

<b>Name:</b> John Enrico M. Amadeo	<b>Date Performed:</b> 9/11/23
<b>Course/Section:</b> CPE232/CPE31S4	<b>Date Submitted:</b>
<b>Instructor:</b> Dr Jonathan Taylar	<b>Semester and SY:</b> 2nd Sem 2023-2024
<b>Activity 4: Running Elevated Ad hoc Commands</b>	
<b>1. Objectives:</b> 1.1 Use commands that makes changes to remote machines 1.2 Use playbook in automating ansible commands	
<b>2. Discussion:</b>  <i>Provide screenshots for each task.</i>  <b>Elevated Ad hoc commands</b> 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.  <b>Playbooks</b> record and execute <b>Ansible's</b> 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. <a href="#">Working with playbooks — Ansible Documentation</a>	
<b>Task 1: Run elevated ad hoc commands</b>  1. Locally, we use the command <i>sudo apt update</i> when we want to download package information from all configured resources. The sources often defined in <i>/etc/apt/sources.list</i> file and other files located in <i>/etc/apt/sources.list.d/</i> 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?

The command got an error

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

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

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

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.

```
amadeoubuntu@manageNode:~/bahayactivities$ ansible all -m apt -a name=vim-nox -
-become --ask-become-pass
BECOME password:
192.168.56.101 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694437827,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following additional packages will be installed:\n fo
nts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n rub
y-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n rubygems-integra
tion vim-runtime\nSuggested packages:\n apache2 | lighttpd | httpd ri ruby-dev
bundler cscope vim-doc\nThe following NEW packages will be installed:\n fonts
-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n ruby-n
et-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n rubygems-integratio
n vim-nox vim-runtime\n0 upgraded, 15 newly installed, 0 to remove and 23 not u
pgraded.\nNeed to get 17.5 MB of archives.\nAfter this operation, 76.4 MB of ad
ditional disk space will be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu ja
mmy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\nGet:2 http://ph.archive.ubuntu
.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5936 B]\nGet:3 http
://ph.archive.ubuntu.com/ubuntu jammy/main amd64 libjs-jquery all 3.6.0+dfsg+~3
.5.13-1 [321 kB]\nGet:4 http://ph.archive.ubuntu.com/ubuntu jammy/universe amd6
4 liblua5.2-0 amd64 5.2.4-2 [125 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu
jammy/main amd64 rubygems-integration all 1.18 [5336 B]\nGet:6 http://ph.archi
```

- 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?

```
amadeoubuntu@manageNode:~/bahayactivities$ which vim
amadeoubuntu@manageNode:~/bahayactivities$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.11 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [install
ed,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`.

```
amadeoubuntu@manageNode:~$ cd /var/log
amadeoubuntu@manageNode:/var/log$ ls
alternatives.log      cups                  gdm3                 speech-dispatcher
alternatives.log.1    dist-upgrade         gpu-manager.log      syslog
apt                   dmesg                hp                   syslog.1
auth.log              dmesg.0              installer             syslog.2.gz
auth.log.1            dmesg.1.gz           journal              ubuntu-advantage.log
auth.log.2.gz         dmesg.2.gz           kern.log              ubuntu-advantage.log.1
boot.log              dmesg.3.gz           kern.log.1           ufw.log
boot.log.1            dpkg.log              kern.log.2.gz        unattended-upgrades
bootstrap.log         dpkg.log.1           lastlog              wtmp
btm                   faillog              openvpn
btm.1                 fontconfig.log       private

amadeoubuntu@manageNode:/var/log$ cd apt
amadeoubuntu@manageNode:/var/log/apt$ less history.log

[1]+  Stopped                  less history.log
amadeoubuntu@manageNode:/var/log/apt$ cat history.log

Start-Date: 2023-09-03  21:14:19
Commandline: apt install net-tools
Requested-By: amadeoubuntu (1000)
Install: net-tools:amd64 (1.60+git20181103.0eebece-1ubuntu5)
End-Date: 2023-09-03  21:14:20

