Al-Najah National University
Computer Network and Information Security
Network Administration Lab

Automation(Ansible)

Instructor:Dr. Ahmed Awad

Dima Bshara
Mohammed Adnan

1 Abstract

This experiment has two sections. The first section aims to install and configure OpenSSH using keys. Also, create a GitHub account and set up and use git on our machine for version control, then install ansible and run simple commands. In the second section, we define a playbook then create it also Use the “When” conditional, then we target specific hosts with our plays. In the end, we use Tags.

2 Introduction

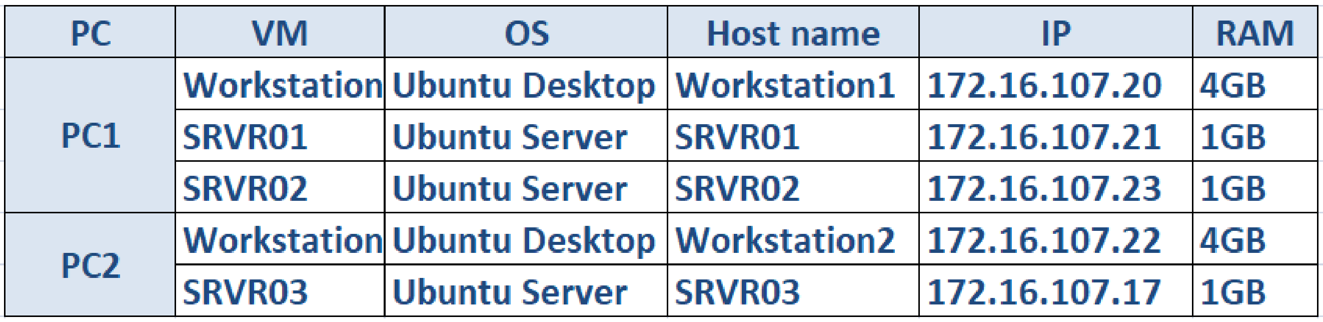
Automation describes a wide range of technologies that reduce human intervention in processes. Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs. Designed for multi-tier deployments since day one Ansible models, our IT infrastructure described how all of our systems interrelate, rather than just managing one system at a time. It uses no agents and no additional custom security infrastructure, so it is easy to deploy - and most importantly, it uses simple language (YAML, in the form of Ansible Playbooks) that allows us to describe our automation jobs. Ansible works by connecting to our nodes and pushing out small programs called (Ansible Modules) to them. These programs are documentary to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default) and removes them when finished. ourr library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically our will work with ourr favorite terminal program, a text editor, and probably a version control system to keep track of changes to ourr content. Passwords are supported, but SSH keys with ssh-agent are one of the best ways to use ansible. Lots of options, root logins are not required so, our can log in as any user and then Su or Sudo to any user. Ansible’s (authorized_key) module is a great way to use ansible to control what machines can access what hosts. Other options, like Kerberos or identity management systems, can also be used. Ansible runs commands on local or remote computers. It can move files around, create files from templates, and run command-line tools primarily used for system administration tasks at scale. It has a push model rather than a pull model like Puppet. If our have used Puppet, Ansible does not evaluate what changes need; to be made and makes those just run through all the commands every time. Inventory file, An Ansible-specific file that defines the systems (hosts) and groups of hosts on which Ansibles should operate. In this inventory, we connect to multiple remote machines. This is common, practice Ansibles playbooks are, stored on ourr laptop or a build server, and from there Ansibles connects out to the remotes machines that are managed. The advantage of running remotely is that our can manage dozens of machines simultaneously, rather than just the local machine. This scaling out to N machines is one of the strengths of ansibles. A task we want to perform on our hosts using ansible is called a play. A playbook contains one or more tasks we want to execute. playbooks; are written using the YAML language, which is a human-readable data serialization language. Yaml; is usually used for configuration files. The real strength of ansibles comes from playbooks. When we write our playbooks, we define the state we want our servers to be in and the commands we want ansibles to perform to bring our servers to that state.

3 Setting up the Work Environment

Required resources:

- 2 PCs with virtualBox [PC1,PC2].
- Internet connection.

We used 5 VMs as listed in the table, which shows the IP address of each one, the operating system that uses, the hostname, and the RAM needs.



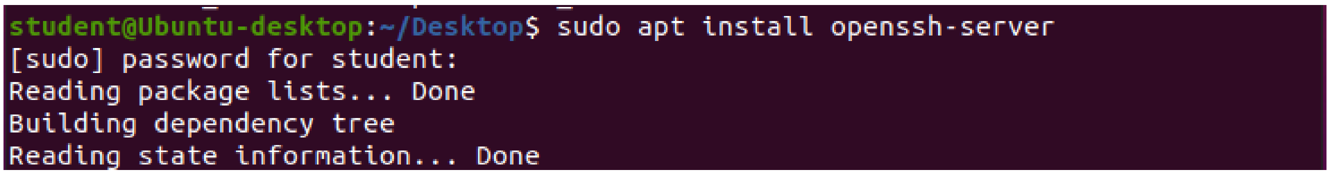
PC	VM	OS	Host name	IP	RAM
PC1	Workstation	Ubuntu Desktop	Workstation1	172.16.107.20	4GB
	SRVR01	Ubuntu Server	SRVR01	172.16.107.21	1GB
	SRVR02	Ubuntu Server	SRVR02	172.16.107.23	1GB
PC2	Workstation	Ubuntu Desktop	Workstation2	172.16.107.22	4GB
	SRVR03	Ubuntu Server	SRVR03	172.16.107.17	1GB

Figure 1: Properties VMs.

4 Procedure

4.1 Install and Configure OpenSSH

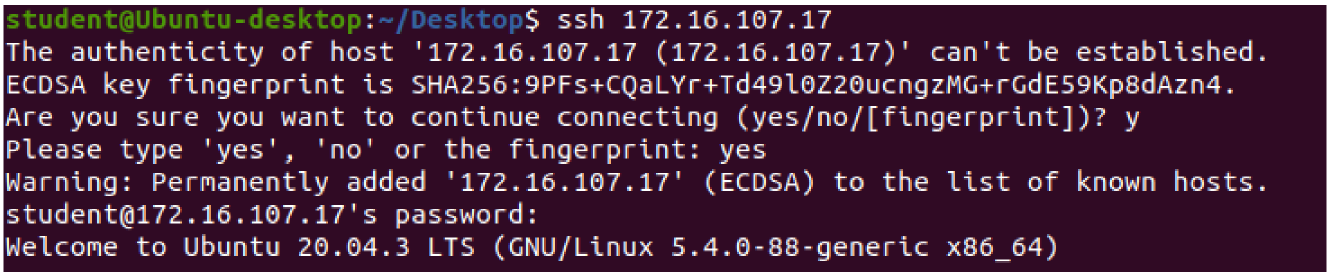
1. At first, we installed an open-ssh server on both workstations as figure 2 show.



```
student@Ubuntu-desktop:~/Desktop$ sudo apt install openssh-server
[sudo] password for student:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 2: Install open-ssh server.

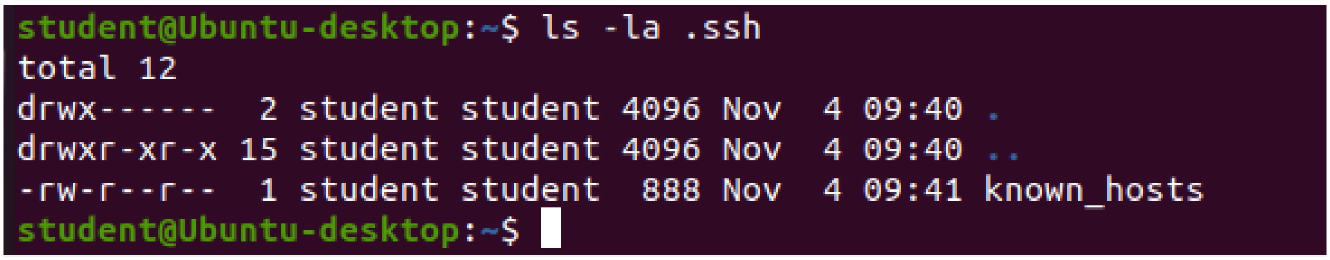
2. Then from both workstations, we open an ssh connection to each of the three servers. This initial connection is very important since we need to accept the servers fingerprint in order for it to be added to our known hosts file in each of the workstations. We accept the server fingerprint for a server as figure 3 show .



```
student@Ubuntu-desktop:~/Desktop$ ssh 172.16.107.17
The authenticity of host '172.16.107.17 (172.16.107.17)' can't be established.
ECDSA key fingerprint is SHA256:9PFs+CQaLYr+Td49l0Z20ucngzMG+rGdE59Kp8dAzn4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '172.16.107.17' (ECDSA) to the list of known hosts.
student@172.16.107.17's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-88-generic x86_64)
```

Figure 3: SSH connection to server.

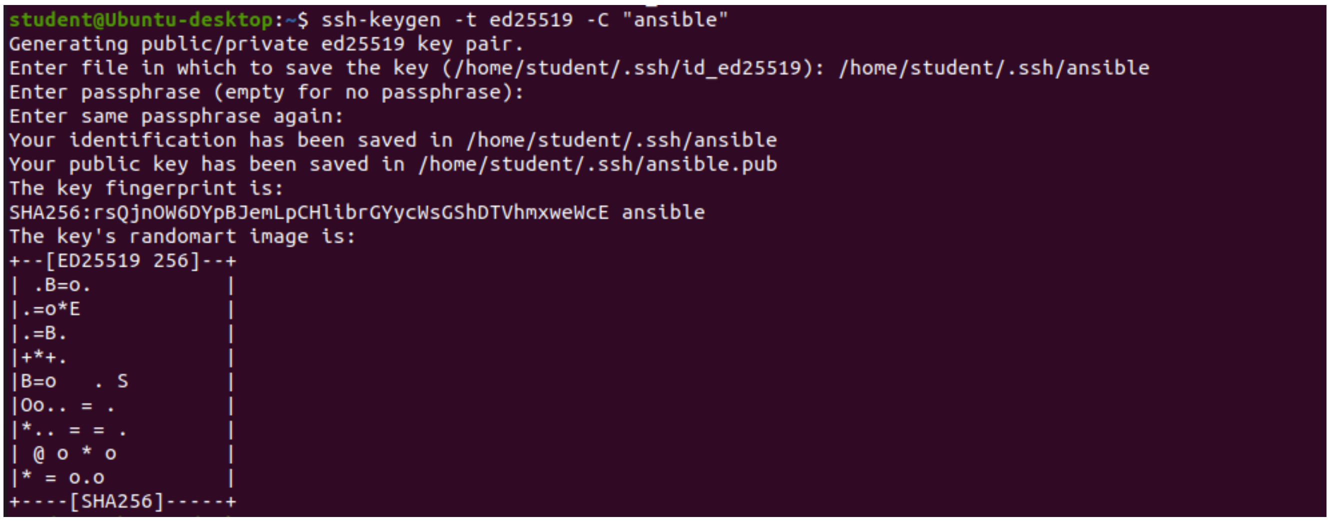
3. After that, on workstation1 we apply the command **ls -la .ssh** to check the keys available. The folder does not contain any keys at the moment as figure 4 shows.



```
student@Ubuntu-desktop:~$ ls -la .ssh
total 12
drwx------  2 student student 4096 Nov  4 09:40 .
drwxr-xr-x 15 student student 4096 Nov  4 09:40 ..
-rw-r--r--  1 student student  888 Nov  4 09:41 known_hosts
student@Ubuntu-desktop:~$
```

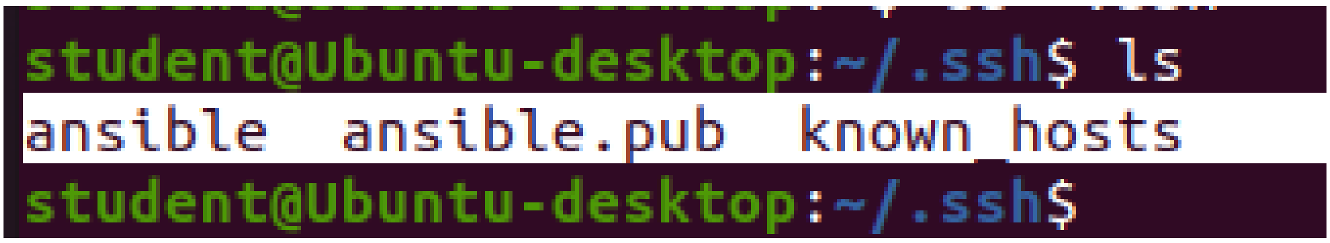
Figure 4: Check available keys.

4. We created an SSH key pair for connecting to the servers using ansible, the command **ssh-keygen -t ed25519 -C "ansible"**
-t: type of key, **-C: comment**, **ed25519** is the most secure key option. Figure 5 shows generates the public and private key and stored in a subdirectory inside the users' home directory named .ssh [/home/student/.ssh]. Then we check contents of the .ssh directory we find two files: **ansible.pub: public key** , **ansible: private key** as figure6 show.



