

<b>Name: Espiritu, Diego Angelo G.</b>	<b>Date Performed: 11/09/2023</b>
<b>Course/Section: CPE232/CPE31S6</b>	<b>Date Submitted: 14/09/2023</b>
<b>Instructor: Dr. Jonathan Vidal Taylar</b>	<b>Semester and SY: 1st sem 2023-2024</b>
<b>Activity 4: Running Elevated Ad hoc Commands</b>	
<b>1. Objectives:</b> 1.1 Use commands that makes changes to remote machines 1.2 Use playbook in automating ansible commands	
<b>2. Discussion:</b>  <i>Provide screenshots for each task.</i>  <b>Elevated Ad hoc commands</b> 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.  <b>Playbooks</b> record and execute <b>Ansible's</b> 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>  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*

```
diego@workstation:~/sysAds6$ ansible all -m apt -a update_cache=true
192.168.56.103 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (13: Permission denied)"
}
192.168.56.102 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (13: Permission denied)"
}
diego@workstation:~/sysAds6$
```

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

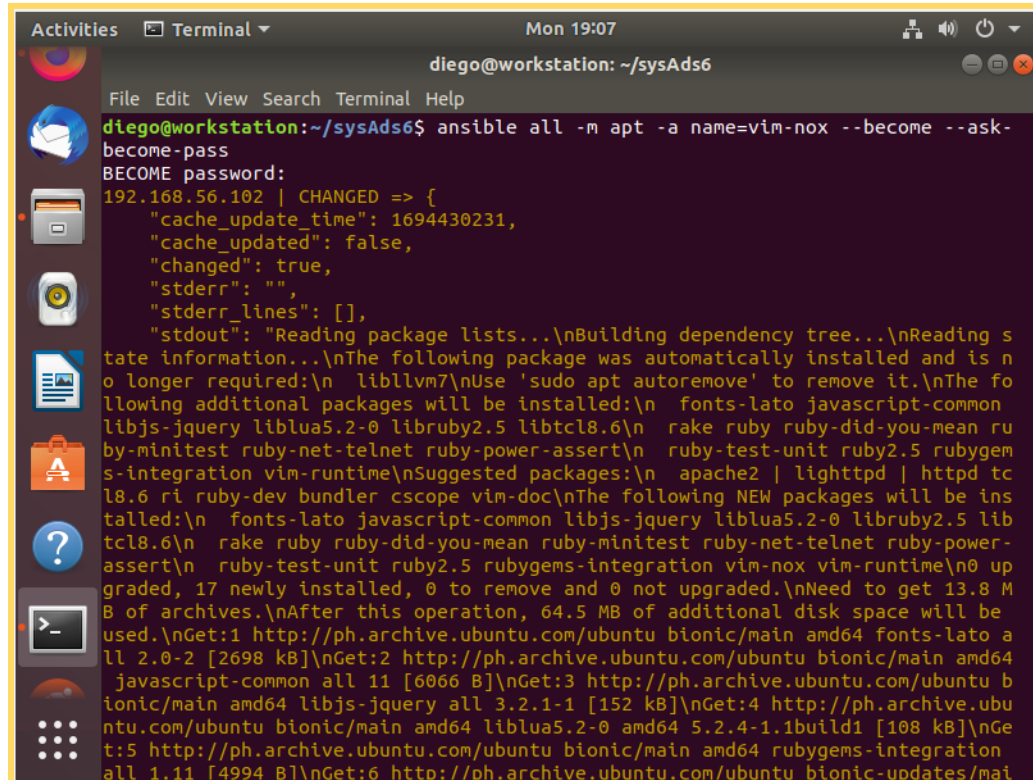
-The command failed because it could not open a lock file.

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.

```
diego@workstation:~/sysAds6$ ansible all -m apt -a update_cache=true --become -
ask-become-pass
BECOME password:
192.168.56.103 | CHANGED => {
  "cache_update_time": 1694430231,
  "cache_updated": true,
  "changed": true
}
192.168.56.102 | CHANGED => {
  "cache_update_time": 1694430231,
  "cache_updated": true,
  "changed": true
}
diego@workstation:~/sysAds6$
```

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.

