Name: Rio Marie G. Suzuki	**Date Performed:** 09/14/2023
Course/Section: CPE232 S6	**Date Submitted:**09/14/2023
Instructor: Dr. Jonathan Taylar	**Semester and SY:** 1st sem 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:**

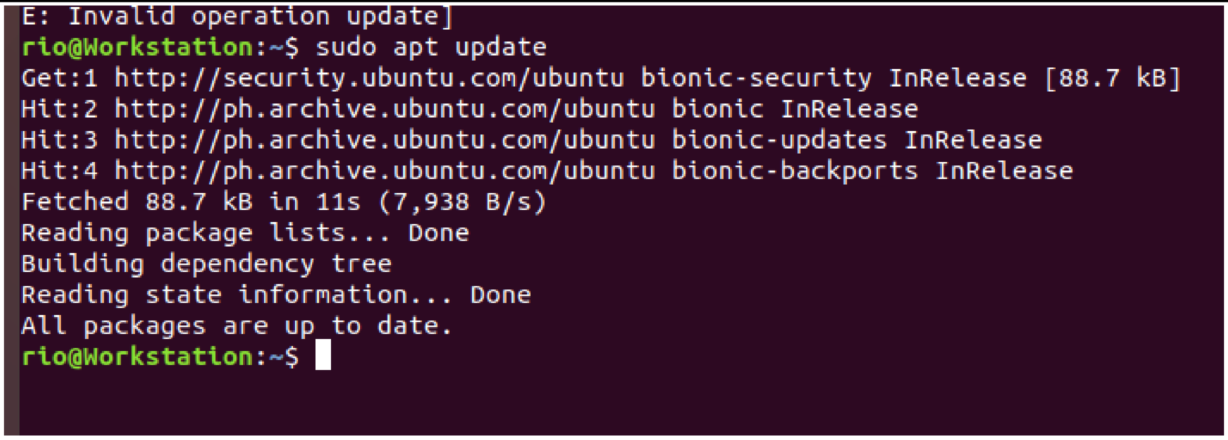
Elevated Ad hoc commands

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

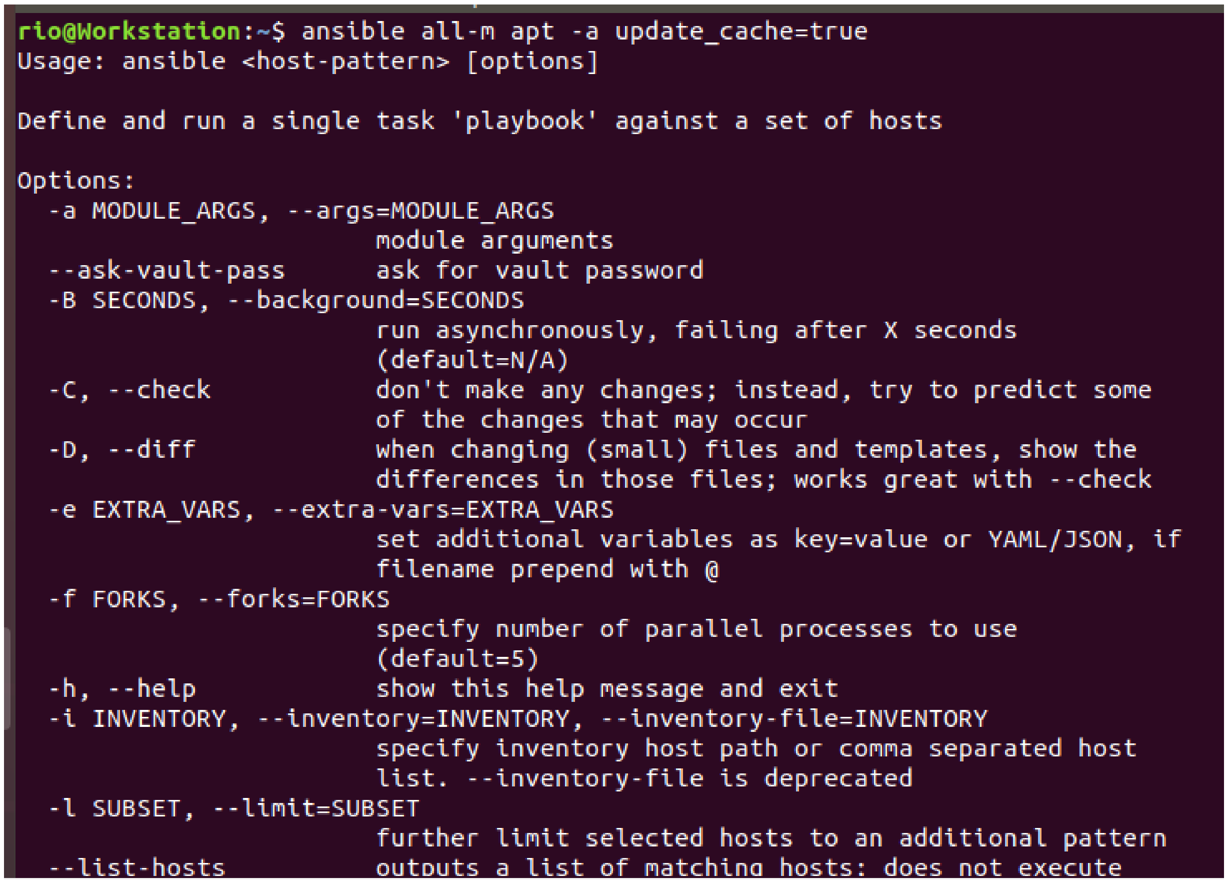
Task 1: Run elevated ad hoc commands

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:

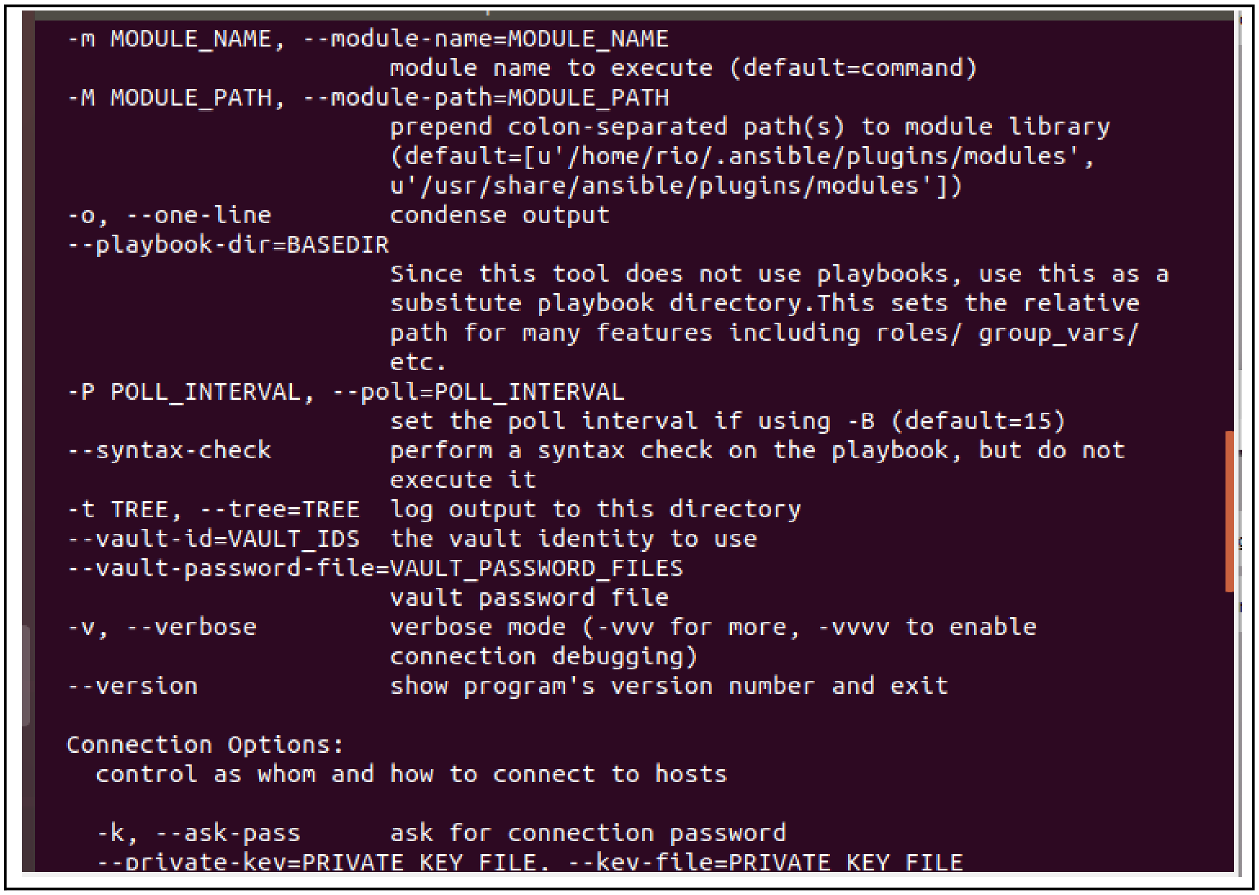


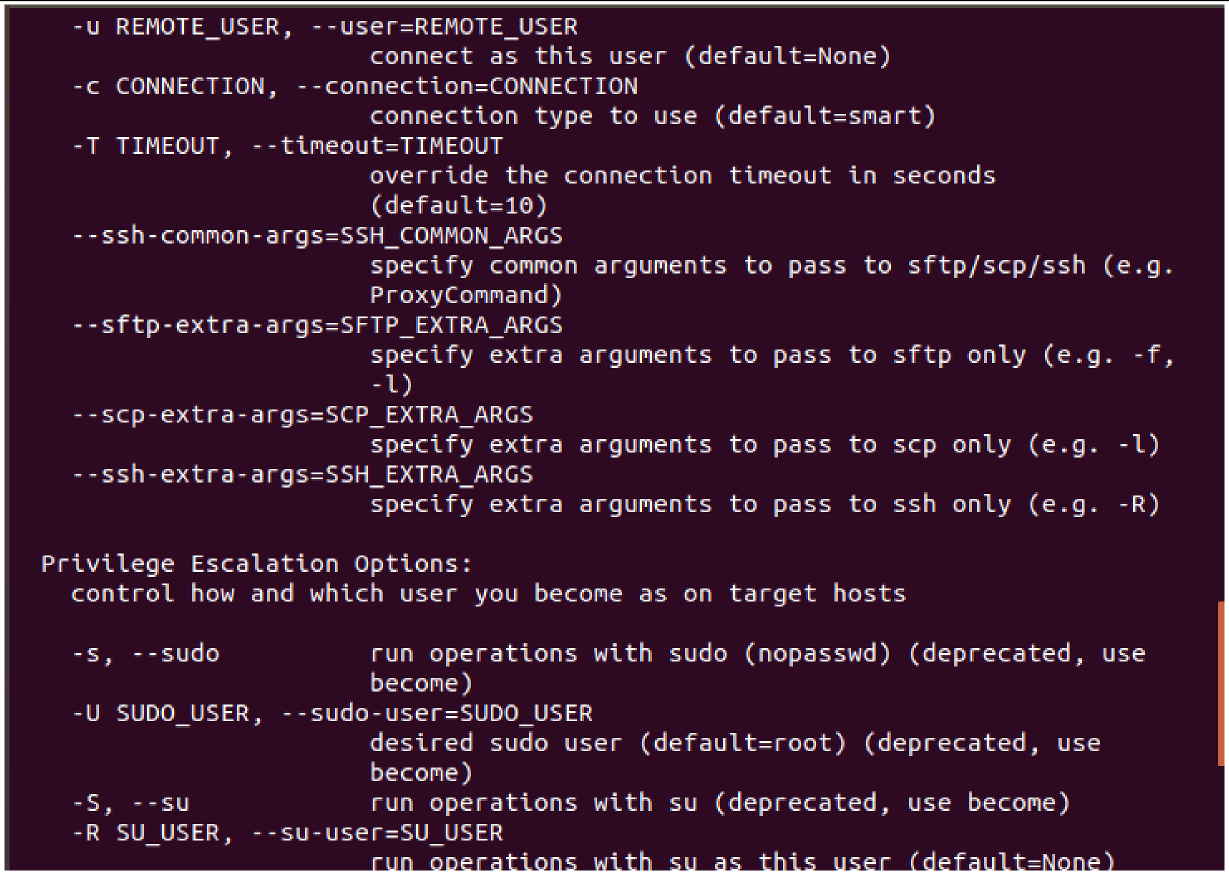
```
E: Invalid operation update]
rio@Workstation:~$ sudo apt update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://ph.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://ph.archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 88.7 kB in 11s (7,938 B/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
rio@Workstation:~$
```

ansible all -m apt -a update_cache=true

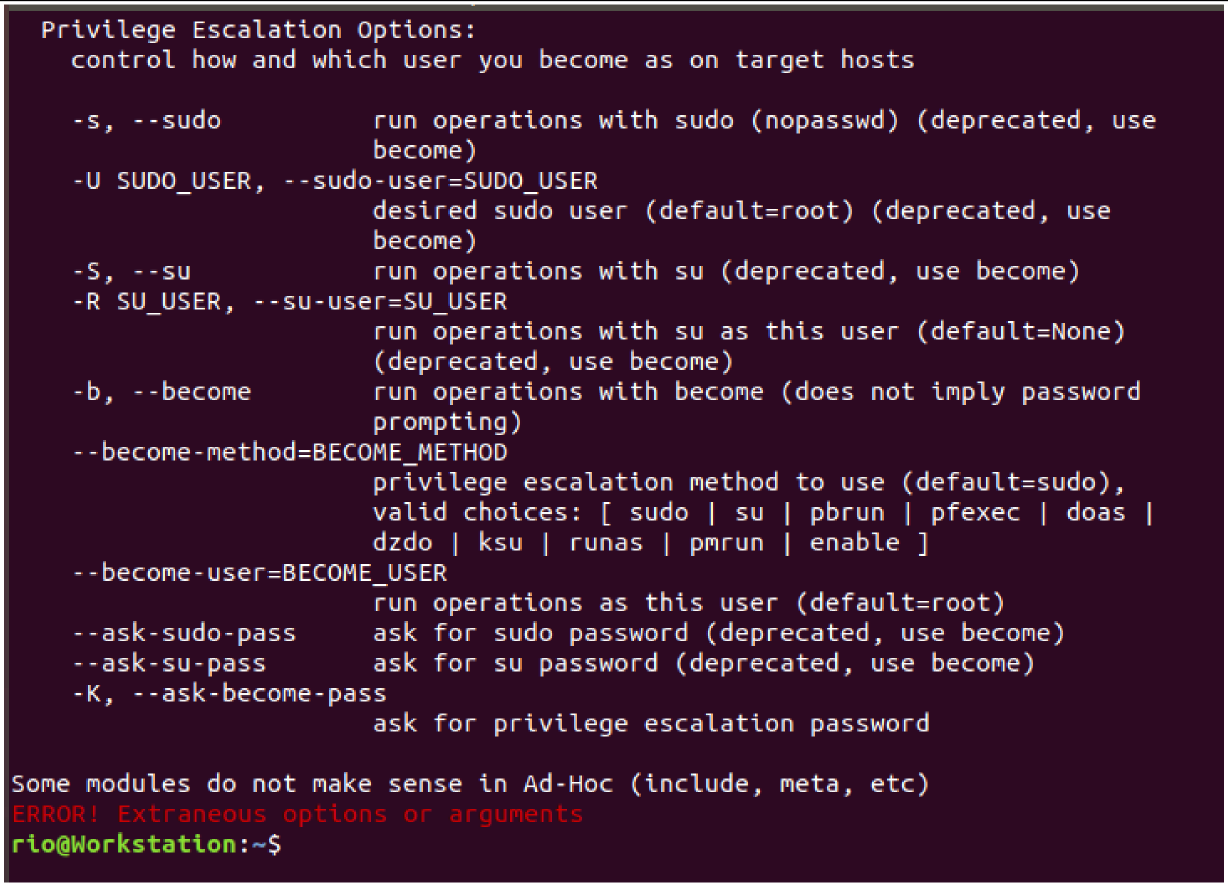


```
rio@Workstation:~$ ansible all-m apt -a update_cache=true
Usage: ansible <host-pattern> [options]

Define and run a single task 'playbook' against a set of hosts

Options:
  -a MODULE_ARGS, --args=MODULE_ARGS
                        module arguments
  --ask-vault-pass      ask for vault password
  -B SECONDS, --background=SECONDS
                        run asynchronously, failing after X seconds
                        (default=N/A)
  -C, --check           don't make any changes; instead, try to predict some
                        of the changes that may occur
  -D, --diff            when changing (small) files and templates, show the
                        differences in those files; works great with --check
  -e EXTRA_VARS, --extra-vars=EXTRA_VARS
                        set additional variables as key=value or YAML/JSON, if
                        filename prepend with @
  -f FORKS, --forks=FORKS
                        specify number of parallel processes to use
                        (default=5)
  -h, --help            show this help message and exit
  -i INVENTORY, --inventory=INVENTORY, --inventory-file=INVENTORY
                        specify inventory host path or comma separated host
                        list. --inventory-file is deprecated
  -l SUBSET, --limit=SUBSET
                        further limit selected hosts to an additional pattern
  --list-hosts          outputs a list of matching hosts; does not execute
```


```
-m MODULE_NAME, --module-name=MODULE_NAME
                      module name to execute (default=command)
-M MODULE_PATH, --module-path=MODULE_PATH
                      prepend colon-separated path(s) to module library
                      (default=[u'/home/rio/.ansible/plugins/modules',
                      u'/usr/share/ansible/plugins/modules'])
-o, --one-line        condense output
--playbook-dir=BASEDIR
                      Since this tool does not use playbooks, use this as a
                      subsitute playbook directory.This sets the relative
                      path for many features including roles/ group_vars/
                      etc.
-P POLL_INTERVAL, --poll=POLL_INTERVAL
                      set the poll interval if using -B (default=15)
--syntax-check        perform a syntax check on the playbook, but do not
                      execute it
-t TREE, --tree=TREE  log output to this directory
--vault-id=VAULT_IDS  the vault identity to use
--vault-password-file=VAULT_PASSWORD_FILES
                      vault password file
-v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                      connection debugging)
--version             show program's version number and exit

Connection Options:
  control as whom and how to connect to hosts

  -k, --ask-pass      ask for connection password
  --private-key=PRIVATE_KEY_FILE, --key-file=PRIVATE_KEY_FILE
```



```
    -u REMOTE_USER, --user=REMOTE_USER
                        connect as this user (default=None)
    -c CONNECTION, --connection=CONNECTION
                        connection type to use (default=smart)
    -T TIMEOUT, --timeout=TIMEOUT
                        override the connection timeout in seconds
                        (default=10)
    --ssh-common-args=SSH_COMMON_ARGS
                        specify common arguments to pass to sftp/scp/ssh (e.g.
                        ProxyCommand)
    --sftp-extra-args=SFTP_EXTRA_ARGS
                        specify extra arguments to pass to sftp only (e.g. -f,
                        -l)
    --scp-extra-args=SCP_EXTRA_ARGS
                        specify extra arguments to pass to scp only (e.g. -l)
    --ssh-extra-args=SSH_EXTRA_ARGS
                        specify extra arguments to pass to ssh only (e.g. -R)

  Privilege Escalation Options:
    control how and which user you become as on target hosts

    -s, --sudo          run operations with sudo (nopasswd) (deprecated, use
                        become)
    -U SUDO_USER, --sudo-user=SUDO_USER
                        desired sudo user (default=root) (deprecated, use
                        become)
    -S, --su            run operations with su (deprecated, use become)
    -R SU_USER, --su-user=SU_USER
                        run operations with su as this user (default=None)
```