Start-Date: 2023-09-11  20:03:26
Commandline: apt-get install xclip
Requested-By: amadeoubuntu (1000)
Install: xclip:amd64 (0.13-2)
```

The `history.log` is like the `history` command but it tells the history of package management actions like installations, updates and removal of packages

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`

```

amadeubuntu@manageNode:~$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694437831,
    "cache_updated": false,
    "changed": false
}
192.168.56.101 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694437827,
    "cache_updated": false,
    "changed": false
}
amadeubuntu@manageNode:~$

```

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

the command is success as seen in the screenshot, the purpose of this command is to prompt the sudo password to executed the installation with elevated privileges.

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.

```

amadeubuntu@manageNode:~$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694437831,
    "cache_updated": false,
    "changed": false
}
192.168.56.101 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694437827,
    "cache_updated": false,
    "changed": false
}

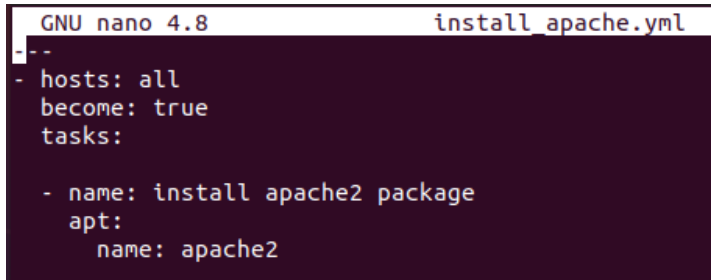
```

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.

```
amadeoubuntu@manageNode: ~/CPE232_AMADEO_HOME
GNU nano 6.2                                install_apache.yml
---
- hosts:all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2