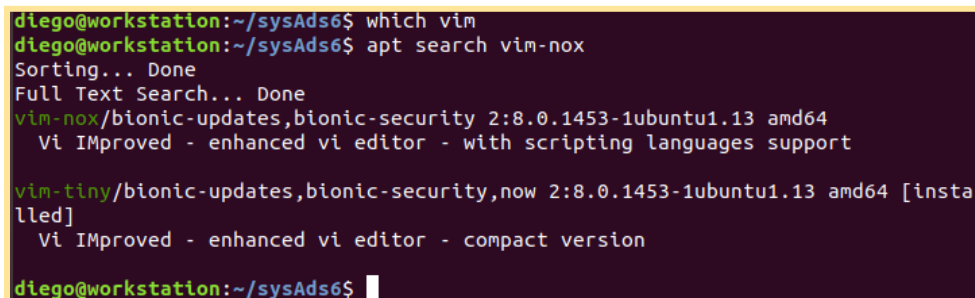


```
diego@workstation: ~/sysAds6
File Edit View Search Terminal Help
diego@workstation:~/sysAds6$ ansible all -m apt -a name=vim-nox --become --ask-become-pass
BECOME password:
192.168.56.102 | CHANGED => {
  "cache_update_time": 1694430231,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "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
ntu.com/ubuntu bionic/main amd64 liblua5.2-0 amd64 5.2.4-1.1build1 [108 kB]\nGe
t:5 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4994 B]\nGet:6 http://ph.archive.ubuntu.com/ubuntu bionic-updates/mai
```

- 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 the command is successful.



```
diego@workstation:~/sysAds6$ which vim
diego@workstation:~/sysAds6$ 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

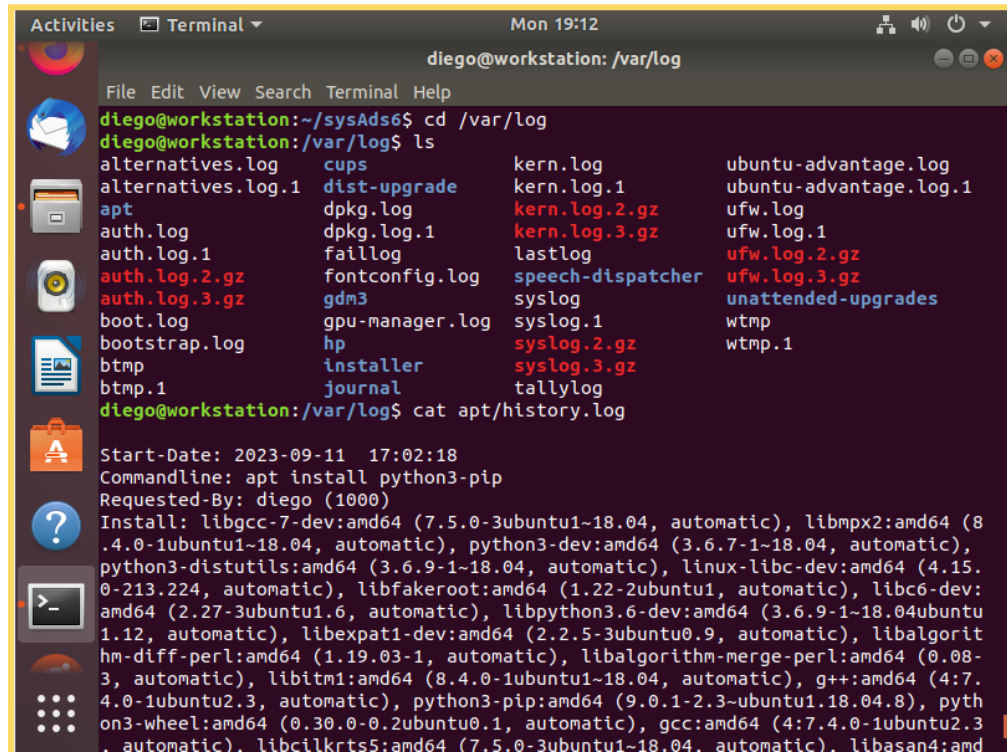
vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled]
Vi IMproved - enhanced vi editor - compact version

diego@workstation:~/sysAds6$
```

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`.

