Name: Buduan, Christian Aaron C.	**Date Performed:** 11/09/2024
Course/Section: CpE31S2	**Date Submitted:** 11.09.2024
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:**

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]

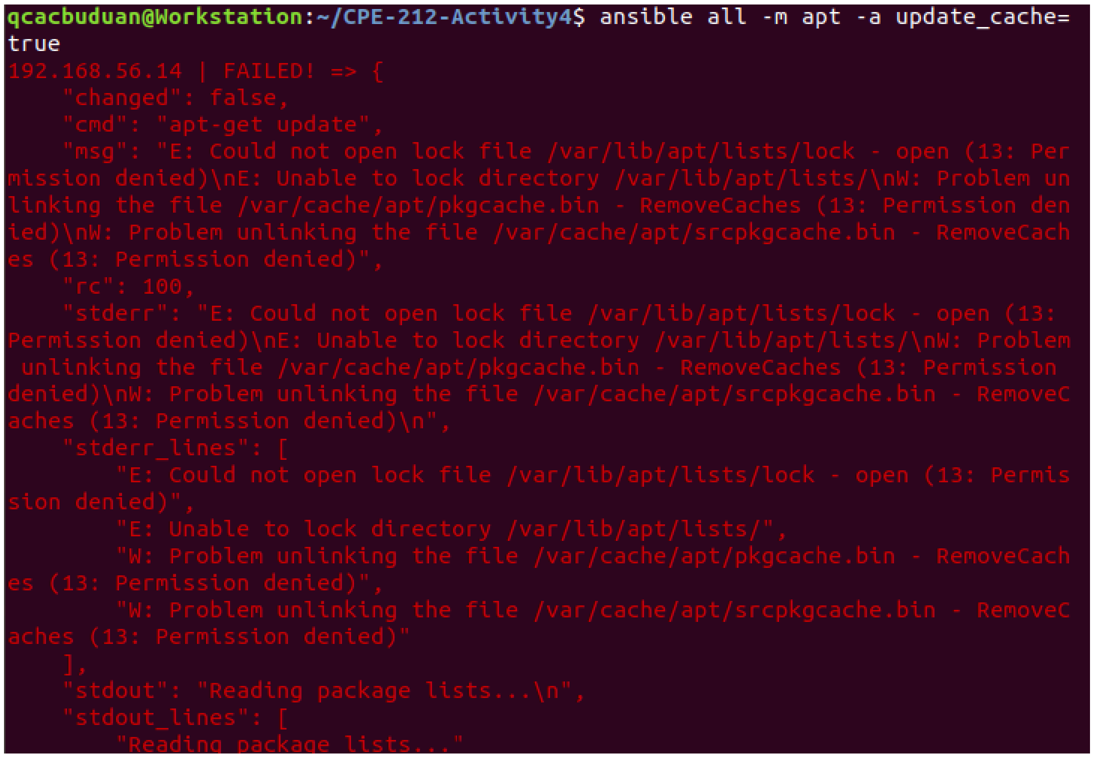
Task 1: Run elevated ad hoc commands

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

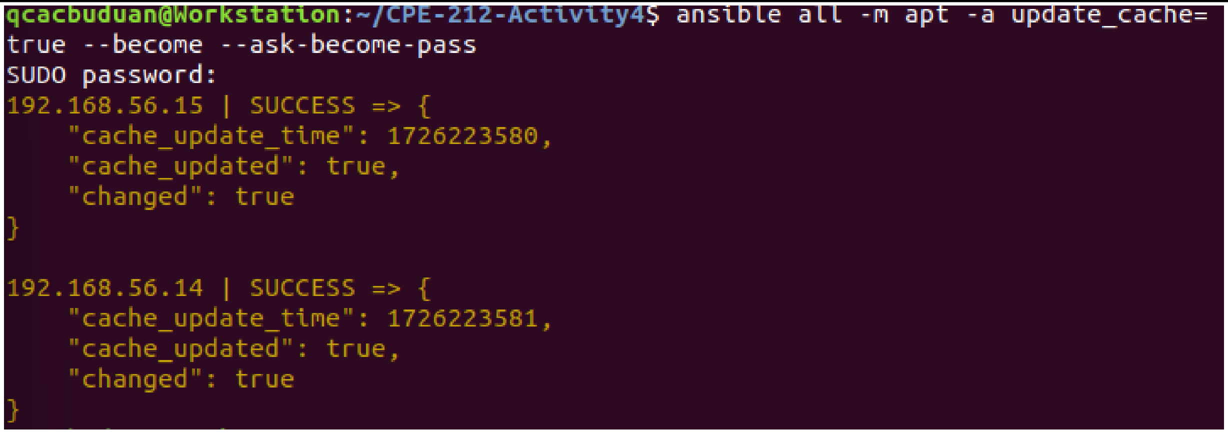
What is the result of the command? Is it successful? No because of I don't have the permission privileges.



```
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible all -m apt -a update_cache=
true
192.168.56.14 | FAILED! => {
    "changed": false,
    "cmd": "apt-get update",
    "msg": "E: Could not open lock file /var/lib/apt/lists/lock - open (13: Per
mission denied)\nE: Unable to lock directory /var/lib/apt/lists/\nW: Problem un
linking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission den
ied)\nW: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCach
es (13: Permission denied)",
    "rc": 100,
    "stderr": "E: Could not open lock file /var/lib/apt/lists/lock - open (13:
Permission denied)\nE: Unable to lock directory /var/lib/apt/lists/\nW: Problem
 unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission
denied)\nW: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveC
aches (13: Permission denied)\n",
    "stderr_lines": [
        "E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permis
sion denied)",
        "E: Unable to lock directory /var/lib/apt/lists/",
        "W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCach
es (13: Permission denied)",
        "W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveC
aches (13: Permission denied)"
    ],
    "stdout": "Reading package lists...\n",
    "stdout_lines": [
        "Reading package lists..."
```

