

Name: JOHN ERA ROLDAN	Date Performed: 9/12/2025
Course/Section: CPE212 - CPE31S2	Date Submitted: 9/12/2025
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem 2025-2026
Activity 4: Running Elevated Ad hoc Commands	
1. Objectives: 1.1 Use commands that makes changes to remote machines 1.2 Use playbook in automating ansible commands	
2. Discussion:	

Provide screenshots for each task.

Elevated Ad hoc commands

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

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

Task 1: Run elevated ad hoc commands

- 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?

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 elevates 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.

```
roldan@workstation:~/CPE212$ ansible all -m apt -a update_cache=true --become  
BECOME password:  
192.168.56.119 | CHANGED => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "cache_update_time": 1757669405,  
    "cache_updated": true,  
    "changed": true  
}  
192.168.56.118 | CHANGED => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "cache_update_time": 1757669412,  
    "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 change 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. **a**

```
roldan@workstation:~/CPE212$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.119 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669405,
    "cache_updated": false,
    "changed": false
}
192.168.56.118 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669412,
    "cache_updated": 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?

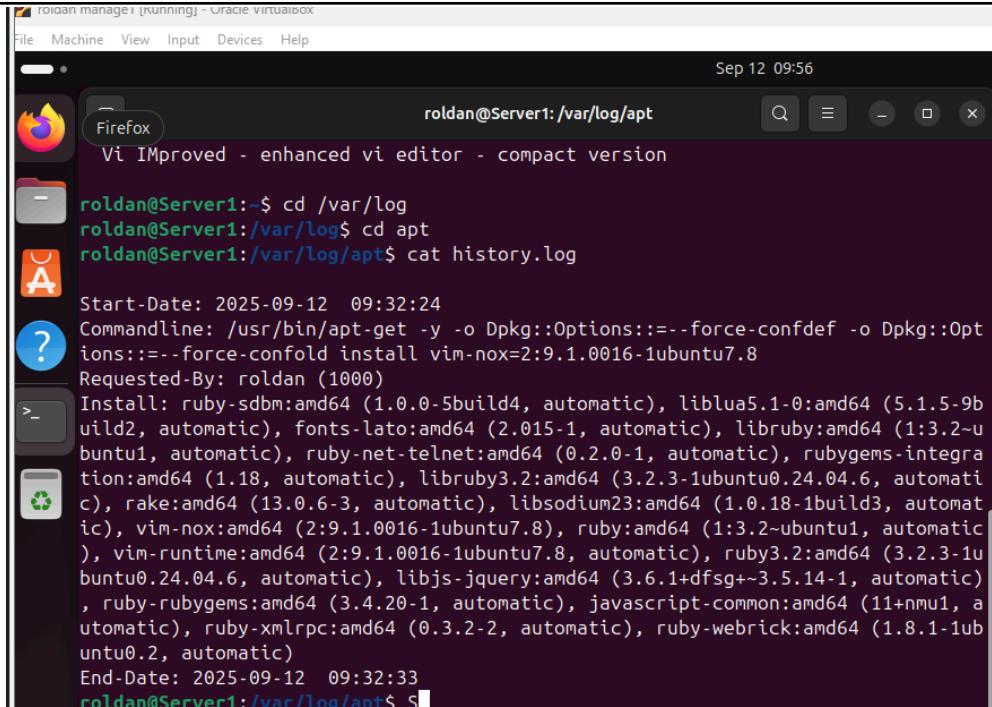
```
roldan@Server1:~$ which vim
/usr/bin/vim
roldan@Server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates,noble-security,now 2:9.1.0016-1ubuntu7.8 amd64 [installed]
  Vi IMproved - enhanced vi editor - with scripting languages support

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

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

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

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*.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "roldan@Server1:/var/log/apt". The terminal content displays the output of the command "cat history.log". The output shows a package installation history from September 12, 2025, at 09:32:24. It lists several packages installed via apt-get, including vim-nox, ruby-sdbm, fonts-lato, libruby, ruby-net-telnet, rubygems-integration, rake, libsodium23, vim-nox, ruby, vim-runtime, libjs-jquery, ruby-rubygems, ruby-xmlrpc, and ruby-webrick.

