

<b>Name:</b> Mark Andrei Ponayo	<b>Date Performed:</b> September 12,2023
<b>Course/Section:</b> BSCPE31S5	<b>Date Submitted:</b>
<b>Instructor:</b> Engr. Roman Richard	<b>Semester and SY:</b>
<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</b> '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. <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:  <i>ansible all -m apt -a update_cache=true</i>	

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

```
ponayo@Workstation:~$ ansible all -m apt -a update_cache=true
127.0.0.1 | 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)"
}
ponayo@Workstation:~$
```

*After executing the command the result is failed*

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.

```
ponayo@Workstation:~$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694518606,
  "cache_updated": true,
  "changed": true
}
ponayo@Workstation:~$
```

*It run successfully since we added the code “—become – ask -become-pass” since it was use to run the codes.*

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.

```
ponayo@Workstation:~$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
127.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694518798,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nTh
e following additional packages will be installed:\n fonts-lato javascript-common libjs-jquery liblua
5.2-0 libruby3.0 rake ruby\n ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n rubyge
ms-integration vim-runtime\nSuggested packages:\n apache2 | lighttpd | httpd ri ruby-dev bundler csc
pe 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-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n
rubygems-integration vim-nox vim-runtime\n0 upgraded, 15 newly installed, 0 to remove and 16 not upgr
aded.\nNeed to get 17.5 MB of archives.\nAfter this operation, 76.4 MB of additional disk space will b
e used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\n
Get:2 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmui [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 amd64 liblua5.2-0 amd64 5.2.4-2 [125 kB]\nG
et:5 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration all 1.18 [5336 B]\nGet:
6 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64 3.0.2-7ubuntu2.4 [50.1 kB
]\nGet:7 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all 3.3.5-2 [228 kB]\nGet:
8 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rake amd64 13.0.6-2 [61.7 kB]\nGet:9 http://ph
.archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7 kB]\nGet:10 http://ph.archive.ubun
tu.com/ubuntu jammy/main amd64 ruby-net-telnet all 0.1.1-2 [12.6 kB]\nGet:11 http://ph.archive.ubun
tu.com/ubuntu jammy/universe amd64 ruby-webrick all 1.7.0-3 [51.8 kB]\nGet:12 http://ph.archive.ubun
tu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all 0.3.2-1ubuntu0.1 [24.9 kB]\nGet:13 http://ph.archive
.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0 amd64 3.0.2-7ubuntu2.4 [5113 kB]\nGet:14 http://
```

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?

```
ponayo@Workstation:~$ which vim
/usr/bin/vim
ponayo@Workstation:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed]
Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed,automatic]
Vi IMproved - enhanced vi editor - compact version

ponayo@Workstation:~$
```

The command is successful since “*which vim*” is found at */usr/bin/vim* and the second one is the command that confirmed that it is installed.

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*.

```
ponayo@Workstation: /var/log/apt
GNU nano 6.2 history.log
Start-Date: 2023-09-12 19:20:28
Commandline: apt install ansible
Requested-By: ponayo (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dns>
End-Date: 2023-09-12 19:21:47

Start-Date: 2023-09-12 19:40:57
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Op>
Requested-By: ponayo (1000)
Install: fonts-lato:amd64 (2.0-2.1, automatic), liblua5.2-0:amd64 (5.2.4-2, aut>
End-Date: 2023-09-12 19:41:05

Read 12 lines
^G Help      ^O Write Out ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File ^N Replace  ^U Paste    ^J Justify  ^_ Go To Line
```

*In history.log, it shows the commands, time and date that I use previously.*

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?

```
ponayo@Workstation:/var/log/apt$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694518798,
  "cache_updated": false,
  "changed": false
}
ponayo@Workstation:/var/log/apt$
```

*The command executed successfully and did not target any remote servers. This also means there are no changes made to any other remote servers.*





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.

```
ponayo@Workstation:/var/log/apt$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1694518798,
  "cache_updated": false,
  "changed": false
}
ponayo@Workstation:/var/log/apt$
```

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

```
ponayo@Workstation:~/CPE232_Ponayo$ git add ansible.txt
ponayo@Workstation:~/CPE232_Ponayo$ git commit -m "Changes"
[main 716dbaf] Changes
1 file changed, 2 insertions(+)
create mode 100644 ansible.txt
ponayo@Workstation:~/CPE232_Ponayo$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ponayooo/CPE232_Ponayo.git
19fb1a9..716dbaf main -> main
ponayo@Workstation:~/CPE232_Ponayo$
```

 Ponayooo Changes	716dbaf 20 minutes ago	 4 commits
 README.md	Boom	2 weeks ago
 ansible.txt	Changes	20 minutes ago

## 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.

```
---
- 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.