What I see in the `history.log` is that there is an install and about ubuntu and Python I'm not exactly sure.



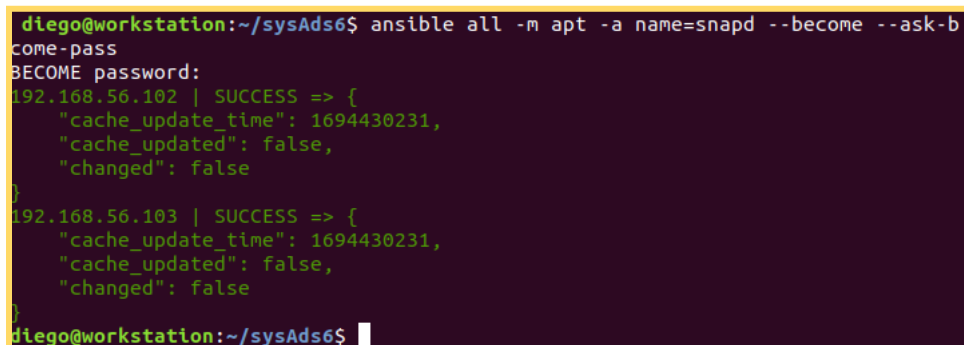
A terminal window titled 'diego@workstation: /var/log' showing the following commands and output:

```
diego@workstation:~/sysAds6$ cd /var/log
diego@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
btmtp                 installer            syslog.3.gz
btmtp.1               journal              tallylog
diego@workstation:/var/log$ cat apt/history.log
```

The output of `cat apt/history.log` shows installation details for various packages, including `libgcc-7-dev:amd64`, `python3-pip:amd64`, and others.

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`



A terminal window showing the execution of an Ansible command:

```
diego@workstation:~/sysAds6$ ansible all -m apt -a name=snapd --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
  "cache_update_time": 1694430231,
  "cache_updated": false,
  "changed": false
}
192.168.56.103 | SUCCESS => {
  "cache_update_time": 1694430231,
  "cache_updated": false,
  "changed": false
}
diego@workstation:~/sysAds6$
```

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

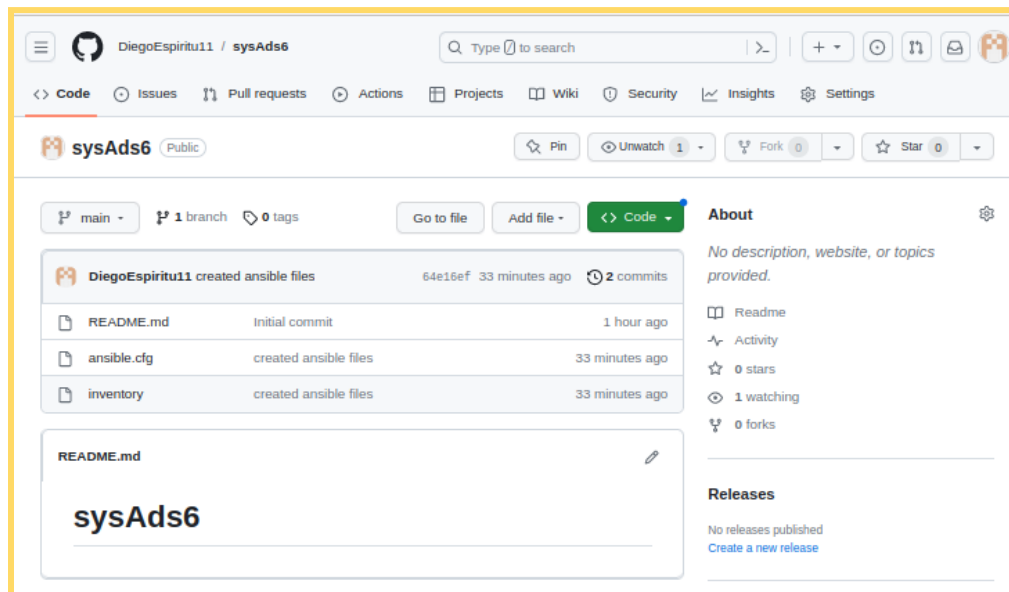
The result is in python format and it didn't change the remote server.