```

roldan@Server1:~$ cd /var/log
roldan@Server1:/var/log$ cd apt
roldan@Server1:/var/log/apt$ cat history.log

Start-Date: 2025-09-12 09:32:24
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold install vim-nox=2:9.1.0016-1ubuntu7.8
Requested-By: roldan (1000)
Install: ruby-sdbm:amd64 (1.0.0-5build4, automatic), liblua5.1-0:amd64 (5.1.5-9b
uild2, automatic), fonts-lato:amd64 (2.015-1, automatic), libruby:amd64 (1:3.2~u
buntu1, automatic), ruby-net-telnet:amd64 (0.2.0-1, automatic), rubygems-integra
tion:amd64 (1.18, automatic), libruby3.2:amd64 (3.2.3-1ubuntu0.24.04.6, automati
c), rake:amd64 (13.0.6-3, automatic), libsodium23:amd64 (1.0.18-1build3, automati
c), vim-nox:amd64 (2:9.1.0016-1ubuntu7.8), ruby:amd64 (1:3.2~ubuntu1, automatic
), vim-runtime:amd64 (2:9.1.0016-1ubuntu7.8, automatic), ruby3.2:amd64 (3.2.3-1u
buntu0.24.04.6, automatic), libjs-jquery:amd64 (3.6.1+dfsg+-3.5.14-1, automatic)
, ruby-rubygems:amd64 (3.4.20-1, automatic), javascript-common:amd64 (11+nmu1, a
utomatic), ruby-xmlrpc:amd64 (0.3.2-2, automatic), ruby-webrick:amd64 (1.8.1-1ub
untu0.2, automatic)
End-Date: 2025-09-12 09:32:33
roldan@Server1:/var/log/apt$ S

```



```

roldan@Server2:~$ cd /var/log
roldan@Server2:/var/log$ cd apt
roldan@Server2:/var/log/apt$ cat history.log

Start-Date: 2025-09-05 10:00:43
Commandline: /usr/bin/unattended-upgrade
Upgrade: libtiff6:amd64 (4.5.1+git230720-4ubuntu2.2, 4.5.1+git230720-4ubuntu2.3)
End-Date: 2025-09-05 10:00:44

Start-Date: 2025-09-05 10:00:46
Commandline: /usr/bin/unattended-upgrade
Install: linux-modules-extra-6.14.0-29-generic:amd64 (6.14.0-29.29~24.04.1, aut
omatic), linux-tools-6.14.0-29-generic:amd64 (6.14.0-29.29~24.04.1, automatic),
linux-image-6.14.0-29-generic:amd64 (6.14.0-29.29~24.04.1, automatic), linux-hwe-
6.14-headers-6.14.0-29:amd64 (6.14.0-29.29~24.04.1, automatic), linux-hwe-6.14-t
ools-6.14.0-29:amd64 (6.14.0-29.29~24.04.1, automatic), linux-headers-6.14.0-29-
generic:amd64 (6.14.0-29.29~24.04.1, automatic), linux-modules-6.14.0-29-generic
:amd64 (6.14.0-29.29~24.04.1, automatic)
Upgrade: linux-headers-generic-hwe-24.04:amd64 (6.14.0-27.27~24.04.1, 6.14.0-29.
29~24.04.1), linux-generic-hwe-24.04:amd64 (6.14.0-27.27~24.04.1, 6.14.0-29.29~2
4.04.1), linux-image-generic-hwe-24.04:amd64 (6.14.0-27.27~24.04.1, 6.14.0-29.29~2
4.04.1)
End-Date: 2025-09-05 10:01:30

```

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?

```
roldan@workstation:~/CPE212$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.119 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669405,
    "cache_updated": false,
    "changed": false
}
192.168.56.118 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669412,
    "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.

```
roldan@workstation:~/CPE212$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.119 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669405,
    "cache_updated": false,
    "changed": false
}
192.168.56.118 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1757669412,
    "cache_updated": false,
    "changed": false
}
roldan@workstation:~/CPE212$
```

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

```
roldan@workstation:~$ sudo cp -r CPE212/* CPE232_Roldan
roldan@workstation:~$ cd CPE232_Roldan
roldan@workstation:~/CPE232_Roldan$ ls
ansible.cfg  CPE212  inventory.ini  inventory.yaml  README.md
roldan@workstation:~/CPE232_Roldan$ git add .
roldan@workstation:~/CPE232_Roldan$ git commit -m "HOA 4.1 Task 1"
[main 2c73645] HOA 4.1 Task 1
 6 files changed, 32 insertions(+)
  create mode 100644 CPE212/ansible.cfg
  create mode 100644 CPE212/inventory.ini
  create mode 100644 CPE212/inventory.yaml
  create mode 100644 ansible.cfg
  create mode 100644 inventory.ini
  create mode 100644 inventory.yaml
roldan@workstation:~/CPE232_Roldan$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 655 bytes | 655.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:johnera98/CPE232_Roldan.git
  b52a5fa..2c73645  main -> main
roldan@workstation:~/CPE232_Roldan$
```

