

Name: Zamora, Angelo E.	Date Performed: 09-11-2024
Course/Section: CPE31S2	Date Submitted: 09-11-2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Semester 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:

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

1. 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? **Command Ansible not found.**

```
zamora@workstation:~$ ansible all -m apt -a update_cache=true
Command 'ansible' not found, but can be installed with:
apt install ansible          # version 2.10.7+merged+base+2.10.8+dfsg-1, or
apt install ansible-core    # version 2.12.0-1ubuntu0.1
Ask your administrator to install one of them.
zamora@workstation:~$
```

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.

```
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible all -m apt -a update_cache=true --become --ask
BECOME password:
192.168.56.102 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726069321,
  "cache_updated": true,
  "changed": true
}
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$
```

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.

```
zanora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726069321,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following additional package
s will be installed:\n fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n ruby-net-telnet ruby-rubygens
ruby-webrick ruby-xmlrpc ruby3.0\n rubygens-integration 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 libr
uby3.0 rake ruby\n ruby-net-telnet ruby-rubygens ruby-webrick ruby-xmlrpc ruby3.0\n rubygens-integration vim-nox vim-runtime\n0 upg
raded, 15 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 17.5 MB of archives.\nAfter this operation, 76.4 MB of addit
ional disk space will be used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2696 kB]\nGet:2 h
ttp://ph.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmv1 [5936 B]\nGet:3 http://ph.archive.ubuntu.com/ubunt
u 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 li
blua5.2-0 amd64 5.2.4-2 [125 kB]\nGet:5 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64 rubygens-integration all 1.18 [5336 B]\n
Get:6 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64 3.0.2-7ubuntu2.7 [50.1 kB]\nGet:7 http://ph.archiv
e.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygens all 3.3.5-2 [228 kB]\nGet:8 http://ph.archive.ubuntu.com/ubuntu jammy/main amd64
ruby amd64 1:3.0-exp1 [5100 B]\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.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all 0.1.1-2 [12.6 kB]\nGet:11 http://ph.archive.ubuntu.com/ubuntu ja
mmy/universe amd64 ruby-webrick all 1.7.0-3 [51.8 kB]\nGet:12 https://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlr
pc all 0.3.2-1ubuntu0.1 [24.9 kB]\nGet:13 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0 amd64 3.0.2-7ubunt
u2.7 [5113 kB]\nGet:14 http://ph.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.18 [6827 kB]
\nGet:15 http://ph.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 vim-nox amd64 2:8.2.3995-1ubuntu2.18 [1941 kB]\nFetched 17
```

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?

```
zanora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ which vim
zanora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security 2:8.2.3995-1ubuntu2.18 amd64
Vi Improved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.18 amd64 [installed,automatic]
Vi Improved - enhanced vi editor - compact version

zanora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$
```

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.

```

GNU nano 6.2                                     history.log

Start-Date: 2024-09-04 13:58:34
Commandline: apt install git
Requested-By: zamora (1000)
Install: git:amd64 (1:2.34.1-1ubuntu1.1), liberror-perl:amd64 (0.17029-1, automatic), git-man:amd64 (1:2.34.1-1ubuntu1.1, automatic)
End-Date: 2024-09-04 13:58:53

Start-Date: 2024-09-11 08:05:12
Commandline: apt install ansible
Requested-By: zamora (1000)
Install: python-babel-localedata:amd64 (2.8.0+dfsg.1-7, automatic), python3-dnspython:amd64 (2.1.0-1, automatic)
End-Date: 2024-09-11 08:06:17

Start-Date: 2024-09-11 09:07:40
Commandline: apt install ansible -y
Requested-By: zamora (1000)
Install: sshpass:amd64 (1.09-1, automatic), python3-resolvelib:amd64 (0.8.1-1, automatic), ansible-core:amd64 (2.16.11-1ppa-jammy, automatic)
Upgrade: ansible:amd64 (2.10.7+merged+base+2.10.8+dfsg-1, 9.9.0-1ppa-jammy)
Error: Sub-process /usr/bin/dpkg returned an error code (1)
End-Date: 2024-09-11 09:08:27

Start-Date: 2024-09-11 09:10:40
Commandline: apt --fix-broken install
Requested-By: zamora (1000)
Install: ansible-core:amd64 (2.16.11-1ppa-jammy, automatic)
End-Date: 2024-09-11 09:11:00

Start-Date: 2024-09-11 20:40:29
Commandline: apt upgrade
Requested-By: zamora (1000)
Upgrade: libext2fs2:amd64 (1.46.5-2ubuntu1.1, 1.46.5-2ubuntu1.2), language-pack-en-base:amd64 (1:22.04-0ubuntu1, automatic)
End-Date: 2024-09-11 20:43:30

Start-Date: 2024-09-11 23:05:45
Commandline: apt install python3.12
Requested-By: zamora (1000)

