Name: John Glen L. Paz	**Date Performed: 9/11/2023**
Course/Section: CPE31S6	**Date Submitted: 9/14/2023**
Instructor: Dr. Jonathan V. Taylar	**Semester and SY: 1st Sem./2023**

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

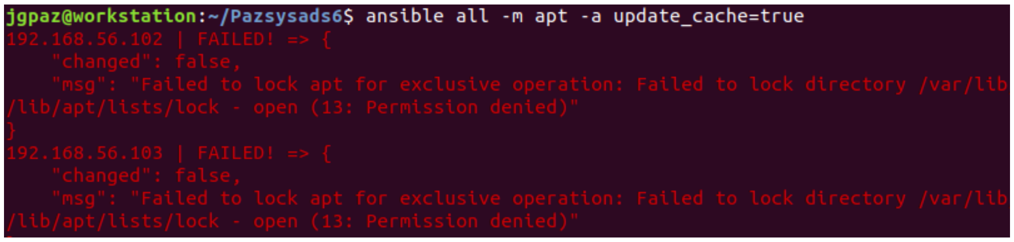
Task 1: Run elevated ad hoc commands

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

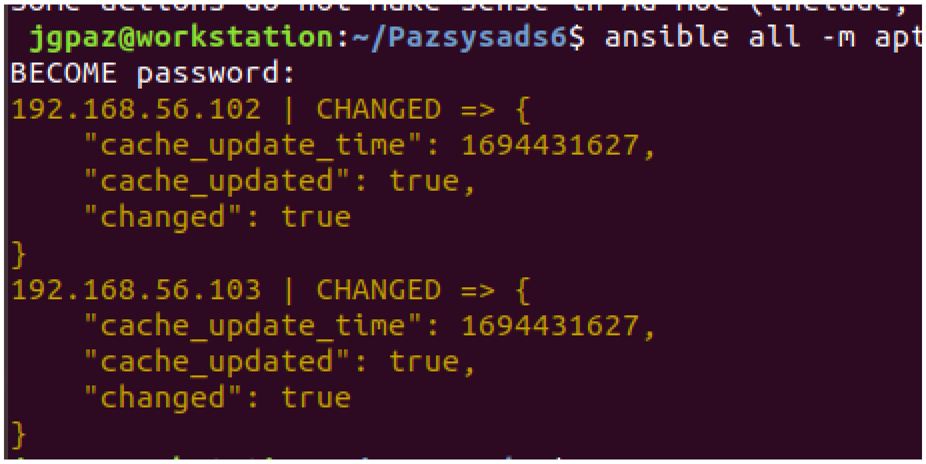
What is the result of the command? Is it successful?



```
jgpaz@workstation:~/Pazsysads6$ ansible all -m apt -a update_cache=true
192.168.56.102 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib
/lib/apt/lists/lock - open (13: Permission denied)"
}
192.168.56.103 | FAILED! => {
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib
/lib/apt/lists/lock - open (13: Permission denied)"
```

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.

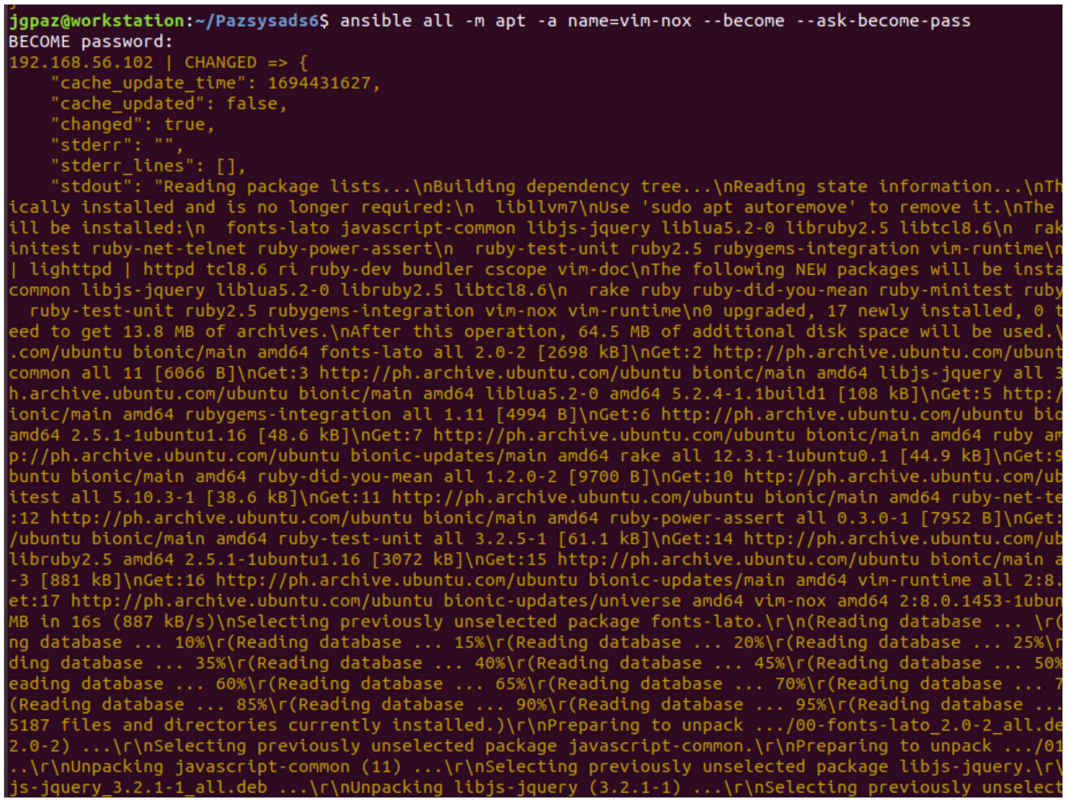
You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.



```
 jgpaz@workstation:~/Pazsysads6$ ansible all -m apt
BECOME password:
192.168.56.102 | CHANGED => {
    "cache_update_time": 1694431627,
    "cache_updated": true,
    "changed": true
}
192.168.56.103 | CHANGED => {
    "cache_update_time": 1694431627,
    "cache_updated": true,
    "changed": true
}
```