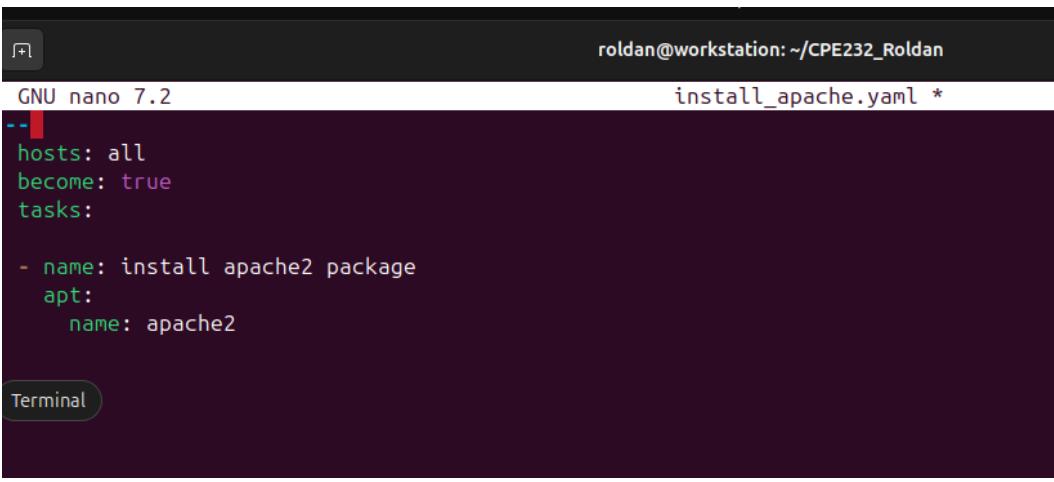
Task 2: Writing our First Playbook

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.yaml *
---
- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: apache2
```

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

```
roldan@workstation:~/CPE232_Roldan$ sudo nano install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ ansible-playbook --ask-become-pass install_apache.yaml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.119]
ok: [192.168.56.118]

TASK [install apache2 package] ****
changed: [192.168.56.119]
changed: [192.168.56.118]

PLAY RECAP ****
192.168.56.118 : ok=2    changed=1    unreachable=0    failed=0    skipped=0
192.168.56.119 : ok=2    changed=1    unreachable=0    failed=0    skipped=0