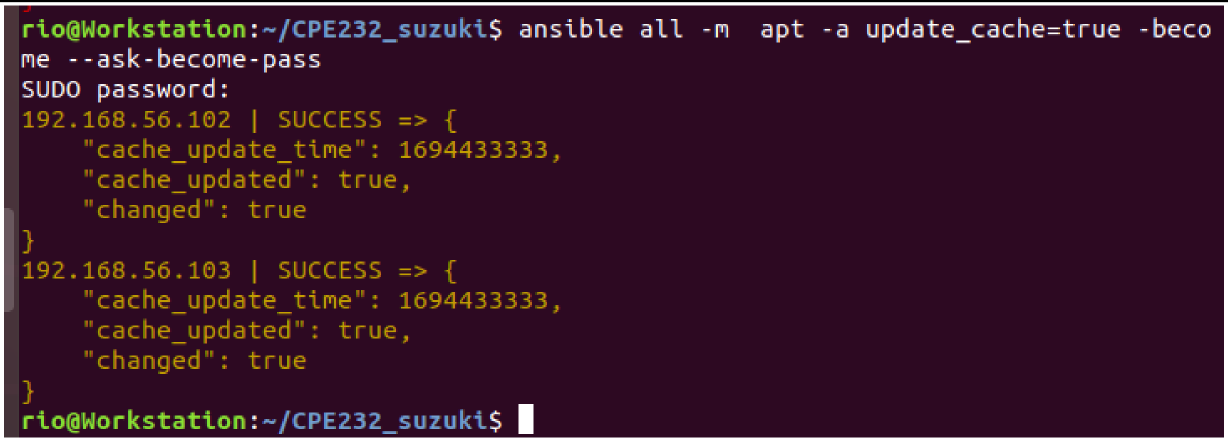


```
  Privilege Escalation Options:
    control how and which user you become as on target hosts

    -s, --sudo          run operations with sudo (nopasswd) (deprecated, use
                        become)
    -U SUDO_USER, --sudo-user=SUDO_USER
                        desired sudo user (default=root) (deprecated, use
                        become)
    -S, --su            run operations with su (deprecated, use become)
    -R SU_USER, --su-user=SU_USER
                        run operations with su as this user (default=None)
                        (deprecated, use become)
    -b, --become        run operations with become (does not imply password
                        prompting)
    --become-method=BECOME_METHOD
                        privilege escalation method to use (default=sudo),
                        valid choices: [ sudo | su | pbrun | pfexec | doas |
                        dzdo | ksu | runas | pmrun | enable ]
    --become-user=BECOME_USER
                        run operations as this user (default=root)
    --ask-sudo-pass     ask for sudo password (deprecated, use become)
    --ask-su-pass       ask for su password (deprecated, use become)
    -K, --ask-become-pass
                        ask for privilege escalation password

Some modules do not make sense in Ad-Hoc (include, meta, etc)
ERROR! Extraneous options or arguments
rio@Workstation:~$
```

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

- **When I execute the command it says error.**

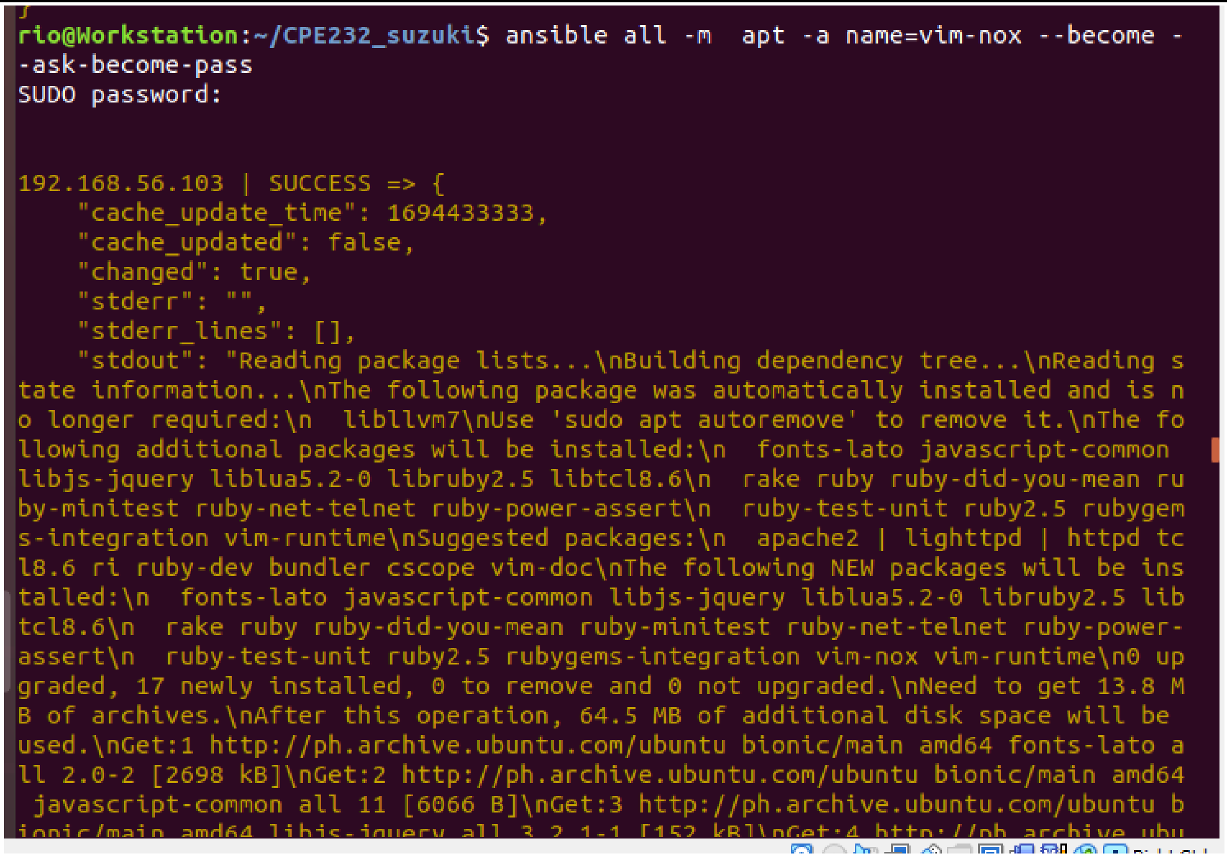
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same 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.



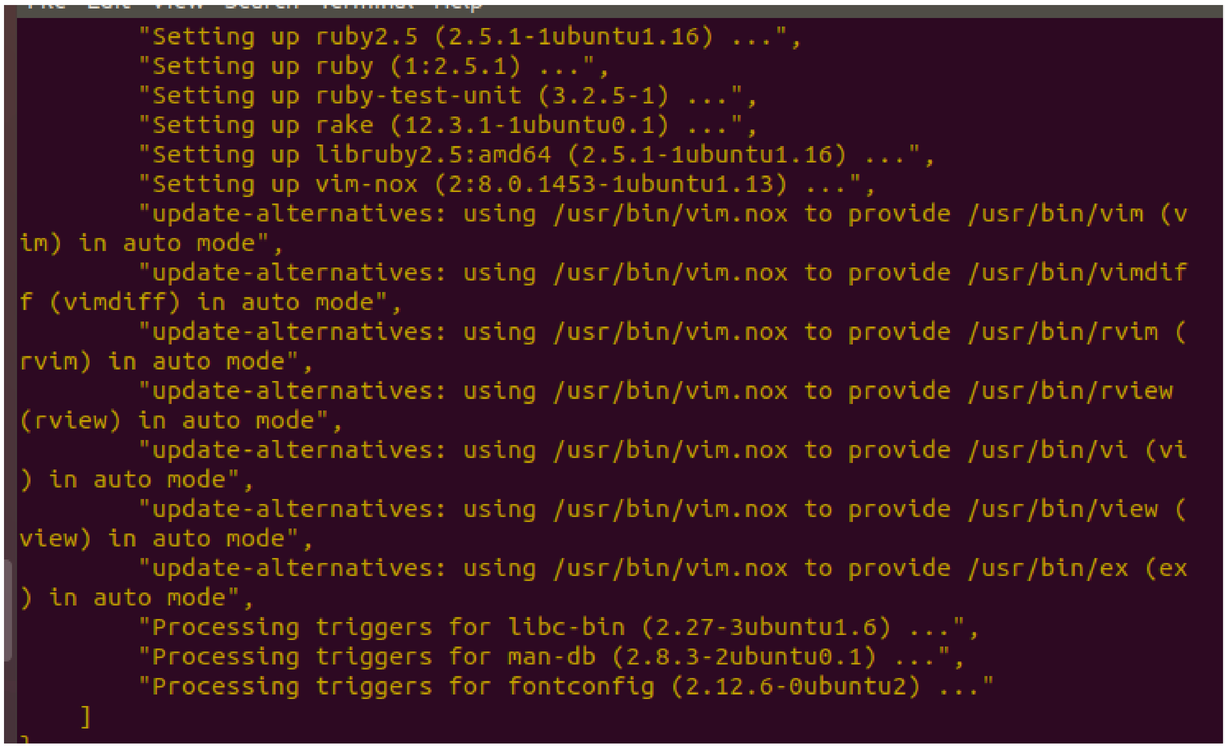
```
rio@Workstation:~/CPE232_suzuki$ ansible all -m  apt -a update_cache=true -beco
me --ask-become-pass
SUDO password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": true,
    "changed": true
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": true,
    "changed": true
}
rio@Workstation:~/CPE232_suzuki$
```

You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

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.

