automatic), libasan4:amd64 (7.5.0-3
ubuntu1~18.04, automatic), libquadmath0:amd64 (8.4.0-1ubuntu1~18.04, automatic),
 build-essential:amd64 (12.4ubuntu1, automatic), libstdc++-7-dev:amd64 (7.5.0-3u
buntu1~18.04, automatic), libtsan0:amd64 (8.4.0-1ubuntu1~18.04, automatic), libu
bsan0:amd64 (7.5.0-3ubuntu1~18.04, automatic), g++-7:amd64 (7.5.0-3ubuntu1~18.04
, automatic), make:amd64 (4.1-9.1ubuntu1, automatic), fakeroot:amd64 (1.22-2ubun
tu1, automatic), gcc-7:amd64 (7.5.0-3ubuntu1~18.04, automatic), python3-lib2to3:
amd64 (3.6.9-1~18.04, automatic), liblsan0:amd64 (8.4.0-1ubuntu1~18.04, automati
```

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?

```
andayalyka@managenode:/var/log/apt$ ansible all -m apt -a name=snapd --become --
ask-become-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:
192.168.56.102 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694509790,
    "cache_updated": false,
    "changed": false
}
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 192.168.56.103 should
use /usr/bin/python3, but is using /usr/bin/python for backward compatibility
with prior Ansible releases. A future Ansible release will default to using the
 discovered platform python for this host. See
https://docs.ansible.com/ansible-
core/2.11/reference_appendices/interpreter_discovery.html for more information.
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
```

```
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694509790,
    "cache_updated": false,
    "changed": false
}
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 192.168.56.103 should
use /usr/bin/python3, but is using /usr/bin/python for backward compatibility
with prior Ansible releases. A future Ansible release will default to using the
 discovered platform python for this host. See
https://docs.ansible.com/ansible-
core/2.11/reference_appendices/interpreter_discovery.html for more information.
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "cache_update_time": 1694509790,
    "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.

```
andayalyka@managenode:/var/log/apt$ ansible all -m apt -a "name=snapd state=late
st" --become --ask-become-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:
192.168.56.102 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694509790,
    "cache_updated": false,
    "changed": false
}
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 192.168.56.103 should
use /usr/bin/python3, but is using /usr/bin/python for backward compatibility
with prior Ansible releases. A future Ansible release will default to using the
 discovered platform python for this host. See
https://docs.ansible.com/ansible-
core/2.11/reference_appendices/interpreter_discovery.html for more information.
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "cache_update_time": 1694509790,
    "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:

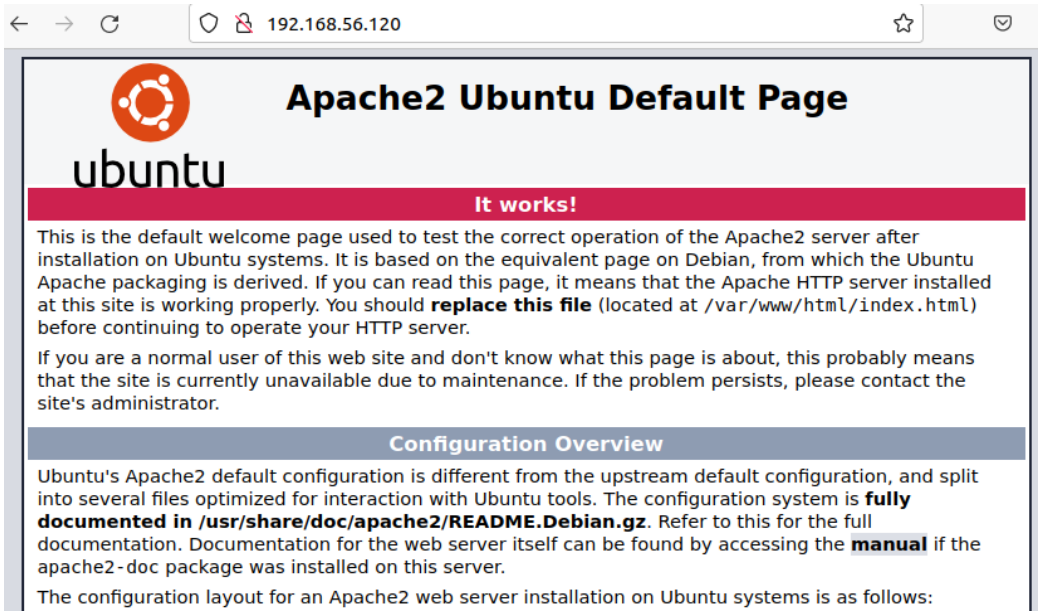
```
  GNU nano 4.8                  install_apache.yml
---
- hosts: all
  become: true
  tasks:

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

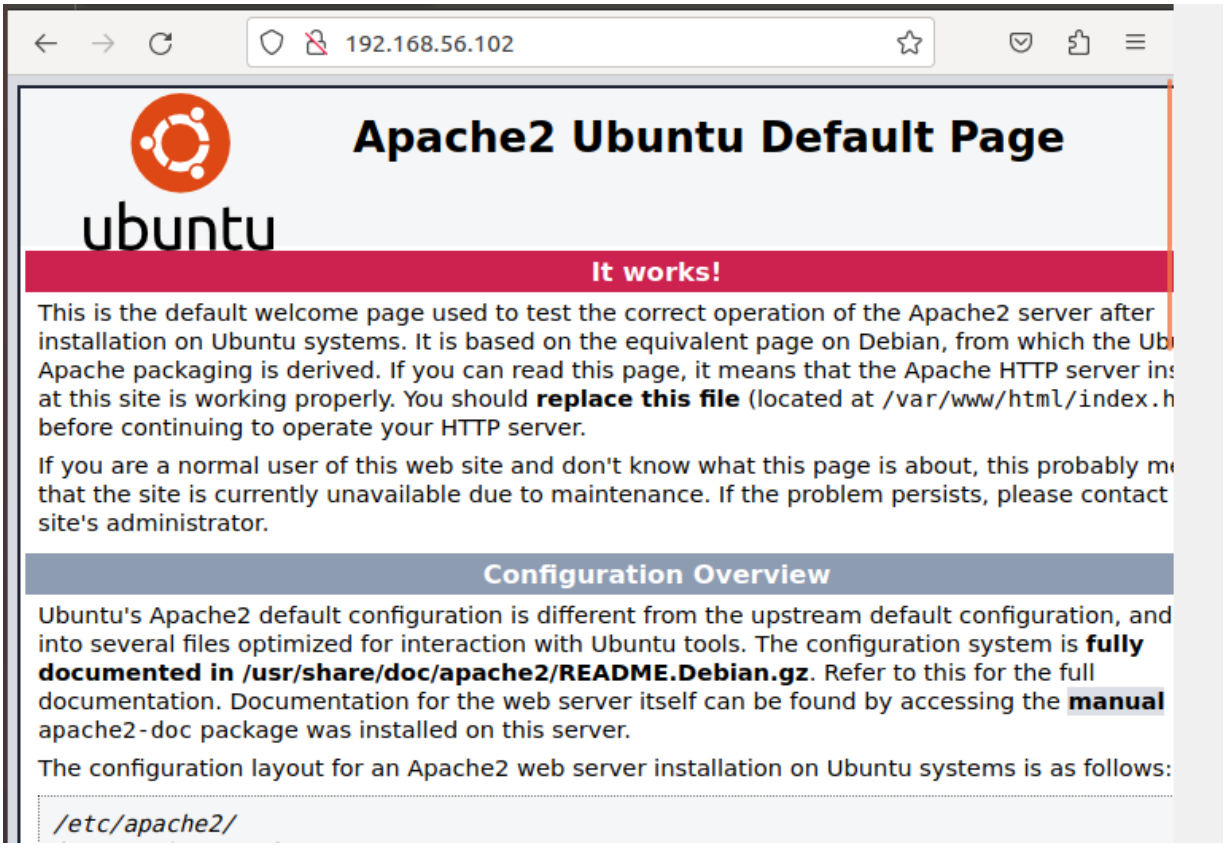
Make sure to save the file. Take note also of the alignments of the texts.