[ Wrote 8 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

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

```
amadeoubuntu@manageNode:~/CPE232_AMADEO_HOME$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.101]

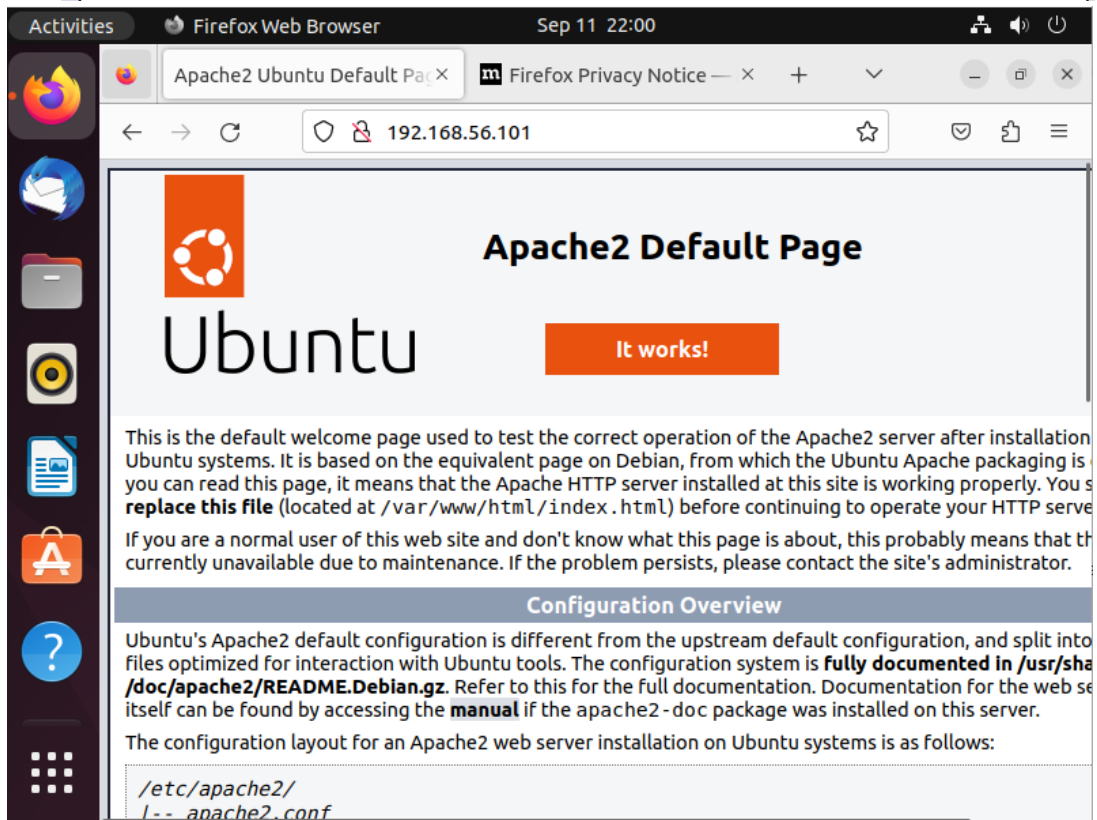
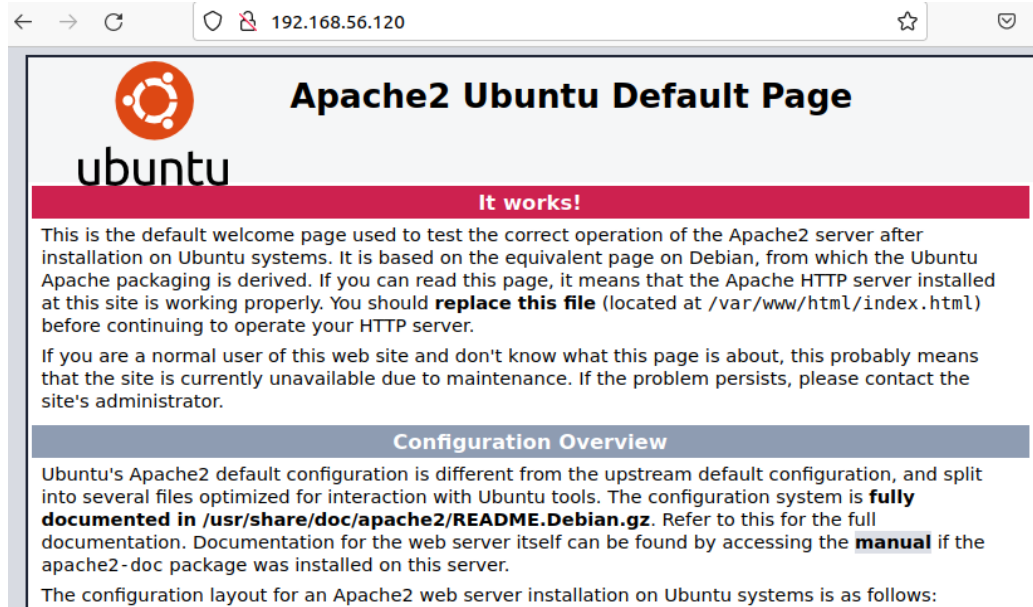
TASK [install apache2 package] *****
changed: [192.168.56.101]