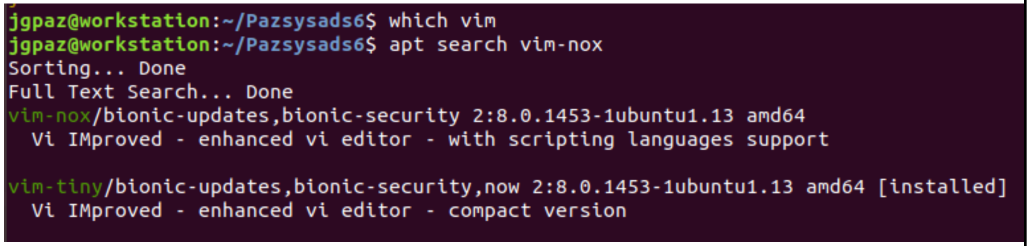
2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction. Here is the command: *ansible all -m apt -a name=vim-nox --become --ask-become-pass.* The command would take some time after typing the password because the local machine instructed the remote servers to

actually install the package.



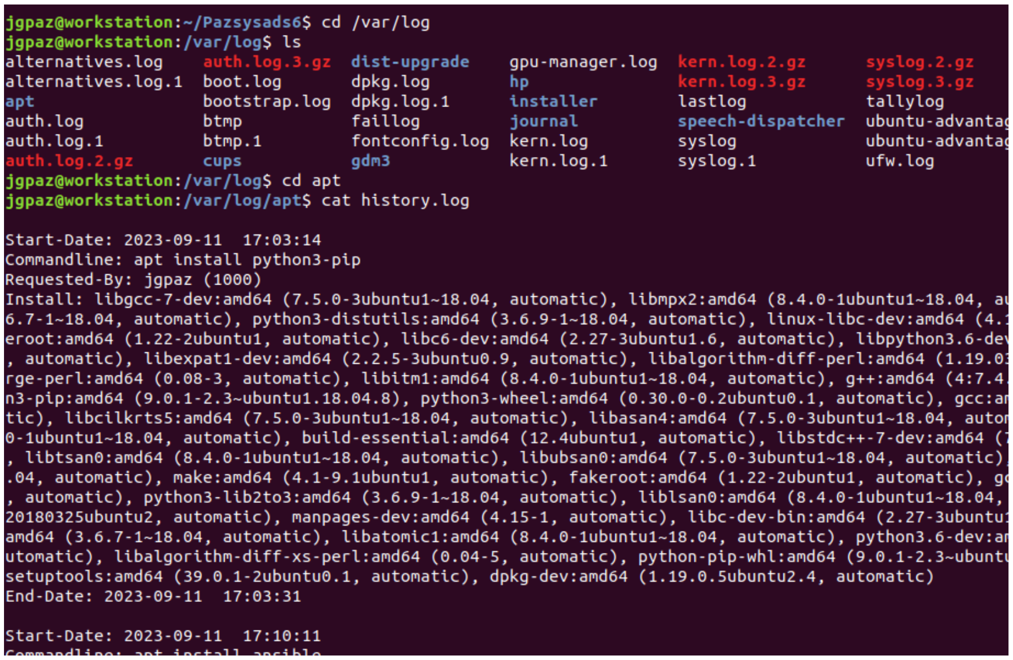
```
jgpaz@workstation:~/Pazsysads6$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
    "cache_update_time": 1694431627,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nTh
ically installed and is no longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe 
ill be installed:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rak
initest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygems-integration vim-runtime\n
| lighttpd | httpd tcl8.6 ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be insta
common libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ruby-minitest ruby
  ruby-test-unit ruby2.5 rubygems-integration vim-nox vim-runtime\n0 upgraded, 17 newly installed, 0 t
eed to get 13.8 MB of archives.\nAfter this operation, 64.5 MB of additional disk space will be used.\
.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2 [2698 kB]\nGet:2 http://ph.archive.ubuntu.com/ubunt
common all 11 [6066 B]\nGet:3 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all 3
h.archive.ubuntu.com/ubuntu bionic/main amd64 liblua5.2-0 amd64 5.2.4-1.1build1 [108 kB]\nGet:5 http:/
ionic/main amd64 rubygems-integration all 1.11 [4994 B]\nGet:6 http://ph.archive.ubuntu.com/ubuntu bio
amd64 2.5.1-1ubuntu1.16 [48.6 kB]\nGet:7 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 ruby am
p://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all 12.3.1-1ubuntu0.1 [44.9 kB]\nGet:9
buntu bionic/main amd64 ruby-did-you-mean all 1.2.0-2 [9700 B]\nGet:10 http://ph.archive.ubuntu.com/ub
itest all 5.10.3-1 [38.6 kB]\nGet:11 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-te
:12 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all 0.3.0-1 [7952 B]\nGet:
/ubuntu bionic/main amd64 ruby-test-unit all 3.2.5-1 [61.1 kB]\nGet:14 http://ph.archive.ubuntu.com/ub
libruby2.5 amd64 2.5.1-1ubuntu1.16 [3072 kB]\nGet:15 http://ph.archive.ubuntu.com/ubuntu bionic/main a
-3 [881 kB]\nGet:16 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 vim-runtime all 2:8.
et:17 http://ph.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 vim-nox amd64 2:8.0.1453-1ubun
MB in 16s (887 kB/s)\nSelecting previously unselected package fonts-lato.\r\n(Reading database ... \r(
ng database ... 10%\r(Reading database ... 15%\r(Reading database ... 20%\r(Reading database ... 25%\r
ding database ... 35%\r(Reading database ... 40%\r(Reading database ... 45%\r(Reading database ... 50%
eading database ... 60%\r(Reading database ... 65%\r(Reading database ... 70%\r(Reading database ... 7
(Reading database ... 85%\r(Reading database ... 90%\r(Reading database ... 95%\r(Reading database ...
5187 files and directories currently installed.)\r\nPreparing to unpack .../00-fonts-lato_2.0-2_all.de
2.0-2) ...\r\nSelecting previously unselected package javascript-common.\r\nPreparing to unpack .../01
..\r\nUnpacking javascript-common (11) ...\r\nSelecting previously unselected package libjs-jquery.\r\
js-jquery_3.2.1-1_all.deb ...\r\nUnpacking libjs-jquery (3.2.1-1) ...\r\nSelecting previously unselect
```