```

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=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? Yes

```
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible all -m apt -a name=snapd --become
BECOME password:
192.168.56.102 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726069321,
  "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.

```

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible all -m apt -a "name=snappd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726069321,
  "cache_updated": false,
  "changed": false
}
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$

```

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

```

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ git commit -m "DONE"
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ansible.cfg
        inventory

nothing added to commit but untracked files present (use "git add" to track)
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ git push
Everything up-to-date
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$

```

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.

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

```
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ sudo nano install_apache.yml
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

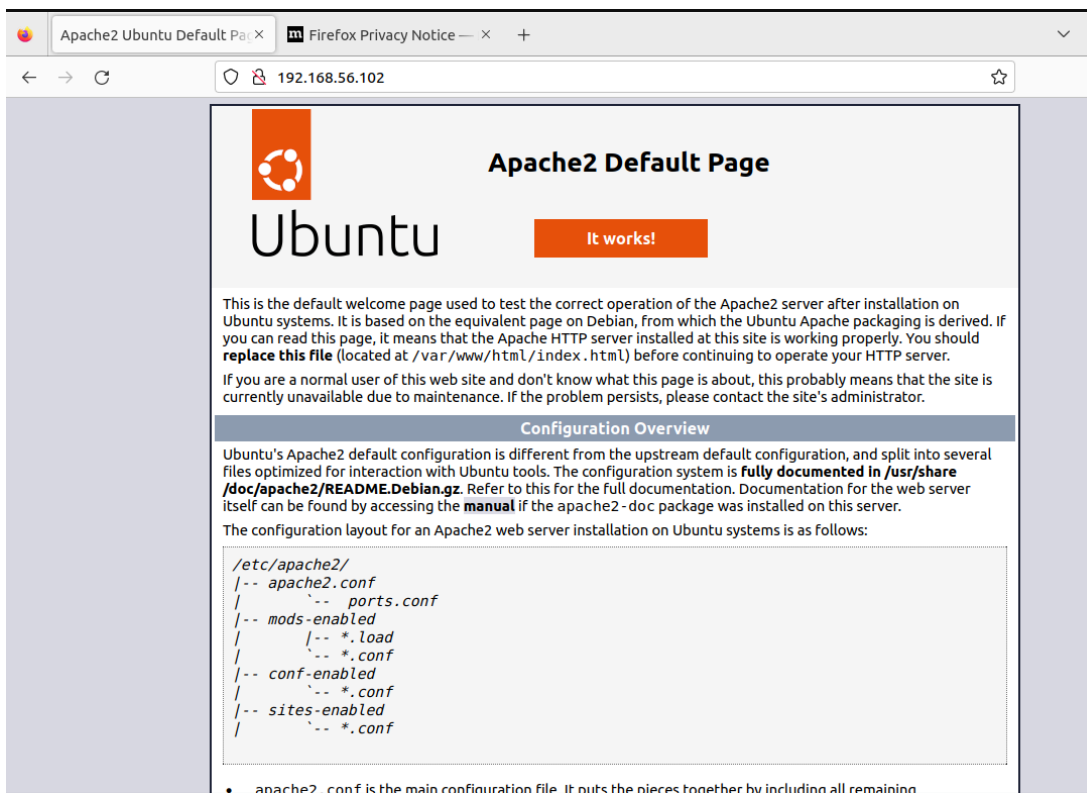
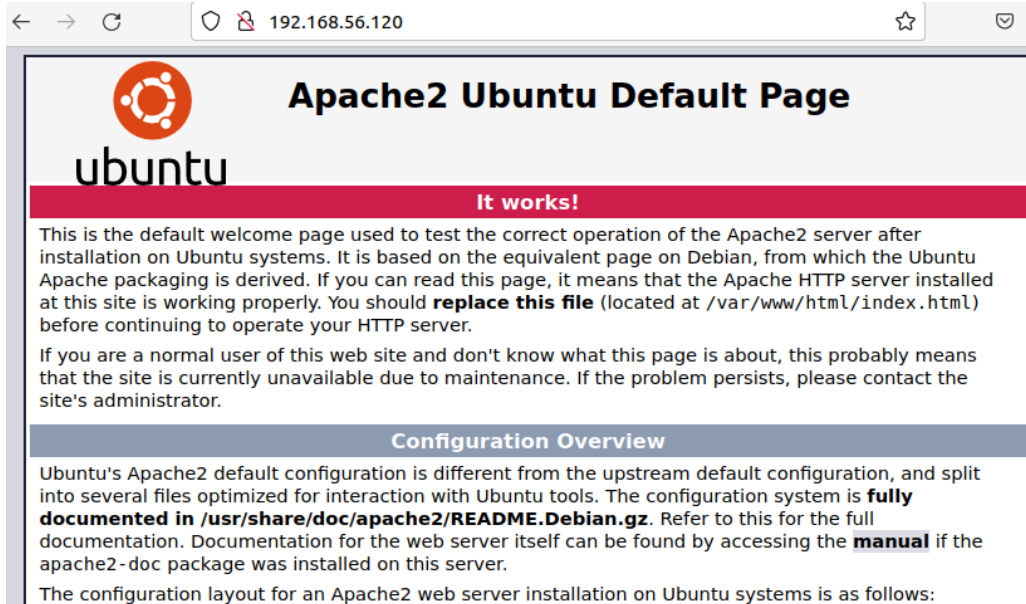
TASK [Gathering Facts] *****
ok: [192.168.56.102]