```

ponayo@Workstation:~/CPE232_Ponayo$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [127.0.0.1]

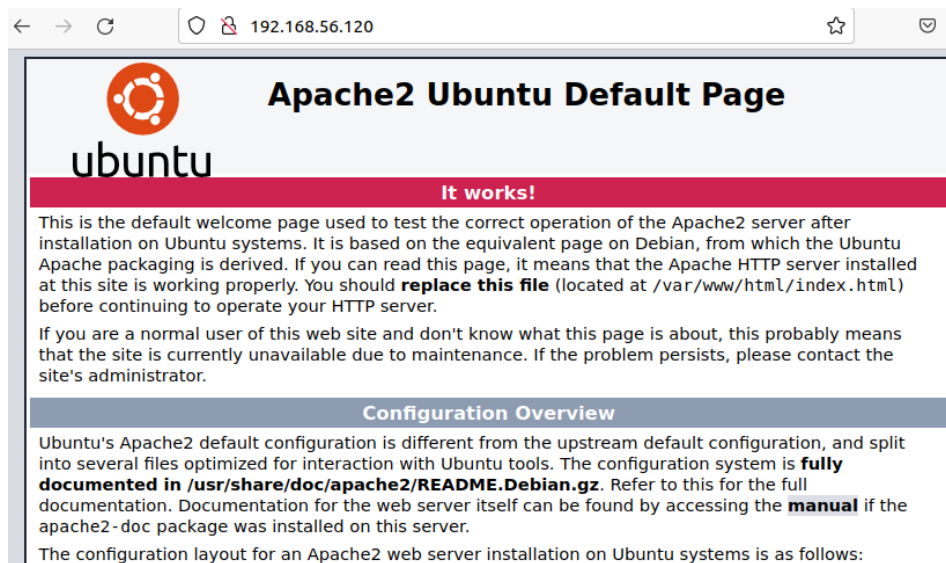
TASK [install apache2 package] *****
changed: [127.0.0.1]

PLAY RECAP *****
127.0.0.1      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0

ponayo@Workstation:~/CPE232_Ponayo$

```

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.



Outcome:



## 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.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- mods-enabled  
|   |-- *.load  
|   |-- *.conf  
|-- conf-enabled  
|   |-- *.conf  
|-- sites-enabled  
|   |-- *.conf
```

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?

```
ponayo@Workstation:~/CPE232_Ponayo$ ansible-playbook --ask-become-pass install_apache.yml  
BECOME password:  
  
PLAY [all] *****  
  
TASK [Gathering Facts] *****  
ok: [127.0.0.1]  
  
TASK [install apache2 package] *****  
fatal: [127.0.0.1]: FAILED! => {"changed": false, "msg": "No package matching 'apacheniponayo' is available"}  
  
PLAY RECAP *****  
127.0.0.1 : ok=1 changed=0 unreachable=0 failed=1 skipped=0 rescued=0 ignored=0  
ponayo@Workstation:~/CPE232_Ponayo$
```

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.

```

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

    - name: update respository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2

```

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

```

ponayo@Workstation:~/CPE232_Ponayo$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [127.0.0.1]

TASK [update repository index] *****
changed: [127.0.0.1]

TASK [install apache2 package] *****
ok: [127.0.0.1]

PLAY RECAP *****
127.0.0.1          : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ponayo@Workstation:~/CPE232_Ponayo$

```

Yes, the command that we input is added since it shows the changed output in the play recap.

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 6.2                                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

```

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

```

ponayo@Workstation:~/CPE232_Ponayo$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [127.0.0.1]

TASK [update repository index] *****
changed: [127.0.0.1]

TASK [install apache2 package] *****
ok: [127.0.0.1]

TASK [add PHP support for apache] *****
changed: [127.0.0.1]


PLAY RECAP *****
127.0.0.1                : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0




ponayo@Workstation:~/CPE232_Ponayo$




```

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

```
ponayo@Workstation:~/CPE232_Ponayo$ git add install_apache.yml
ponayo@Workstation:~/CPE232_Ponayo$ git commit -m "update install_apache.yml with PHP"
[main 3bbce8b] update install_apache.yml with PHP
1 file changed, 17 insertions(+)
create mode 100644 install_apache.yml
ponayo@Workstation:~/CPE232_Ponayo$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 481 bytes | 481.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Ponayooo/CPE232_Ponayo.git
716dbaf..3bbce8b main -> main
ponayo@Workstation:~/CPE232_Ponayo$
```

 **CPE232\_Ponayo** Public Pin Unwatch 1

 main  1 branch  0 tags Go to file Add file Code

Ponayooo update install_apache.yml with PHP		3bbce8b now 5 commits
 README.md	Boom	2 weeks ago
 ansible.txt	Changes	50 minutes ago
 install_apache.yml	update install_apache.yml with PHP	now

[CPE232\\_Ponayo/install\\_apache.yml at main · Ponayooo/CPE232\\_Ponayo \(github.com\)](https://github.com/Ponayooo/CPE232_Ponayo/blob/main/install_apache.yml)

## Reflections:

Answer the following:

1. What is the importance of using a playbook?
  - The importance of using a playbook is we can use it to automate. Since, we can use it to run multiple tasks, assign roles, and define configuration and etc. It also offers a repeatable, reusable, and simple configuration.
2. Summarize what we have done on this activity.
  - During this activity we're able to use commands that changes to remote machines and use playbook in an automating ansible commands. We we're able to understand of what use and benefits of ansible.