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?



```
jgpaz@workstation:~/Pazsysads6$ which vim
jgpaz@workstation:~/Pazsysads6$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security 2:8.0.1453-1ubuntu1.13 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

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

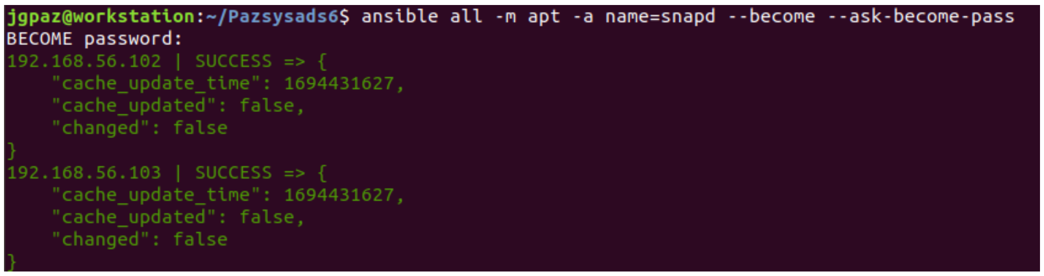
2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.



```
jgpaz@workstation:~/Pazsysads6$ cd /var/log
jgpaz@workstation:/var/log$ ls
alternatives.log    auth.log.3.gz  dist-upgrade    gpu-manager.log  kern.log.2.gz      syslog.2.gz
alternatives.log.1  boot.log       dpkg.log        hp               kern.log.3.gz      syslog.3.gz
apt                 bootstrap.log  dpkg.log.1      installer        lastlog            tallylog
auth.log            btmp           faillog         journal          speech-dispatcher  ubuntu-advantag
auth.log.1          btmp.1         fontconfig.log  kern.log         syslog             ubuntu-advantag
auth.log.2.gz       cups           gdm3            kern.log.1       syslog.1           ufw.log
jgpaz@workstation:/var/log$ cd apt
jgpaz@workstation:/var/log/apt$ cat history.log

Start-Date: 2023-09-11  17:03:14
Commandline: apt install python3-pip
Requested-By: jgpaz (1000)
Install: libgcc-7-dev:amd64 (7.5.0-3ubuntu1~18.04, automatic), libmpx2:amd64 (8.4.0-1ubuntu1~18.04, au
6.7-1~18.04, automatic), python3-distutils:amd64 (3.6.9-1~18.04, automatic), linux-libc-dev:amd64 (4.1
eroot:amd64 (1.22-2ubuntu1, automatic), libc6-dev:amd64 (2.27-3ubuntu1.6, automatic), libpython3.6-dev
, automatic), libexpat1-dev:amd64 (2.2.5-3ubuntu0.9, automatic), libalgorithm-diff-perl:amd64 (1.19.03
rge-perl:amd64 (0.08-3, automatic), libitm1:amd64 (8.4.0-1ubuntu1~18.04, automatic), g++:amd64 (4:7.4.
n3-pip:amd64 (9.0.1-2.3~ubuntu1.18.04.8), python3-wheel:amd64 (0.30.0-0.2ubuntu0.1, automatic), gcc:am
tic), libcilkrts5:amd64 (7.5.0-3ubuntu1~18.04, automatic), libasan4:amd64 (7.5.0-3ubuntu1~18.04, autom
0-1ubuntu1~18.04, automatic), build-essential:amd64 (12.4ubuntu1, automatic), libstdc++-7-dev:amd64 (7
, libtsan0:amd64 (8.4.0-1ubuntu1~18.04, automatic), libubsan0:amd64 (7.5.0-3ubuntu1~18.04, automatic),
.04, automatic), make:amd64 (4.1-9.1ubuntu1, automatic), fakeroot:amd64 (1.22-2ubuntu1, automatic), gc
, automatic), python3-lib2to3:amd64 (3.6.9-1~18.04, automatic), liblsan0:amd64 (8.4.0-1ubuntu1~18.04,
20180325ubuntu2, automatic), manpages-dev:amd64 (4.15-1, automatic), libc-dev-bin:amd64 (2.27-3ubuntu1
amd64 (3.6.7-1~18.04, automatic), libatomic1:amd64 (8.4.0-1ubuntu1~18.04, automatic), python3.6-dev:am
utomatic), libalgorithm-diff-xs-perl:amd64 (0.04-5, automatic), python-pip-whl:amd64 (9.0.1-2.3~ubuntu
setuptools:amd64 (39.0.1-2ubuntu0.1, automatic), dpkg-dev:amd64 (1.19.0.5ubuntu2.4, automatic)
End-Date: 2023-09-11  17:03:31

