

|  |  |
|--|--|
| <b>Name:</b> Ilagan, Carlo Hideki D.   | <b>Date Performed:</b> 9/11/2024                 |
| <b>Course/Section:</b> CpE31S2 - CpE212  | <b>Date Submitted:</b> 9/11/2024                 |
| <b>Instructor:</b> Mr. Robin Valenzuela  | <b>Semester and SY:</b> 1st Semester / 2024-2025 |
| <b>Activity 4: Running Elevated Ad hoc Commands</b>  |  |
| <b>1. Objectives:</b><br>1.1 Use commands that makes changes to remote machines<br>1.2 Use playbook in automating ansible commands   |  |
| <b>2. Discussion:</b><br><br><i>Provide screenshots for each task.</i><br><br><b>Elevated Ad hoc commands</b><br>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.<br><br><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><br><br>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:

*ansible all -m apt -a update\_cache=true*

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

```
hideki@workstation:~$ ansible all -m apt -a update_cache=true
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
```

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.

```
hideki@workstation:~/Hands-on-Activity-4.1$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726055873,
  "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

```
hideki@workstation:~/Hands-on-Activity-4.1$ 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": 1726053046,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nThe following packages were automatically installed and are
no longer required:\n  linux-headers-6.8.0-31 linux-headers-6.8.0-31-generic\n
linux-image-6.8.0-31-generic linux-modules-6.8.0-31-generic\n  linux-modules-ext
ra-6.8.0-31-generic linux-tools-6.8.0-31\n  linux-tools-6.8.0-31-generic\nUse 's
udo apt autoremove' to remove them.\nThe following additional packages will be i
nstalled:\n  fonts-lato javascript-common libjs-jquery liblua5.1-0 libruby libru
by3.2\n  libsodium23 rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-webr
ick\n  ruby-xmlrpc ruby3.2 rubygems-integration vim-runtime\nSuggested packages:
```

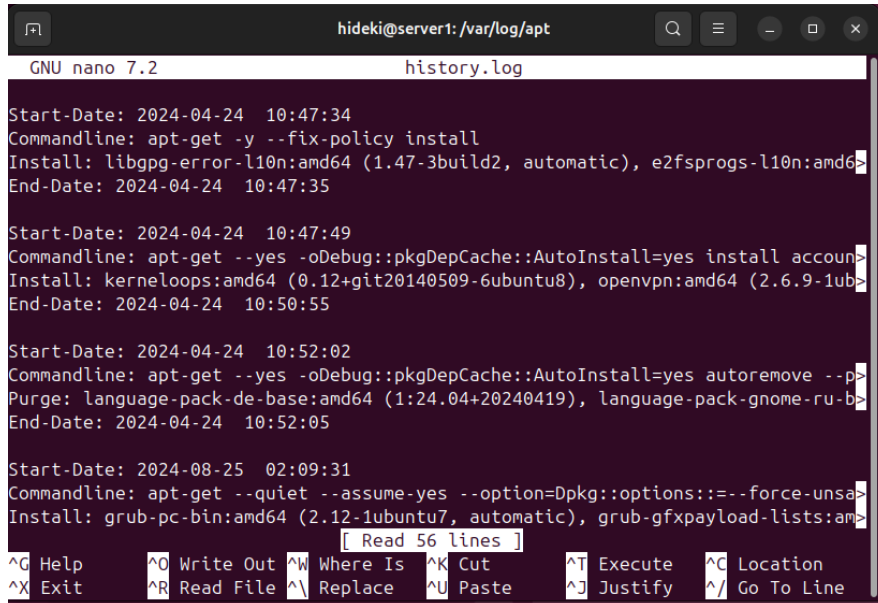
- 2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?

- Yes