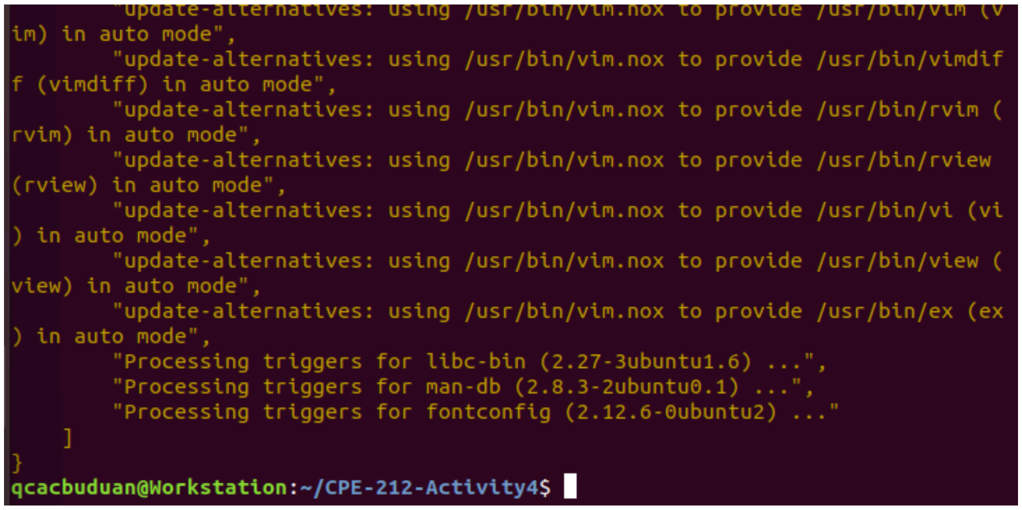
Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same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



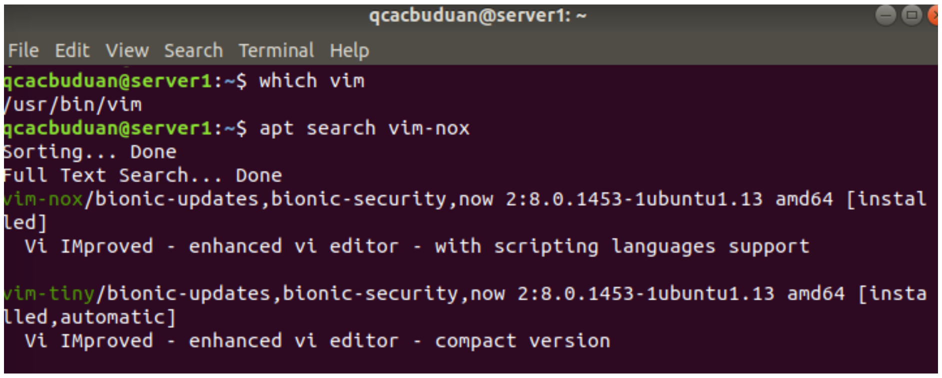
```
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible all -m apt -a update_cache=
true --become --ask-become-pass
SUDO password:
192.168.56.15 | SUCCESS => {
    "cache_update_time": 1726223580,
    "cache_updated": true,
    "changed": true
}

192.168.56.14 | SUCCESS => {
    "cache_update_time": 1726223581,
    "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.

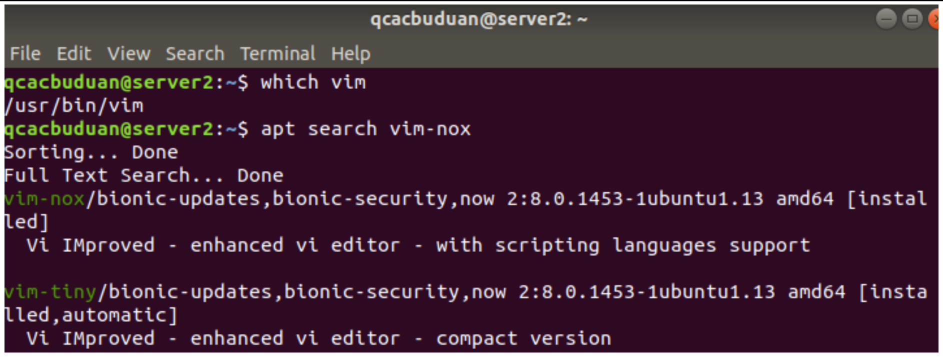