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

```
diego@workstation:~/sysAds6$ ansible all -m apt -a "name=snapt state=latest" --become --ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
  "cache_update_time": 1694430231,
  "cache_updated": false,
  "changed": false
}
192.168.56.102 | SUCCESS => {
  "cache_update_time": 1694430231,
  "cache_updated": false,
  "changed": false
}
diego@workstation:~/sysAds6$
```

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.



## 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
---
- hosts: all
  become: true
  task:

    - name: install apache2 package
      apt:
        name: apache2

[ Read 8 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
```

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.

It installs the file name of the playbook and it shows that in the server 1 and 2 have changes after installing.

```
diego@workstation:~/sysAds6$ nano install_apache.yml
diego@workstation:~/sysAds6$ nano install_apache.yml
diego@workstation:~/sysAds6$ ansible-playbook --ask-become-pass install_apache.
yml
BECOME password:

PLAY [all] *****
*

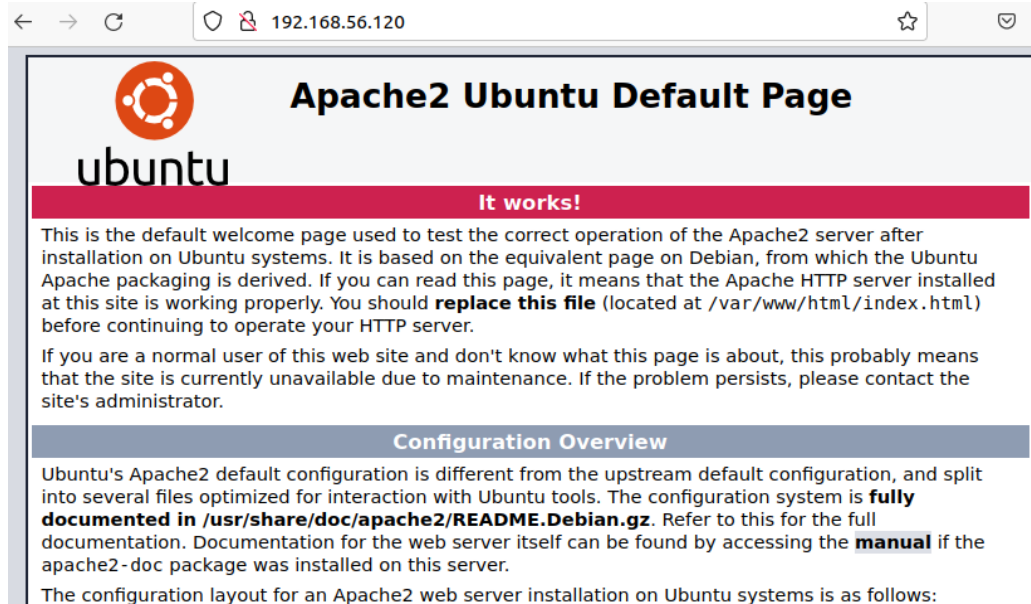
TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