```
student@Ubuntu-desktop:~$ ssh-keygen -t ed25519 -C "ansible"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/student/.ssh/id_ed25519): /home/student/.ssh/ansible
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/ansible
Your public key has been saved in /home/student/.ssh/ansible.pub
The key fingerprint is:
SHA256:rsQjnOW6DYpBJemLpCHlibrGYycWsGShDTVhmxweWcE ansible
The key's randomart image is:
+--[ED25519 256]--+
| .B=o.           |
|.=o*E            |
|.=B.             |
|+*+.             |
|B=o    . S       |
|Oo.. = .         |
|*.. = = .        |
| @ o * o         |
|* = o.o          |
+----[SHA256]-----+
```

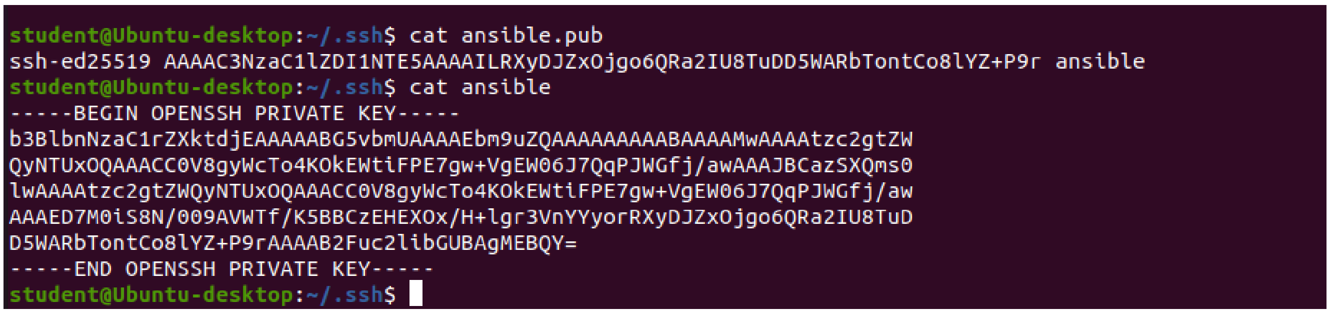
Figure 5: Generates Ansible keys.



```
student@Ubuntu-desktop:~/.ssh$ ls
ansible  ansible.pub  known_hosts
student@Ubuntu-desktop:~/.ssh$
```

Figure 6: Check the keys.

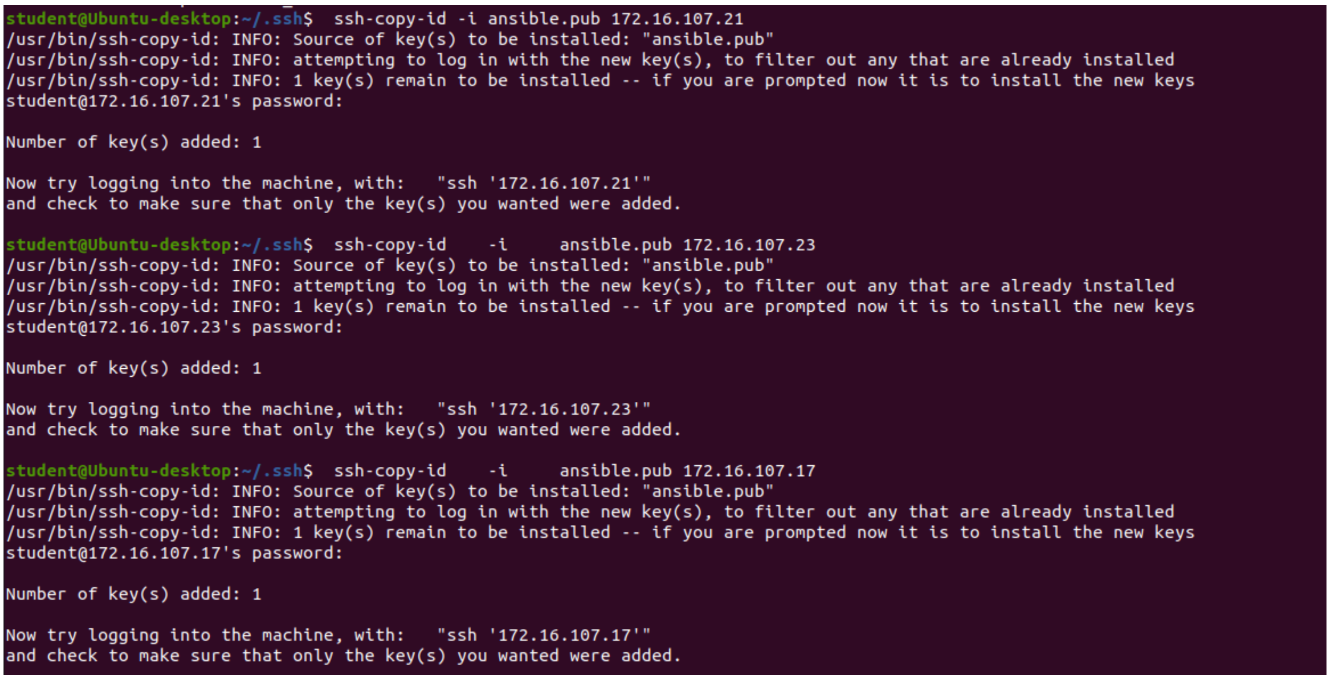
5. We used the command **cat** to see contents of both keys and notice the difference as figure 7 show that the private key have more content than public key.



```
student@Ubuntu-desktop:~/.ssh$ cat ansible.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILRXyDJZxOjgo6QRa2IU8TuDD5WARbTontCo8lYZ+P9r ansible
student@Ubuntu-desktop:~/.ssh$ cat ansible
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACC0V8gyWcTo4KOkEWtiFPE7gw+VgEW06J7QqPJWGfj/awAAAJBCazSXQms0
lwAAAAtzc2gtZWQyNTUxOQAAACC0V8gyWcTo4KOkEWtiFPE7gw+VgEW06J7QqPJWGfj/aw
AAAED7M0iS8N/009AVWTf/K5BBCzEHEXOx/H+lgr3VnYYyorRXyDJZxOjgo6QRa2IU8TuD
D5WARbTontCo8lYZ+P9rAAAAB2Fuc2libGUBAgMEBQY=
-----END OPENSSH PRIVATE KEY-----
student@Ubuntu-desktop:~/.ssh$ 
```

Figure 7: Check the keys.

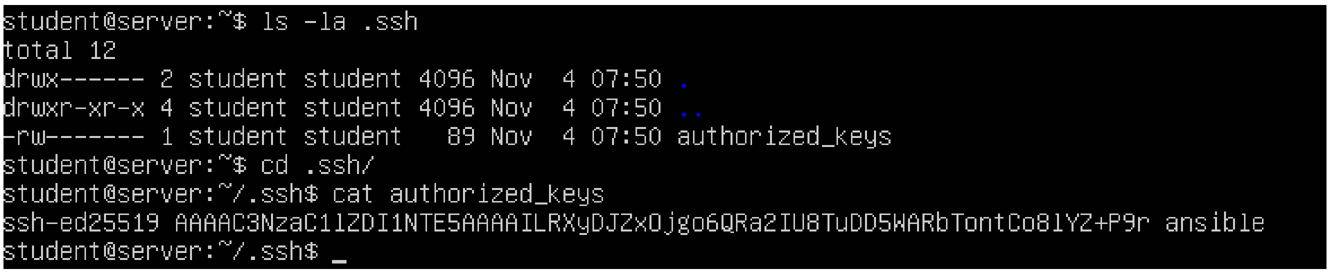
6. Also, we copied the public key to each of the 3 servers using the commands as figure 8 shows, to allow connection to the server using public key "ansible.pub" without asking for the password.



```
student@Ubuntu-desktop:~/.ssh$  ssh-copy-id -i ansible.pub 172.16.107.21
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "ansible.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
student@172.16.107.21's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.16.107.21'"
and check to make sure that only the key(s) you wanted were added.

student@Ubuntu-desktop:~/.ssh$  ssh-copy-id    -i     ansible.pub 172.16.107.23
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "ansible.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
student@172.16.107.23's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.16.107.23'"
and check to make sure that only the key(s) you wanted were added.

student@Ubuntu-desktop:~/.ssh$  ssh-copy-id    -i     ansible.pub 172.16.107.17
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "ansible.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
student@172.16.107.17's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.16.107.17'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 8: Copy the public key to 3 servers.

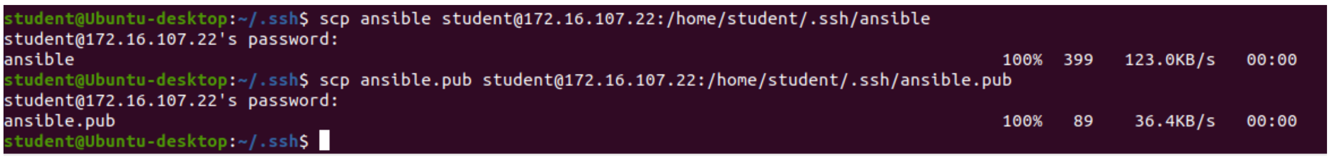
7. We check the .ssh directory on a server to confirm that the public key has been added to the file **/.ssh/authorized_keys.** as figure 9 show .



```
student@server:~$ ls -la .ssh
total 12
drwx------ 2 student student 4096 Nov  4 07:50 .
drwxr-xr-x 4 student student 4096 Nov  4 07:50 ..
-rw------- 1 student student   89 Nov  4 07:50 authorized_keys
student@server:~$ cd .ssh/
student@server:~/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILRXyDJZxOjgo6QRa2IU8TuDD5WARbTontCo8lYZ+P9r ansible
student@server:~/.ssh$ _
```

Figure 9: Confirm the public key added.

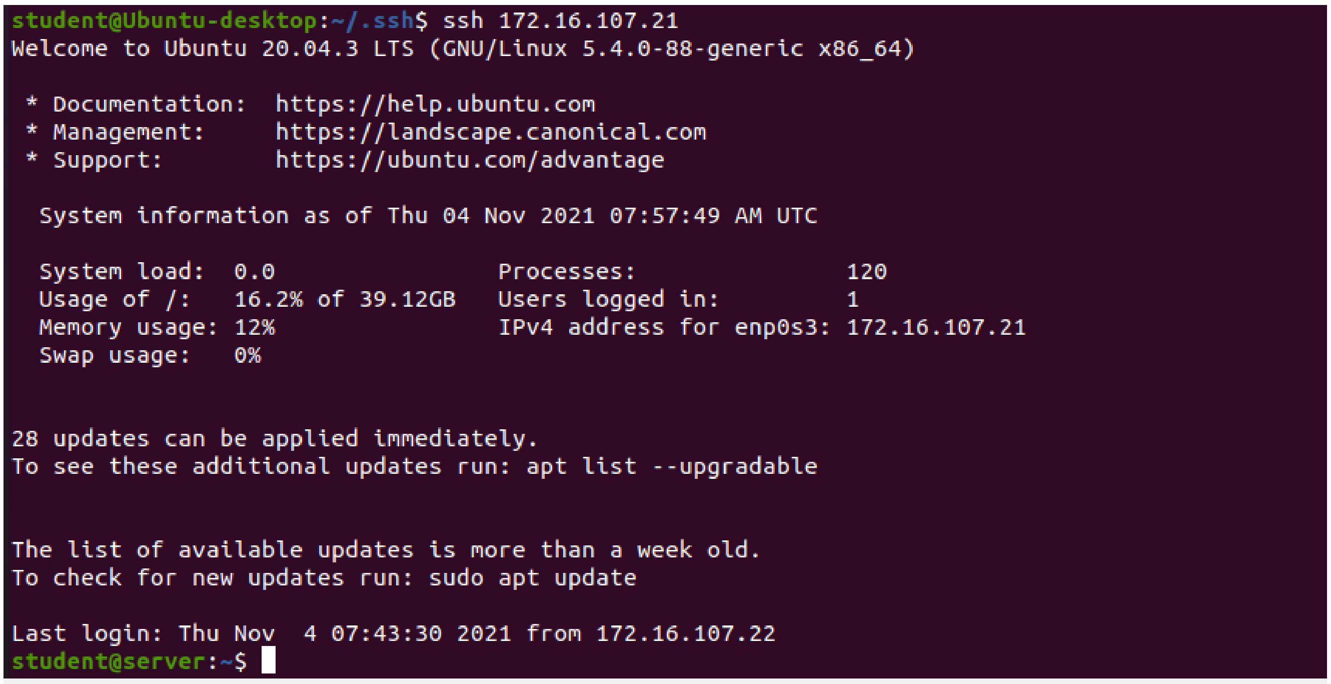
8. We copied both keys public and private to workstation2 using Secure Copy Protocol (scp) as figure 10 show, since these must be identical because they identify a single user ID which is student.



```
student@Ubuntu-desktop:~/.ssh$ scp ansible student@172.16.107.22:/home/student/.ssh/ansible
student@172.16.107.22's password:
ansible                                                                             100%  399   123.0KB/s   00:00
student@Ubuntu-desktop:~/.ssh$ scp ansible.pub student@172.16.107.22:/home/student/.ssh/ansible.pub
student@172.16.107.22's password:
ansible.pub                                                                         100%   89    36.4KB/s   00:00
student@Ubuntu-desktop:~/.ssh$
```

Figure 10: copy public and private keys to workstation2.

9. Then we test our environment by connect it via ssh to a server without the need to enter a password because the public key was saved in the server, as shown in figure 11.



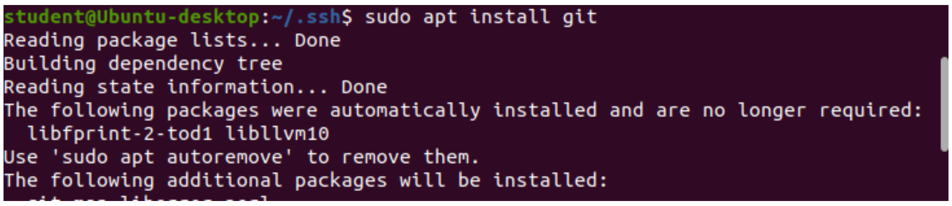
```
student@Ubuntu-desktop:~/.ssh$ ssh 172.16.107.21
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 04 Nov 2021 07:57:49 AM UTC

  System load:  0.0                Processes:               120
  Usage of /:   16.2% of 39.12GB   Users logged in:         1
  Memory usage: 12%                IPv4 address for enp0s3: 172.16.107.21
  Swap usage:   0%


28 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Nov  4 07:43:30 2021 from 172.16.107.22
student@server:~$
```

Figure 11: SSH connection.

4.2 Version Control

In this section of the lab we will create a GitHub account and use it's repositories to share configuration files between both workstations.