```
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vim (v
im) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vimdif
f (vimdiff) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/rvim (
rvim) in auto 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) ..."
    ]
}
qcacbuduan@Workstation:~/CPE-212-Activity4$
```

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.**

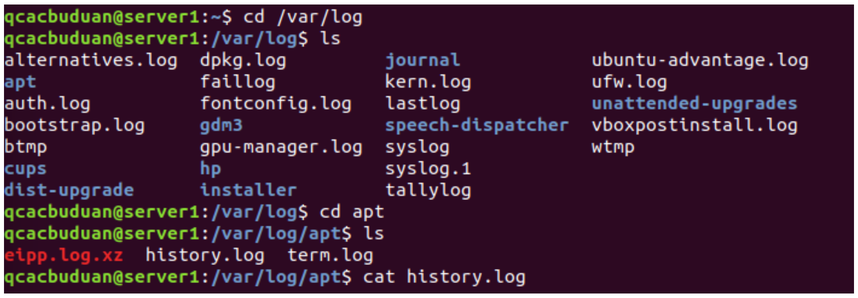


```
qcacbuduan@server1:~$ which vim
/usr/bin/vim
qcacbuduan@server1:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

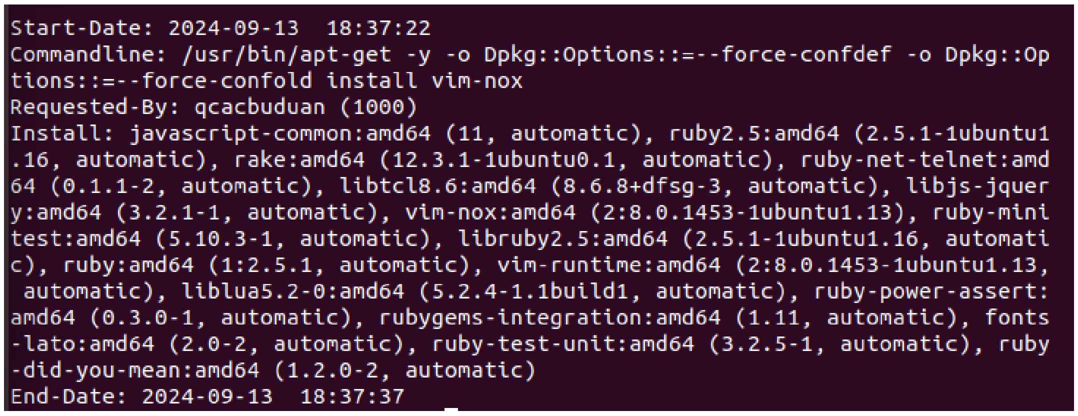


```
qcacbuduan@server2:~$ which vim
/usr/bin/vim
qcacbuduan@server2:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [instal
led]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled,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. **In the history log, it shows all the package changes that were made to the system via apt.**



```
qcacbuduan@server1:~$ cd /var/log
qcacbuduan@server1:/var/log$ ls
alternatives.log  dpkg.log         journal            ubuntu-advantage.log
apt               faillog          kern.log           ufw.log
auth.log          fontconfig.log   lastlog            unattended-upgrades
bootstrap.log     gdm3             speech-dispatcher  vboxpostinstall.log
btmp              gpu-manager.log  syslog             wtmp
cups              hp               syslog.1
dist-upgrade      installer        tallylog
qcacbuduan@server1:/var/log$ cd apt
qcacbuduan@server1:/var/log/apt$ ls
eipp.log.xz  history.log  term.log
qcacbuduan@server1:/var/log/apt$ cat history.log
```

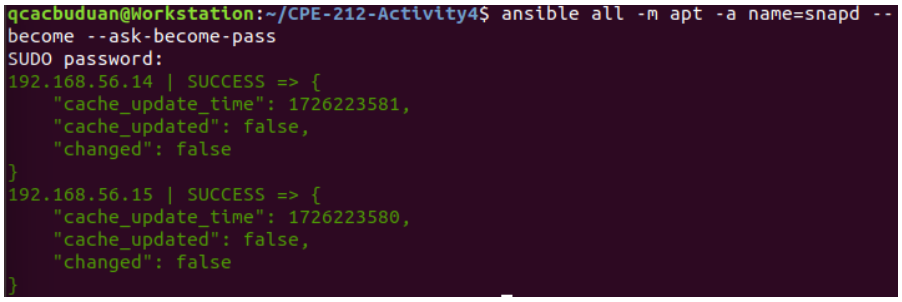


