

Name: FRIAS, ABEGAIL L.	Date Performed: sept 13, 2024
Course/Section: CPE212 - CPE31S21	Date Submitted: sept 13, 2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem/2024-2025
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 infoall -mrmation 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?

- **No, the command did not run because it lacked the higher permissions necessary for its execution.**

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.

```
abegailfrias01@workstation:~/Activity_4.1$ ansible all -m apt -a update_cache=true
192.168.56.118 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation"
}
workstation | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation"
}
```

```
abegailfrias01@workstation:~/Activity_4.1$ ansible all -m apt -a name=vim-nox -
-become --ask-become-pass
SUDO password:
workstation | SUCCESS => {
  "cache_update_time": 1726197013,
  "cache_updated": false,
  "changed": false
}
192.168.56.118 | SUCCESS => {
  "cache_update_time": 1726197034,
  "cache_updated": false,
  "changed": false
}
abegailfrias01@workstation:~/Activity_4.1$
```

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.

```
abegailfrias01@workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
SUDO password:
192.168.56.118 | SUCCESS => {
  "cache_update_time": 1726197034,
  "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 tc\nl8.6 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 3 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
```

```
abegailfrias01@workstation:~/Activity_4.1$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
SUDO password:
workstation | SUCCESS => {
  "cache_update_time": 1726197013,
  "cache_updated": false,
  "changed": false
}
192.168.56.118 | SUCCESS => {
  "cache_update_time": 1726197034,
  "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?

- Yes it was successful.

```

abegailfrias01@workstation:~/Activity_4.1$ which vim
/usr/bin/vim
abegailfrias01@workstation:~/Activity_4.1$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-check-new-release-gtk/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled]
  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`.

```

atic), python-pyasn1:amd64 (0.4.2-3, automatic), python-idna:amd64 (2.6-1, auto
matic), ansible:amd64 (2.5.1+dfsg-1ubuntu0.1), python-minimal:amd64 (2.7.15-rc1
-1, automatic), python-asn1crypto:amd64 (0.24.0-1, automatic), python-httplib2:
amd64 (0.9.2+dfsg-1ubuntu0.3, automatic)
End-Date: 2024-09-13 08:06:10

Start-Date: 2024-09-13 09:43:45
Commandline: apt-get install python-apt -y -q
Requested-By: abegailfrias01 (1000)
Install: python-apt:amd64 (1.6.6)
End-Date: 2024-09-13 09:43:50

Start-Date: 2024-09-13 11:13:21
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Op
tions::=--force-confold install vim-nox
Requested-By: abegailfrias01 (1000)
Install: javascript-common:amd64 (11, automatic), ruby2.5:amd64 (2.5.1-1ubuntu1
.16, automatic), rake:amd64 (12.3.1-1ubuntu0.1, automatic), ruby-net-telnet:amd
64 (0.1.1-2, automatic), libtcl8.6:amd64 (8.6.8+dfsg-3, automatic), libjs-jquer
y:amd64 (3.2.1-1, automatic), vim-nox:amd64 (2:8.0.1453-1ubuntu1.13), ruby-mini
test:amd64 (5.10.3-1, automatic), libruby2.5:amd64 (2.5.1-1ubuntu1.16, automati
c), ruby:amd64 (1:2.5.1, automatic), vim-runtime:amd64 (2:8.0.1453-1ubuntu1.13,
automatic), liblua5.2-0:amd64 (5.2.4-1.1build1, automatic), ruby-power-assert:
amd64 (0.3.0-1, automatic), rubygems-integration:amd64 (1.11, automatic), fonts
-lato:amd64 (2.0-2, automatic), ruby-test-unit:amd64 (3.2.5-1, automatic), ruby
-did-you-mean:amd64 (1.2.0-2, automatic)
End-Date: 2024-09-13 11:14:27
abegailfrias01@workstation:/var/log/apt$

```

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`

```
abegailfrias01@workstation:~/Activity_4.1$ ansible all -m apt -a name=snapd --become --ask-become-pass
SUDO password:
192.168.56.118 | SUCCESS => {
  "cache_update_time": 1726197034,
  "cache_updated": false,
  "changed": false
}
workstation | SUCCESS => {
  "cache_update_time": 1726197013,
  "cache_updated": false,
  "changed": false
}
abegailfrias01@workstation:~/Activity_4.1$
```

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

- The snapd package was successfully installed as a result of the command. The outcome was successful, but the remote servers haven't updated as of yet.