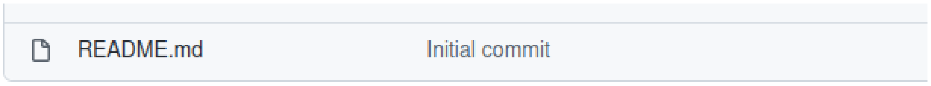
1. On both workstations we install a **git** tool as shown in figure 12



```
student@Ubuntu-desktop:~/.ssh$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm10
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
```

Figure 12: Install git.

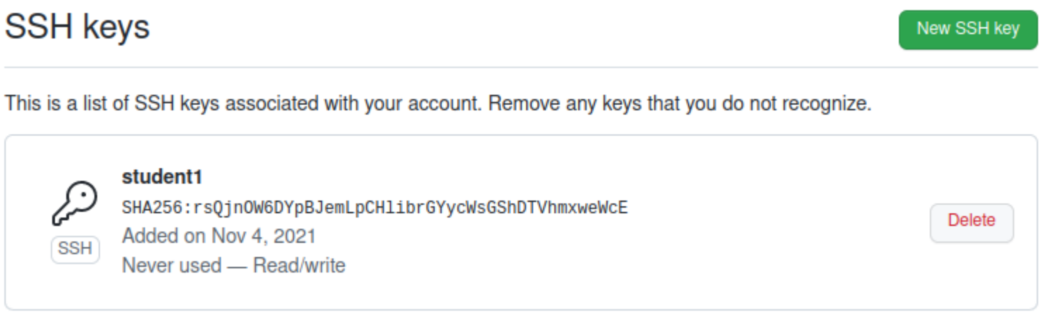
2. After that, on workstation navigate to www.github.com and sign up. create a repository. Name the repository nislab. Figure 13 show **README** file which is inside nislab repository .



README.md Initial commit

Figure 13: README file.

3. Then, we open the SSH and GPG keys and deleted any keys if there, and we add New SSH Key. The title will be "Student1" as shown in figure 14.



SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

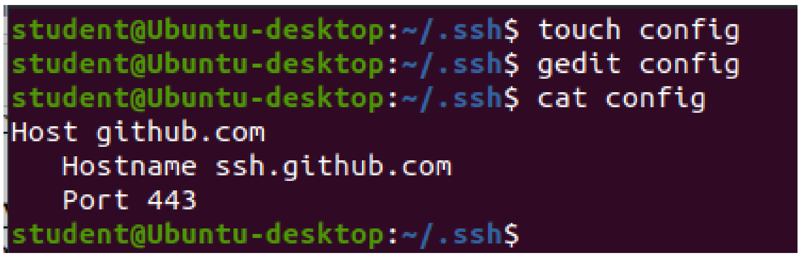
SSH

student1
SHA256:rsQjnOW6DYpBJemLpCHlibrGYycWsGShDTVhmxweWcE
Added on Nov 4, 2021
Never used — Read/write

Delete

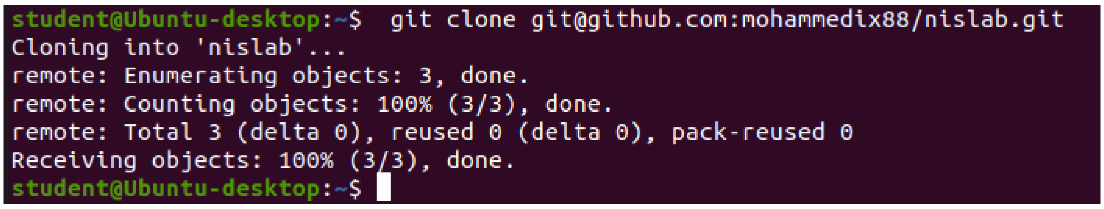
Figure 14: Add public key in profile .

4. Now, Inside the .ssh directory of both workstations create a new file named config with the following contents as figure 15 shows. Then we copy the ssh link from the repository to download a copy of the repository inside our home directory on both workstations, as shown in figure 16.



```
student@Ubuntu-desktop:~/.ssh$ touch config
student@Ubuntu-desktop:~/.ssh$ gedit config
student@Ubuntu-desktop:~/.ssh$ cat config
Host github.com
   Hostname ssh.github.com
   Port 443
student@Ubuntu-desktop:~/.ssh$
```

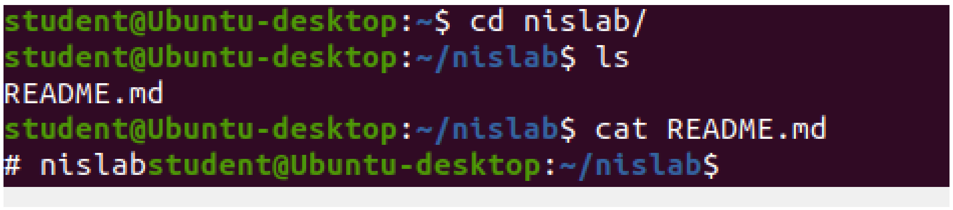
Figure 15: edit config file .



```
student@Ubuntu-desktop:~$  git clone git@github.com:mohammedix88/nislab.git
Cloning into 'nislab'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
student@Ubuntu-desktop:~$
```

Figure 16: git clone using ssh .

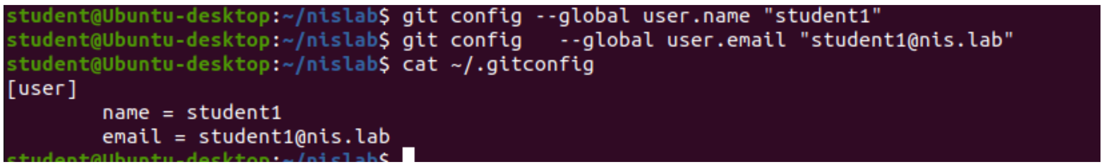
5. After that, we list our home directory contents on both workstations, Nislab directory will be present on both of them. So made sure to open the Nislab dir and display his content as shown in figure 17.



```
student@Ubuntu-desktop:~$ cd nislab/
student@Ubuntu-desktop:~/nislab$ ls
README.md
student@Ubuntu-desktop:~/nislab$ cat README.md
# nislabstudent@Ubuntu-desktop:~/nislab$
```

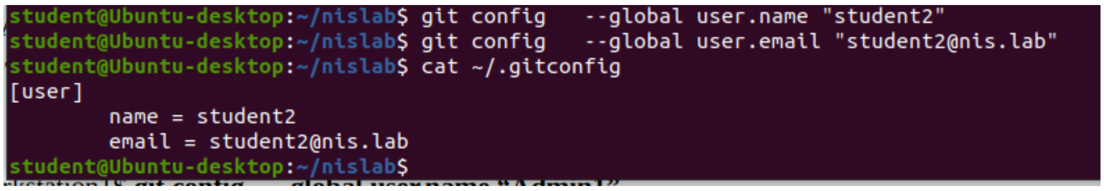
Figure 17: Made sure of nislab .

6. Then, We need to tell git who we are on both machines as shown in figures 18 and 19.



```
student@Ubuntu-desktop:~/nislab$ git config --global user.name "student1"
student@Ubuntu-desktop:~/nislab$ git config   --global user.email "student1@nis.lab"
student@Ubuntu-desktop:~/nislab$ cat ~/.gitconfig
[user]
        name = student1
        email = student1@nis.lab
```

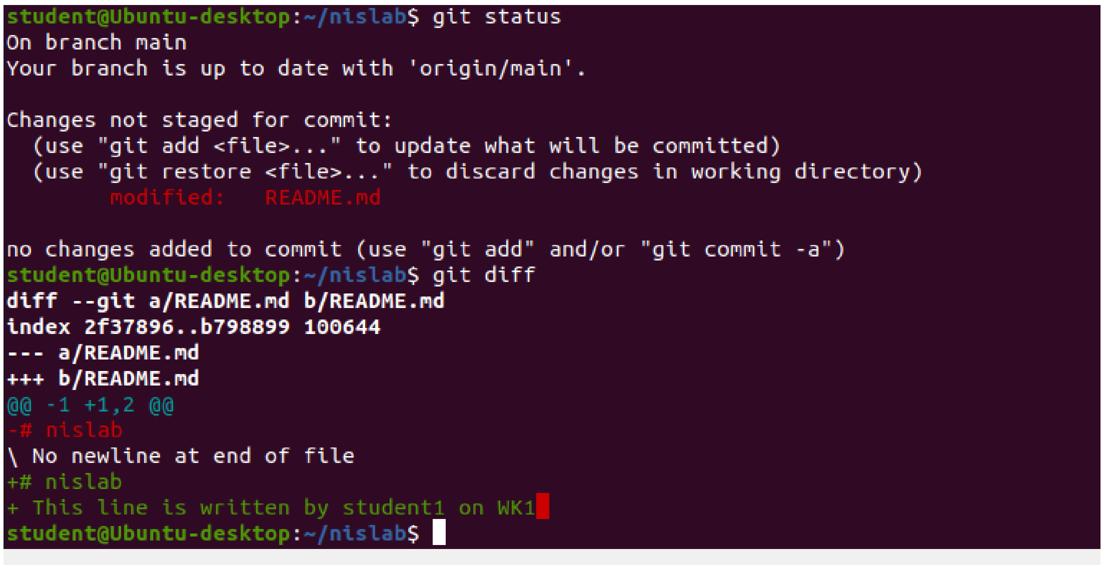
Figure 18: Student1 user.



```
student@Ubuntu-desktop:~/nislab$ git config   --global user.name "student2"
student@Ubuntu-desktop:~/nislab$ git config   --global user.email "student2@nis.lab"
student@Ubuntu-desktop:~/nislab$ cat ~/.gitconfig
[user]
        name = student2
        email = student2@nis.lab
student@Ubuntu-desktop:~/nislab$
```

Figure 19: Student2 user .

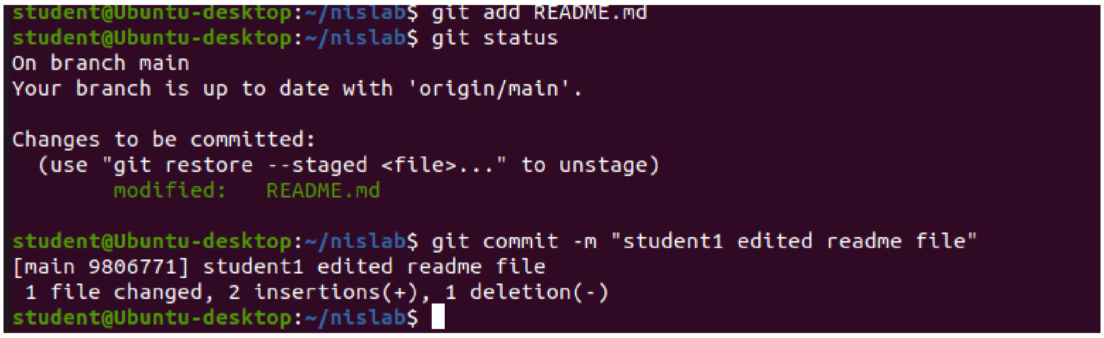
7. Now, we edit the readme.MD file on workstation1 using a text editor to add a few words to the end of the file like “ This line is written by admin1 on workstation1”. Then we will use the status command this will show that a change was done as figure 20 shows and then we use the diff command to see the difference between those changes.



```
student@Ubuntu-desktop:~/nislab$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
student@Ubuntu-desktop:~/nislab$ git diff
diff --git a/README.md b/README.md
index 2f37896..b798899 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# nislab
\ No newline at end of file
+# nislab
+ This line is written by student1 on WK1
student@Ubuntu-desktop:~/nislab$
```

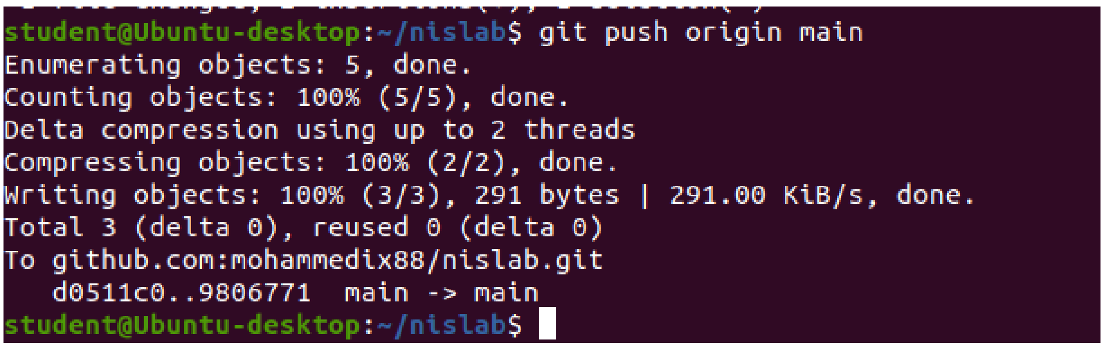
Figure 20: Status & diff command .