```
Start-Date: 2024-09-13  18:37:22
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Op
tions::=--force-confold install vim-nox
Requested-By: qcacbuduan (1000)
Install: javascript-common:amd64 (11, automatic), ruby2.5:amd64 (2.5.1-1ubuntu1
.16, automatic), rake:amd64 (12.3.1-1ubuntu0.1, 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  18:37:37
```

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? **It did not change anything in the**

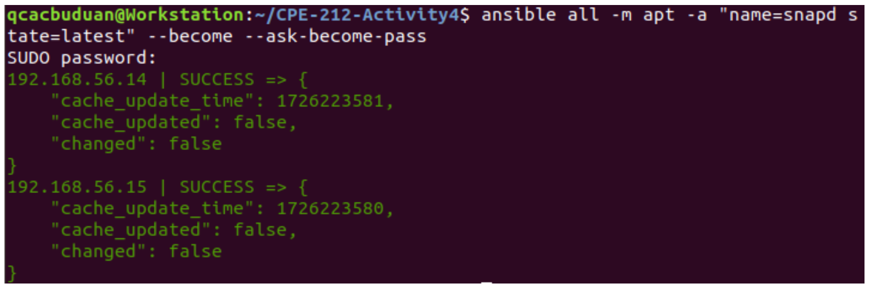
remote servers because the package is already installed on the servers.



```
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible all -m apt -a name=snapd --
become --ask-become-pass
SUDO password:
192.168.56.14 | SUCCESS => {
    "cache_update_time": 1726223581,
    "cache_updated": false,
    "changed": false
}
192.168.56.15 | SUCCESS => {
    "cache_update_time": 1726223580,
    "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. **In this command, it indicates that it would install the latest version of snapd, however, since I just installed it, there were no changes in the remote servers.**



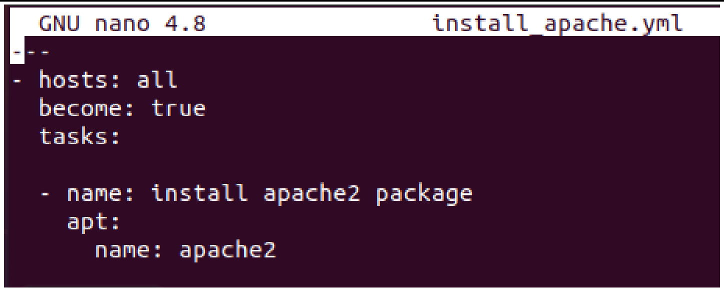
```
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible all -m apt -a "name=snapd s
tate=latest" --become --ask-become-pass
SUDO password:
192.168.56.14 | SUCCESS => {
    "cache_update_time": 1726223581,
    "cache_updated": false,
    "changed": false
}
192.168.56.15 | SUCCESS => {
    "cache_update_time": 1726223580,
    "cache_updated": false,
    "changed": false
}
```

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

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.

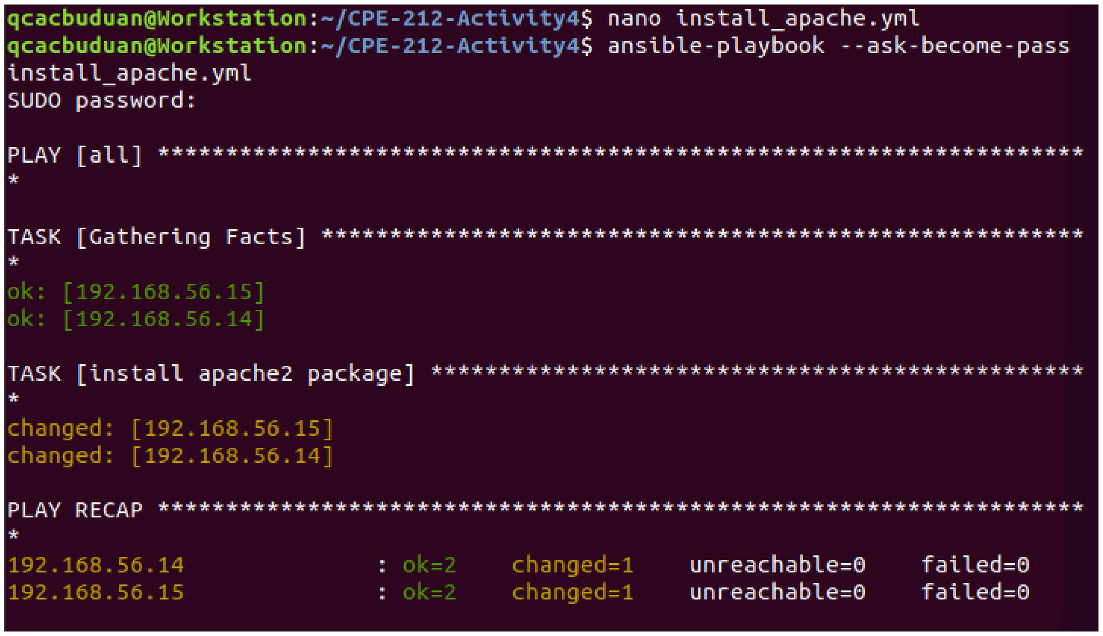
   When the editor appears, type the following:



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

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

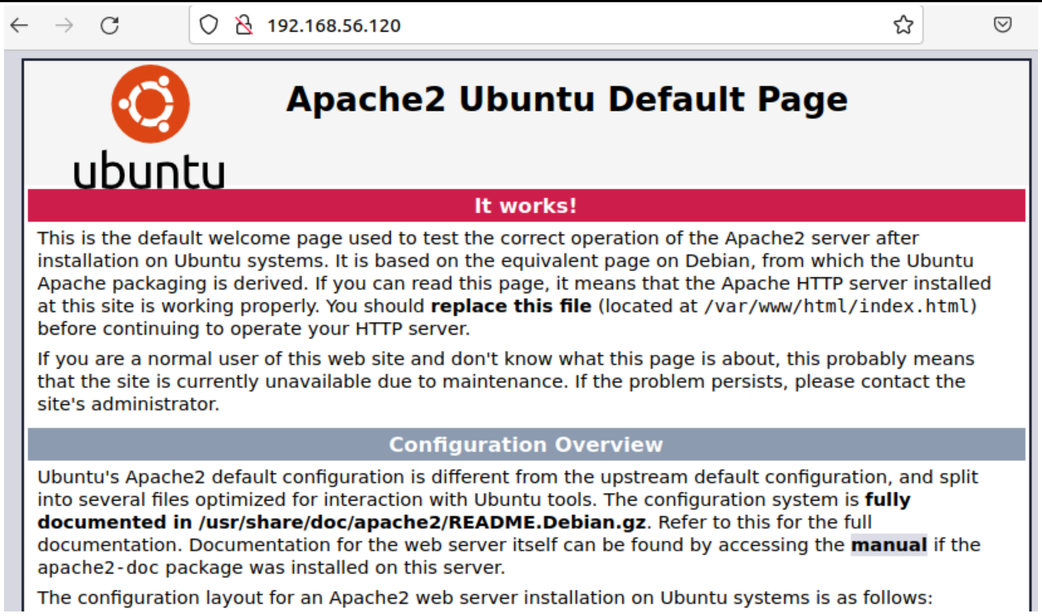
Make sure to save the file. Take note also of the alignments of the texts.