Start-Date: 2023-09-11  17:10:11
Commandline: apt install ansible
```

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.

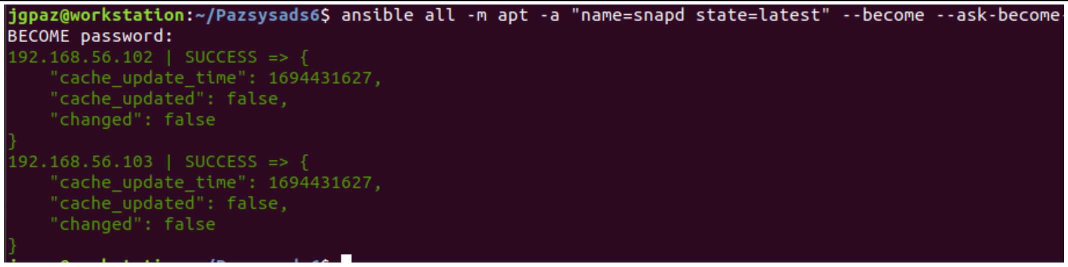
   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*



```
jgpaz@workstation:~/Pazsysads6$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694431627,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694431627,
    "cache_updated": false,
    "changed": false
}
```

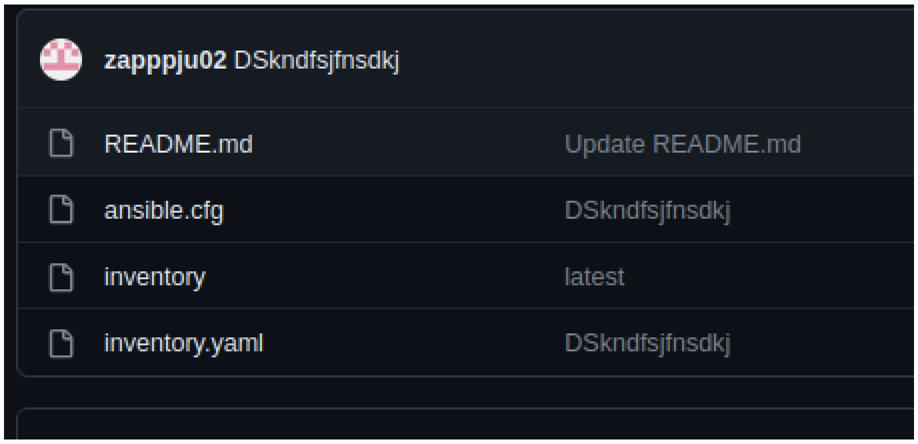
   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers?

   3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*
   Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.



```
jgpaz@workstation:~/Pazsysads6$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694431627,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694431627,
    "cache_updated": false,
    "changed": false
}
```

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



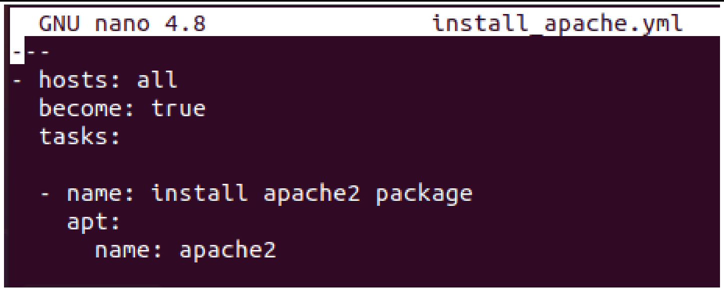
zapppju02 DSkndfsjfnsdkj

README.md	Update README.md
ansible.cfg	DSkndfsjfnsdkj
inventory	latest
inventory.yaml	DSkndfsjfnsdkj

Task 2: Writing our First Playbook

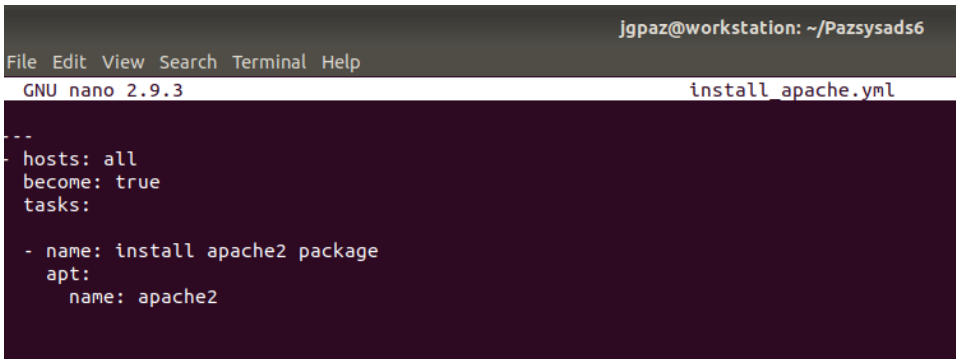
1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and 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
```

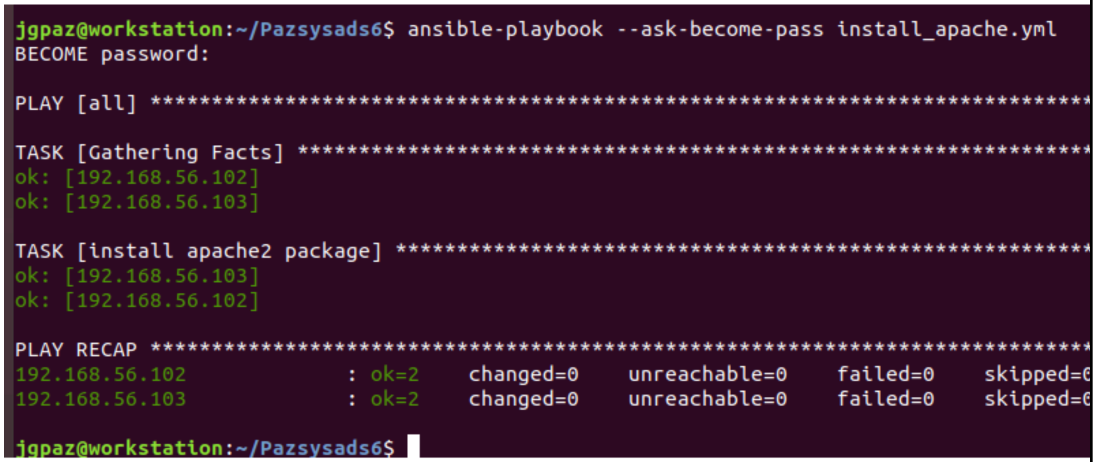
Make sure to save the file. Take note also of the alignments of the texts.