```
hideki@workstation:~/Hands-on-Activity-4.1$ which vim
hideki@workstation:~/Hands-on-Activity-4.1$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/noble-updates 2:9.1.0016-1ubuntu7.1 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/noble-updates,now 2:9.1.0016-1ubuntu7.1 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`.



```
hideki@server1: /var/log/apt
GNU nano 7.2 history.log
Start-Date: 2024-04-24 10:47:34
Commandline: apt-get -y --fix-policy install
Install: libgpg-error-l10n:amd64 (1.47-3build2, automatic), e2fsprogs-l10n:amd64 (1.47-3build2, automatic)
End-Date: 2024-04-24 10:47:35

Start-Date: 2024-04-24 10:47:49
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes install accountsservice
Install: kerneloops:amd64 (0.12+git20140509-6ubuntu8), openvpn:amd64 (2.6.9-1ubuntu1)
End-Date: 2024-04-24 10:50:55

Start-Date: 2024-04-24 10:52:02
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes autoremove --purge
Purge: language-pack-de-base:amd64 (1:24.04+20240419), language-pack-gnome-ru-base:amd64 (1:24.04+20240419)
End-Date: 2024-04-24 10:52:05

Start-Date: 2024-08-25 02:09:31
Commandline: apt-get --quiet --assume-yes --option=Dpkg::options::=-force-unsafe-io install grub-pc-bin:amd64 (2.12-1ubuntu7, automatic), grub-gfxpayload-lists:amd64 (2.12-1ubuntu7, automatic)
[ Read 56 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

- History of application installed in the server.

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?

- It shows that ansible is inside the servers.



```
hideki@workstation:~/Hands-on-Activity-4.1$ ansible all -m apt -a name=snapt --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726053046,
  "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.

```
hideki@workstation:~/Hands-on-Activity-4.1$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1726053046,
  "cache_updated": false,
  "changed": false
}
```

- It did not change, it means that the ansible is in latest update.

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

```
hideki@workstation:~/Hands-on-Activity-4.1$ git commit -m "DONE"
[main ec5a25f] DONE
2 files changed, 6 insertions(+)
create mode 100644 ansible.cfg
create mode 100644 inventory
hideki@workstation:~/Hands-on-Activity-4.1$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 424 bytes | 424.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:chilagan-github/Hands-on-Activity-4.1.git
578149e..ec5a25f  main -> main
```