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

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

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$
```

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? **Unsuccessful**

```
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

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

TASK [install apache2 package] *****
fatal: [192.168.56.102]: FAILED! => ("changed": false, "msg": "No package matching 'apache1' is available")

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

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ^C
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$
```

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? **It includes the update means it was successful**

```
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

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

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

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

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

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$
```

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.

- Run the playbook and describe the output. Did the new command change anything on the remote servers? **Yes it did**

```

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ sudo nano install_apache.yml
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

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

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

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

TASK [add PHP support for apache] *****
changed: [192.168.56.102]

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

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$






```

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

```

zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ git commit -m "DONE"
[main f140b64] DONE
3 files changed, 24 insertions(+)
create mode 100644 ansible.cfg
create mode 100644 install_apache.yml
create mode 100644 inventory
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 625 bytes | 625.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:GeloaceRT/CPE212_Hands-On-Activity-4.1_ZAMORA.git
172f13c..f140b64  main -> main
zamora@workstation:~/CPE212_Hands-On-Activity-4.1_ZAMORA$

```

 GeloaceRT	DONE	f140b64 · 2 minutes ago	2 Commits
 README.md	first commit		3 hours ago
 ansible.cfg	DONE		2 minutes ago
 install_apache.yml	DONE		2 minutes ago
 inventory	DONE		2 minutes ago

Reflections:

Answer the following:

1. What is the importance of using a playbook?

Using a playbook is crucial because it allows you to install numerous packages automatically without having to wait for each one to finish before starting another one. In essence, it's a script that can execute tasks with just one query.

2. Summarize what we have done on this activity.

The use of Ansible was the primary emphasis of this exercise. Task 2 presents the playbook and explains how it operates. Basically, as long as the commands inside the yml file are correct, the playbook can install numerous packages in a single query, saving the user from having to manually type out each package install and wait for its download.