jgpaz@workstation: ~/Pazsysads6
File Edit View Search Terminal Help

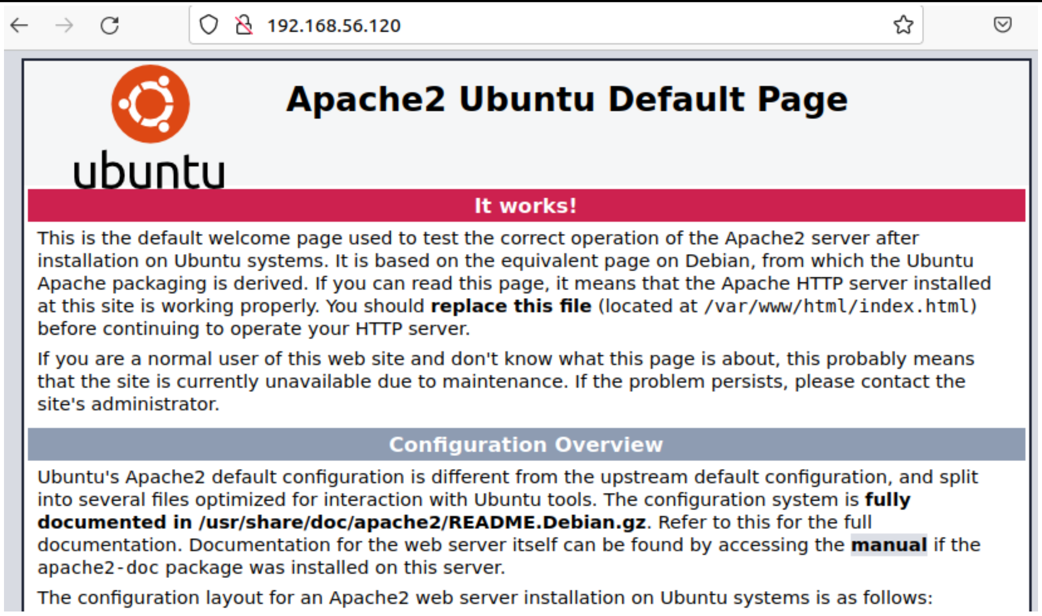
```
  GNU nano 2.9.3                                        install_apache.yml

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

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

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



```
jgpaz@workstation:~/Pazsysads6$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [install apache2 package] ******************************************************
ok: [192.168.56.103]
ok: [192.168.56.102]

PLAY RECAP **************************************************************************
192.168.56.102             : ok=2    changed=0    unreachable=0    failed=0    skipped=
192.168.56.103             : ok=2    changed=0    unreachable=0    failed=0    skipped=