## 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.

```
hideki@workstation:~/Hands-on-Activity-4.1$ ansible-playbook --ask-become-pass i
ninstall_apache.yml
BECOME password:

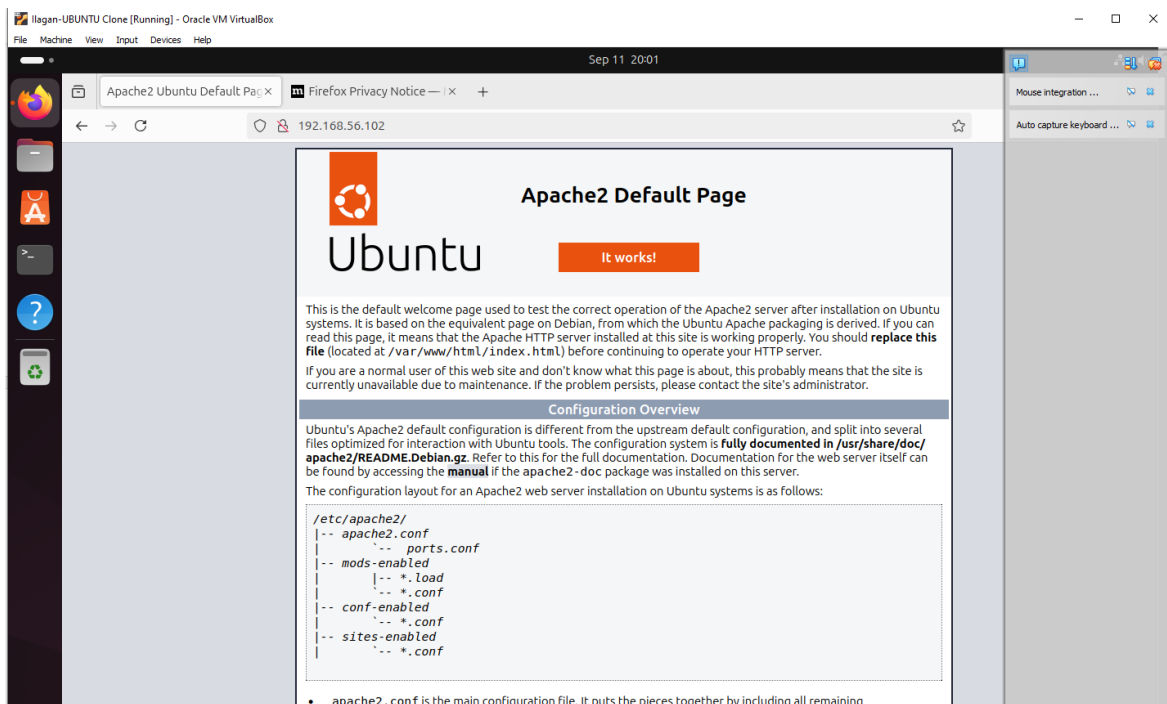
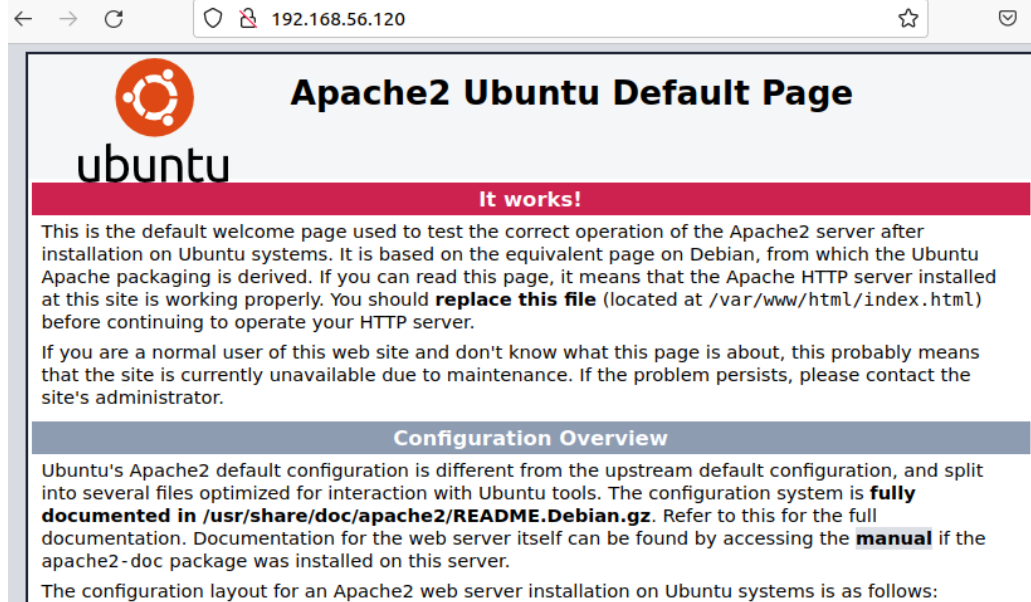
PLAY [all] *****

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

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

PLAY RECAP *****
192.168.56.102      : ok=2    changed=0    unreachable=0    failed=0    s
kipped=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.



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?

```
hideki@workstation:~/Hands-on-Activity-4.1$ 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 'league of legends' is available"}

PLAY RECAP *****
192.168.56.102      : ok=1    changed=0    unreachable=0    failed=1    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.