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

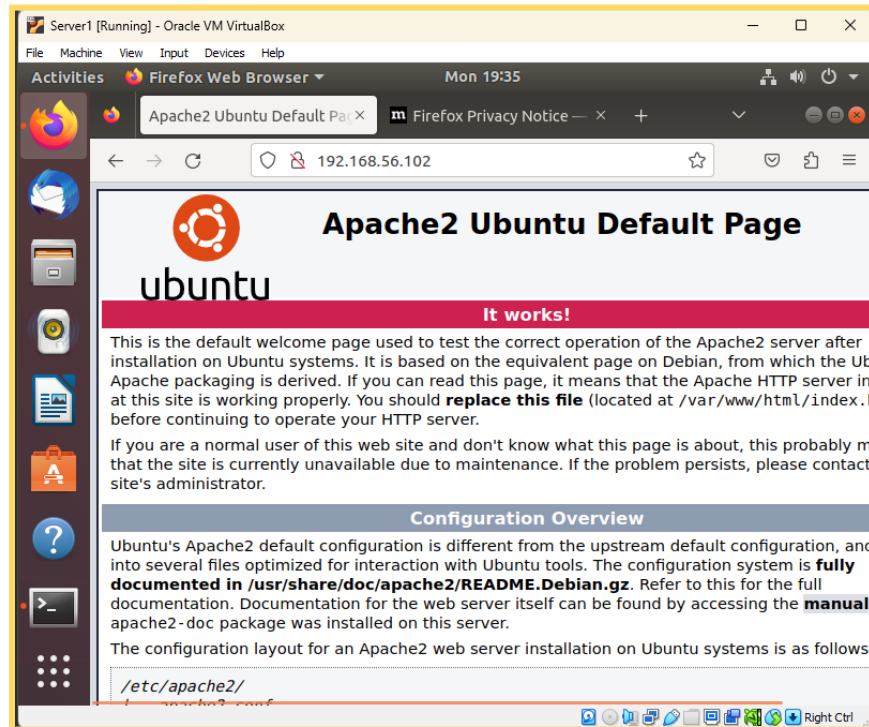
PLAY RECAP *****
*
192.168.56.102      : ok=2    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=2    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
diego@workstation:~/sysAds6$
```



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.

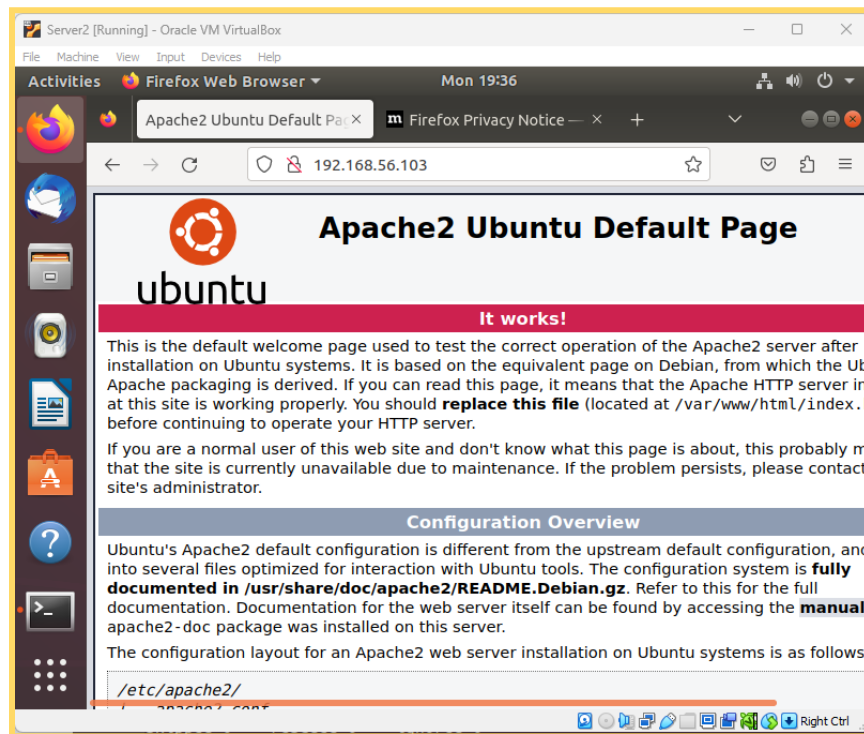


## Server 1





## Server 2



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?

As seen in the result after changing the name of the package it could not be recognized since it is not a name of any packages.