8. Then, we add the file to the list of files to be committed, as shown in figure 21. And we push it to be available for any user to pull it as shown in figure 22.



```
student@Ubuntu-desktop:~/nislab$ git add README.md
student@Ubuntu-desktop:~/nislab$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

student@Ubuntu-desktop:~/nislab$ git commit -m "student1 edited readme file"
[main 9806771] student1 edited readme file
 1 file changed, 2 insertions(+), 1 deletion(-)
student@Ubuntu-desktop:~/nislab$
```

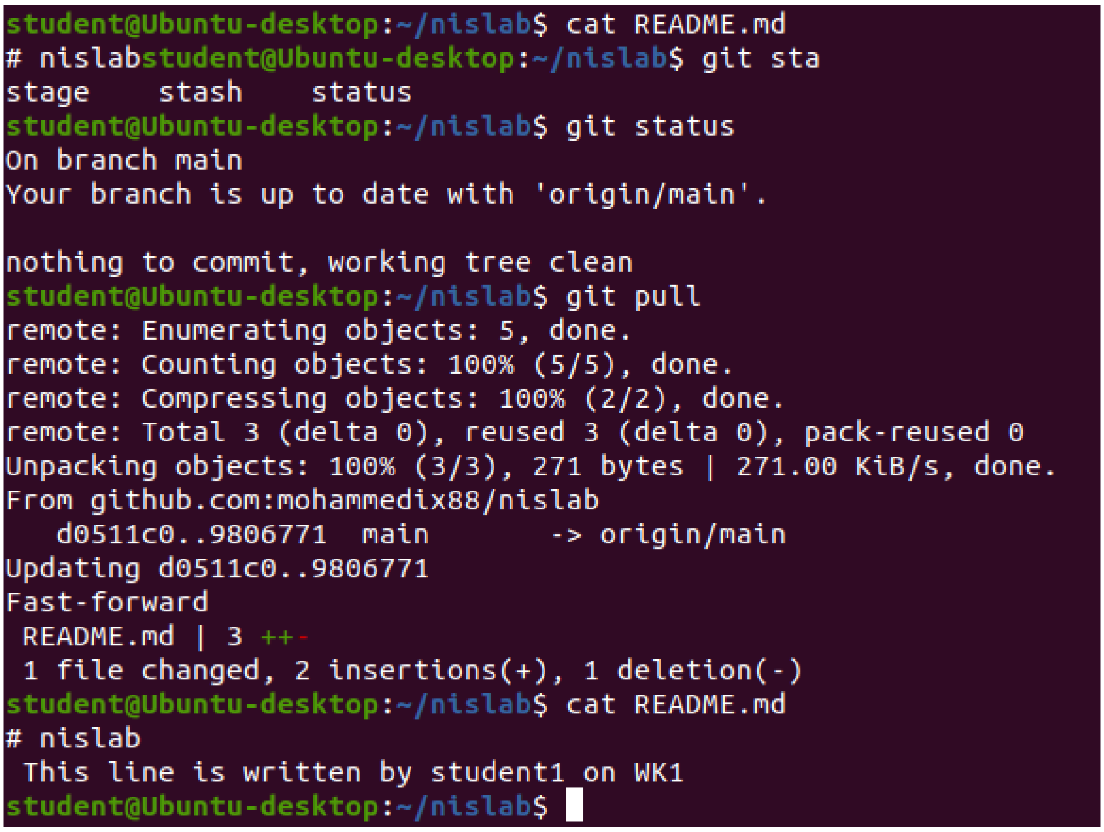
Figure 21: Add file to be commet .



```
student@Ubuntu-desktop:~/nislab$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:mohammedix88/nislab.git
   d0511c0..9806771  main -> main
student@Ubuntu-desktop:~/nislab$
```

Figure 22: Add file to be commet .

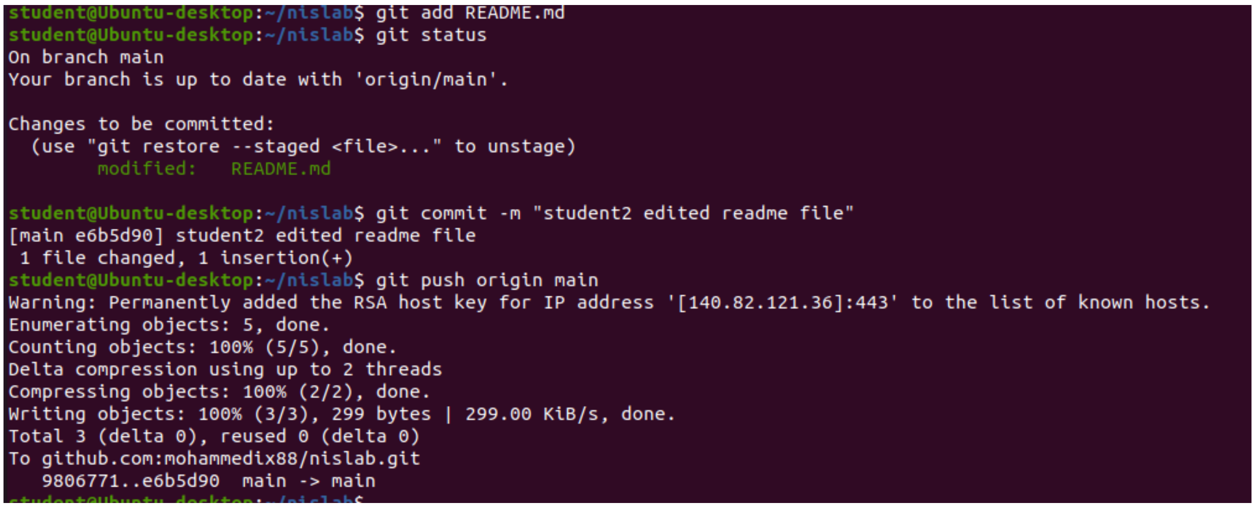
9. After we check online, we will see a change to the file. But workstaion2 is still out of sync. At first, we show the status of the git as shown in figure 23, as we see that there is no change or modification in the file, so we pull to see if the file has changed or not.



```
student@Ubuntu-desktop:~/nislab$ cat README.md
# nislabstudent@Ubuntu-desktop:~/nislab$ git sta
stage    stash    status
student@Ubuntu-desktop:~/nislab$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
student@Ubuntu-desktop:~/nislab$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
From github.com:mohammedix88/nislab
   d0511c0..9806771  main       -> origin/main
Updating d0511c0..9806771
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
student@Ubuntu-desktop:~/nislab$ cat README.md
# nislab
 This line is written by student1 on WK1
student@Ubuntu-desktop:~/nislab$ 
```

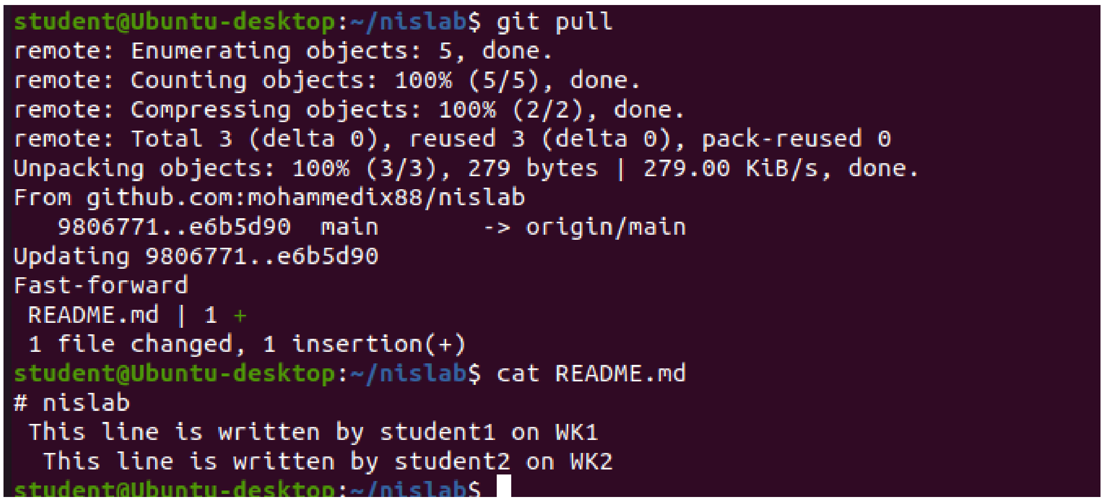
Figure 23: Pull file .

10. On workstation2, we edit the file by adding “Admin2 says hi”. Add, commit, and push the file as shown in figure 24. Then we check git status at workstation1 and do a pull then cat the contents as figure 25 shown.



```
student@Ubuntu-desktop:~/nislab$ git add README.md
student@Ubuntu-desktop:~/nislab$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

student@Ubuntu-desktop:~/nislab$ git commit -m "student2 edited readme file"
[main e6b5d90] student2 edited readme file
 1 file changed, 1 insertion(+)
student@Ubuntu-desktop:~/nislab$ git push origin main
Warning: Permanently added the RSA host key for IP address '[140.82.121.36]:443' to the list of known hosts.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:mohammedix88/nislab.git
   9806771..e6b5d90  main -> main
```

Figure 24: push file from Wk2 .

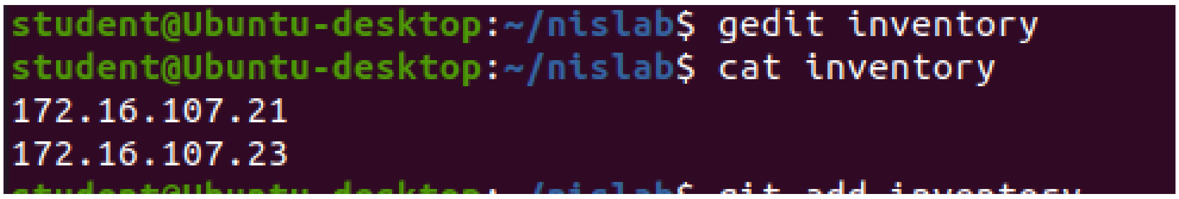


```
student@Ubuntu-desktop:~/nislab$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 279 bytes | 279.00 KiB/s, done.
From github.com:mohammedix88/nislab
   9806771..e6b5d90  main       -> origin/main
Updating 9806771..e6b5d90
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
student@Ubuntu-desktop:~/nislab$ cat README.md
# nislab
 This line is written by student1 on WK1
  This line is written by student2 on WK2
student@Ubuntu-desktop:~/nislab$
```

Figure 25: Pull file from WK1 .

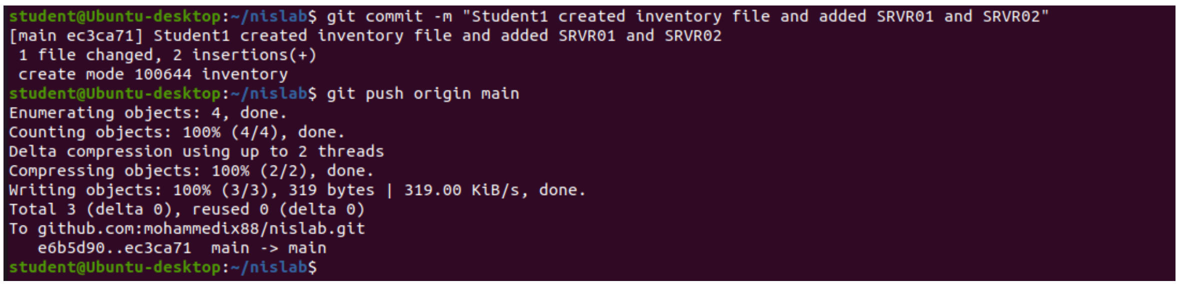
4.3 Ansible [ad-hoc commands]

1. On both workstations, we install ansible, which is an automated tool. Then we edit the inventory file which, will be located in Nislab dir by adding IP addresses of SRVR01 and SRVR02, each one in a line as figure 26 show. After that, we add the file to the list of files to be, committed then we push it to be available to pull for other users. Figure 27.



```
student@Ubuntu-desktop:~/nislab$ gedit inventory
student@Ubuntu-desktop:~/nislab$ cat inventory
172.16.107.21
172.16.107.23
```

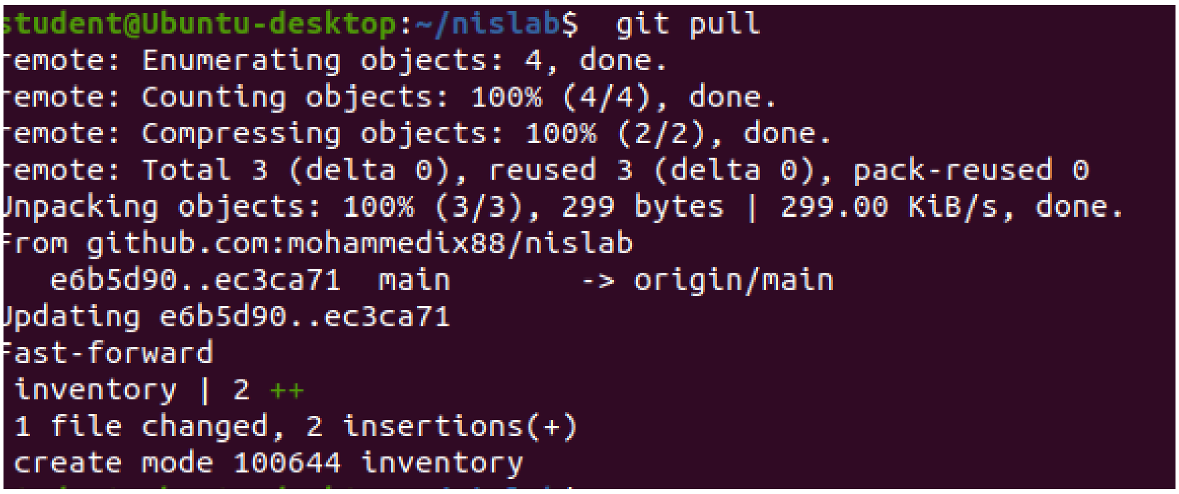
Figure 26: Edit inventory file.



```
student@Ubuntu-desktop:~/nislab$ git commit -m "Student1 created inventory file and added SRVR01 and SRVR02"
[main ec3ca71] Student1 created inventory file and added SRVR01 and SRVR02
 1 file changed, 2 insertions(+)
 create mode 100644 inventory
student@Ubuntu-desktop:~/nislab$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:mohammedix88/nislab.git
   e6b5d90..ec3ca71  main -> main
student@Ubuntu-desktop:~/nislab$
```

Figure 27: push inventory file.