jgpaz@workstation:~/Pazsysads6$
```

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!

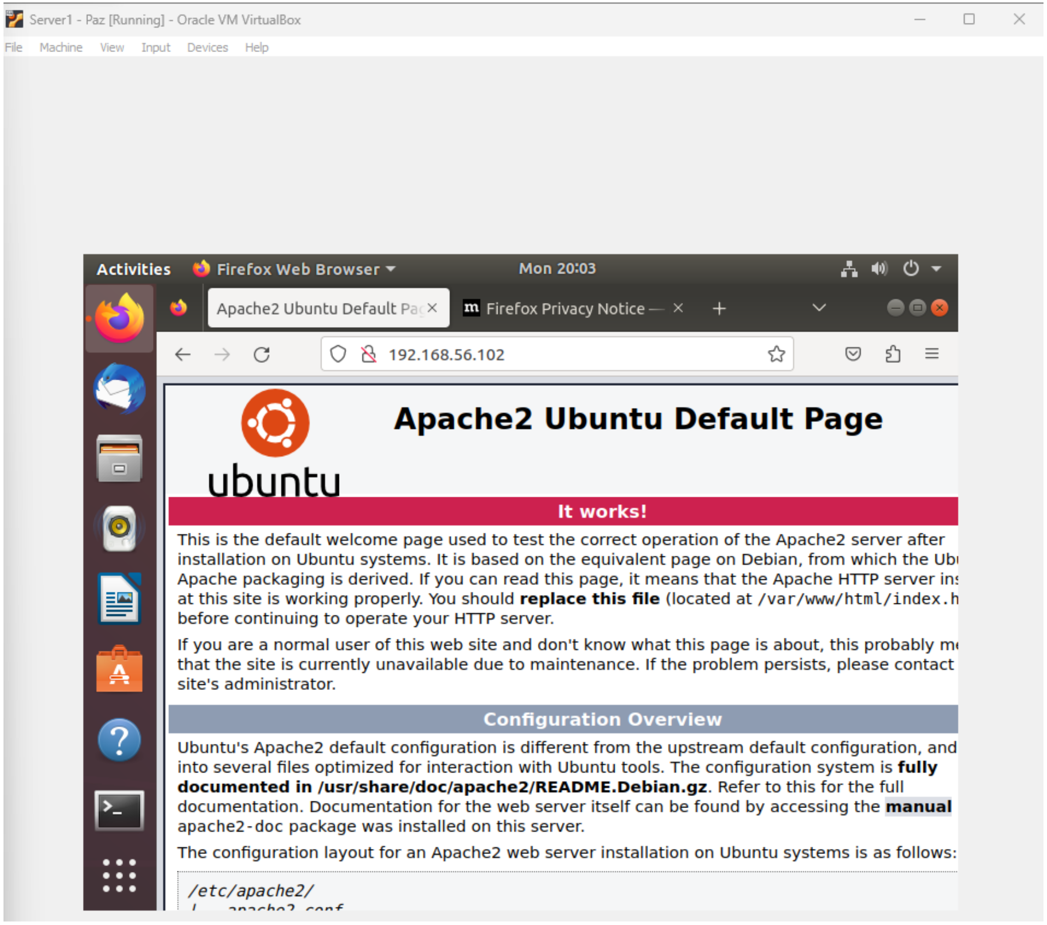
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at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


Server1 - Paz [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Mon 20:03
Apache2 Ubuntu Default Pa
Firefox Privacy Notice —

192.168.56.102

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

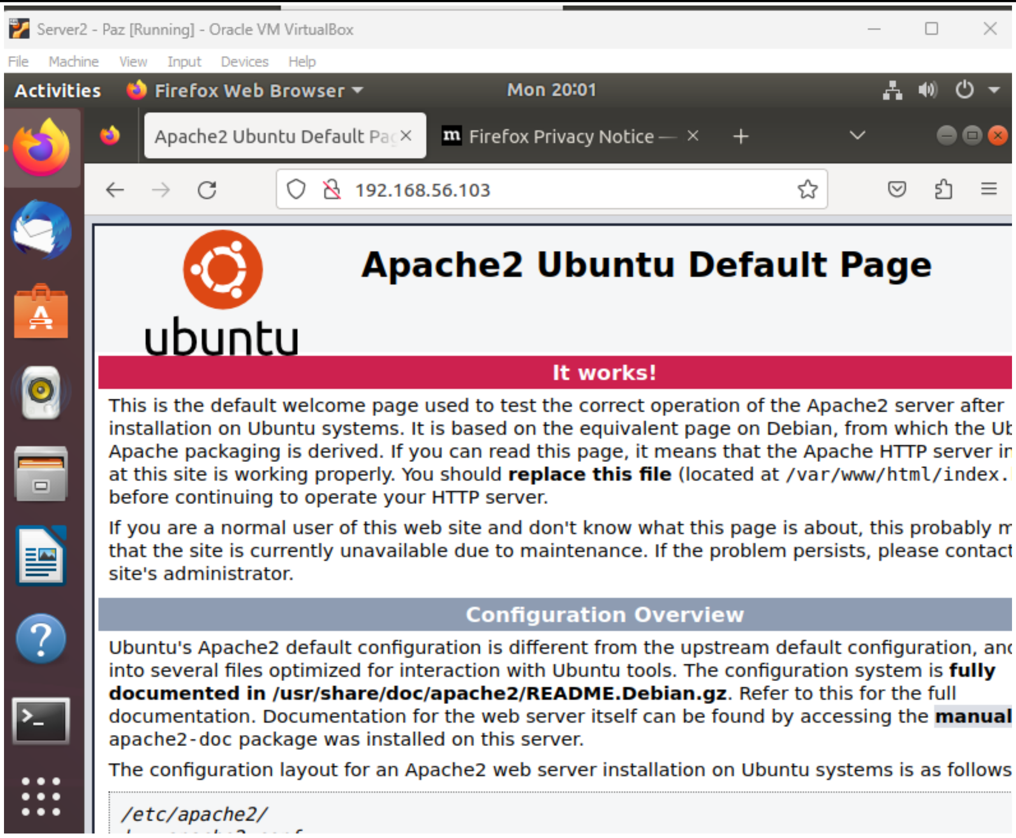
If you are a normal user of this web site and don't know what this page is about, this probably m that the site is currently unavailable due to maintenance. If the problem persists, please contact site's administrator.

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/
```


Server2 - Paz [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Mon 20:01
Apache2 Ubuntu Default Pa
Firefox Privacy Notice —
192.168.56.103
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 ir 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 m that the site is currently unavailable due to maintenance. If the problem persists, please contact site's administrator.

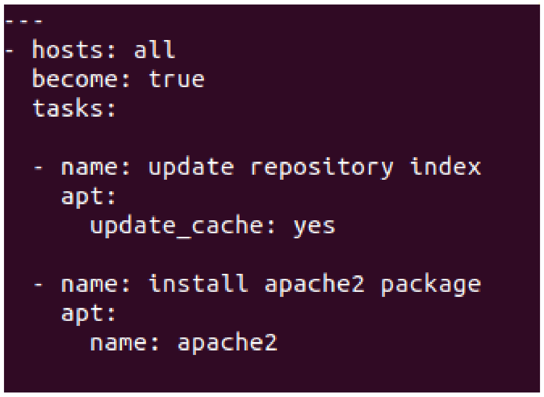
Configuration Overview

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

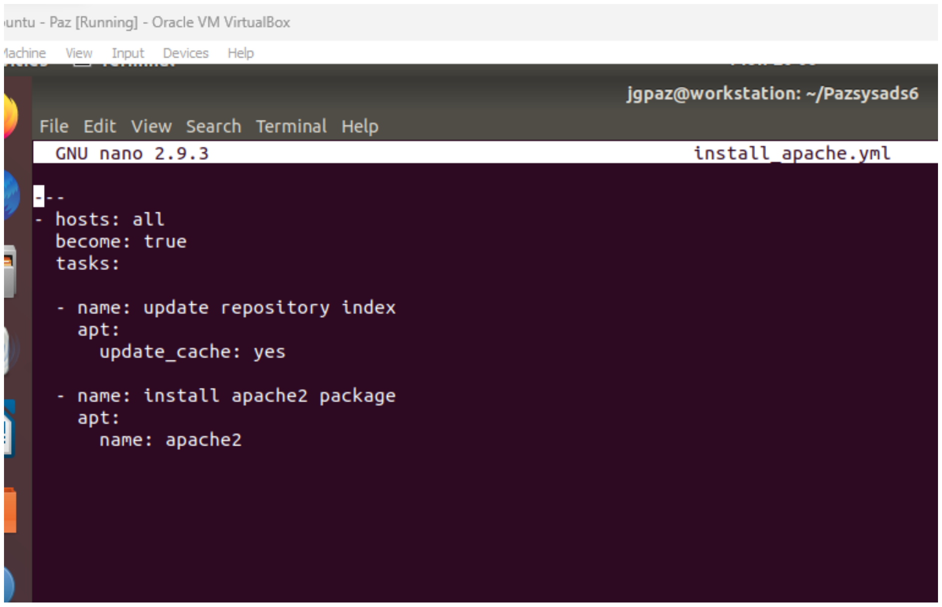
```
/etc/apache2/
```

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?
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.



