

Name: Ferrer, Joseph Bryan M.	Date Performed: 9/11/2023
Course/Section: CPE31S6	Date Submitted: 9/14/2023
Instructor: Dr. Jonathan V. Taylar	Semester and SY: 1st Semester / 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: <i>Provide screenshots for each task.</i> 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 <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: <i>ansible all -m apt -a update_cache=true</i> What is the result of the command? Is it successful?	

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible all -m apt -a update_cache=true
192.168.56.103 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)"
}
192.168.56.102 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)"
}
```

The result of the command will lead to error. The error said that it failed to lock apt for exclusive operations such as locking the directory and it denied permission to enter.

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.

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
  "cache_update_time": 1694429929,
  "cache_updated": true,
  "changed": true
}
192.168.56.103 | CHANGED => {
  "cache_update_time": 1694429937,
  "cache_updated": true,
  "changed": true
}
```

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

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
  "cache_update_time": 1694429929,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following package was automatically installed and is no longer required:\n libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe following additional packages will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ruby-minitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygems-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd tcl8.6 ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ruby-minitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygems-integration vim-nox vim-runtime\n0 upgraded, 17 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 13.8 MB of archives.\nAfter this operation, 64.5 MB of additional disk space will be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2 [2698 kB]\nGet:2 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all 11 [6066 B]\nGet:3 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all 3.2.1-1 [152 kB]\nGet:4 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 liblua5.2-0 amd64 5.2.4-1.1build1 [108 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration all 1.11 [4994 B]\nGet:6 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main
```

- 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?

```
josephferrer@workstation:~/CPE232_Ferrer$ which vim
josephferrer@workstation:~/CPE232_Ferrer$ 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 [installed]
Vi IMproved - enhanced vi editor - compact version
```

Yes, the command was successful.

- 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`.

```
josephferrer@workstation:~/CPE232_Ferrer$ cd /var/log
josephferrer@workstation:/var/log$ ls
alternatives.log  cups          kern.log       ubuntu-advantage.log
alternatives.log.1 dist-upgrade  kern.log.1    ubuntu-advantage.log.1
apt              dpkg.log     kern.log.2.gz ufw.log
auth.log         dpkg.log.1   kern.log.3.gz ufw.log.1
auth.log.1      faillog      lastlog       ufw.log.2.gz
auth.log.2.gz   fontconfig.log speech-dispatcher ufw.log.3.gz
auth.log.3.gz   gdm3        syslog        unattended-upgrades
boot.log        gpu-manager.log syslog.1       wtmp
bootstrap.log   hp          syslog.2.gz   wtmp.1
btmtp           installer    syslog.3.gz
btmtp.1         journal     tallylog
josephferrer@workstation:/var/log$ cd apt
josephferrer@workstation:/var/log/apt$ ls
eipp.log.xz  history.log  history.log.1.gz  term.log  term.log.1.gz
josephferrer@workstation:/var/log/apt$ cat history.log

Start-Date: 2023-09-11 17:02:49
Commandline: apt install python3-pip
Requested-By: josephferrer (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),
python3-distutils:amd64 (3.6.9-1~18.04, automatic), linux-libc-dev:amd64 (4.15.
0-213.224, automatic), libfakeroot:amd64 (1.22-2ubuntu1, automatic), libc6-dev:
amd64 (2.27-3ubuntu1.6, automatic), libpython3.6-dev:amd64 (3.6.9-1~18.04ubuntu
1.12, automatic), libexpat1-dev:amd64 (2.2.5-3ubuntu0.9, automatic), libalgorit
hm-diff-perl:amd64 (1.19.03-1, automatic), libalgorithm-merge-perl:amd64 (0.08-
3, automatic), libitm1:amd64 (8.4.0-1ubuntu1~18.04, automatic), g++:amd64 (4:7.
```

As you can see, you can see the recent apt activity (installation, removal, upgrade) in the `history.log`.

3. This time, we will install a package called `snapt`. 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=snapt --become --ask-become-pass` Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible all -m apt -a name=snapt --be
come --ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
  "cache_update_time": 1694429937,
  "cache_updated": false,
  "changed": false
}
192.168.56.102 | SUCCESS => {
  "cache_update_time": 1694429929,
  "cache_updated": false,
  "changed": false
}
```

Yes it is success but it doesn't change anything in the remote servers.