2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command. The result first shows that it is gathering facts, which means it is checking the information of the servers, and then it executes the install apache package and at the last part shows the remote server's status.



```
qcacbuduan@Workstation:~/CPE-212-Activity4$ nano install_apache.yml
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible-playbook --ask-become-pass
install_apache.yml
SUDO password:

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

TASK [Gathering Facts] ********************************************************
*
ok: [192.168.56.15]
ok: [192.168.56.14]

TASK [install apache2 package] ************************************************
*
changed: [192.168.56.15]
changed: [192.168.56.14]

PLAY RECAP ********************************************************************
*
192.168.56.14              : ok=2    changed=1    unreachable=0    failed=0
192.168.56.15              : ok=2    changed=1    unreachable=0    failed=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.


192.168.56.120

Apache2 Ubuntu Default Page

It works!

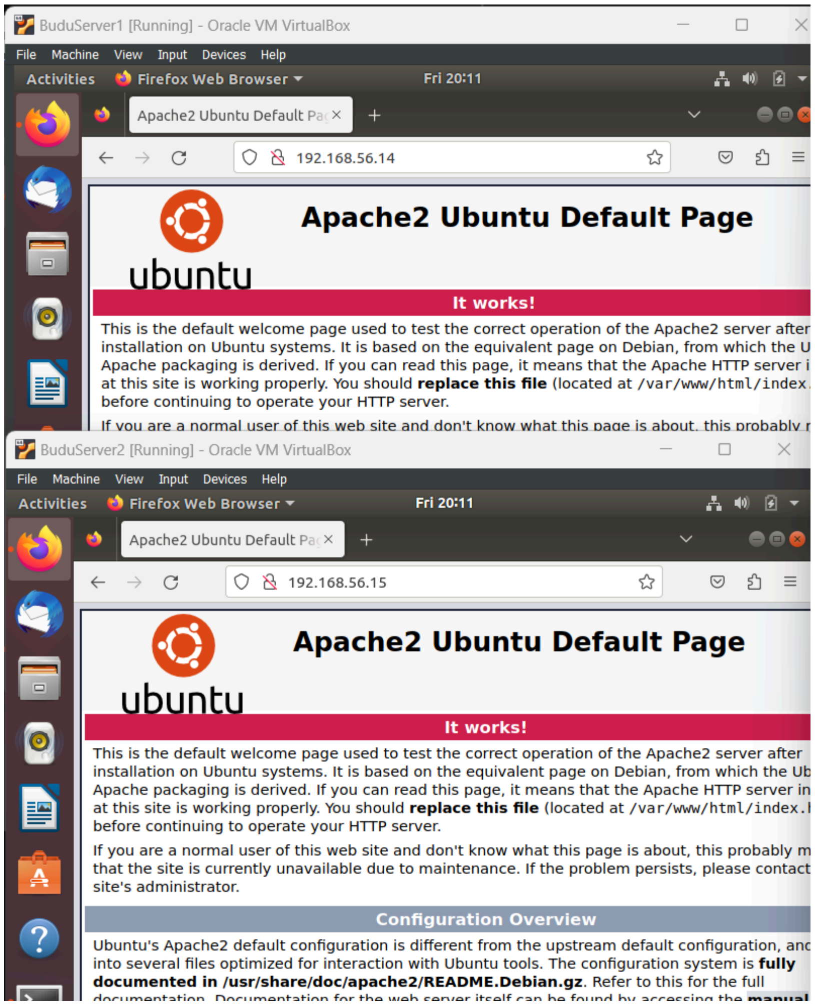
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at 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:


BuduServer1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Firefox Web Browser Fri 20:11

Apache2 Ubuntu Default Pa

192.168.56.14

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

BuduServer2 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Firefox Web Browser Fri 20:11

Apache2 Ubuntu Default Pa

192.168.56.15

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.l` before continuing to operate your HTTP server.

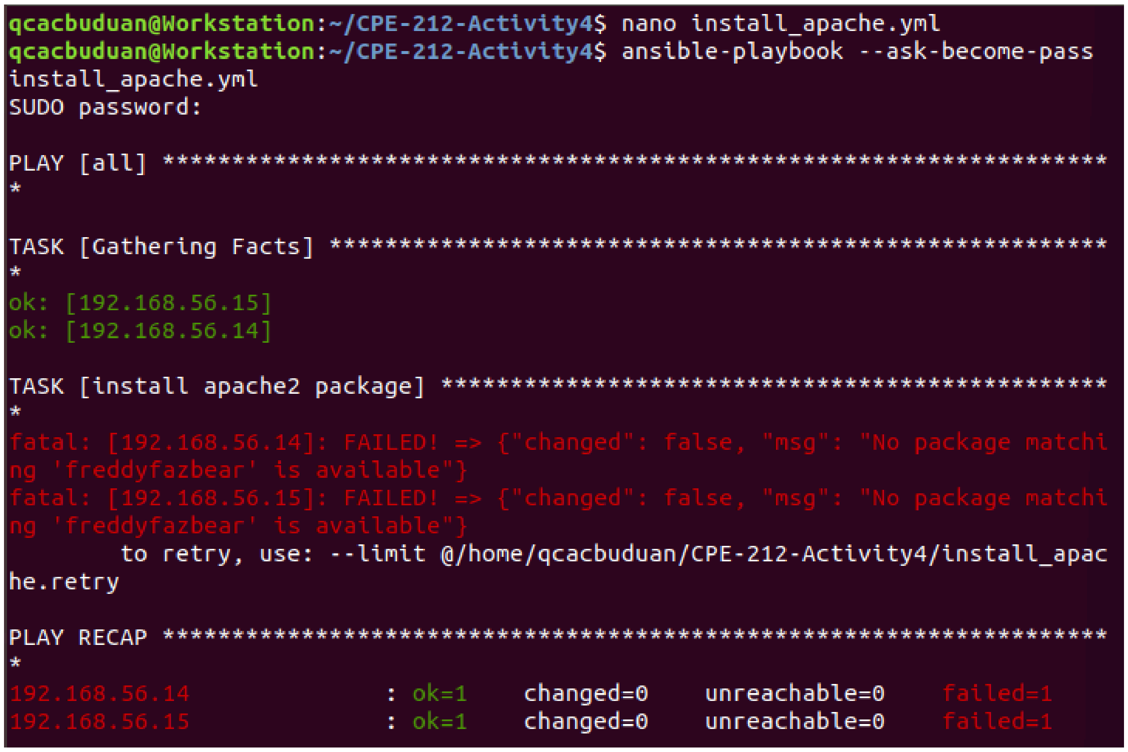
If you are a normal user of this web site and don't know what this page is about, this probably m that the site is currently unavailable due to maintenance. If the problem persists, please contact site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, anc 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

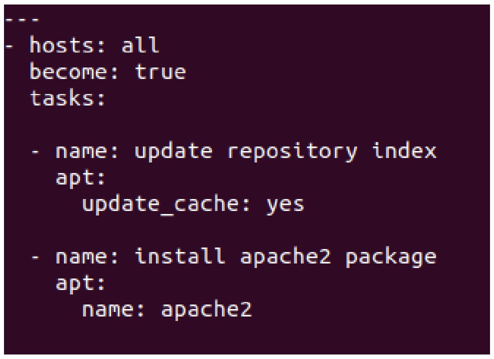
server 1(192.168.56.14) and server 2(192.168.56.15)

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?