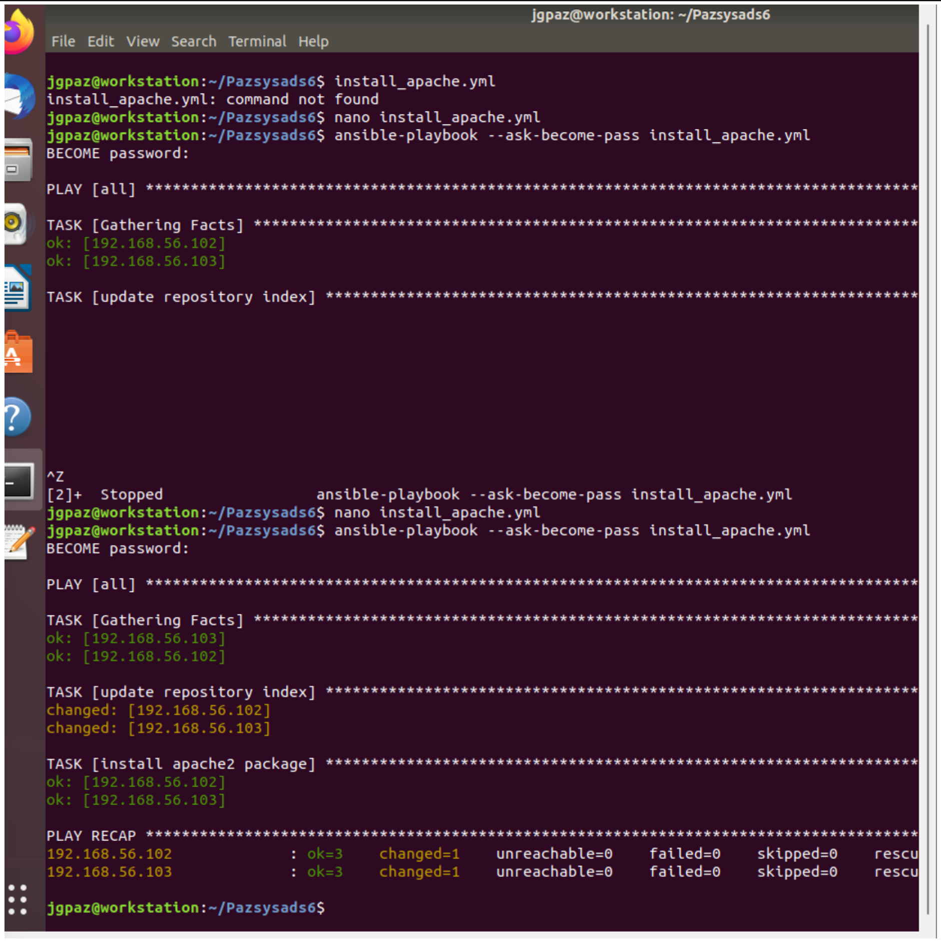
```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

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

Save the changes to this file and exit.


untu - Paz [Running] - Oracle VM VirtualBox
Machine View Input Devices Help
jgpaz@workstation: ~/Pazsysads6
File Edit View Search Terminal Help

```
  GNU nano 2.9.3                                               install_apache.yml

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

  - name: update repository index
    apt:
      update_cache: yes

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

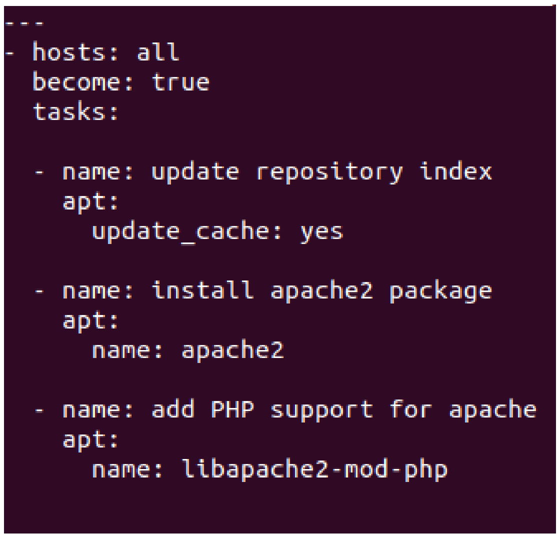
6. Run the playbook and describe the output. Did the new command change anything on the remote servers?



```
jgpaz@workstation:~/Pazsysads6$ install_apache.yml
install_apache.yml: command not found
jgpaz@workstation:~/Pazsysads6$ nano install_apache.yml
jgpaz@workstation:~/Pazsysads6$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [update repository index] ********************************************************







^Z
[2]+  Stopped                 ansible-playbook --ask-become-pass install_apache.yml
jgpaz@workstation:~/Pazsysads6$ nano install_apache.yml
jgpaz@workstation:~/Pazsysads6$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [update repository index] ********************************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] ********************************************************
ok: [192.168.56.102]
ok: [192.168.56.103]

PLAY RECAP ****************************************************************************
192.168.56.102             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescu
192.168.56.103             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescu

jgpaz@workstation:~/Pazsysads6$
```