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

```
abegailfrias01@workstation:~/Activity_4.1$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
SUDO password:
192.168.56.118 | SUCCESS => {
  "cache_update_time": 1726197034,
  "cache_updated": false,
  "changed": false
}
workstation | SUCCESS => {
  "cache_update_time": 1726197013,
  "cache_updated": false,
  "changed": false
}
Show Applications
```

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.

```
abegailfrias01@workstation:~/Activity_4.1$ git status
On branch main
Your branch is based on 'origin/master', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean
abegailfrias01@workstation:~/Activity_4.1$
```

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
```

```
GNU nano 2.9.3          install_apache.yml          Modified
--
- 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.


```

abegailfrias01@workstation:~$ sudo nano install_apache.yml
abegailfrias01@workstation:~$ ansible-playbook --ask-become-pass install_apache
.yml
SUDO password:

PLAY [all] *****
*

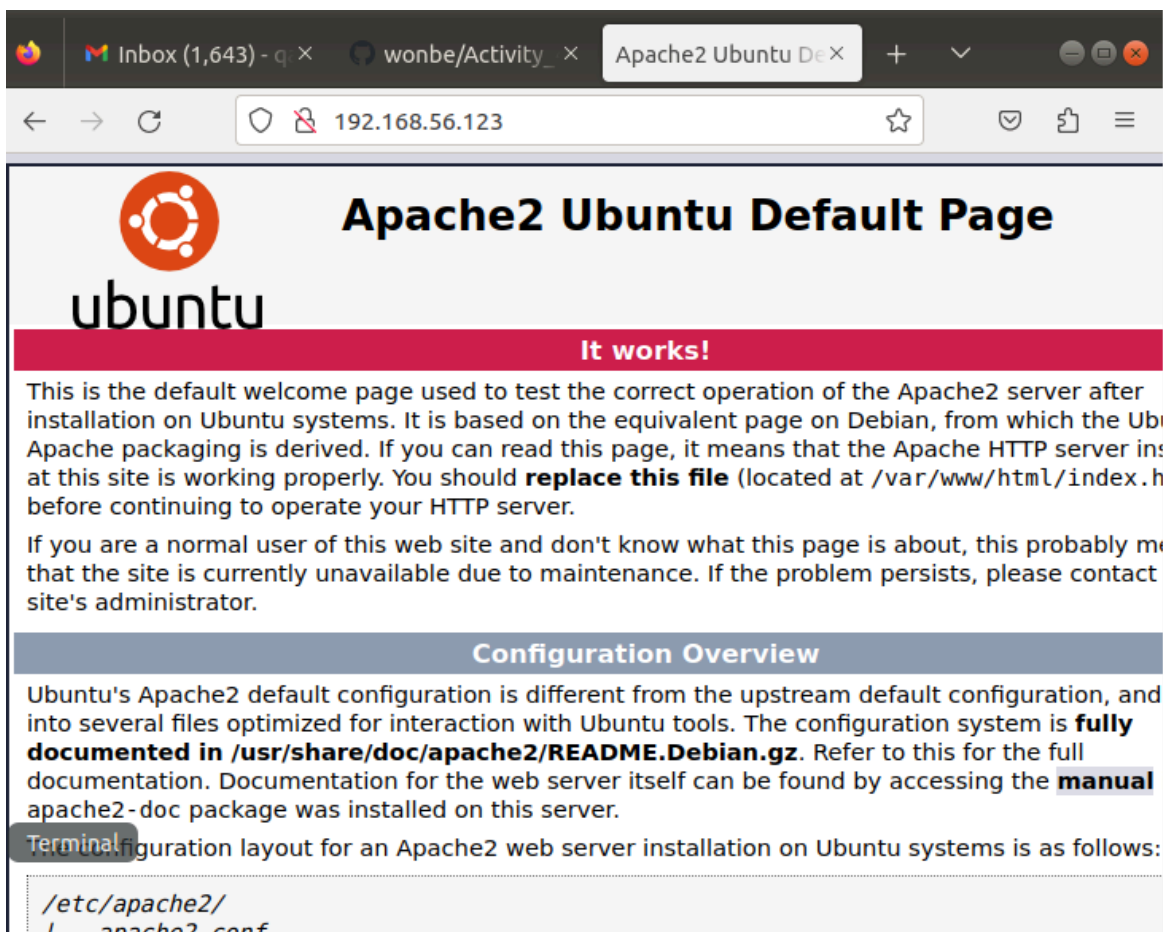
TASK [Gathering Facts] *****
*
ok: [192.168.56.118]
ok: [workstation]

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

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

abegailfrias01@workstation:~$

```



Inbox (1,643) - q x wonbe/Activity_ x Apache2 Ubuntu De x

192.168.56.123

ubuntu

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 your site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and is 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** page. The `apache2-doc` package was installed on this server.