3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694429929,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694429937,
    "cache_updated": false,
    "changed": false
}
```

It is also successful however it doesn't change anything in the remote servers.

Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.

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

```
josephferrer@workstation:~/CPE232_Ferrer$ git add ansible.cfg
josephferrer@workstation:~/CPE232_Ferrer$ git add inventory
josephferrer@workstation:~/CPE232_Ferrer$ git push
Everything up-to-date
josephferrer@workstation:~/CPE232_Ferrer$ git status
On branch main
Your branch is up to date with 'origin/main'.

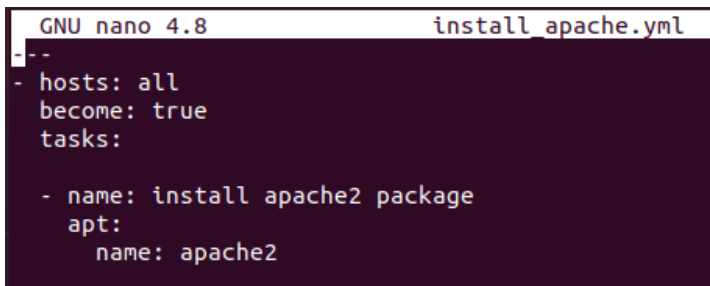
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   ansible.cfg
        modified:   inventory
```

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.

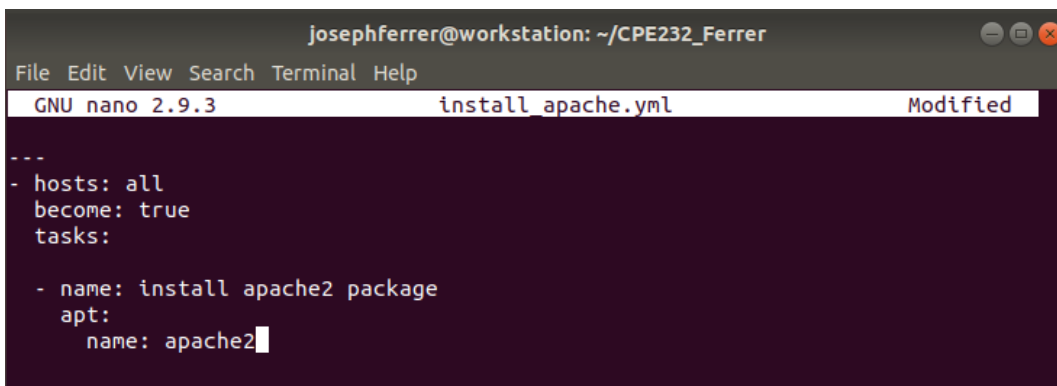
When the editor appears, type the following:



```
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.



```
josephferrer@workstation: ~/CPE232_Ferrer
File Edit View Search Terminal Help
GNU nano 2.9.3      install_apache.yml      Modified
---
- hosts: all
  become: true
  tasks:

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

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

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible-playbook --ask-become-pass in
stall_apache.yml
BECOME password:

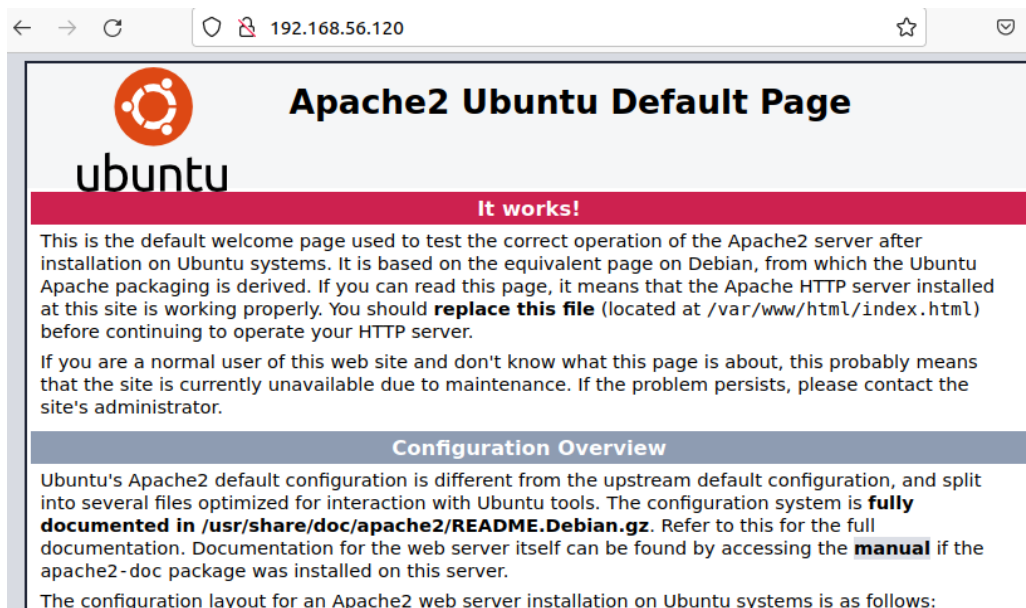
PLAY [all] *****
*

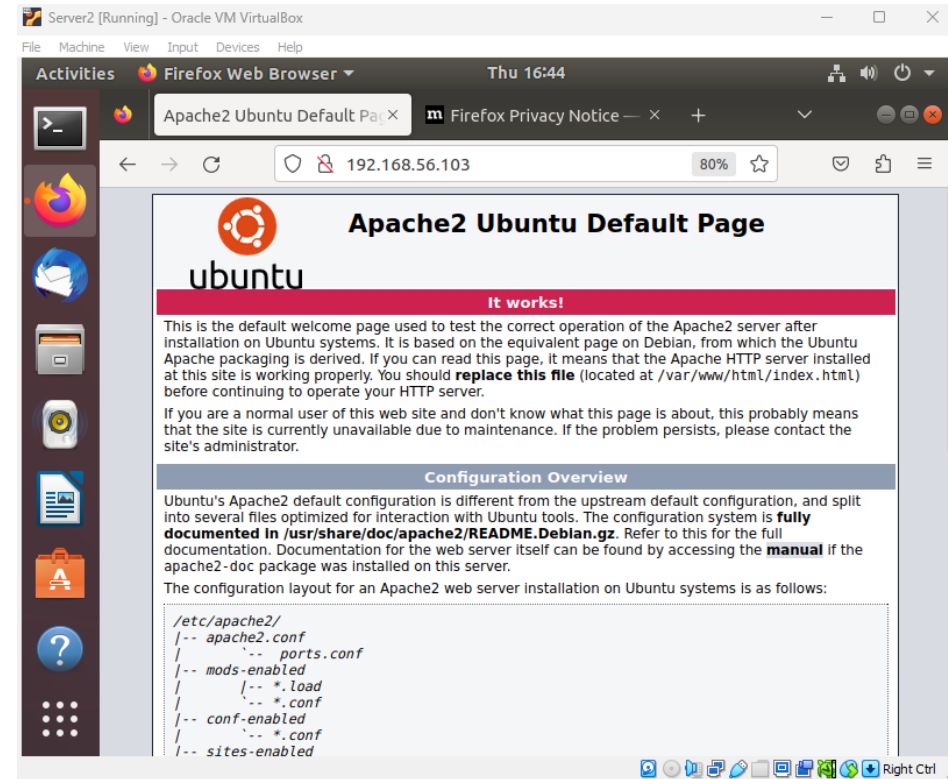
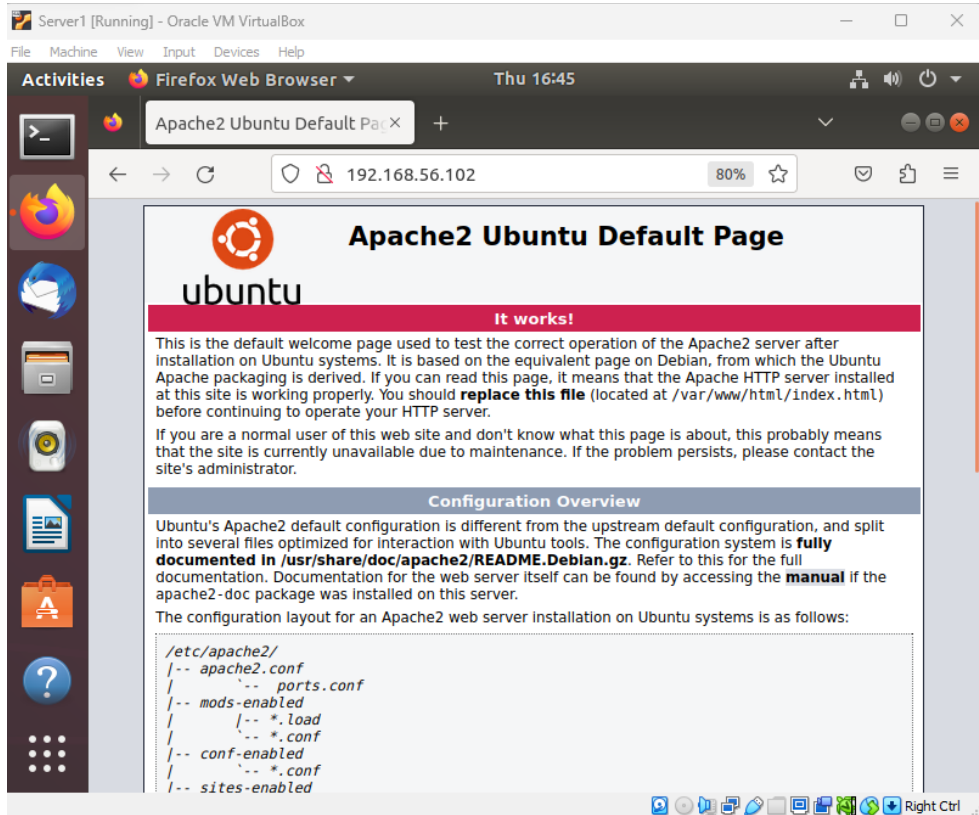
TASK [Gathering Facts] *****
*
ok: [192.168.56.103]
ok: [192.168.56.102]

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

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





SU

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?

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible-playbook --ask-become-pass in
stall_apache.yml
BECOME password:

PLAY [all] *****
*

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

TASK [install ubuntu2 package] *****
*
ok: [192.168.56.103]
ok: [192.168.56.102]

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

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