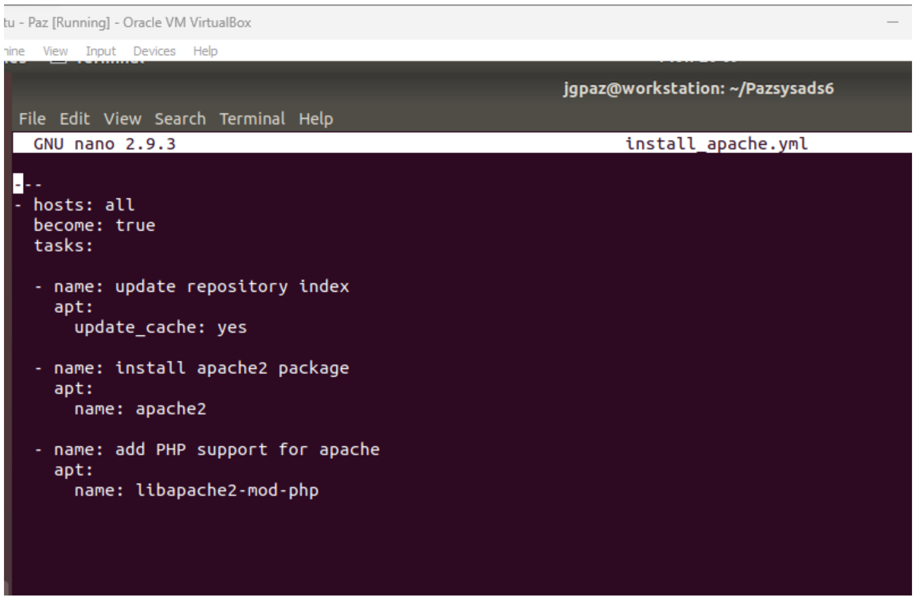
Yes, it changed.

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



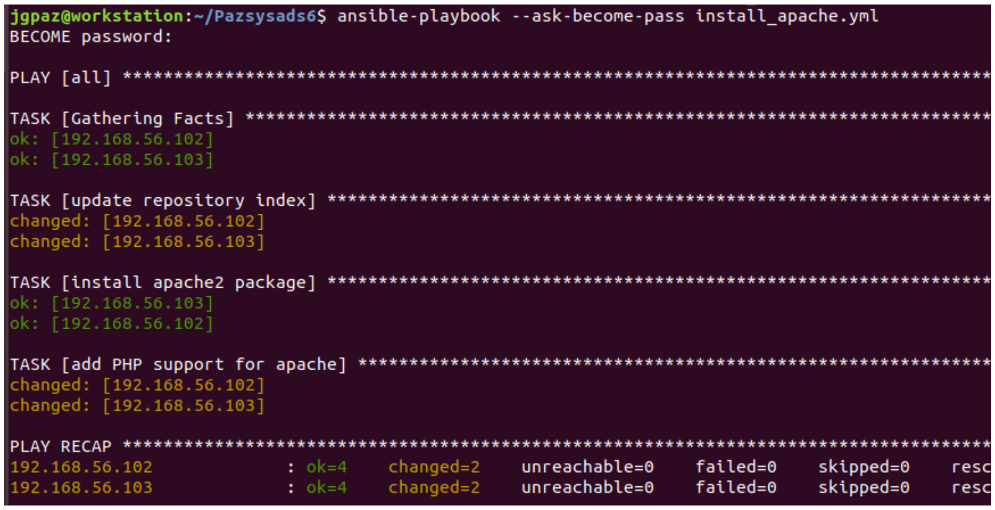
```
---
- hosts: all
  become: true
  tasks:

  - 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.


tu - Paz [Running] - Oracle VM VirtualBox
nine View Input Devices Help
jgpaz@workstation: ~/Pazsysads6
File Edit View Search Terminal Help

```
  GNU nano 2.9.3                                           install_apache.yml

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

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

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



```
jgpaz@workstation:~/Pazsysads6$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

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

TASK [update repository index] ***************************************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

TASK [install apache2 package] ***************************************************************
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [add PHP support for apache] ************************************************************
changed: [192.168.56.102]
changed: [192.168.56.103]

PLAY RECAP ***********************************************************************************
192.168.56.102             : ok=4    changed=2    unreachable=0    failed=0    skipped=0    resc
192.168.56.103             : ok=4    changed=2    unreachable=0    failed=0    skipped=0    resc
```

9. Finally, make sure that we are in sync with GitHub. Provide the link of your GitHub repository.
   - https://github.com/zapppju02/Pazsysads6.git

Reflections:
Answer the following:

1. What is the importance of using a playbook?
   - it allows for the automation of tasks on remote tasks on remote hosts, which saves time and reduces the manual overhead of creating everything from scratch.

2. Summarize what we have done on this activity.
   - We ping each server by using ansible commands and these commands were executed as a superuser (root) or another privileged user on target hosts. We know that ansible is an agentless system that uses remote SSH to execute actions. This elevated access is often necessary for performing system-level tasks, configuration changes, and software installations that require administrative permissions.

Conclusion:

- In this activity, I learned that the importance of using ansible can simplify our course, enhance its efficiency, and allow for the automation of a wide range of tasks, from server provisioning to software deployment and updates. It's a valuable tool for managing Ubuntu-based infrastructure effectively and ensuring that systems are secure and well-maintained. Playbooks also allow you to describe the desired state of your infrastructure and the sequence of tasks required to achieve that state. It is quite difficult for me at the moment but I managed to finish it within the day. Overall, learning Ansible and utilizing its full potential for managing big infrastructures require a shift from ad hoc commands to playbook-based automation. It encourages best practices in infrastructure management and automation while providing enhanced organization, repeatability, and scalability.