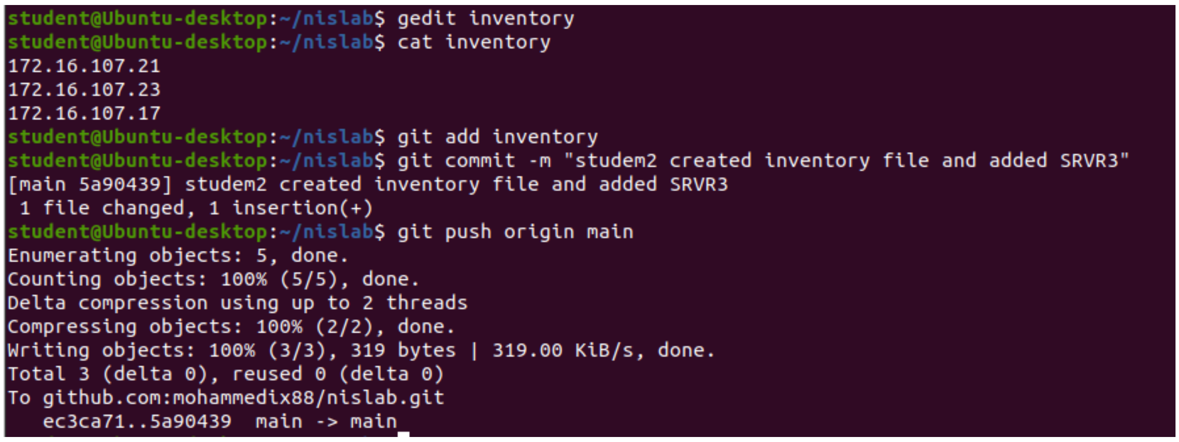
2. On the workstation2 we pull for get the new change of the file, as shown in figure 28.



```
student@Ubuntu-desktop:~/nislab$  git pull
emote: Enumerating objects: 4, done.
emote: Counting objects: 100% (4/4), done.
emote: Compressing objects: 100% (2/2), done.
emote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
npacking objects: 100% (3/3), 299 bytes | 299.00 KiB/s, done.
rom github.com:mohammedix88/nislab
   e6b5d90..ec3ca71  main       -> origin/main
pdating e6b5d90..ec3ca71
ast-forward
 inventory | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 inventory
```

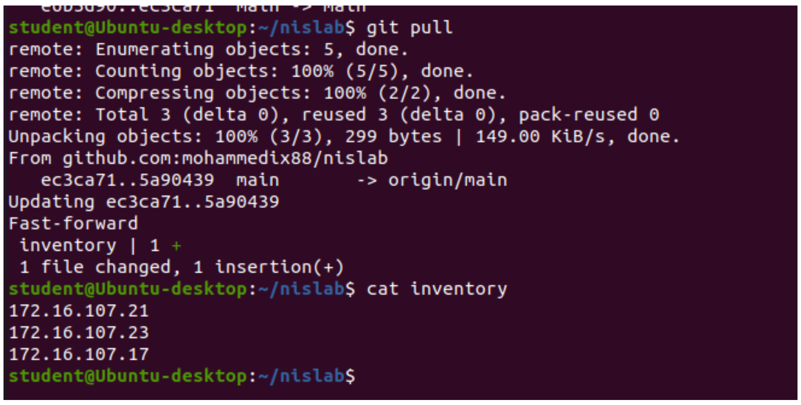
Figure 28: Pull in workstation2.

3. on workstation2 we edit the inventory file and add the IP address of SRVR03 save and push to GitHub with a comment “Admin2 modified the inventory and added SRVR03” then pull it on workstation1. Figures 29 and 30 shows those steps.



```
student@Ubuntu-desktop:~/nislab$ gedit inventory
student@Ubuntu-desktop:~/nislab$ cat inventory
172.16.107.21
172.16.107.23
172.16.107.17
student@Ubuntu-desktop:~/nislab$ git add inventory
student@Ubuntu-desktop:~/nislab$ git commit -m "studem2 created inventory file and added SRVR3"
[main 5a90439] studem2 created inventory file and added SRVR3
 1 file changed, 1 insertion(+)
student@Ubuntu-desktop:~/nislab$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:mohammedix88/nislab.git
   ec3ca71..5a90439  main -> main
```

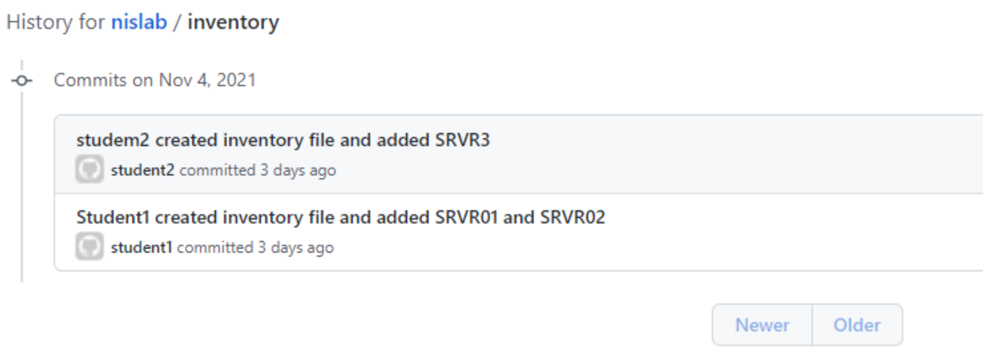
Figure 29: push in workstation2.



```
student@Ubuntu-desktop:~/nislab$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 299 bytes | 149.00 KiB/s, done.
From github.com:mohammedix88/nislab
   ec3ca71..5a90439  main       -> origin/main
Updating ec3ca71..5a90439
Fast-forward
 inventory | 1 +
 1 file changed, 1 insertion(+)
student@Ubuntu-desktop:~/nislab$ cat inventory
172.16.107.21
172.16.107.23
172.16.107.17
student@Ubuntu-desktop:~/nislab$
```

Figure 30: pull in workstation.

4. On GitHub web page we click the inventory file then click the history link we will see all versions of this file each with the comment that was written by each admin. as shown in figure 31.



History for nislab / inventory

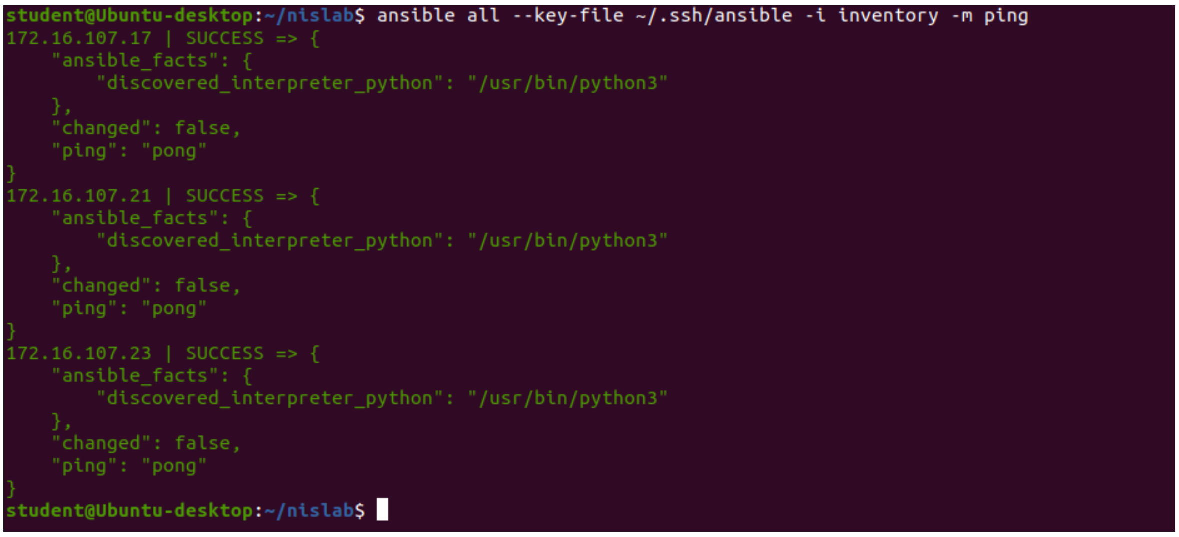
Commits on Nov 4, 2021

studem2 created inventory file and added SRVR3
student2 committed 3 days ago

Student1 created inventory file and added SRVR01 and SRVR02
student1 committed 3 days ago

Figure 31: History in github.

5. Now To run a command on all servers we use the command as shown in figure 32. *Note*: The ping here is NOT ICMP ping it's an ansible module that tests for a successful SSH connection to each of the servers in the inventory list we created.

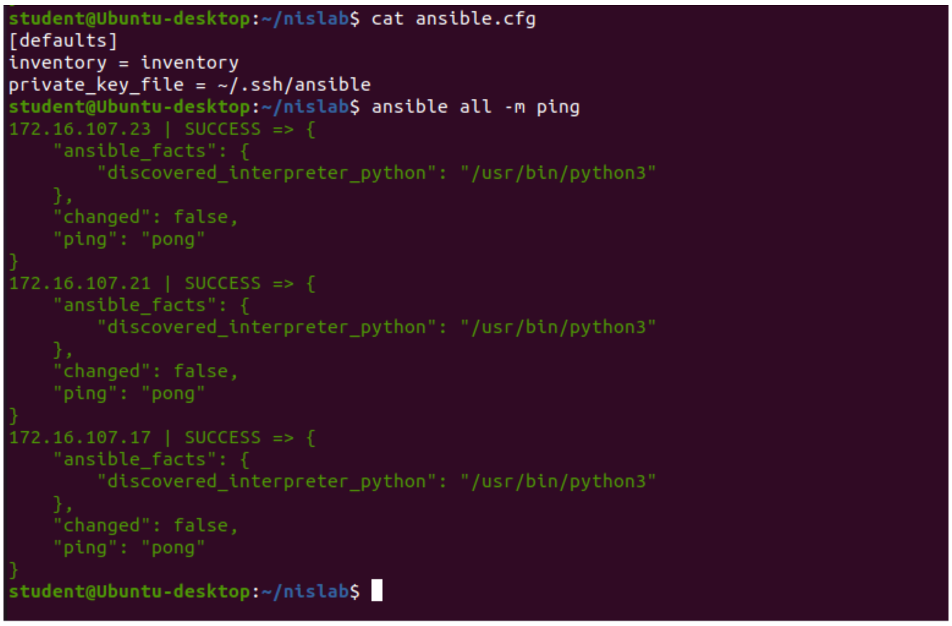


```
student@Ubuntu-desktop:~/nislab$ ansible all --key-file ~/.ssh/ansible -i inventory -m ping
172.16.107.17 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.21 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.23 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
student@Ubuntu-desktop:~/nislab$
```

Figure 32: ssh all server by one command.

all: command will run on all servers
/.ssh/ansible: key used to connect to the servers
-i inventory: server IP list
-m: module to use in this case ping

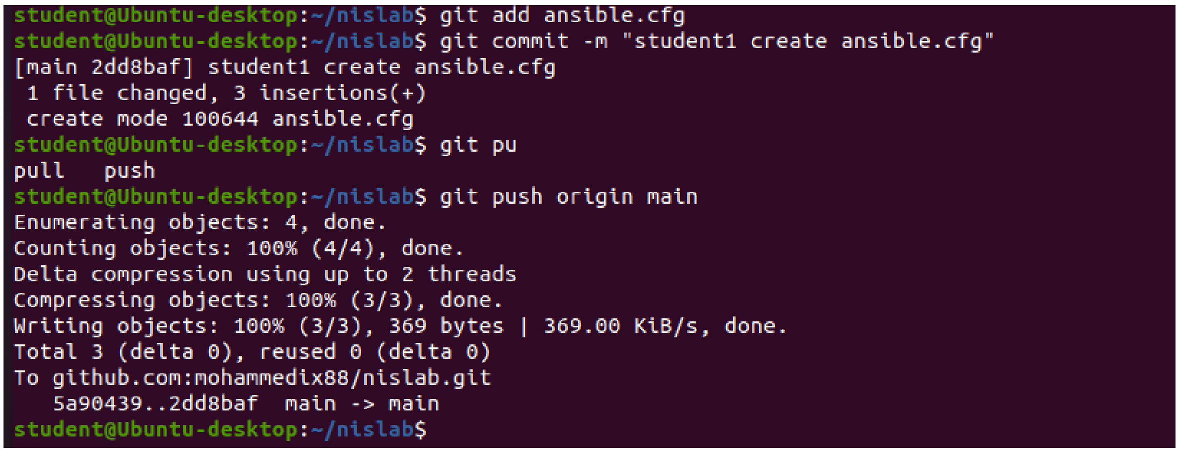
6. The last command was too long to make it shorter we will store the inventory file name and key to be used in a configuration file after that, let's run the new version of the previous ansible ping command, as shown in figure 33.



```
student@Ubuntu-desktop:~/nislab$ cat ansible.cfg
[defaults]
inventory = inventory
private_key_file = ~/.ssh/ansible
student@Ubuntu-desktop:~/nislab$ ansible all -m ping
172.16.107.23 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.21 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.17 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
student@Ubuntu-desktop:~/nislab$
```

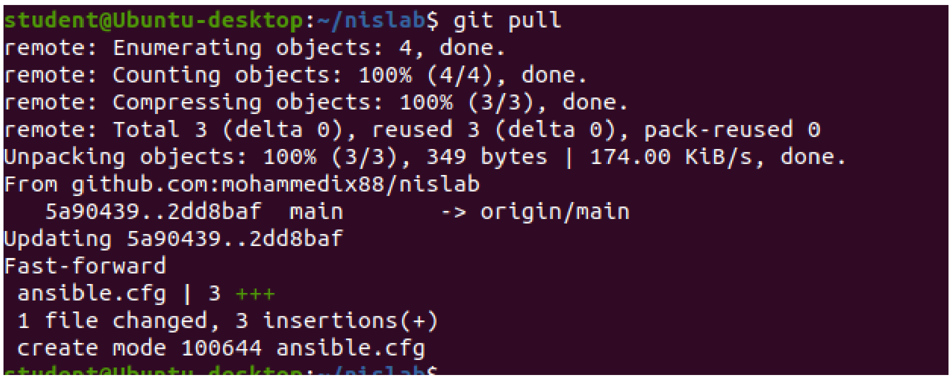
Figure 33: create cfg file and ssh using new version

7. Then we Push the files to GitHub and pull them on workstation2 and run the previous command to test. As shown in figure 34& 35 and 36.