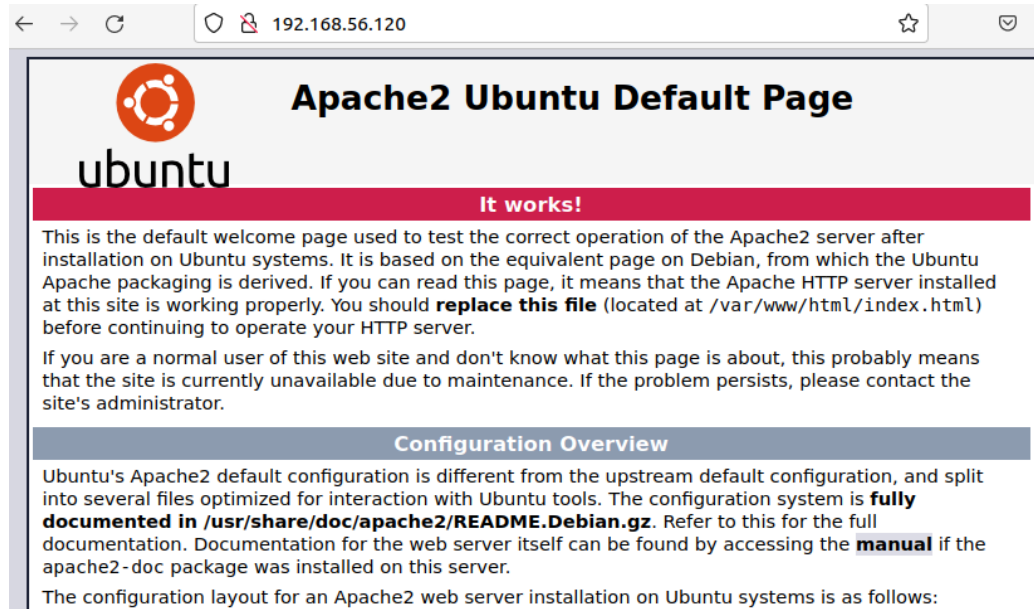
Terminal Configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```

/etc/apache2/
/etc/apache2/conf

```

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?
 - The second task failed as a result of this modification's output since the package did not match anything that was accessible.

```
GNU nano 2.9.3      install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apacheyou2 package
    apt:
      name: apacheyou2

Welcome to Ubuntu
```



```

abegailfrias01@workstation:~/Activity_4.1$ ansible-playbook --ask-become-pass i
ninstall_apache.yml
SUDO password:

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [workstation]
ok: [192.168.56.118]

TASK [install apacheyou2 package] *****
*
fatal: [workstation]: FAILED! => {"changed": false, "msg": "No package matching
'apacheyou2' is available"}
fatal: [192.168.56.118]: FAILED! => {"changed": false, "msg": "No package match
ing 'apacheyou2' is available"}
    to retry, use: --limit @/home/abegailfrias01/Activity_4.1/install_apach
e.retry

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

abegailfrias01@workstation:~/Activity_4.1$

```

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.

```

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

```

```

---
- 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?
 - The revised repository index of the remote servers was the result of the recent modification. The repository index was modified by the new command.

```

abegailfrias01@workstation:~/Activity_4.1$ ansible-playbook --ask-become-pass i
ninstall_apache.yml
SUDO password:

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [192.168.56.118]
ok: [workstation]

TASK [update repository index] *****
*
changed: [192.168.56.118]
changed: [workstation]

TASK [install apache2 package] *****
*
ok: [192.168.56.118]
ok: [workstation]

PLAY RECAP *****
*
192.168.56.118      : ok=3    changed=1    unreachable=0    failed=0
workstation        : ok=3    changed=1    unreachable=0    failed=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.

```
abegailfrias01@workstation: ~/Activity_4.1
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
```

```
---
- 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?
 - The new command change enable the remote servers to have PHP support for apache thus the page now has PHP when the IP is looked up from the remote server affected.

```
abegailfrias01@workstation: ~/Activity_4.1
File Edit View Search Terminal Help
*
TASK [Gathering Facts] *****
*
ok: [192.168.56.118]
ok: [workstation]

TASK [update repository index] *****
*
changed: [192.168.56.118]
changed: [workstation]

TASK [install apache2 package] *****
*
ok: [192.168.56.118]
ok: [workstation]

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

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

abegailfrias01@workstation:~/Activity_4.1$
```

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

https://github.com/wonbe/Activity_4.1

Reflections:

Answer the following:

1. What is the importance of using a playbook?
 - The ability for server administrators to run a series of instructions from a single file is what makes a playbook so important. Consequently, updating or implementing changes is quicker and easier.
2. Summarize what we have done on this activity.
 - Through this exercise, we were able to update and run modifications from the workstation to the other server using Ansible. Using Ansible, we updated and upgraded in a manner similar to that of sudo. Furthermore, we were able to use playbooks to carry out tasks more quickly and in a logical order. Additionally, we learnt how to set up Ansible and how to utilize and code on a playbook, where it is imperative that all capitalizations and spaces be used appropriately.