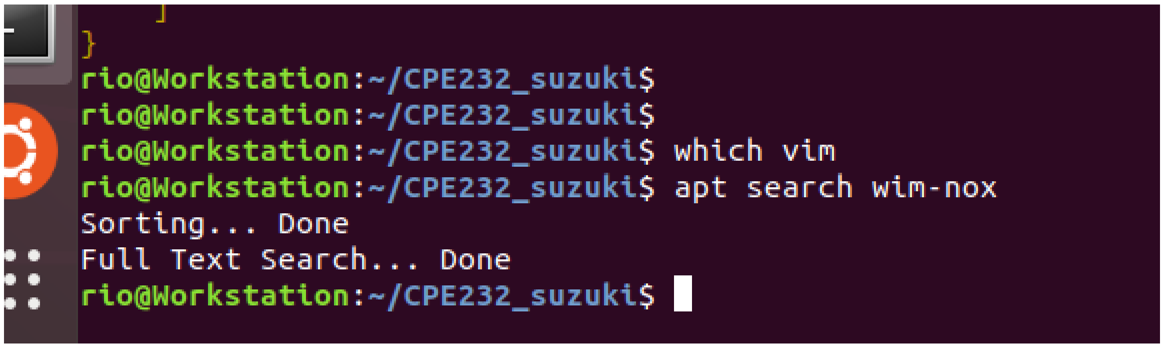


```
rio@Workstation:~/CPE232_suzuki$ ansible all -m  apt -a name=vim-nox --become -
-ask-become-pass
SUDO password:


192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe fo
llowing 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 ru
by-minitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygem
s-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd tc
l8.6 ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be ins
talled:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 lib
tcl8.6\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 up
graded, 17 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 13.8 M
B 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 a
ll 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 b
ionic/main amd64 libjs-jquery all 3.2.1-1 [152 kB]\nGet:4 http://ph.archive.ubu
```



```
        "Setting up ruby2.5 (2.5.1-1ubuntu1.16) ...",
        "Setting up ruby (1:2.5.1) ...",
        "Setting up ruby-test-unit (3.2.5-1) ...",
        "Setting up rake (12.3.1-1ubuntu0.1) ...",
        "Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.16) ...",
        "Setting up vim-nox (2:8.0.1453-1ubuntu1.13) ...",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vim (v
im) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vimdif
f (vimdiff) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/rvim (
rvim) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/rview
(rview) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vi (vi
) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/view (
view) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/ex (ex
) in auto mode",
        "Processing triggers for libc-bin (2.27-3ubuntu1.6) ...",
        "Processing triggers for man-db (2.8.3-2ubuntu0.1) ...",
        "Processing triggers for fontconfig (2.12.6-0ubuntu2) ..."
    ]
```

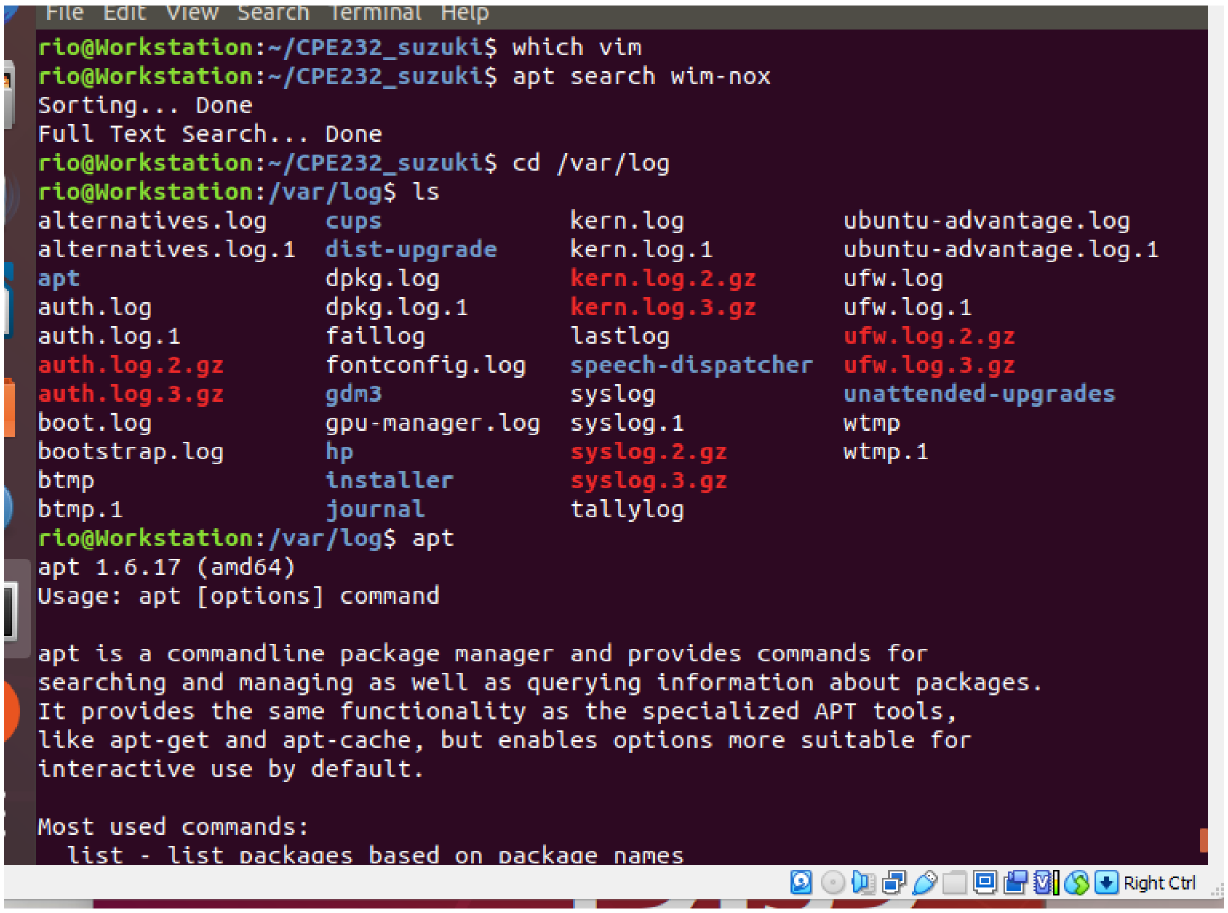
2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful?



```
    ]
}
rio@Workstation:~/CPE232_suzuki$
rio@Workstation:~/CPE232_suzuki$
rio@Workstation:~/CPE232_suzuki$ which vim
rio@Workstation:~/CPE232_suzuki$ apt search wim-nox
Sorting... Done
Full Text Search... Done
rio@Workstation:~/CPE232_suzuki$ 
```

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.

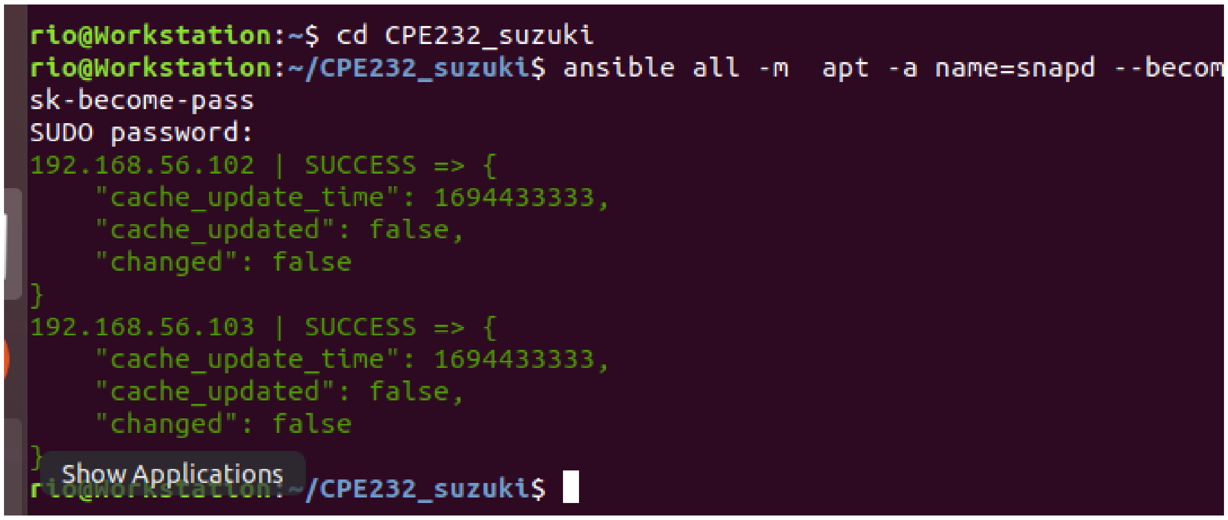
- **It shows what was done throughout the system, wherein in this instance, ansible was installed.**



```
rio@Workstation:~/CPE232_suzuki$ which vim
rio@Workstation:~/CPE232_suzuki$ apt search wim-nox
Sorting... Done
Full Text Search... Done
rio@Workstation:~/CPE232_suzuki$ cd /var/log
rio@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
btmp                installer        syslog.3.gz
btmp.1              journal          tallylog
rio@Workstation:/var/log$ apt
apt 1.6.17 (amd64)
Usage: apt [options] command

apt is a commandline package manager and provides commands for
searching and managing as well as querying information about packages.
It provides the same functionality as the specialized APT tools,
like apt-get and apt-cache, but enables options more suitable for
interactive use by default.

Most used commands:
  list - list packages based on package names
```

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.