roldan@workstation:~/CPE232_Roldan$
```

- 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:

Sep 12 10:41

buntu Default Pac × Firefox Privacy Notice — / × +

Not Secure http://192.168.56.118 ☆

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.

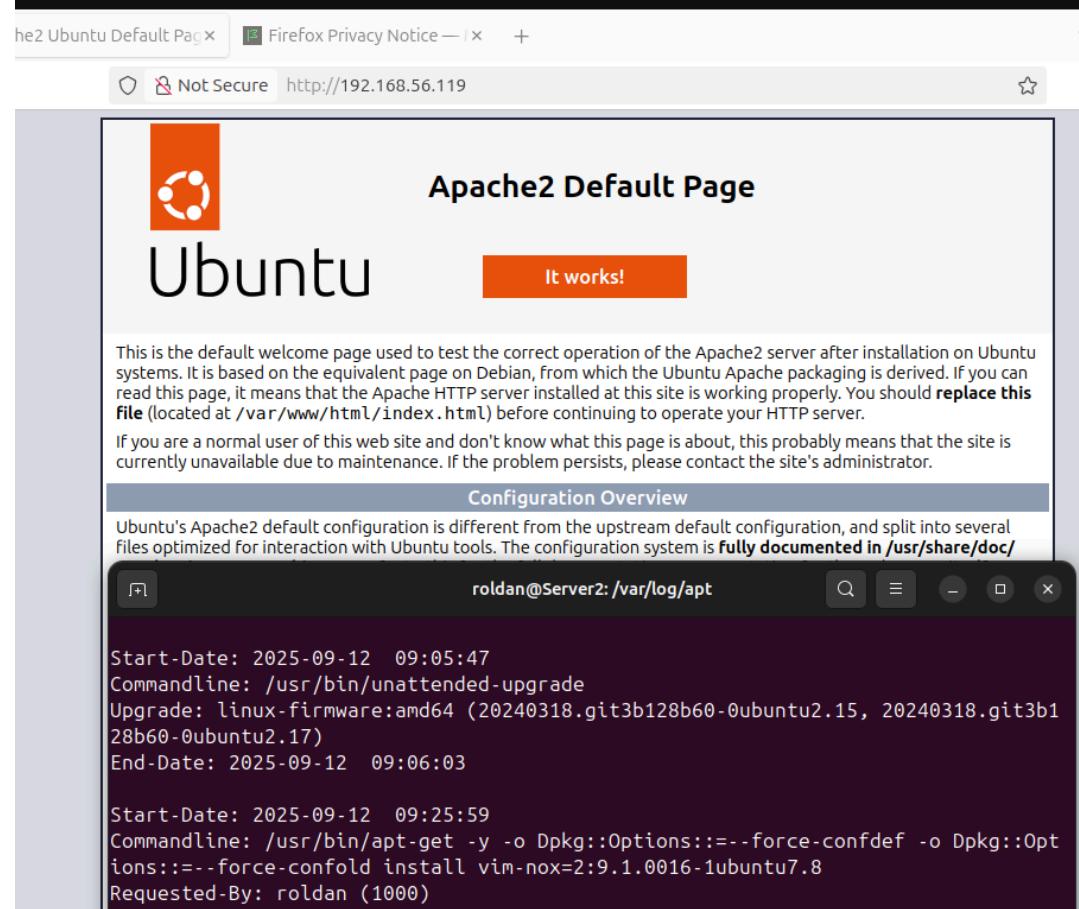
Configuration Overview

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.

```
+ roldan@Server1: /var/log/apt Q - X
Vi IMproved - enhanced vi editor - compact version

roldan@Server1:~$ cd /var/log
roldan@Server1:/var/log$ cd apt
roldan@Server1:/var/log/apt$ cat history.log

Start-Date: 2025-09-12 09:32:24
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold install vim-nox=2:9.1.0016-1ubuntu7.8
Requested-By: roldan (1000)
Install: ruby-sdbm:amd64 (1.0.0-5build4, automatic), liblua5.1-0:amd64 (5.1.5-9b)
```



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?

```
Firefox                               roldan@workstation:~/CPE232_Roldan
GNU nano 7.2                           install_apache.yaml *
---
- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: xdpackage
```

```
roldan@workstation:~/CPE232_Roldan$ sudo nano install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ ansible-playbook -ask-become-pass install_apache.yaml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.119]
ok: [192.168.56.118]

TASK [install apache2 package] ****
fatal: [192.168.56.118]: FAILED! => {"changed": false, "msg": "No package matching 'xdpackage' found"}
fatal: [192.168.56.119]: FAILED! => {"changed": false, "msg": "No package matching 'xdpackage' found"}

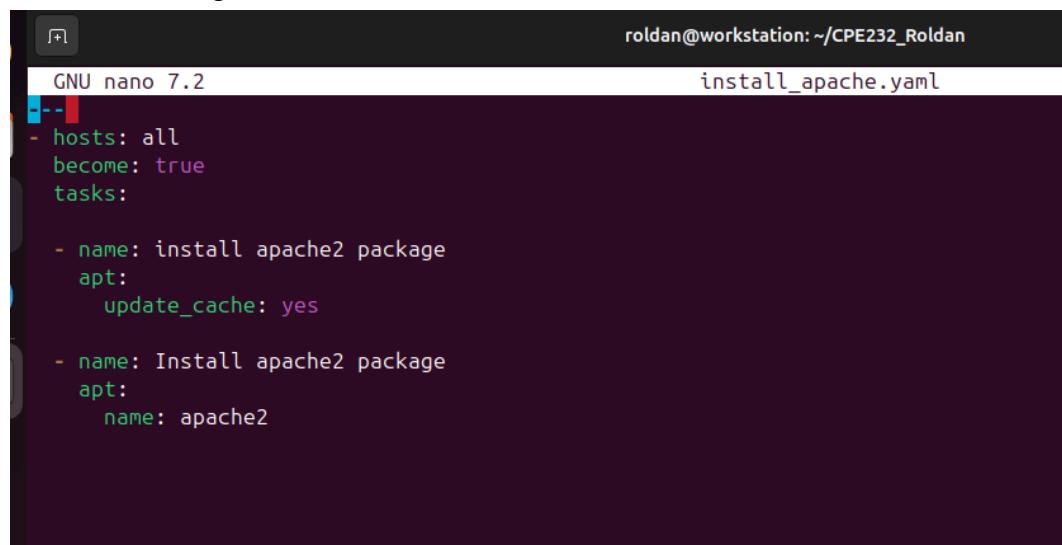
PLAY RECAP ****
192.168.56.118 : ok=1    changed=0    unreachable=0    failed=1    skipped=0
192.168.56.119 : ok=1    changed=0    unreachable=0    failed=1    skipped=0