changed: [192.168.56.103]

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

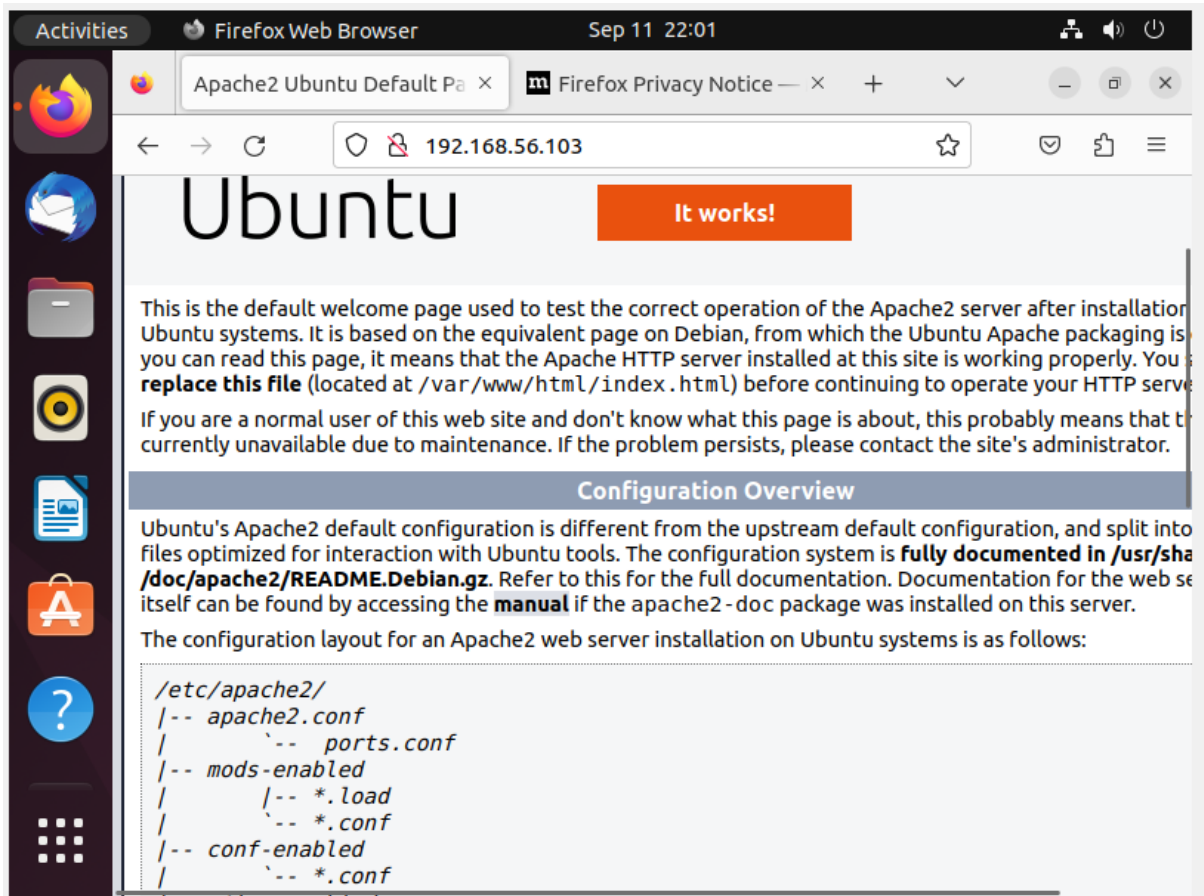
the command `ansible-playbook --ask-become-pass install_apache.yml` installs the apache2 package to the 2 remote servers



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.







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