```
josephferrer@workstation: ~/CPE232_Ferrer
File Edit View Search Terminal Help
GNU nano 2.9.3 install_apache.yml

--
- 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?

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible-playbook --ask-become-pass in
stall_apache.yml
BECOME password:

PLAY [all] *****
*

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

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

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

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

Yes it updates repository index.

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

```
josephferrer@workstation: ~/CPE232_Ferrer
File Edit View Search Terminal Help
GNU nano 2.9.3      install_apache.yml

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

    - name: update repository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2

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

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?

```
josephferrer@workstation:~/CPE232_Ferrer$ ansible-playbook --ask-become-pass in
stall_apache.yml
BECOME password:

PLAY [all] *****
*

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

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

Yes, the task to add PHP support for apache is successful and the changes are applied on each remote server.

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

```
josephferrer@workstation:~/CPE232_Ferrer$ git commit -m "apache"
[main 7085d4a] apache
 2 files changed, 3 insertions(+), 3 deletions(-)
josephferrer@workstation:~/CPE232_Ferrer$ git add install_apache.yml
josephferrer@workstation:~/CPE232_Ferrer$ git push origin
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 416 bytes | 416.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:qjbferrer/CPE232_Ferrer.git
 1243349..7085d4a  main -> main
```

```
josephferrer@workstation:~/CPE232_Ferrer$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   install_apache.yml
```

https://github.com/qjbferrer/CPE232_Ferrer

Reflections:

Answer the following:

1. What is the importance of using a playbook?

There is a lot of importance in using a playbook. It allows me to automate the configuration and management of my Ubuntu servers. Using a playbook actually saves a lot of time and effort and can also help to ensure that every server is always in a consistent state. Playbooks can help me to automate repetitive tasks such as software installing, services configuration and managing user accounts. It can also reduce the risk of errors by configuring the servers in a consistent way.

2. Summarize what we have done on this activity.

The summary of this activity is all about creation of ansible playbooks. We connected the 2 servers in ansible which is a popular tool for managing Ubuntu servers with playbooks. Ansible does not need to install any software on every server in order to use it. Ansible playbooks are also written in YAML, which is a human-readable format. The playbook that we created is also connected to our GitHub. For every task that I made, it is a must to git push and git commit in our Ubuntu workstation in order to sync the progress in GitHub.

Conclusion:

- In this activity, we have two objectives which is to use commands that make changes to remote machines and use playbook in automating ansible commands. I have been able to attain the first objective by running the elevated ad hoc commands in my Ubuntu. These are long commands or syntaxes and you must be careful in typing these commands because it can lead to error or failure. By following the syntax given, I have been able to finish this activity. The second objective is to use playbook in automating ansible commands. I have been able to finish this objective by creating an Ansible playbook. This ansible playbook is written in YAML which is a human-readable format. It contains three tasks which are: updating the repository index, installing apache2 package, and adding PHP support for apache.