```
diego@workstation:~/sysAds6$ nano install_apache.yml
diego@workstation:~/sysAds6$ 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 [install apache2 package] *****
*
fatal: [192.168.56.103]: FAILED! => {"changed": false, "msg": "No package match
ing 'geng_geng' is available"}
fatal: [192.168.56.102]: FAILED! => {"changed": false, "msg": "No package match
ing 'geng_geng' is available"}

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

diego@workstation:~/sysAds6$
```

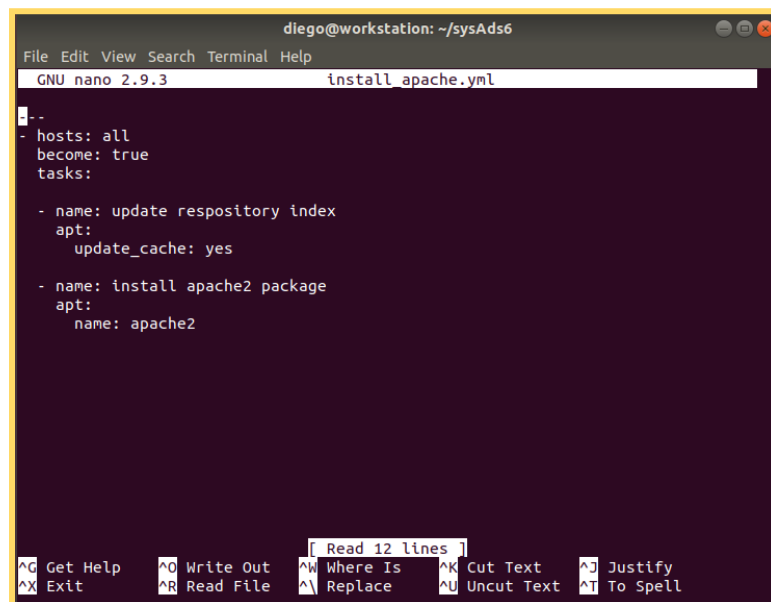
5. This time, we are going to put additional task to our playbook. Edit the *install\_apache.yml*. As you can see, we are now adding an additional command, which is the *update\_cache*. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities)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



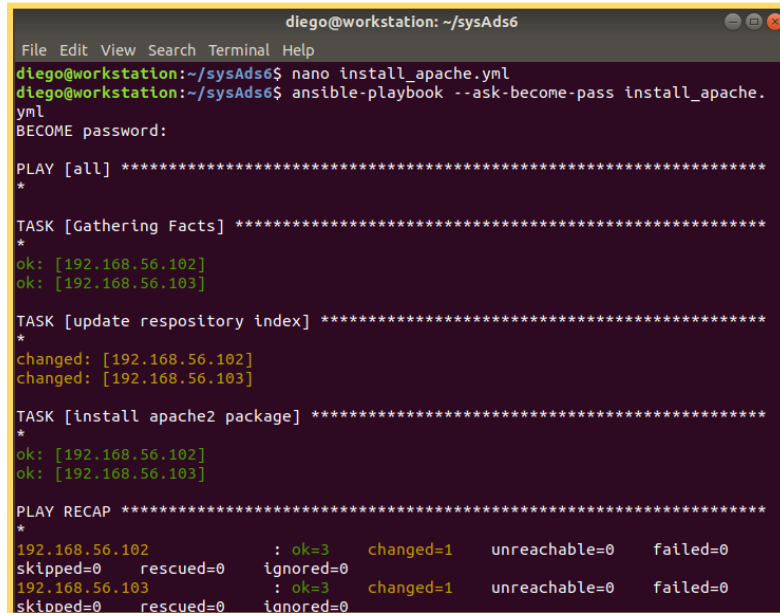
```
diego@workstation: ~/sysAds6
File Edit View Search Terminal Help
GNU nano 2.9.3 install_apache.yml
--
- hosts: all
  become: true
  tasks:

    - name: update respository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2

^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
```

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