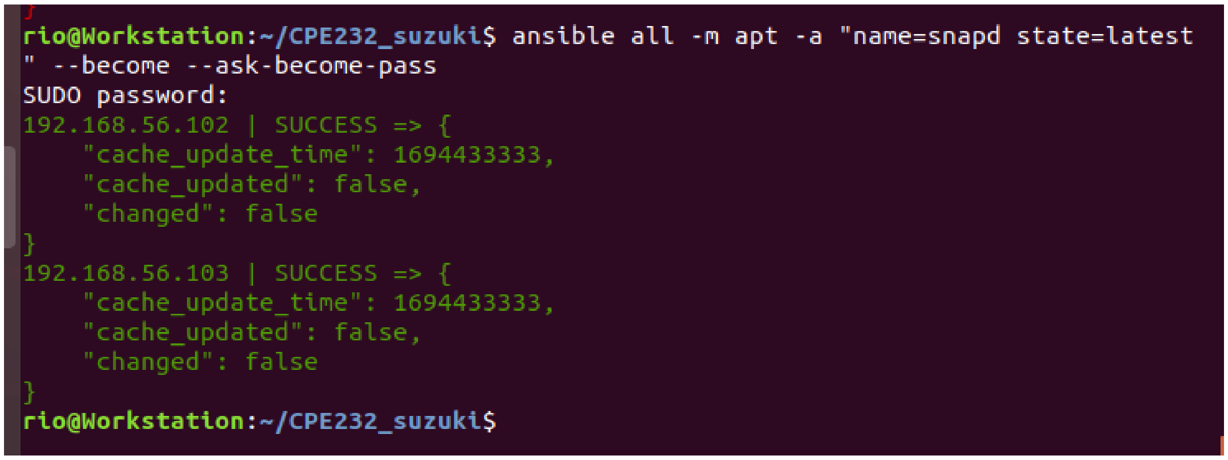
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*



```
rio@Workstation:~$ cd CPE232_suzuki
rio@Workstation:~/CPE232_suzuki$ ansible all -m  apt -a name=snapd --becom
sk-become-pass
SUDO password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": false,
    "changed": false
}
rio@Workstation:~/CPE232_suzuki$
```

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

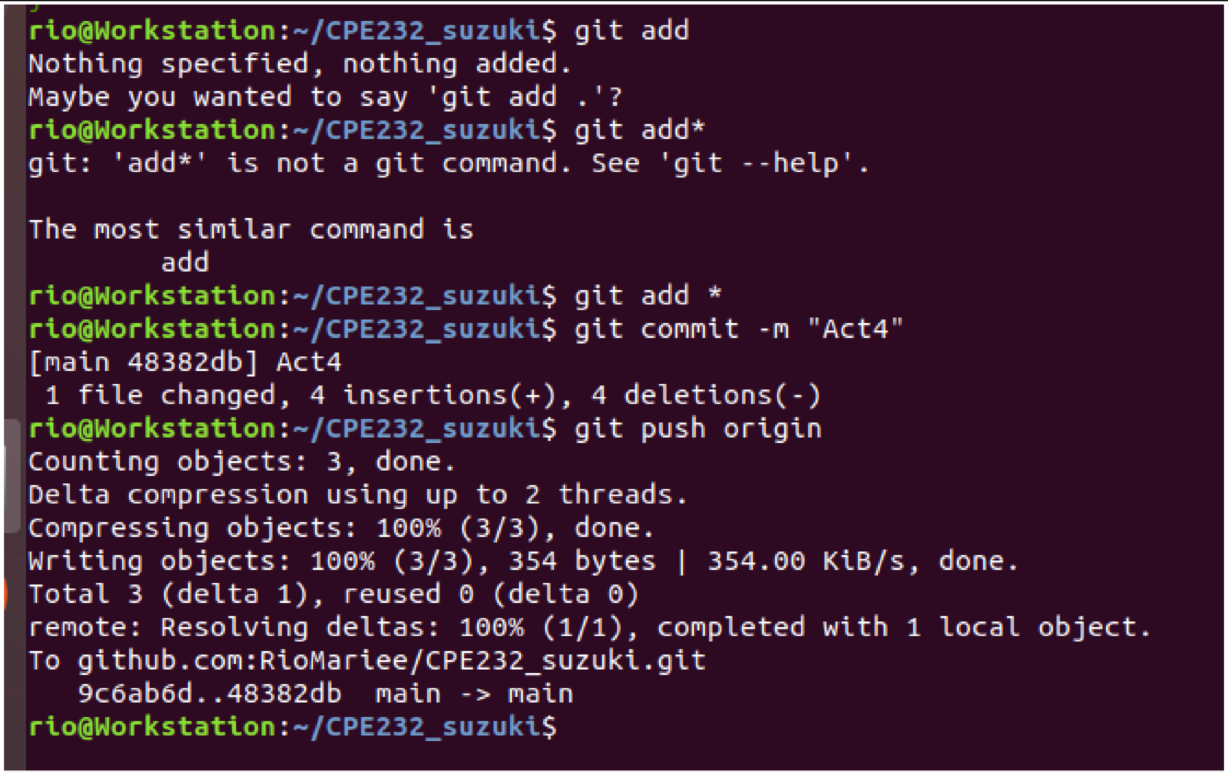
   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*



```
rio@Workstation:~/CPE232_suzuki$ ansible all -m apt -a "name=snapd state=latest
" --become --ask-become-pass
SUDO password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694433333,
    "cache_updated": false,
    "changed": false
}
rio@Workstation:~/CPE232_suzuki$
```

   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.

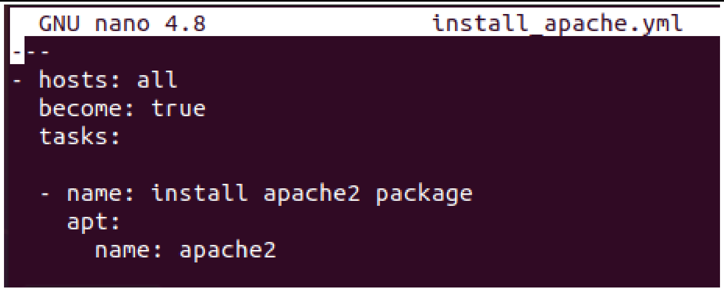


```
rio@Workstation:~/CPE232_suzuki$ git add
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'?
rio@Workstation:~/CPE232_suzuki$ git add*
git: 'add*' is not a git command. See 'git --help'.

The most similar command is
        add
rio@Workstation:~/CPE232_suzuki$ git add *
rio@Workstation:~/CPE232_suzuki$ git commit -m "Act4"
[main 48382db] Act4
 1 file changed, 4 insertions(+), 4 deletions(-)
rio@Workstation:~/CPE232_suzuki$ git push origin
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 354 bytes | 354.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:RioMariee/CPE232_suzuki.git
   9c6ab6d..48382db  main -> main
rio@Workstation:~/CPE232_suzuki$
```

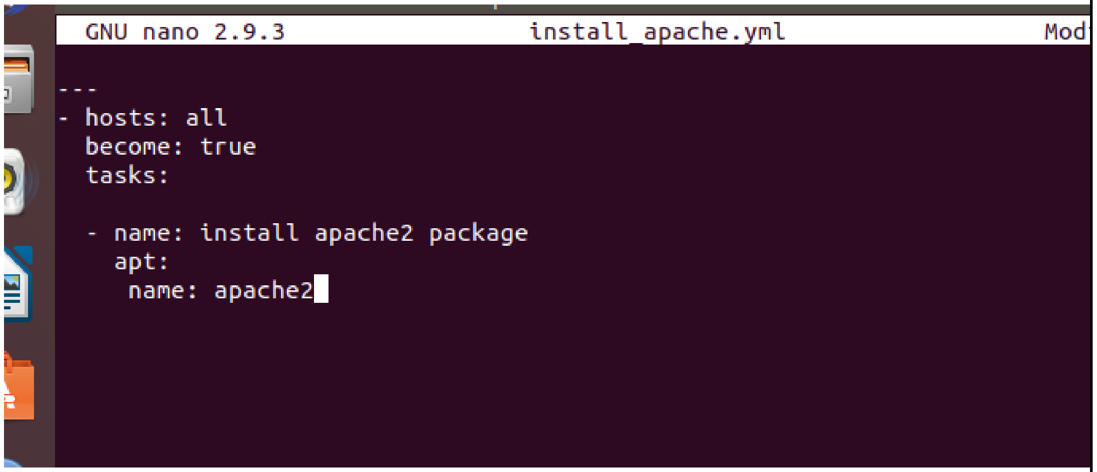
Task 2: Writing our First Playbook

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:



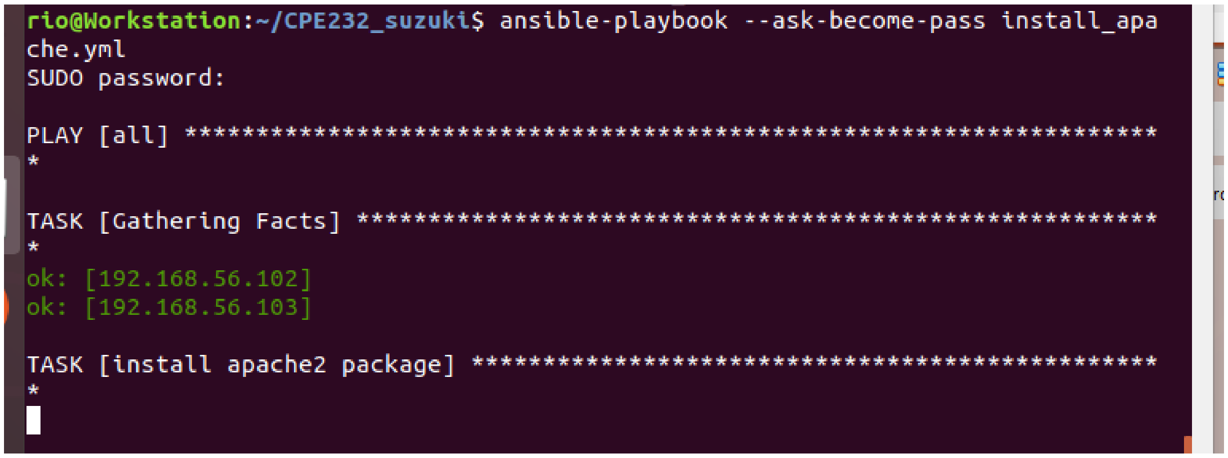
```
  GNU nano 4.8                    install_apache.yml
---
- hosts: all
  become: true
  tasks:

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



```
  GNU nano 2.9.3                    install_apache.yml                    Mod

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

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

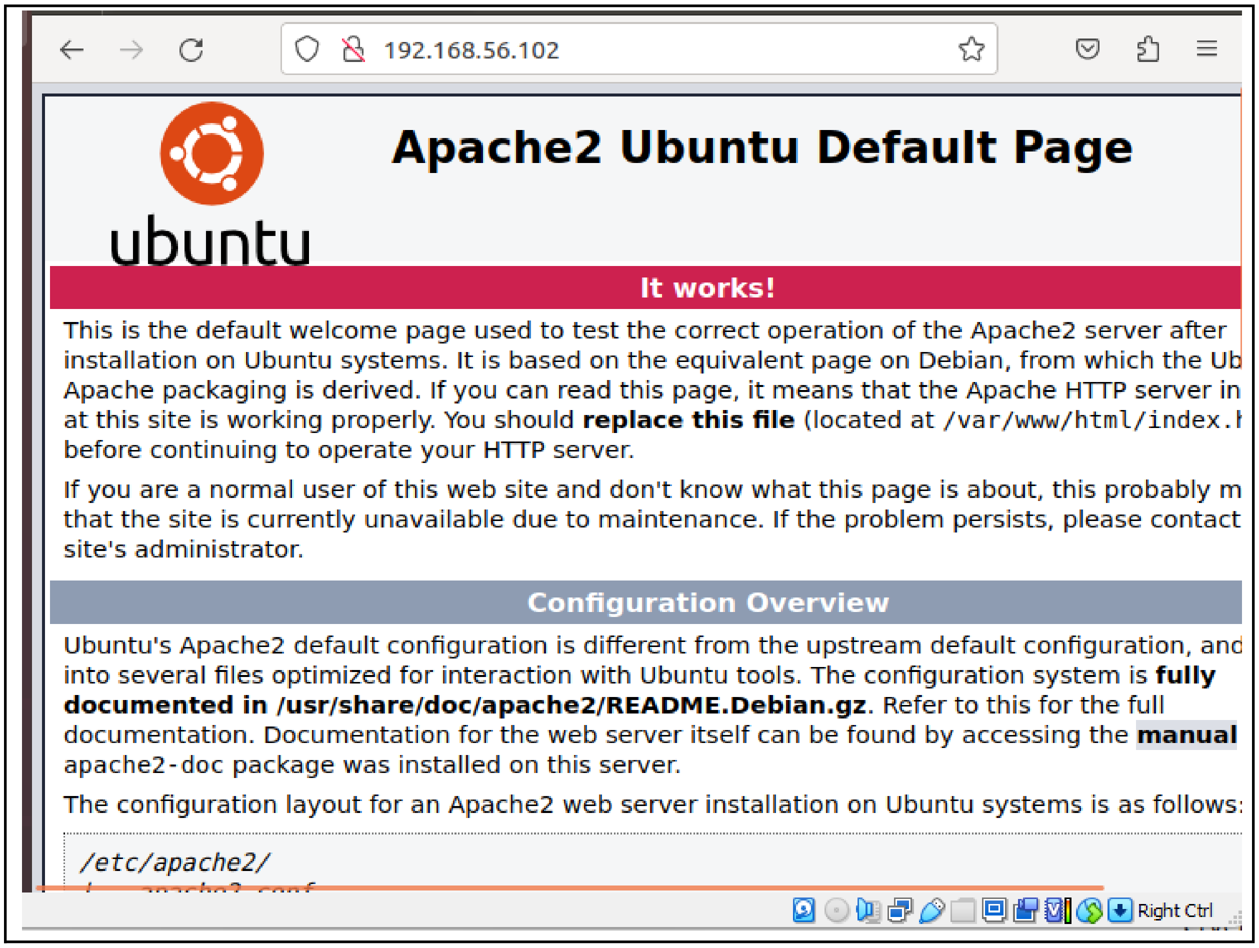
Make sure to save the file. Take note also of the alignments of the texts.

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



```
rio@Workstation:~/CPE232_suzuki$ ansible-playbook --ask-become-pass install_apa
che.yml
SUDO password:

PLAY [all] *********************************************************************
*

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

TASK [install apache2 package] *************************************************
*
```

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.



192.168.56.102

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 Ub Apache packaging is derived. If you can read this page, it means that the Apache HTTP server in at this site is working properly. You should **replace this file** (located at `/var/www/html/index.h` before continuing to operate your HTTP server.

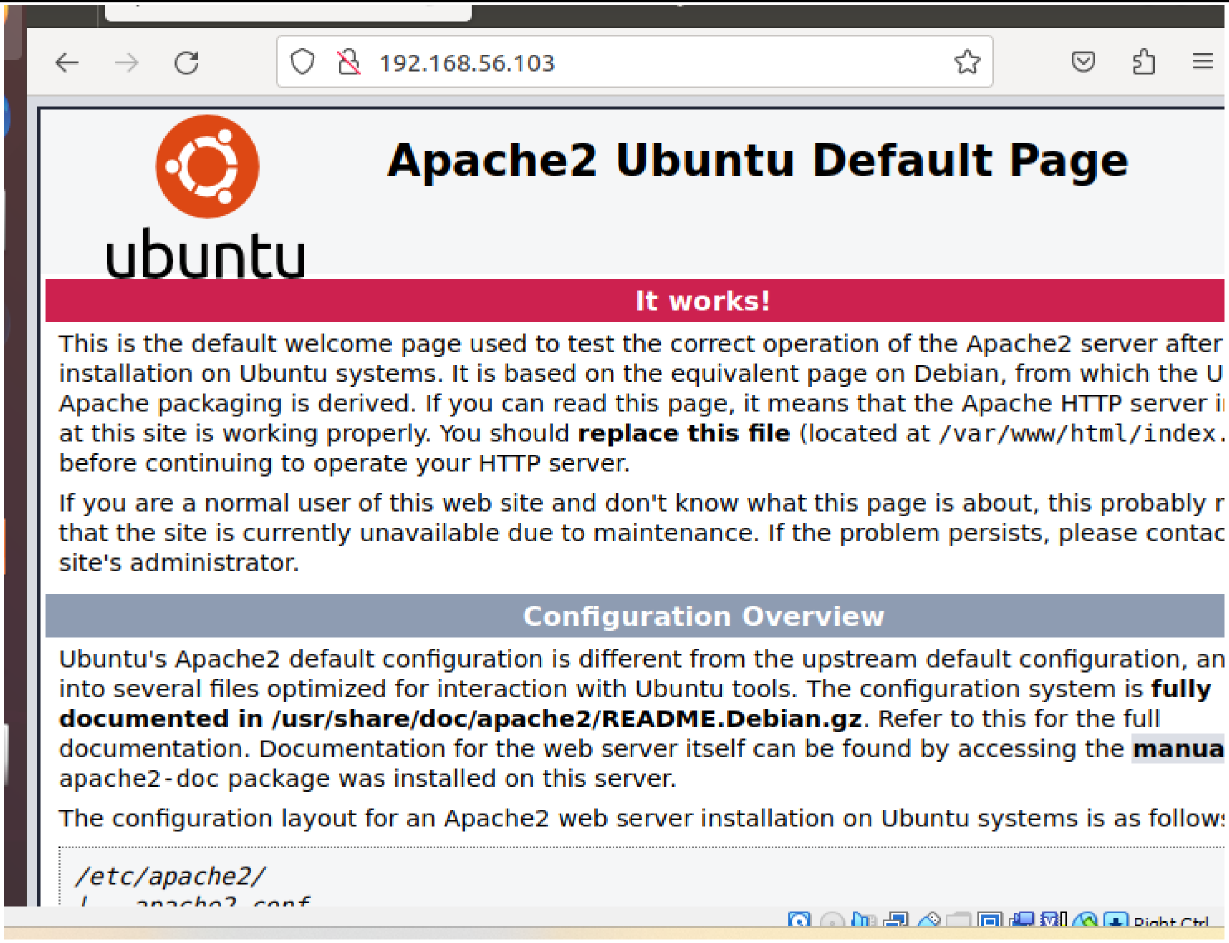
If you are a normal user of this web site and don't know what this page is about, this probably m that the site is currently unavailable due to maintenance. If the problem persists, please contact site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and 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** 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/
```



192.168.56.103

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 U Apache packaging is derived. If you can read this page, it means that the Apache HTTP server i at this site is working properly. You should **replace this file** (located at `/var/www/html/index.` before continuing to operate your HTTP server.

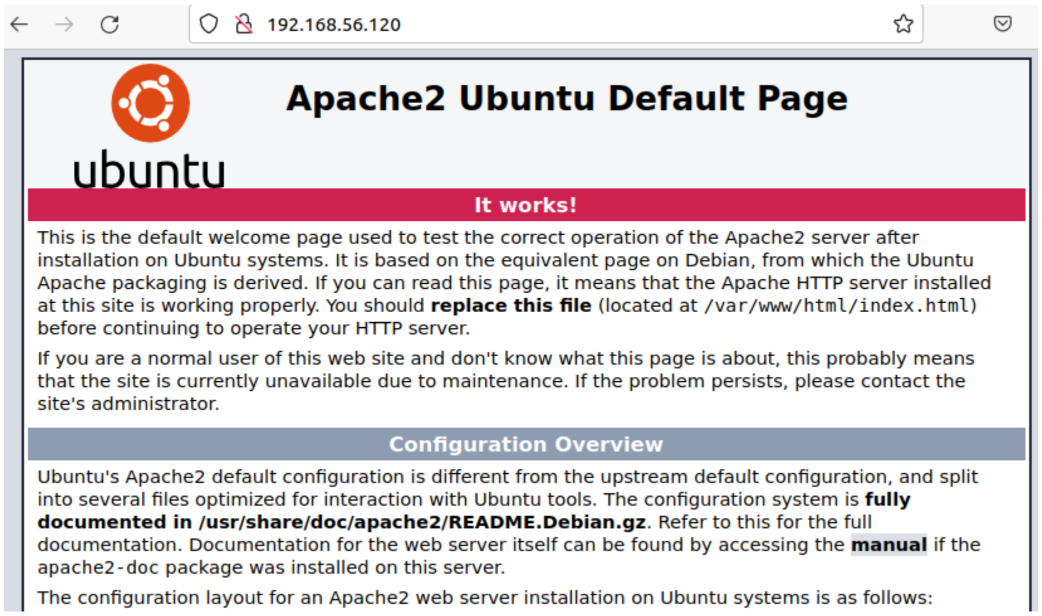
If you are a normal user of this web site and don't know what this page is about, this probably r that the site is currently unavailable due to maintenance. If the problem persists, please contac site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, an 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 **manua** apache2-doc package was installed on this server.

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