roldan@workstation:~/CPE232_Roldan$
```

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.



The screenshot shows a terminal window with the title bar "roldan@workstation: ~/CPE232_Roldan". The window contains the YAML code from the previous block, with some syntax highlighting. The file is named "install_apache.yaml". The terminal interface includes a status bar at the bottom.

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

```
roldan@workstation:~/CPE232_Roldan$ sudo nano install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ ansible-playbook --ask-become-pass install_apache.yaml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.119]
ok: [192.168.56.118]

TASK [install apache2 package] ****
changed: [192.168.56.119]
changed: [192.168.56.118]

TASK [Install apache2 package] ****
ok: [192.168.56.118]
ok: [192.168.56.119]

PLAY RECAP ****
192.168.56.118 : ok=3    changed=1    unreachable=0    failed=0    skipped=0
192.168.56.119 : ok=3    changed=1    unreachable=0    failed=0    skipped=0

roldan@workstation:~/CPE232_Roldan$
```

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                               roldan@workstation: ~/CPE232_Roldan
-----                                     install_apache.yaml *
- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      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?

```
roldan@workstation:~/CPE232_Roldan$ PLAY RECAP ****
192.168.56.118      : ok=3    changed=1    unreachable=0    failed=0    skipped=0
192.168.56.119      : ok=3    changed=1    unreachable=0    failed=0    skipped=0

roldan@workstation:~/CPE232_Roldan$ sudo nano install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ sudo nano install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ ansible-playbook --ask-become-pass install_apache.yaml
BECOME password:

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [192.168.56.119]
ok: [192.168.56.118]

TASK [install apache2 package] ****
changed: [192.168.56.118]
changed: [192.168.56.119]

TASK [Install apache2 package] ****
ok: [192.168.56.118]
ok: [192.168.56.119]

TASK [add PHP support for apache] ****
changed: [192.168.56.118]
changed: [192.168.56.119]

PLAY RECAP ****
192.168.56.118      : ok=4    changed=2    unreachable=0    failed=0    skipped=0
192.168.56.119      : ok=4    changed=2    unreachable=0    failed=0    skipped=0

roldan@workstation:~/CPE232_Roldan$
```

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

```
roldan@workstation:~/CPE232_Roldan$ git commit -m "Finished HOA4.1"
[main 1471580] Finished HOA4.1
 1 file changed, 16 insertions(+)
  create mode 100644 install_apache.yaml
roldan@workstation:~/CPE232_Roldan$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 413 bytes | 413.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:johnera98/CPE232_Roldan.git
 2c73645..1471580  main -> main
roldan@workstation:~/CPE232_Roldan$
```

Reflections:

Answer the following:

1. What is the importance of using a playbook?
 - The reason why using a playbook is important is that it will enable you to automate processes on your servers in a well-organized manner. You have to write down the steps once in the playbook instead of rewriting them each time you want to do the same steps, and Ansible runs them automatically. This saves time, minimizes mistakes and ensures that all things are done in the same manner on a consistent basis. It is particularly handy in instances of dealing with a large number of servers or repeating the tasks frequently.
2. Summarize what we have done on this activity.
 - In this activity, we were taught to use commands to do changes on the remote machines. That is to say that we ran instructions originating on our computer to update or repair other servers without having to log into each one of them individually. We also learnt how to automate these commands using a playbook. We don't want to type commands over and over, we instead enter them into a playbook, a list of instructions we can have Ansible follow. This simplifies and increases the speed of managing a large number of servers.