```
diego@workstation: ~/sysAds6
File Edit View Search Terminal Help
diego@workstation:~/sysAds6$ nano install_apache.yml
diego@workstation:~/sysAds6$ ansible-playbook --ask-become-pass install_apache.
yaml
BECOME password:

PLAY [all] *****
*

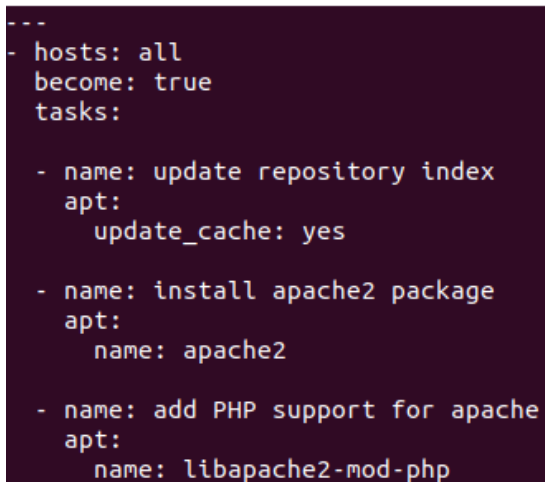
TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

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

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

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



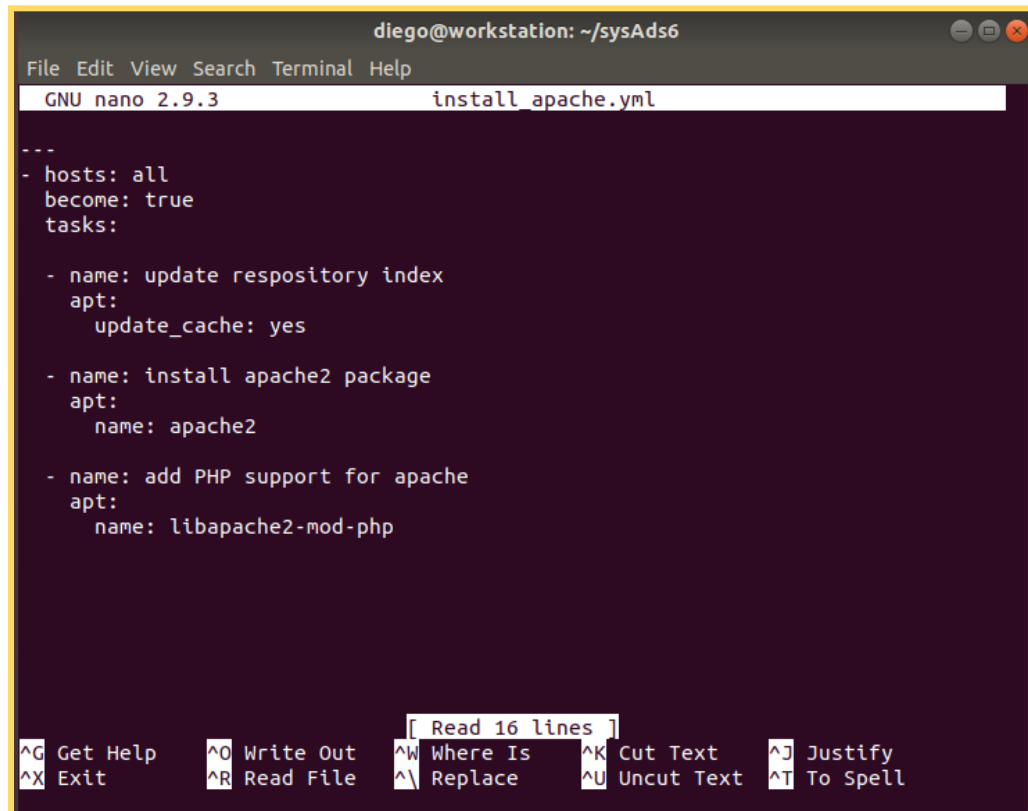
```
---
- hosts: all
  become: true
  tasks:

    - 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.