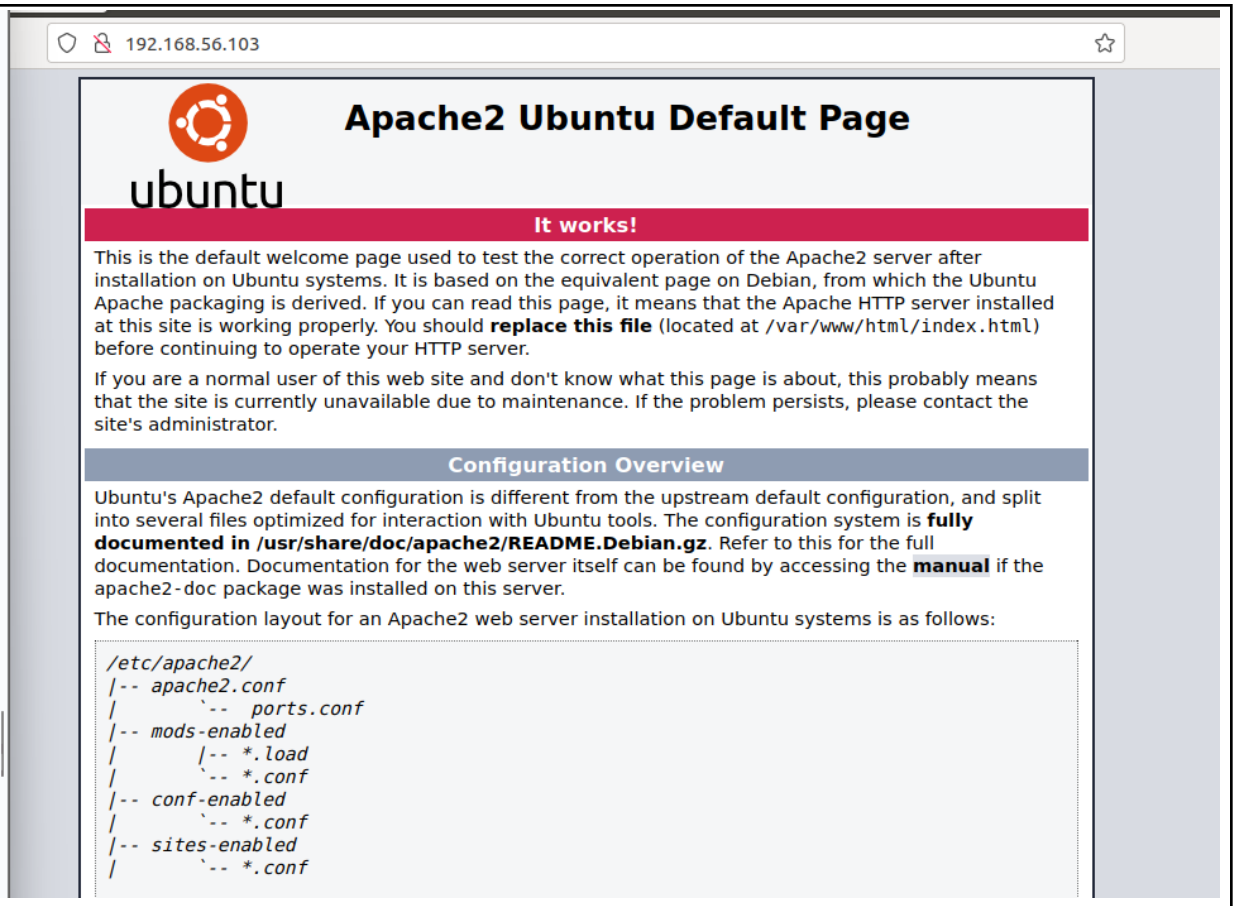
2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.

```
andayalyka@managenode:~/CPE232_ANDAYA$ ansible-playbook --ask-become-pass instal
l_apache.yml
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the
controller starting with Ansible 2.12. Current version: 3.6.9 (default, Mar 10
2023, 16:46:00) [GCC 8.4.0]. This feature will be removed from ansible-core in
version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
BECOME password:

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

TASK [Gathering Facts] *********************************************************
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 192.168.56.103 should
use /usr/bin/python3, but is using /usr/bin/python for backward compatibility
with prior Ansible releases. A future Ansible release will default to using the
 discovered platform python for this host. See
https://docs.ansible.com/ansible-
core/2.11/reference_appendices/interpreter_discovery.html for more information.
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.56.103]
ok: [192.168.56.102]

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

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
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:


192.168.56.102
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 Ubu
Apache packaging is derived. If you can read this page, it means that the Apache HTTP server ins
at this site is working properly. You should replace this file (located at /var/www/html/index.h
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 m
that the site is currently unavailable due to maintenance. If the problem persists, please contact
site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and
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
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/



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?
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?

```
with prior Ansible releases. A future Ansible release will default to using the
 discovered platform python for this host. See
https://docs.ansible.com/ansible-
core/2.11/reference_appendices/interpreter_discovery.html for more information.
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.56.103]
ok: [192.168.56.102]

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

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

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

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

```
 This feature will be removed in version 2.12. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.56.103]
ok: [192.168.56.102]

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

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

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

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

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

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - **A playbook is a structured document or manual that outlines a series of steps, instructions, and procedures to accomplish a specific task or set of tasks. It is commonly used in various fields, including sports, business, project management, and cybersecurity. Playbooks provide a standardized approach to performing tasks. This ensures that procedures are carried out consistently and uniformly, reducing the likelihood of errors or omissions. Playbooks are valuable tools for standardizing processes, transferring knowledge, managing risks, and ensuring efficiency in various domains. They contribute to organizational effectiveness and are particularly crucial in tasks where consistency, accuracy, and compliance are paramount.**
2. Summarize what we have done on this activity.
   - **Executing commands that enact changes on remote machines involves sending directives or instructions from a central control system to effect**

alterations, configurations, or execute tasks on other devices within a network. This constitutes a fundamental principle in system administration and automation. When making changes to remote systems, it is crucial to use caution and secure the appropriate authorization because incorrect or illegal changes may have unanticipated effects and even cause service outages. To further assure the security and integrity of the remote management process, it is crucial to employ strong security measures, such as secure authentication and encrypted communication channels. Utilizing a playbook within Ansible represents a potent method for automating and coordinating intricate operations and commands across multiple remote systems. A playbook, structured in YAML format, encompasses a sequence of tasks and directives for Ansible to follow. Playbooks empower you to define the desired configuration of your infrastructure and automate actions to attain that desired state.

CONCLUSION:

I have learned that when altering remote systems, caution and proper authorization are essential to avoid unintended consequences or service disruptions. To enhance security, robust measures like secure authentication and encrypted communication channels should be implemented. Using a playbook in Ansible is a powerful way to automate complex operations across multiple remote systems. A playbook, written in YAML, outlines a series of tasks and instructions for Ansible to execute. It allows you to define and automate the desired configuration of your infrastructure. This approach streamlines management and ensures systems operate in the intended state.