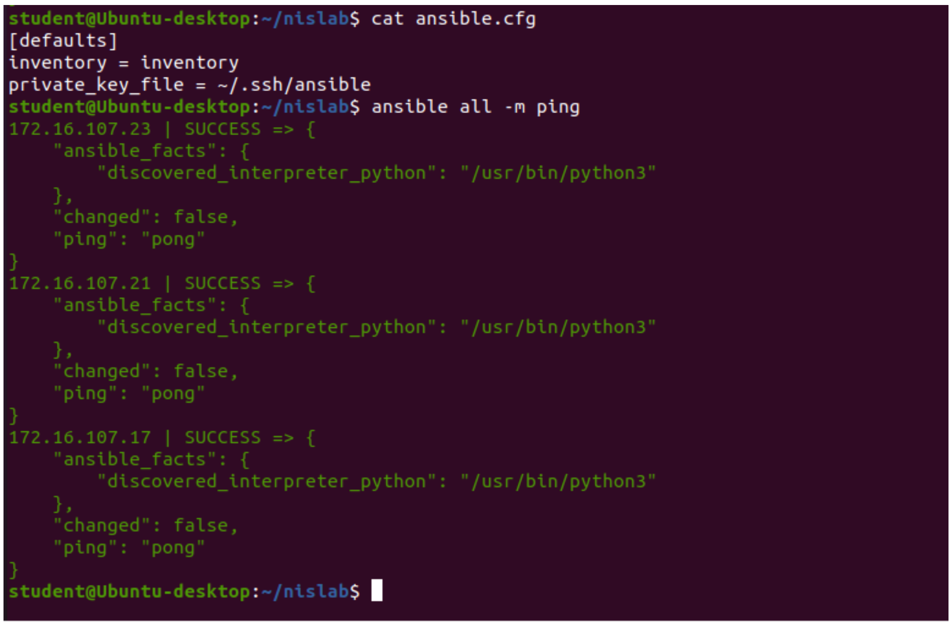
```
student@Ubuntu-desktop:~/nislab$ git add ansible.cfg
student@Ubuntu-desktop:~/nislab$ git commit -m "student1 create ansible.cfg"
[main 2dd8baf] student1 create ansible.cfg
 1 file changed, 3 insertions(+)
 create mode 100644 ansible.cfg
student@Ubuntu-desktop:~/nislab$ git pu
pull   push
student@Ubuntu-desktop:~/nislab$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 369 bytes | 369.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:mohammedix88/nislab.git
   5a90439..2dd8baf  main -> main
student@Ubuntu-desktop:~/nislab$
```

Figure 34: Push the files to GitHub



```
student@Ubuntu-desktop:~/nislab$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 349 bytes | 174.00 KiB/s, done.
From github.com:mohammedix88/nislab
   5a90439..2dd8baf  main       -> origin/main
Updating 5a90439..2dd8baf
Fast-forward
 ansible.cfg | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 ansible.cfg
```

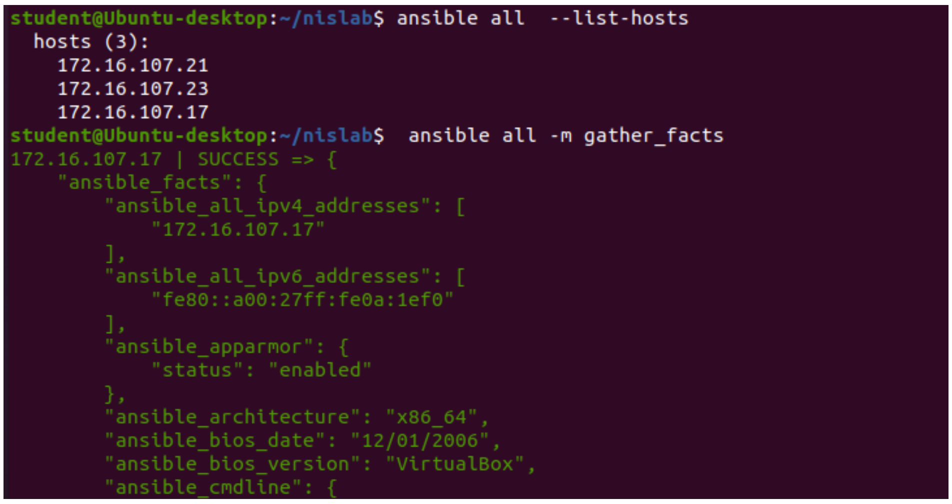
Figure 35: Pull file from Workstation2



```
student@Ubuntu-desktop:~/nislab$ cat ansible.cfg
[defaults]
inventory = inventory
private_key_file = ~/.ssh/ansible
student@Ubuntu-desktop:~/nislab$ ansible all -m ping
172.16.107.23 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.21 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.16.107.17 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
student@Ubuntu-desktop:~/nislab$
```

Figure 36: run the previous command in WK2

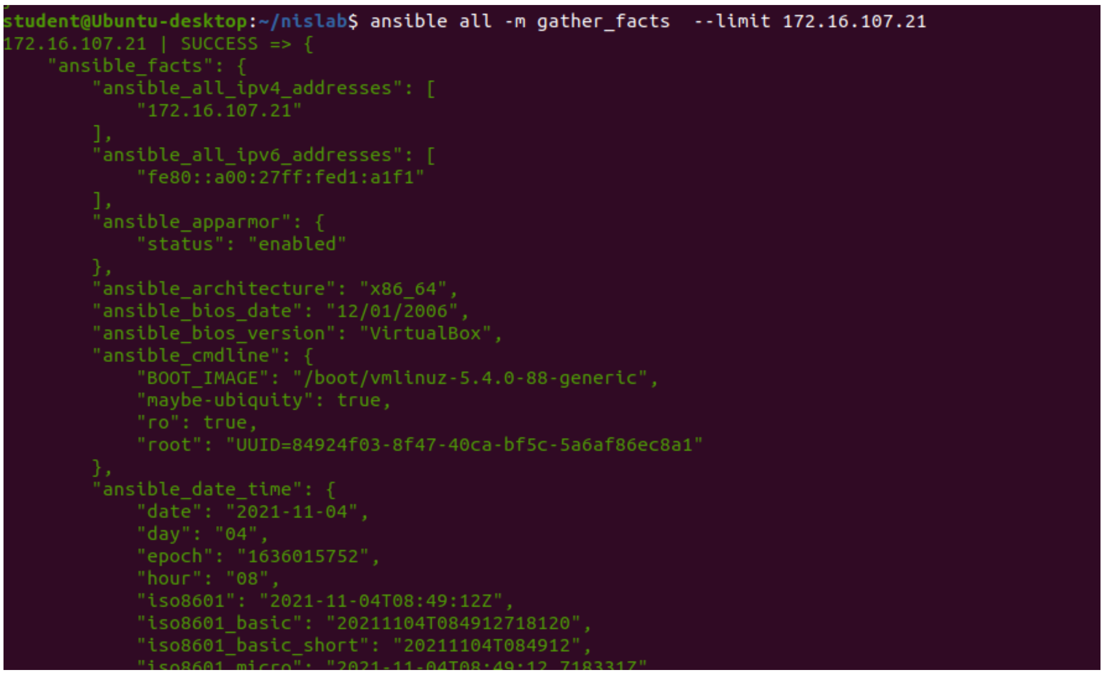
8. Then we will list the current hosts we are managing with ansible and information gathering about hosts we are managing. As shown in figure 37.



```
student@Ubuntu-desktop:~/nislab$ ansible all  --list-hosts
  hosts (3):
    172.16.107.21
    172.16.107.23
    172.16.107.17
student@Ubuntu-desktop:~/nislab$  ansible all -m gather_facts
172.16.107.17 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.16.107.17"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::a00:27ff:fe0a:1ef0"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/01/2006",
        "ansible_bios_version": "VirtualBox",
        "ansible_cmdline": {
```

Figure 37: List all host & information about it

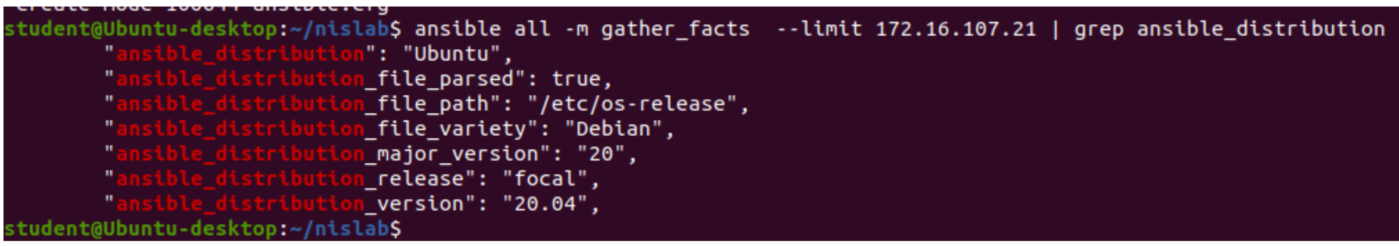
9. The output of the previous command is overwhelming but we can limit that using the –limit option. As shown in figure 39. the output of this commands contains all information we can use in ansible when we use the “when” conditional if we use playbooks to target certain hosts in our inventory with commands that we want to limit to a certain OS or distribution for example.



```
student@Ubuntu-desktop:~/nislab$ ansible all -m gather_facts  --limit 172.16.107.21
172.16.107.21 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.16.107.21"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::a00:27ff:fed1:a1f1"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/01/2006",
        "ansible_bios_version": "VirtualBox",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-5.4.0-88-generic",
            "maybe-ubiquity": true,
            "ro": true,
            "root": "UUID=84924f03-8f47-40ca-bf5c-5a6af86ec8a1"
        },
        "ansible_date_time": {
            "date": "2021-11-04",
            "day": "04",
            "epoch": "1636015752",
            "hour": "08",
            "iso8601": "2021-11-04T08:49:12Z",
            "iso8601_basic": "20211104T084912718120",
            "iso8601_basic_short": "20211104T084912",
            "iso8601_micro": "2021-11-04T08:49:12.718331Z"
```

Figure 38: Gather information for specific host

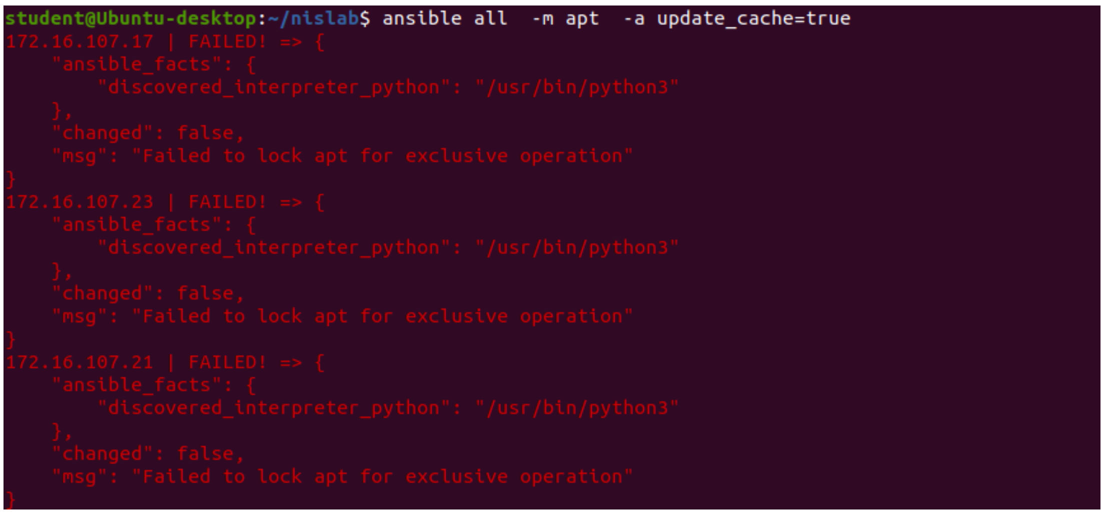
10. Then, we do the following command in workstation2 as shown in figure 41 to get the OS distribution running on our SRVR01 server.



```
student@Ubuntu-desktop:~/nislab$ ansible all -m gather_facts  --limit 172.16.107.21 | grep ansible_distribution
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/os-release",
        "ansible_distribution_file_variety": "Debian",
        "ansible_distribution_major_version": "20",
        "ansible_distribution_release": "focal",
        "ansible_distribution_version": "20.04",
student@Ubuntu-desktop:~/nislab$
```

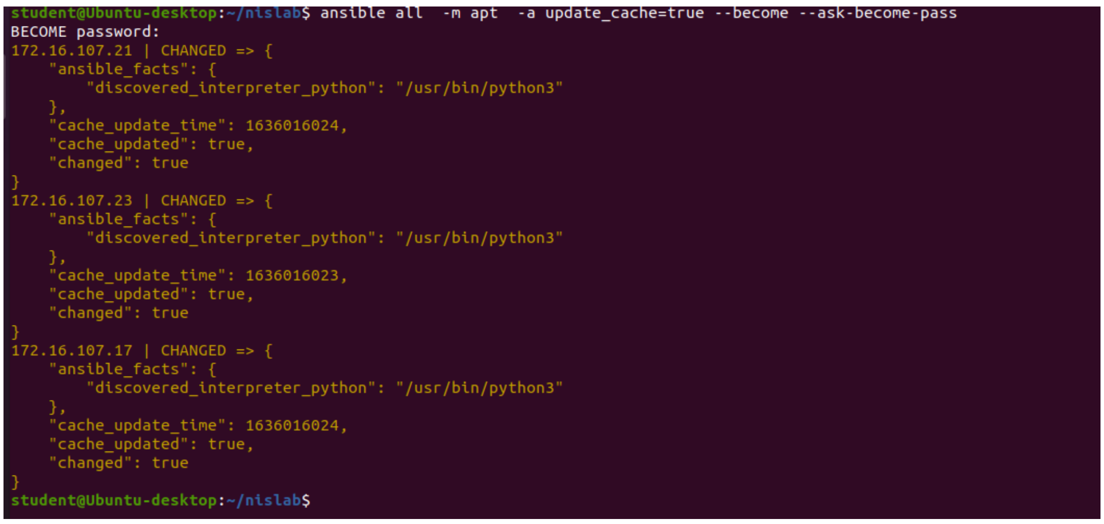
Figure 39: OS distribution for SRVR01

11. Now use the apt module to update the repository index on all servers. As shown in figure 41. So we conclude that apt update command can't be run like that it should be sudo apt update. So we need to elevate the privileges of the command when ansible runs it as shown in figure 42.



```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a update_cache=true
172.16.107.17 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
172.16.107.23 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
172.16.107.21 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation"
}
```