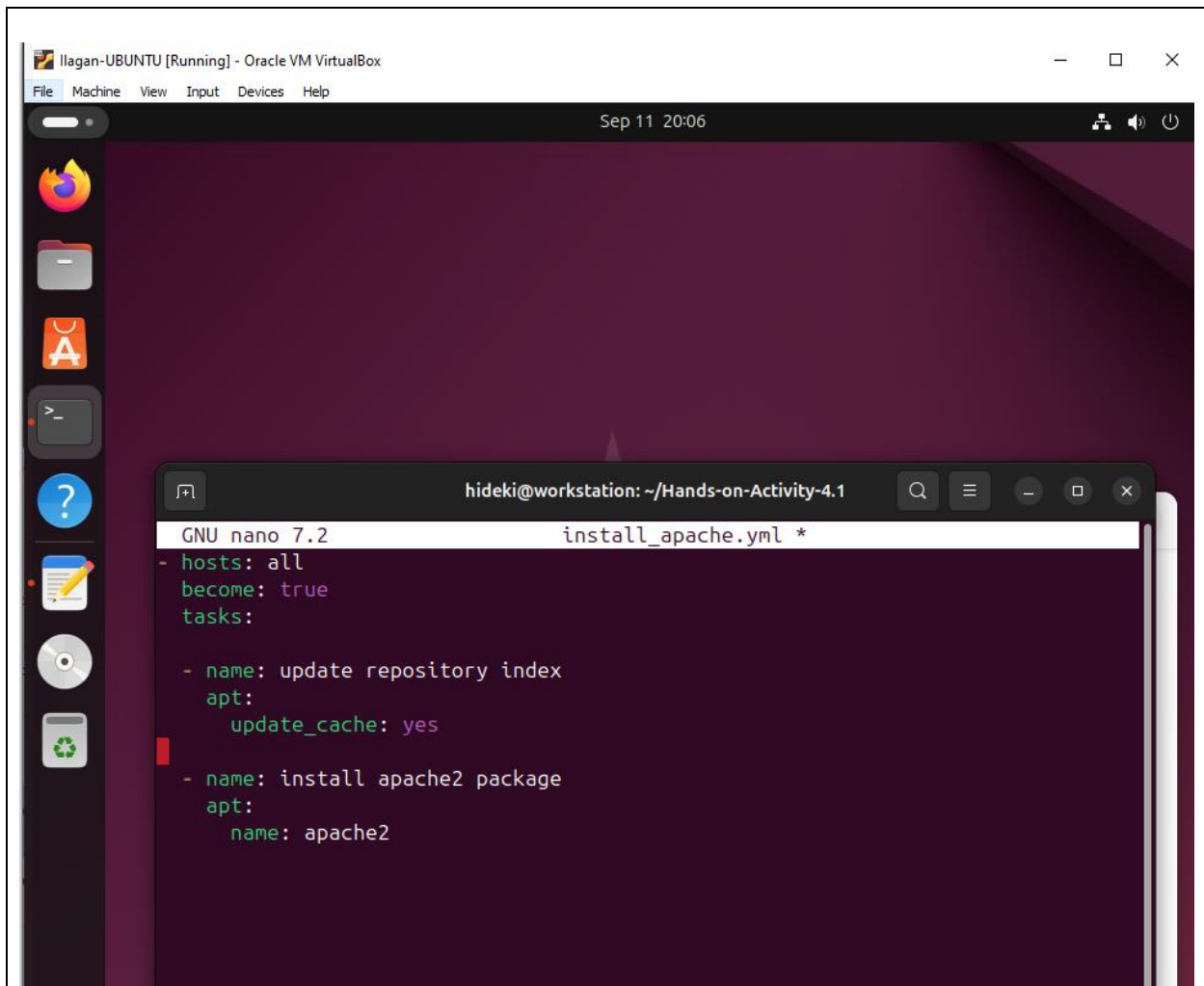
```
---
- hosts: all
  become: true
  tasks:

    - 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?

- Yes, since there is a new command added in the install\_apache.yml file