```
diego@workstation: ~/sysAds6
File Edit View Search Terminal Help
GNU nano 2.9.3 install_apache.yml

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

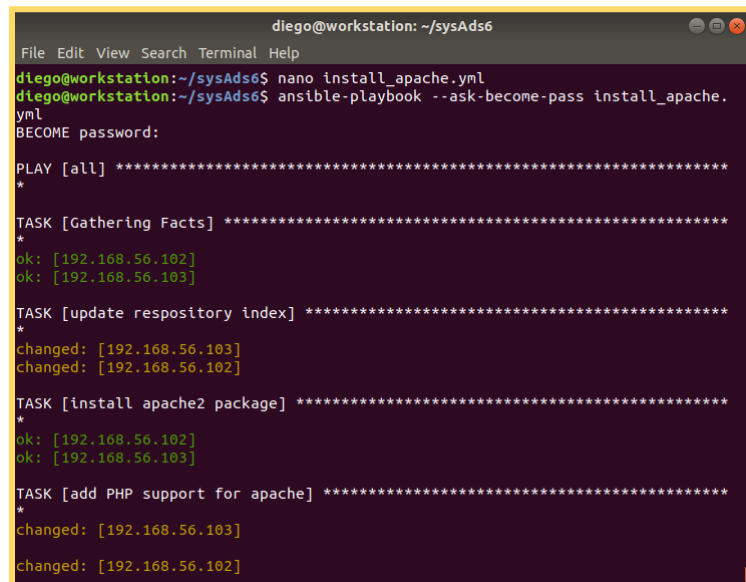
    - name: update repository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2

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

[ Read 16 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
```

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



```
diego@workstation: ~/sysAds6
File Edit View Search Terminal Help
diego@workstation:~/sysAds6$ nano install_apache.yml
diego@workstation:~/sysAds6$ ansible-playbook --ask-become-pass install_apache.
yml
BECOME password:

PLAY [all] *****
*

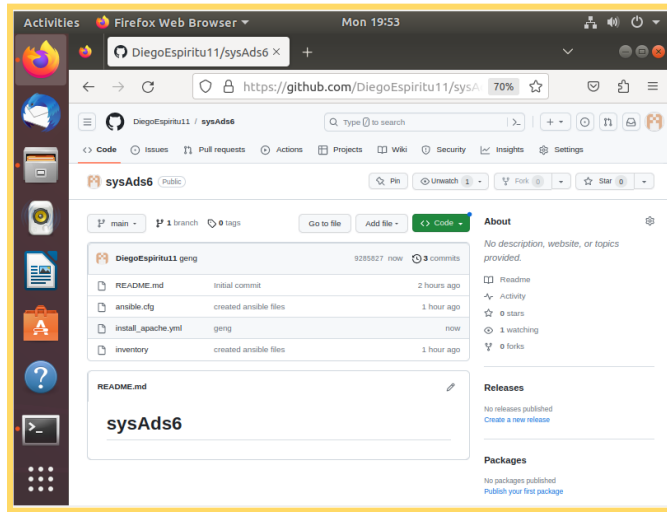
TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

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

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]
```

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



[GitHub - DiegoEspiritu11/sysAds6](https://github.com/DiegoEspiritu11/sysAds6)

### Reflections:

Answer the following:

1. What is the importance of using a playbook?

Playbooks are useful for automating repetitive tasks that would take too long to do manually. They help ensure the same actions are done consistently on multiple systems, and make it easier to develop business skills. Playbooks are also frameworks for prewritten code that us developers can use for ad hoc or basic templates.

2. Summarize what we have done on this activity.

In this activity, we learned how to create and execute an Ansible playbook. The activity provides a fundamental understanding of creating a playbook and highlights errors that may occur in the code. Playbooks help ensure that the same actions are performed consistently across multiple systems. They also codify operational knowledge and serve as frameworks of prewritten code.

### Conclusion:

I learned a lot about creating an Ansible playbook in this activity. The other activities we did in the past are now more sensible to me because they are all connected to each other and serve as the basic fundamentals and building blocks of system administration. I am excited to learn more about this subject in the near future.