```
qcacbuduan@Workstation:~/CPE-212-Activity4$ nano install_apache.yml
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible-playbook --ask-become-pass
install_apache.yml
SUDO password:

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

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.15]
ok: [192.168.56.14]

TASK [install apache2 package] *************************************************
*
fatal: [192.168.56.14]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'freddyfazbear' is available"}
fatal: [192.168.56.15]: FAILED! => {"changed": false, "msg": "No package matchi
ng 'freddyfazbear' is available"}
        to retry, use: --limit @/home/qcacbuduan/CPE-212-Activity4/install_apac
he.retry

PLAY RECAP *********************************************************************
*
192.168.56.14              : ok=1    changed=0    unreachable=0    failed=1
192.168.56.15              : ok=1    changed=0    unreachable=0    failed=1
```

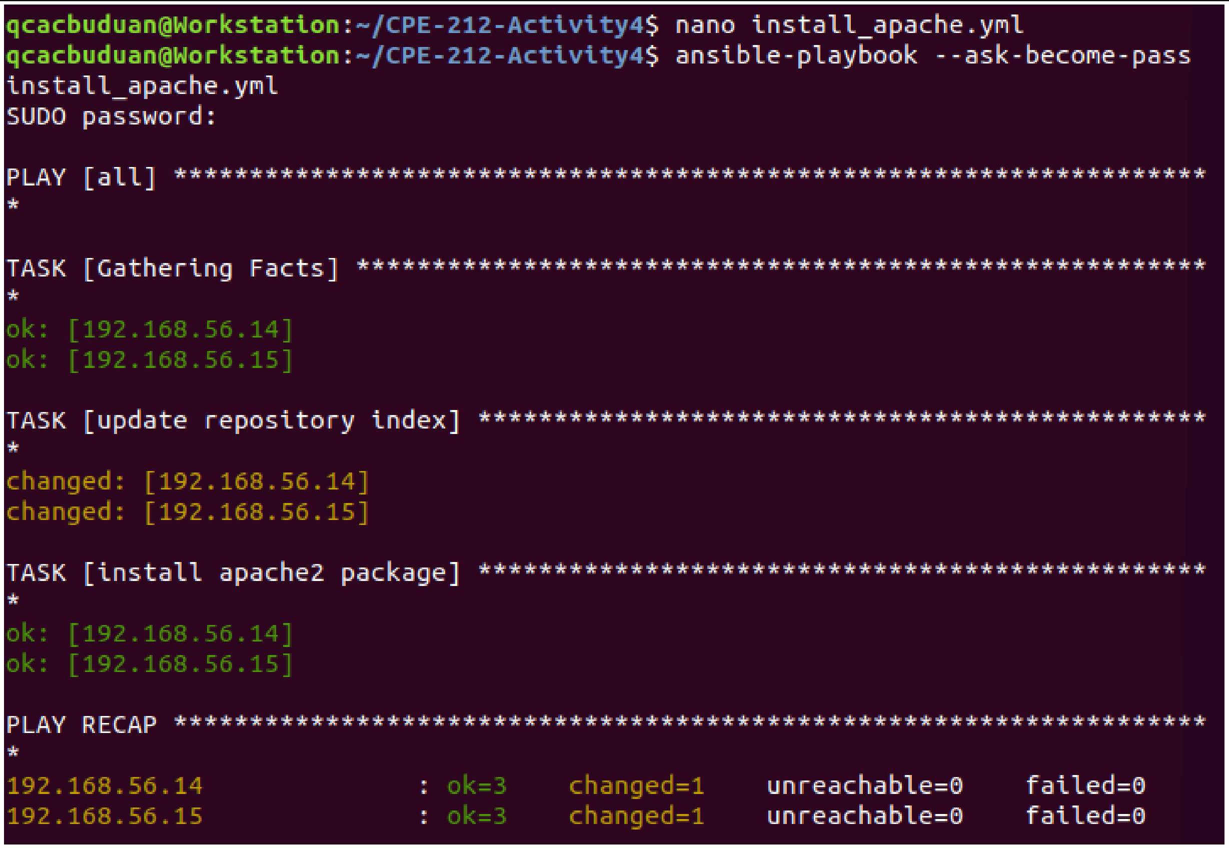
5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.



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

  - name: update repository index
    apt:
      update_cache: yes

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

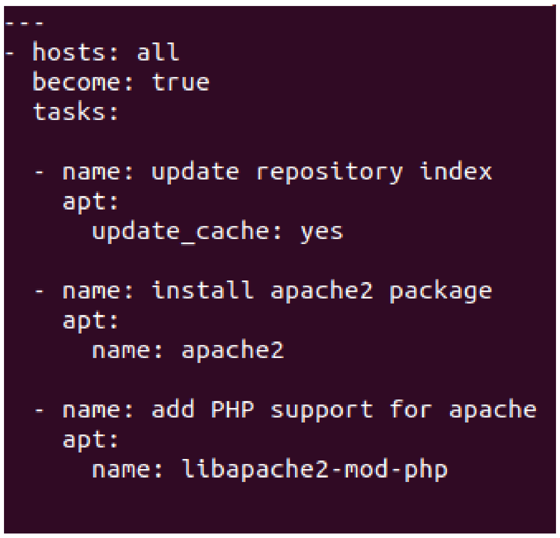
Save the changes to this file and exit.

6. Run the playbook and describe the output. Did the new command change anything on the remote servers? **Yes because of the update repository index.**