```
GNU nano 6.2 install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install python3-pip package
    apt:
      name: test
```

```

amadeoubuntu@manageNode:~/CPE232_AMADEO_HOME$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.101]

TASK [install python3-pip package] *****
fatal: [192.168.56.101]: FAILED! => {"changed": false, "msg": "No package matching 'test' is available"}
fatal: [192.168.56.103]: FAILED! => {"changed": false, "msg": "No package matching 'test' is available"}

PLAY RECAP *****
192.168.56.101      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0
ignored=0
192.168.56.103      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0
ignored=0

```

The output is failed because there's no test apt

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?

```
amadeoubuntu@manageNode:~/CPE232_AMADEO_HOME$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.103]
ok: [192.168.56.101]

TASK [update repository index] *****
changed: [192.168.56.103]
changed: [192.168.56.101]

TASK [install apache2 package] *****
ok: [192.168.56.103]
ok: [192.168.56.101]

PLAY RECAP *****
192.168.56.101      : 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
```

yes according to the task [update repository index] and play recap there is a change

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?

```
amadeoubuntu@manageNode:~/CPE232_AMADEO_HOME$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.56.101]
ok: [192.168.56.103]

TASK [update repository index] *****
changed: [192.168.56.101]
changed: [192.168.56.103]

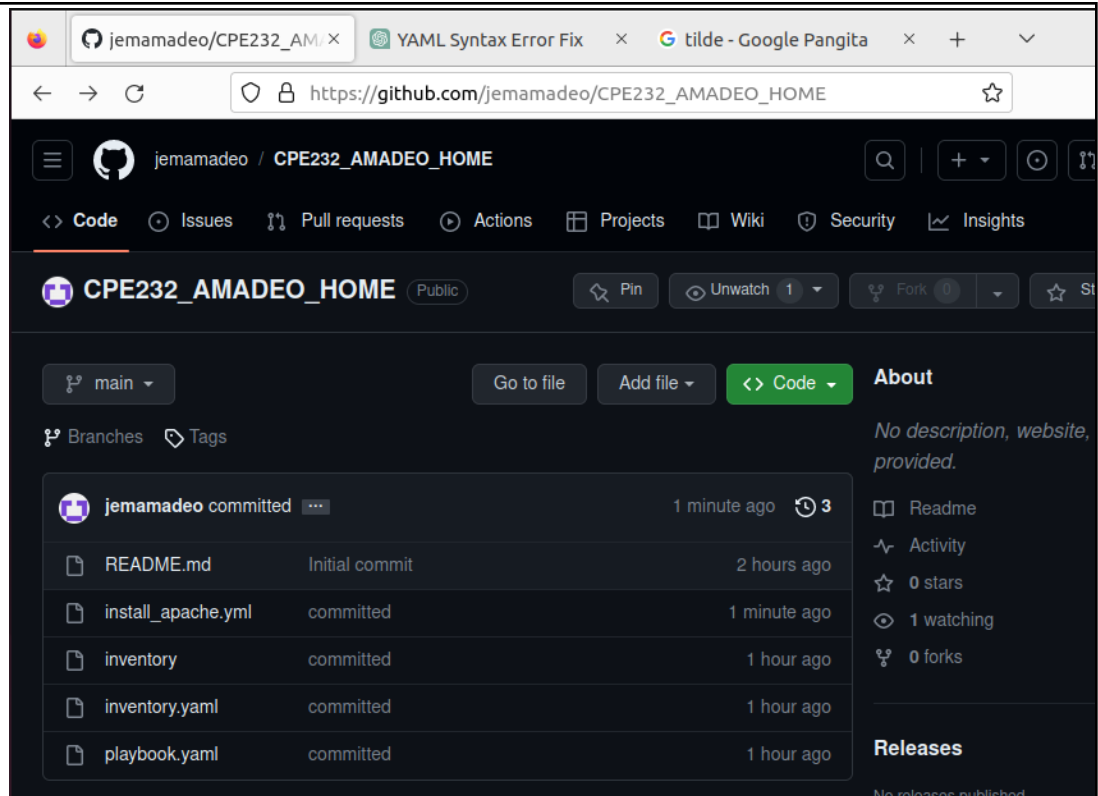
TASK [install apache2 package] *****
ok: [192.168.56.103]
ok: [192.168.56.101]

TASK [add PHP support for apache] *****
changed: [192.168.56.103]
changed: [192.168.56.101]

PLAY RECAP *****
192.168.56.101      : 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
```

there is 2 new changes when the command is executed which is the task that update the repository index and the new task which is add php support for apache

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



[https://github.com/jemamadeo/CPE232\\_AMADEO\\_HOME](https://github.com/jemamadeo/CPE232_AMADEO_HOME)

### Reflections:

Answer the following:

1. What is the importance of using a playbook?

Playbook plays a big role in the System Administration wherein its like a script and manages other computers and it also automates a certain task to other computers which is my control nodes.

2. Summarize what we have done on this activity.

in this activity i make use of the ansible commands wherein i install some packages like vim-nox and snapd. In the last task i created a playbook and its task is to update repository index, install apache2 package and add PHP support for apache.

### Conclusion:

In this activity i learned how important playbook is and also the use of ansible to update instead of sudo apt update. in Task 1 i learned how to troubleshoot and install the packages vim-nox and snapd. In Task 2 i learned how important playbook is and i created an automated script where its task is to update repository index, install apache2 package and add PHP support for apache.