```
/etc/apache2/
```



192.168.56.120

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 the site's administrator.

Configuration Overview

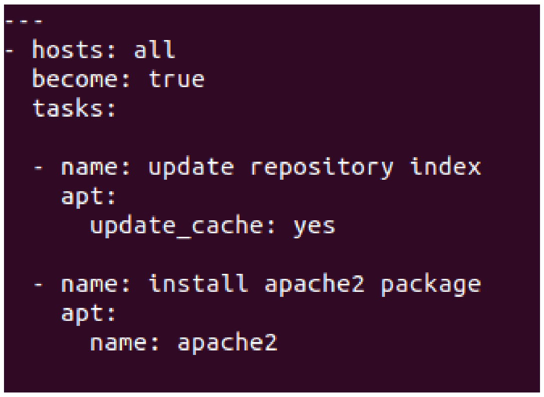
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. 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:

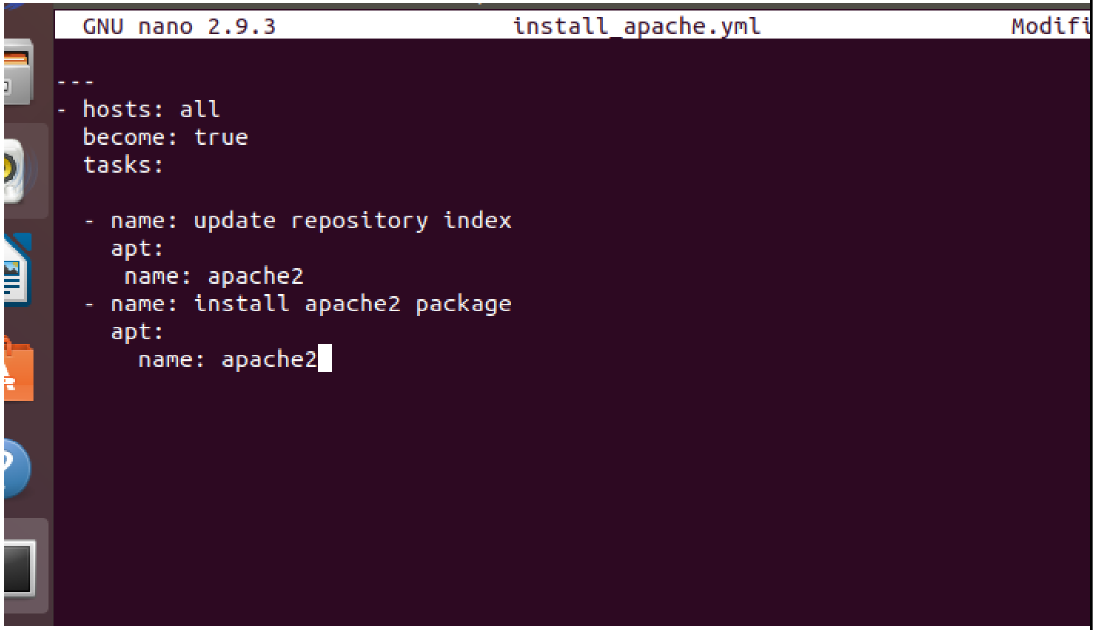
4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output?

- **It will not work because the package name has been changed.**

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.



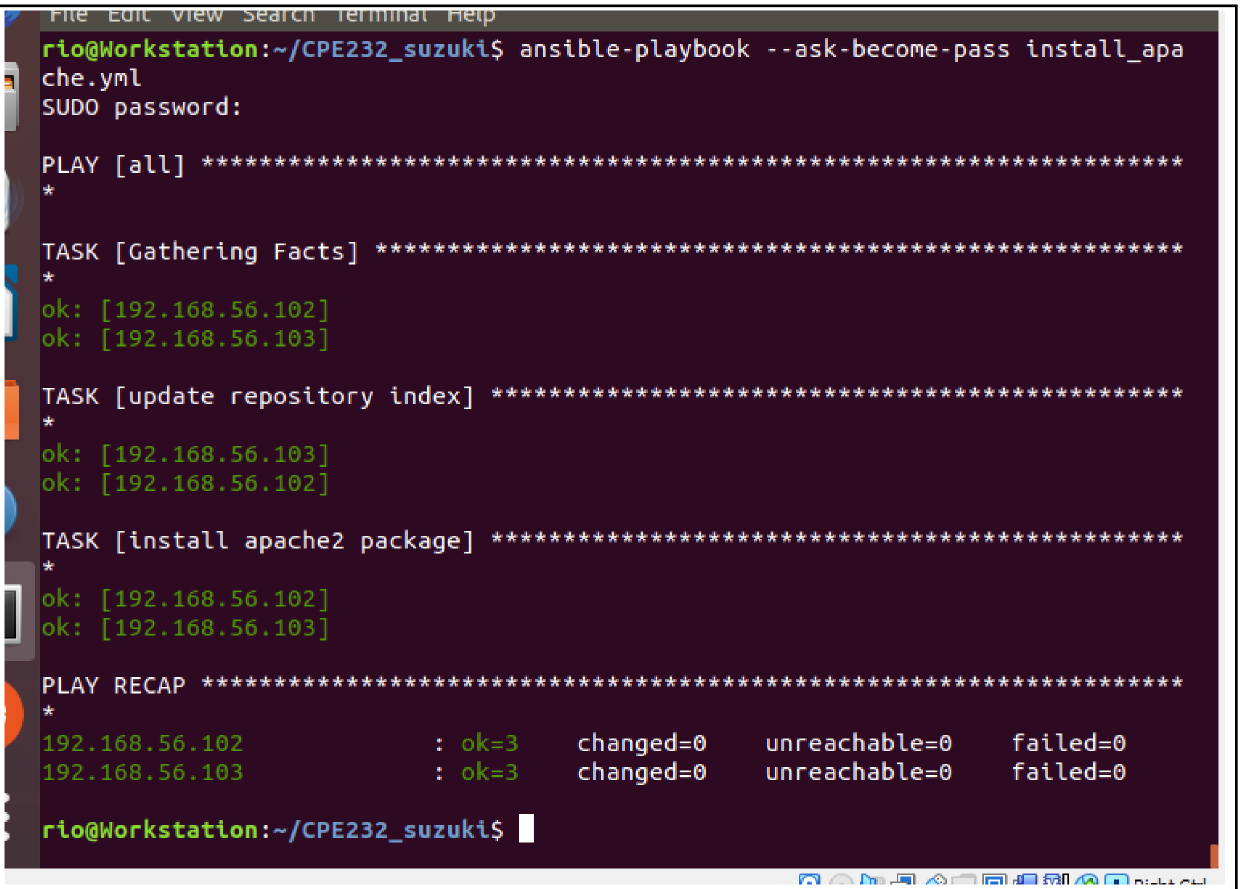
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

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