Figure 40: apt module withot root

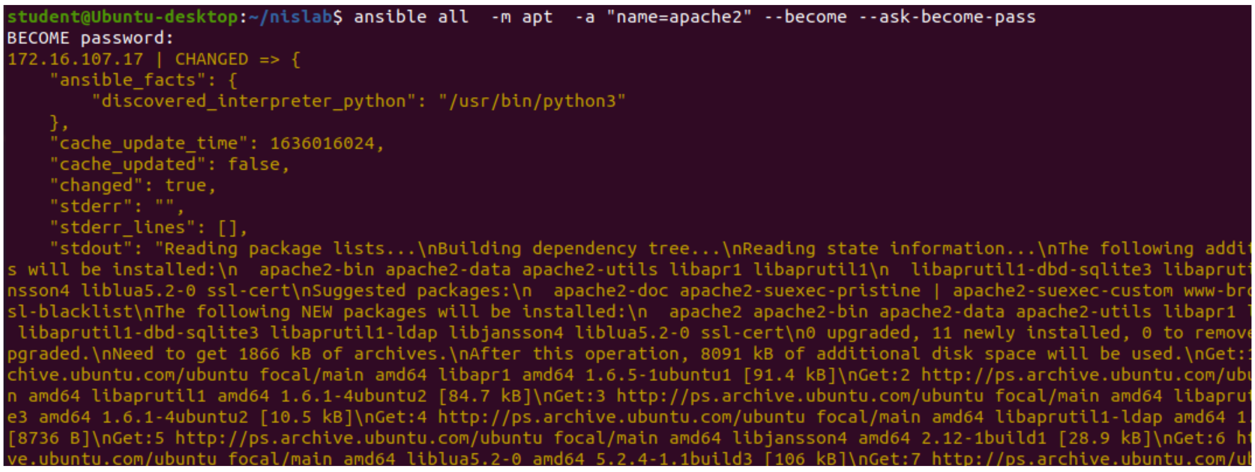


```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a update_cache=true --become --ask-become-pass
BECOME password:
172.16.107.21 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016024,
    "cache_updated": true,
    "changed": true
}
172.16.107.23 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016023,
    "cache_updated": true,
    "changed": true
}
172.16.107.17 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016024,
    "cache_updated": true,
    "changed": true
}
student@Ubuntu-desktop:~/nislab$
```

Figure 41: apt module with root

-m apt: same as apt **-a update_cache=true** : same as update –**become:** same as sudo –**ask-become-pass:** so that ansible asks us for the sudo password.

12. Now we will install apache package in all server using one command as shown in figure 42. this should install Apache web server on all 3 servers. Then we open the web browser and open http://[SRVR01_IP] as figure 43 show.



```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a "name=apache2" --become --ask-become-pass
BECOME password:
172.16.107.17 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016024,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following addi
s will be installed:\n  apache2-bin apache2-data apache2-utils libapr1 libaprutil1\n  libaprutil1-dbd-sqlite3 libaprut
nsson4 liblua5.2-0 ssl-cert\nSuggested packages:\n  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-br
sl-blacklist\nThe following NEW packages will be installed:\n  apache2 apache2-bin apache2-data apache2-utils libapr1 
 libaprutil1-dbd-sqlite3 libaprutil1-ldap libjansson4 liblua5.2-0 ssl-cert\n0 upgraded, 11 newly installed, 0 to remove
pgraded.\nNeed to get 1866 kB of archives.\nAfter this operation, 8091 kB of additional disk space will be used.\nGet:
chive.ubuntu.com/ubuntu focal/main amd64 libapr1 amd64 1.6.5-1ubuntu1 [91.4 kB]\nGet:2 http://ps.archive.ubuntu.com/ub
n amd64 libaprutil1 amd64 1.6.1-4ubuntu2 [84.7 kB]\nGet:3 http://ps.archive.ubuntu.com/ubuntu focal/main amd64 libapru
e3 amd64 1.6.1-4ubuntu2 [10.5 kB]\nGet:4 http://ps.archive.ubuntu.com/ubuntu focal/main amd64 libaprutil1-ldap amd64 1
[8736 B]\nGet:5 http://ps.archive.ubuntu.com/ubuntu focal/main amd64 libjansson4 amd64 2.12-1build1 [28.9 kB]\nGet:6 h
ve.ubuntu.com/ubuntu focal/main amd64 liblua5.2-0 amd64 5.2.4-1.1build3 [106 kB]\nGet:7 http://ps.archive.ubuntu.com/u
```

Figure 42: install apache in all server

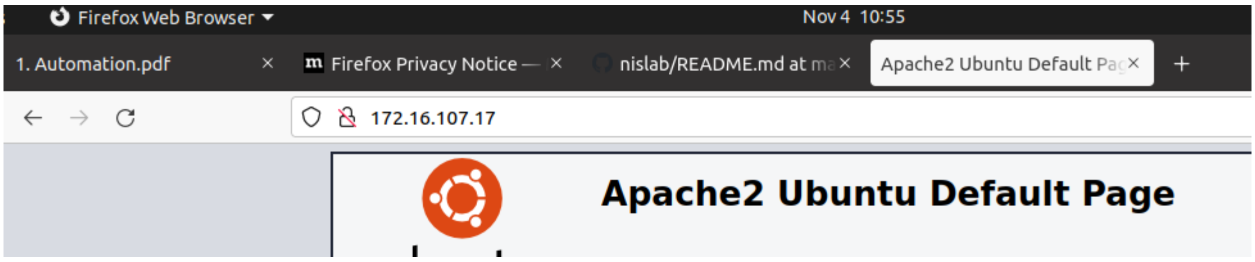
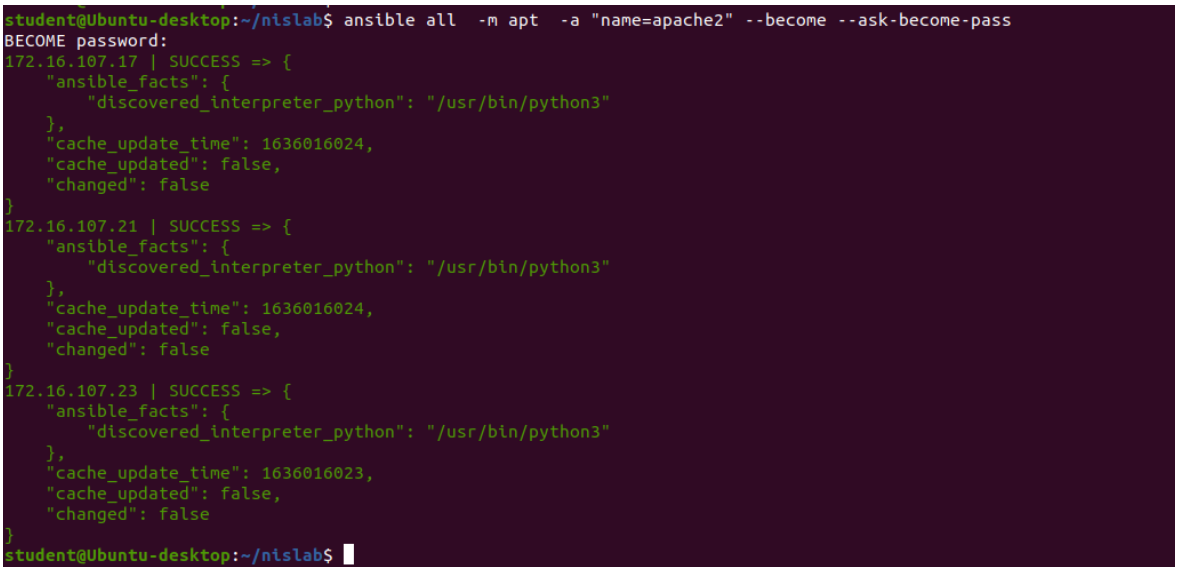


Figure 43: open apache web on server1

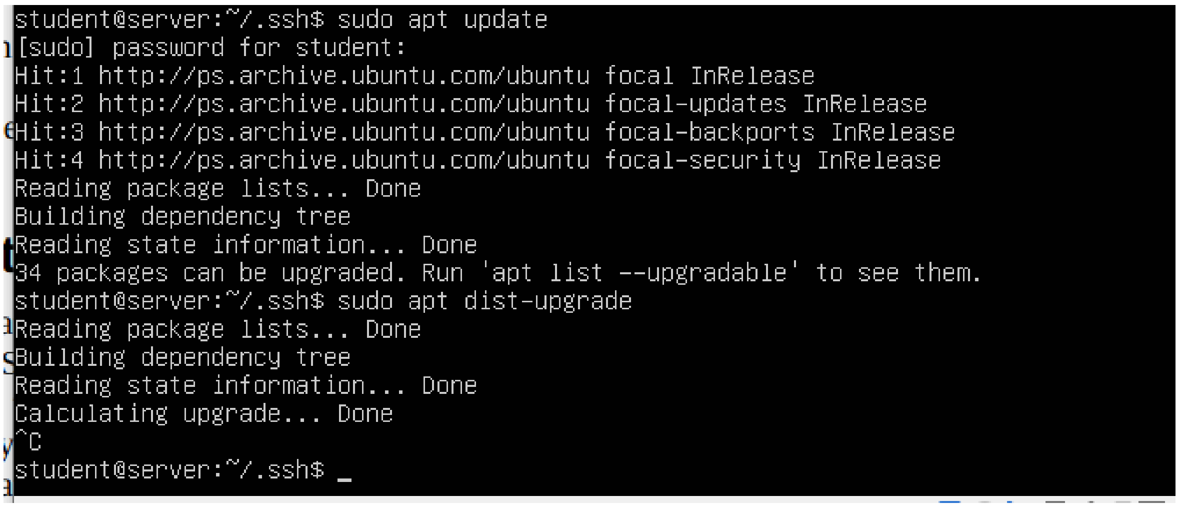
13. We repeat the last command and notice that the output of the command will give a no change message. This command will install a package if it's absent but will not update that package if it has an update. As shown in figure 44.



```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a "name=apache2" --become --ask-become-pass
BECOME password:
172.16.107.17 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016024,
    "cache_updated": false,
    "changed": false
}
172.16.107.21 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016024,
    "cache_updated": false,
    "changed": false
}
172.16.107.23 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016023,
    "cache_updated": false,
    "changed": false
}
student@Ubuntu-desktop:~/nislab$ 
```

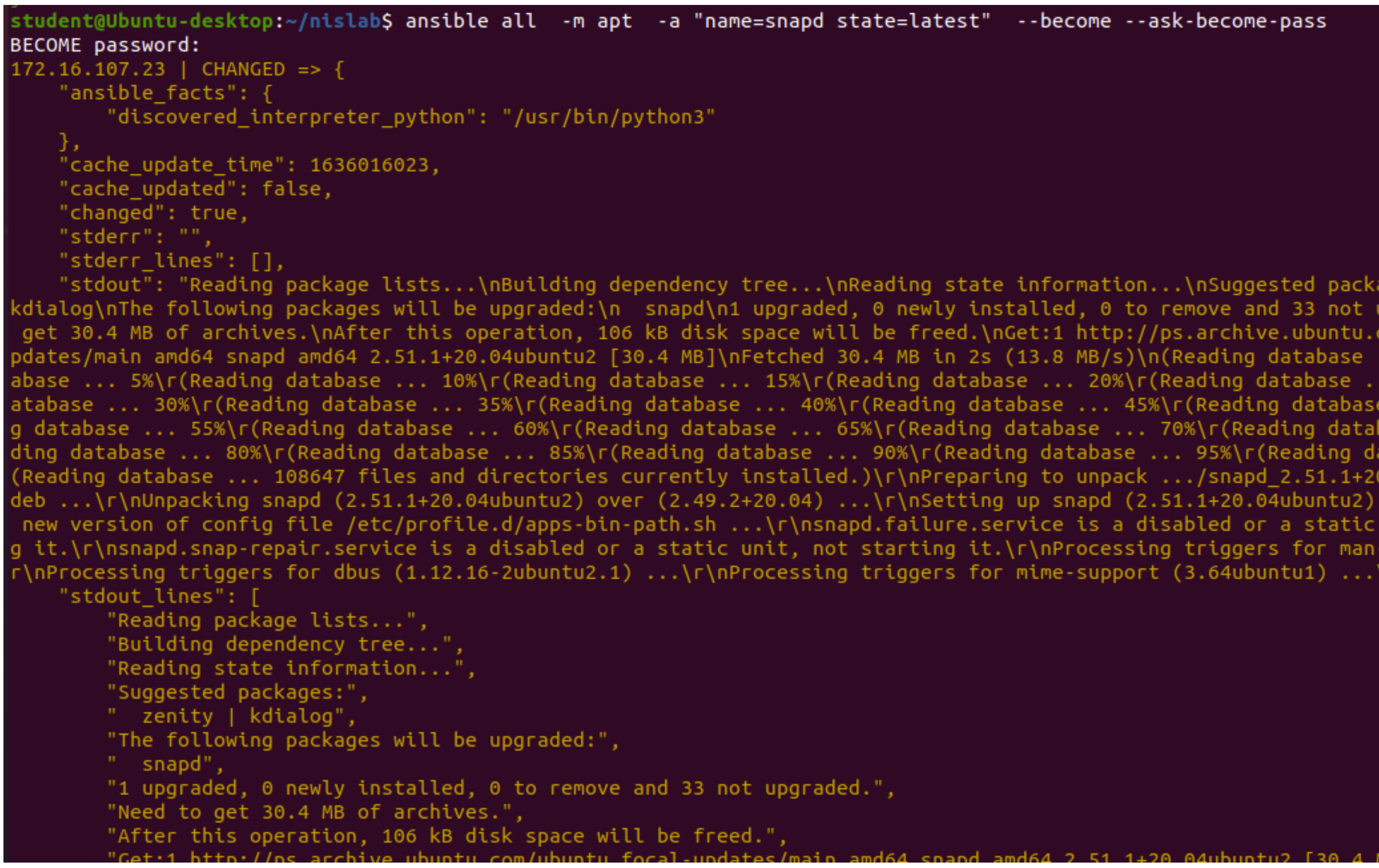
Figure 44: repeat the install command

14. Then, we check if there is any possible update for SRV01 as shown in figure 45, as we see that there is an update and upgrade for that server, so we update all servers using one command as shown in figure 46 after that we repeat the command in the SRV01, to check if it updates or not as figure 47 shows that the update done but there is an upgrade yet So we do the following command in workstation1 as shown in figure 48 to upgrade all server in the end, we verify that all package is up to date in SRV01 as shown in figure 49.