```
qcacbuduan@Workstation:~/CPE-212-Activity4$ nano install_apache.yml
qcacbuduan@Workstation:~/CPE-212-Activity4$ ansible-playbook --ask-become-pass
install_apache.yml
SUDO password:

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

TASK [Gathering Facts] ********************************************************
*
ok: [192.168.56.14]
ok: [192.168.56.15]

TASK [update repository index] ************************************************
*
changed: [192.168.56.14]
changed: [192.168.56.15]

TASK [install apache2 package] ************************************************
*
ok: [192.168.56.14]
ok: [192.168.56.15]

PLAY RECAP ********************************************************************
*
192.168.56.14              : ok=3    changed=1    unreachable=0    failed=0
192.168.56.15              : ok=3    changed=1    unreachable=0    failed=0
```

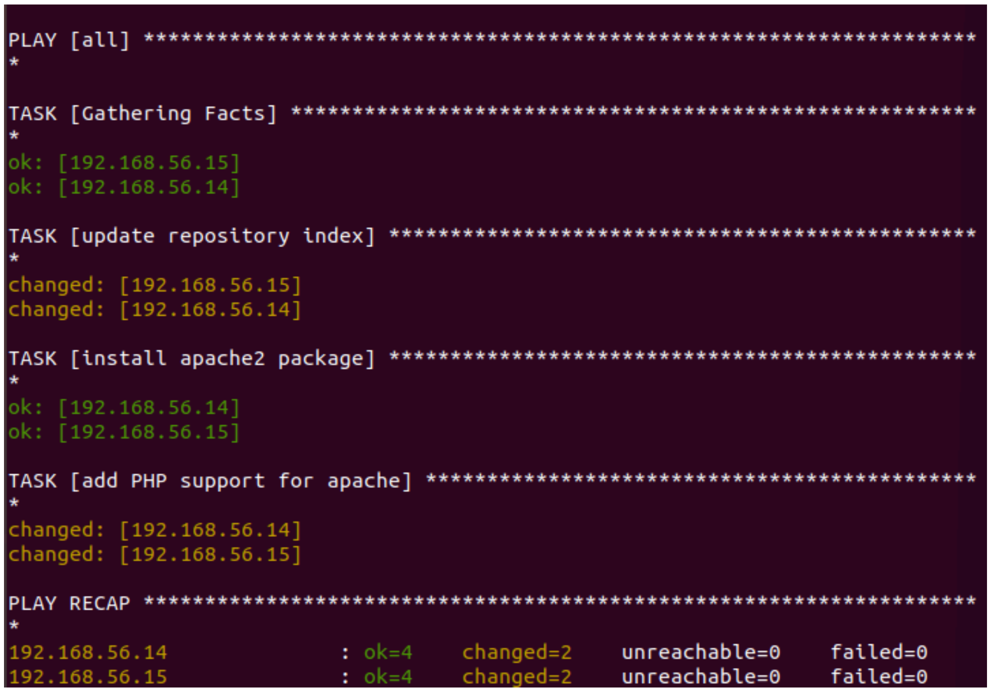
7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.



```
---
- 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? **Yes, we installed the apache mod**



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

TASK [Gathering Facts] *****************************************************
*
ok: [192.168.56.15]
ok: [192.168.56.14]

TASK [update repository index] *********************************************
*
changed: [192.168.56.15]
changed: [192.168.56.14]

TASK [install apache2 package] *********************************************
*
ok: [192.168.56.14]
ok: [192.168.56.15]

TASK [add PHP support for apache] ******************************************
*
changed: [192.168.56.14]
changed: [192.168.56.15]

PLAY RECAP *****************************************************************
*
192.168.56.14              : ok=4    changed=2    unreachable=0    failed=0
192.168.56.15              : ok=4    changed=2    unreachable=0    failed=0
```

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

   Github Repository link: *https://github.com/buduman/CPE-212-Activity4.git*

Reflections:

Answer the following:

1. What is the importance of using a playbook?
   - In ansible, using playbook is important because when we write a playbook, we can define the state that we want our servers to be in and the command that ansible will carry out to bring our servers to that state.
2. Summarize what we have done on this activity.
   - in this activity, I was able to learn the basics of ansible, how to create an inventory and made default configurations before being introduced to running ad-hoc commands, which made installing programs to remote servers easier, I was also able to learn about playbook, in which you can write scripts and automate commands which can help you automate and manage servers effectively.