```
  GNU nano 2.9.3                install_apache.yml                    Modifi

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

  - name: update repository index
    apt:
     name: apache2
  - name: install apache2 package
    apt:
      name: apache2
```

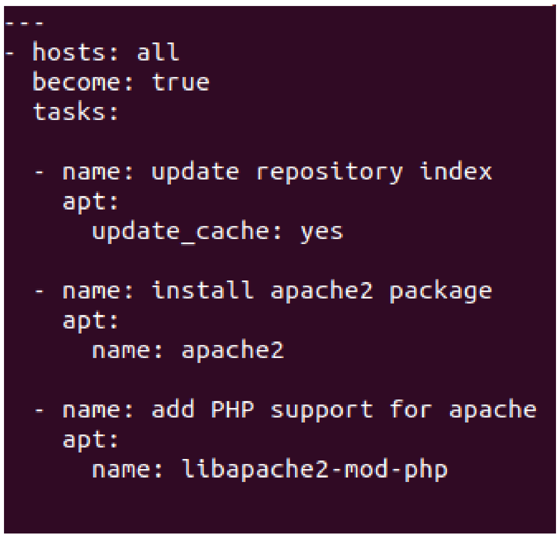
Save the changes to this file and exit.

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

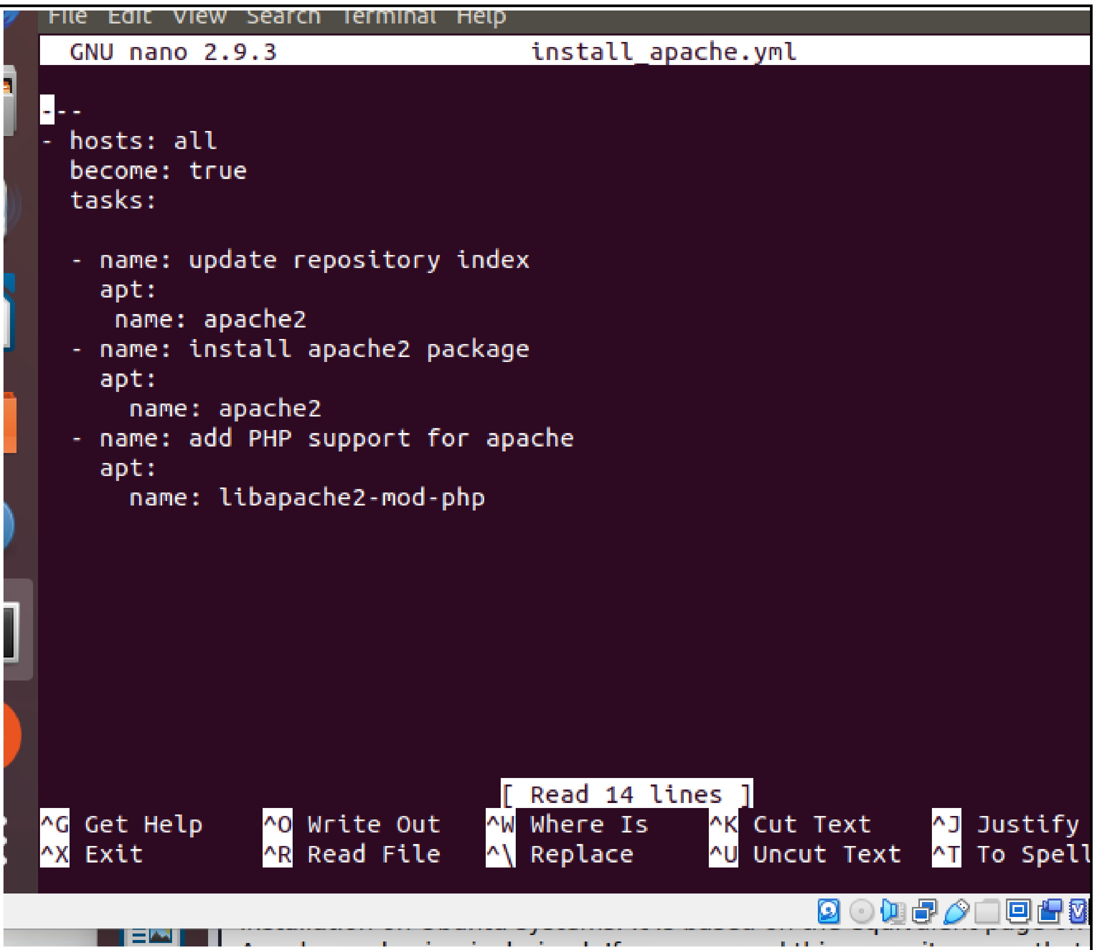


```
rio@Workstation:~/CPE232_suzuki$ ansible-playbook --ask-become-pass install_apa
che.yml
SUDO password:

PLAY [all] *********************************************************************
*

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

TASK [update repository index] *************************************************
*
ok: [192.168.56.103]
ok: [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=0    unreachable=0    failed=0
192.168.56.103             : ok=3    changed=0    unreachable=0    failed=0

rio@Workstation:~/CPE232_suzuki$
```

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



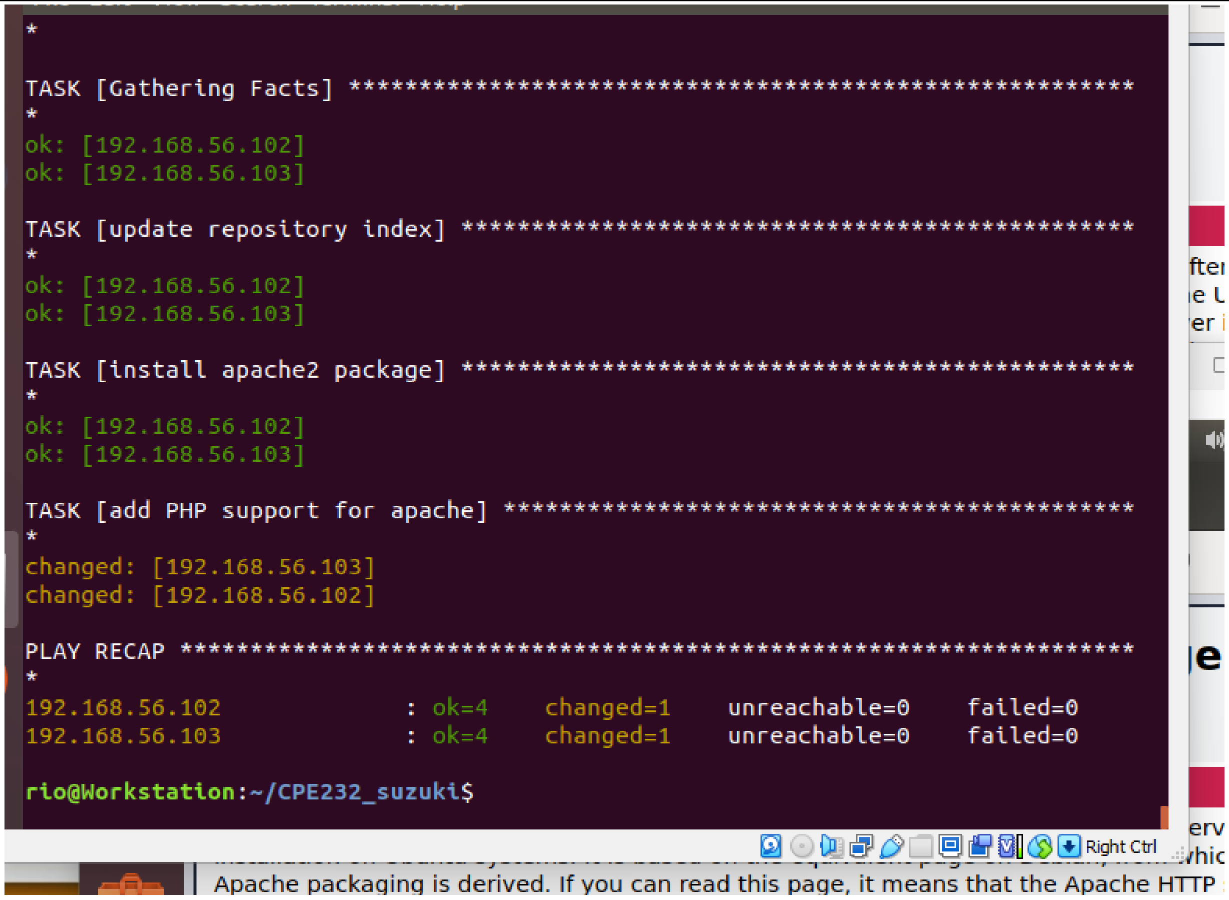
```
---
- hosts: all
  become: true
  tasks:

  - 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
```



```
  GNU nano 2.9.3                              install_apache.yml

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

  - name: update repository index
    apt:
     name: apache2
  - name: install apache2 package
    apt:
      name: apache2
  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php

                                  [ Read 14 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

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?



```
*

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

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

PLAY RECAP ***************************************************************************
*
192.168.56.102             : ok=4    changed=1    unreachable=0    failed=0
192.168.56.103             : ok=4    changed=1    unreachable=0    failed=0

rio@Workstation:~/CPE232_suzuki$
```

Apache packaging is derived. If you can read this page, it means that the Apache HTTP

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

https://github.com/RioMariee/CPE232_suzuki.git

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - The importance of the playbooks can define the state we desire for all the remote servers that we manage. Also, The condition that we added in the playbook can be saved, and it means that the play can be shared and used again.
2. Summarize what we have done on this activity.
   - In this activity, we created a .cfg and file for the ansible and put a script inside of it. We also used various commands that made changes to the remote machine. Throughout the activity, I was able to learn ad hoc commands that would install,update and upgrade packages in the remote machine and also created playbooks that record and execute ansible's configuration.

Conclusion
To conclude,this activity aims for the student to utilize the skills on problem solving since most of the commands don't work on my part, I encounter a lot of problems specially on the installation part. However, I was able to resolve it. Utilizing the commands to make some changes in the remote machines and in creating playbooks is important since it helps the user to have a control/manage on the remote systems and this makes the installation and configuration of each device way more easier, one example of this that we did in the class is the installation of the app called “vlc” using the ansible command.