```

hideki@workstation:~/Hands-on-Activity-4.1$ ansible-playbook --ask-become-pass 1
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    s
kipped=0    rescued=0    ignored=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.

```

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

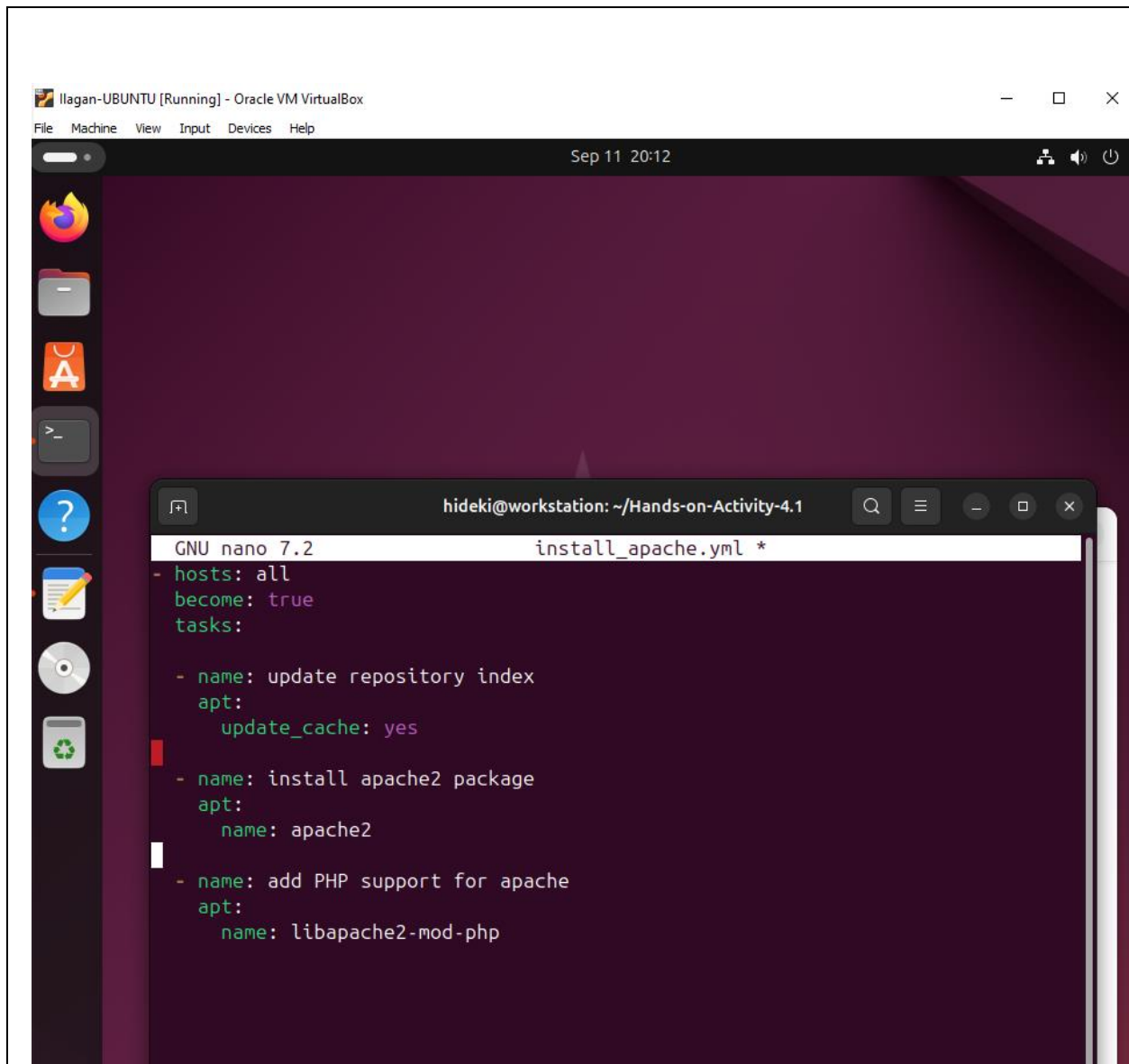
    - name: update repository index
      apt:
        update_cache: yes

    - 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?

```

hideki@workstation:~/Hands-on-Activity-4.1$ ansible-playbook --ask-become-pass i
ninstall_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] *****
ok: [192.168.56.102]

PLAY RECAP *****
192.168.56.102      : ok=4    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

```

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

```

hideki@workstation:~/Hands-on-Activity-4.1$ git add *
hideki@workstation:~/Hands-on-Activity-4.1$ git commit -m "DONE"
[main 8896469] DONE
1 file changed, 15 insertions(+)
create mode 100644 install_apache.yml
hideki@workstation:~/Hands-on-Activity-4.1$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 487 bytes | 487.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:chilagan-github/Hands-on-Activity-4.1.git
ec5a25f..8896469  main -> main

```

The screenshot shows a GitHub repository page for 'Hands-on-Activity-4.1' by user 'chilagan-github'. The repository is public and has 0 stars, 0 forks, and 1 watcher. The main branch is 'main'. The commit history shows four commits by user 'hideki':

| File               | Status         | Time         |
|--------------------|----------------|--------------|
| README.md          | Initial commit | 12 hours ago |
| ansible.cfg        | DONE           | 1 hour ago   |
| install_apache.yml | DONE           | 1 minute ago |
| inventory          | DONE           | 1 hour ago   |

The repository has no description, website, or topics provided. The 'About' section lists the README, Activity, 0 stars, 1 watching, and 0 forks. The 'Releases' section states 'No releases published' and provides a link to 'Create a new release'.

## Reflections:

Answer the following:

1. What is the importance of using a playbook?
  - The importance of using a playbook is that you can automatically install multiple packages without manually waiting for each of them to finish before running another install. It is basically a script that can run commands in a single query.
2. Summarize what we have done on this activity.
  - This activity mainly focused on the use of ansible. The task 2 introduces the playbook and how it works where rather than manually typing each package install and waiting for its download, the playbook can easily install multiple packages in one query as long as the commands inside the yml file is right.