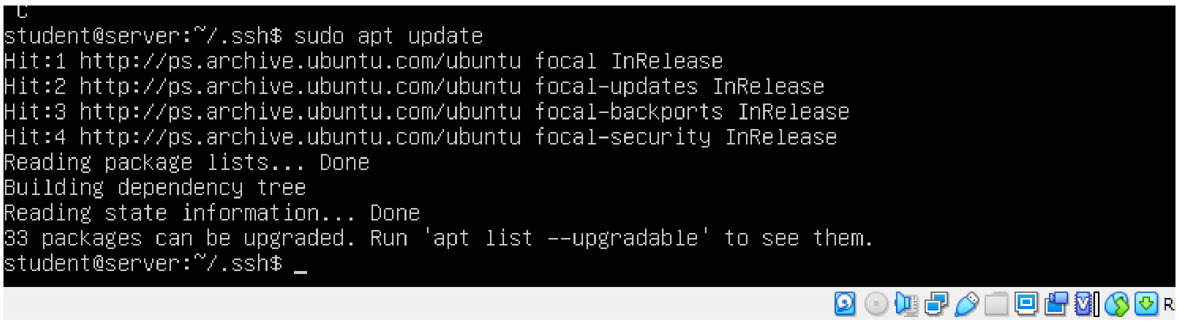
```
student@server:~/.ssh$ sudo apt update
[sudo] password for student:
Hit:1 http://ps.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://ps.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://ps.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://ps.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
34 packages can be upgraded. Run 'apt list --upgradable' to see them.
student@server:~/.ssh$ sudo apt dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
^C
student@server:~/.ssh$ _
```

Figure 45: Check for update on SRV01



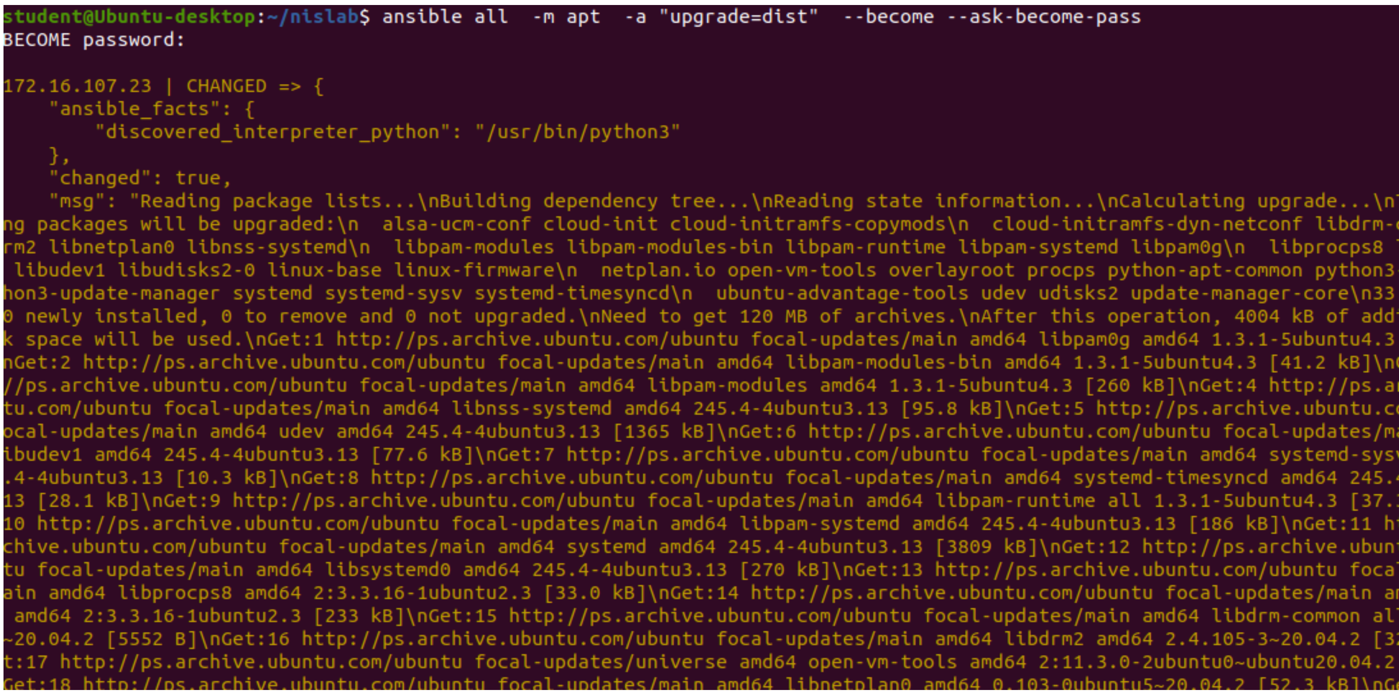
```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a "name=snapd state=latest"  --become --ask-become-pass
BECOME password:
172.16.107.23 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1636016023,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nSuggested pack
kdialog\nThe following packages will be upgraded:\n  snapd\n1 upgraded, 0 newly installed, 0 to remove and 33 not
 get 30.4 MB of archives.\nAfter this operation, 106 kB disk space will be freed.\nGet:1 http://ps.archive.ubuntu.
pdates/main amd64 snapd amd64 2.51.1+20.04ubuntu2 [30.4 MB]\nFetched 30.4 MB in 2s (13.8 MB/s)\n(Reading database
abase ... 5%\r(Reading database ... 10%\r(Reading database ... 15%\r(Reading database ... 20%\r(Reading database .
atabase ... 30%\r(Reading database ... 35%\r(Reading database ... 40%\r(Reading database ... 45%\r(Reading databas
g database ... 55%\r(Reading database ... 60%\r(Reading database ... 65%\r(Reading database ... 70%\r(Reading data
ding database ... 80%\r(Reading database ... 85%\r(Reading database ... 90%\r(Reading database ... 95%\r(Reading d
(Reading database ... 108647 files and directories currently installed.)\r\nPreparing to unpack .../snapd_2.51.1+2
deb ...\r\nUnpacking snapd (2.51.1+20.04ubuntu2) over (2.49.2+20.04) ...\r\nSetting up snapd (2.51.1+20.04ubuntu2)
 new version of config file /etc/profile.d/apps-bin-path.sh ...\r\nsnapd.failure.service is a disabled or a static
g it.\r\nsnapd.snap-repair.service is a disabled or a static unit, not starting it.\r\nProcessing triggers for man
r\nProcessing triggers for dbus (1.12.16-2ubuntu2.1) ...\r\nProcessing triggers for mime-support (3.64ubuntu1) ...
    "stdout_lines": [
        "Reading package lists...",
        "Building dependency tree...",
        "Reading state information...",
        "Suggested packages:",
        "  zenity | kdialog",
        "The following packages will be upgraded:",
        "  snapd",
        "1 upgraded, 0 newly installed, 0 to remove and 33 not upgraded.",
        "Need to get 30.4 MB of archives.",
        "After this operation, 106 kB disk space will be freed.",
        "Get:1 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 snapd amd64 2.51.1+20.04ubuntu2 [30.4
```

Figure 46: Update all servers



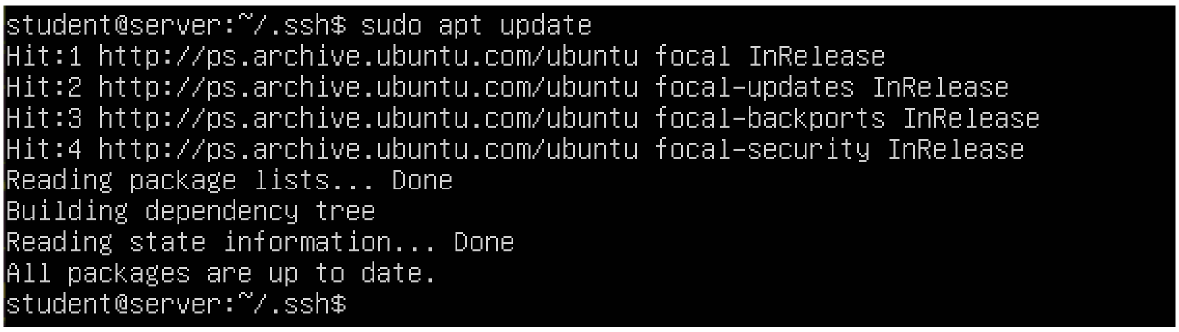
```
C
student@server:~/.ssh$ sudo apt update
Hit:1 http://ps.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://ps.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://ps.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://ps.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
33 packages can be upgraded. Run 'apt list --upgradable' to see them.
student@server:~/.ssh$ _
```

Figure 47: Check for update & upgrade on SRV01



```
student@Ubuntu-desktop:~/nislab$ ansible all  -m apt  -a "upgrade=dist"  --become --ask-become-pass
BECOME password:

172.16.107.23 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "msg": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nCalculating upgrade...\n
ng packages will be upgraded:\n  alsa-ucm-conf cloud-init cloud-initramfs-copymods\n  cloud-initramfs-dyn-netconf libdrm-c
rm2 libnetplan0 libnss-systemd\n  libpam-modules libpam-modules-bin libpam-runtime libpam-systemd libpam0g\n  libprocps8 
 libudev1 libudisks2-0 linux-base linux-firmware\n  netplan.io open-vm-tools overlayroot procps python-apt-common python3
hon3-update-manager systemd systemd-sysv systemd-timesyncd\n  ubuntu-advantage-tools udev udisks2 update-manager-core\n33
0 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 120 MB of archives.\nAfter this operation, 4004 kB of add
k space will be used.\nGet:1 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam0g amd64 1.3.1-5ubuntu4.3
nGet:2 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-modules-bin amd64 1.3.1-5ubuntu4.3 [41.2 kB]\n
//ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-modules amd64 1.3.1-5ubuntu4.3 [260 kB]\nGet:4 http://ps.a
tu.com/ubuntu focal-updates/main amd64 libnss-systemd amd64 245.4-4ubuntu3.13 [95.8 kB]\nGet:5 http://ps.archive.ubuntu.c
ocal-updates/main amd64 udev amd64 245.4-4ubuntu3.13 [1365 kB]\nGet:6 http://ps.archive.ubuntu.com/ubuntu focal-updates/ma
ibudev1 amd64 245.4-4ubuntu3.13 [77.6 kB]\nGet:7 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 systemd-sysv
.4-4ubuntu3.13 [10.3 kB]\nGet:8 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 systemd-timesyncd amd64 245.
13 [28.1 kB]\nGet:9 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-runtime all 1.3.1-5ubuntu4.3 [37.
10 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-systemd amd64 245.4-4ubuntu3.13 [186 kB]\nGet:11 h
chive.ubuntu.com/ubuntu focal-updates/main amd64 systemd amd64 245.4-4ubuntu3.13 [3809 kB]\nGet:12 http://ps.archive.ubun
tu focal-updates/main amd64 libsystemd0 amd64 245.4-4ubuntu3.13 [270 kB]\nGet:13 http://ps.archive.ubuntu.com/ubuntu foca
ain amd64 libprocps8 amd64 2:3.3.16-1ubuntu2.3 [33.0 kB]\nGet:14 http://ps.archive.ubuntu.com/ubuntu focal-updates/main a
 amd64 2:3.3.16-1ubuntu2.3 [233 kB]\nGet:15 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libdrm-common al
~20.04.2 [5552 B]\nGet:16 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libdrm2 amd64 2.4.105-3~20.04.2 [3
t:17 http://ps.archive.ubuntu.com/ubuntu focal-updates/universe amd64 open-vm-tools amd64 2:11.3.0-2ubuntu0~ubuntu20.04.2
Get:18 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnetplan0 amd64 0.103-0ubuntu5~20.04.2 [52.3 kB]\nG
```

Figure 48: Upgrade all servers



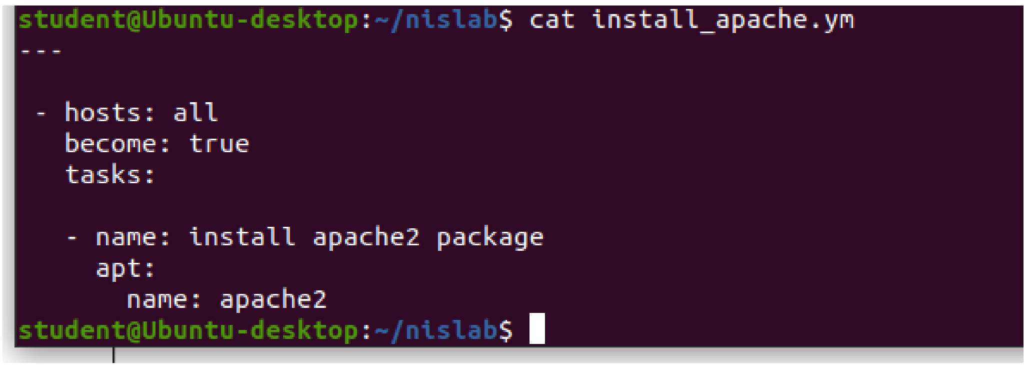
```
student@server:~/.ssh$ sudo apt update
Hit:1 http://ps.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://ps.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://ps.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://ps.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
student@server:~/.ssh$
```

Figure 49: Check for update & upgrade on SRV01

5 Section 2

5.1 Introduction to Playbooks

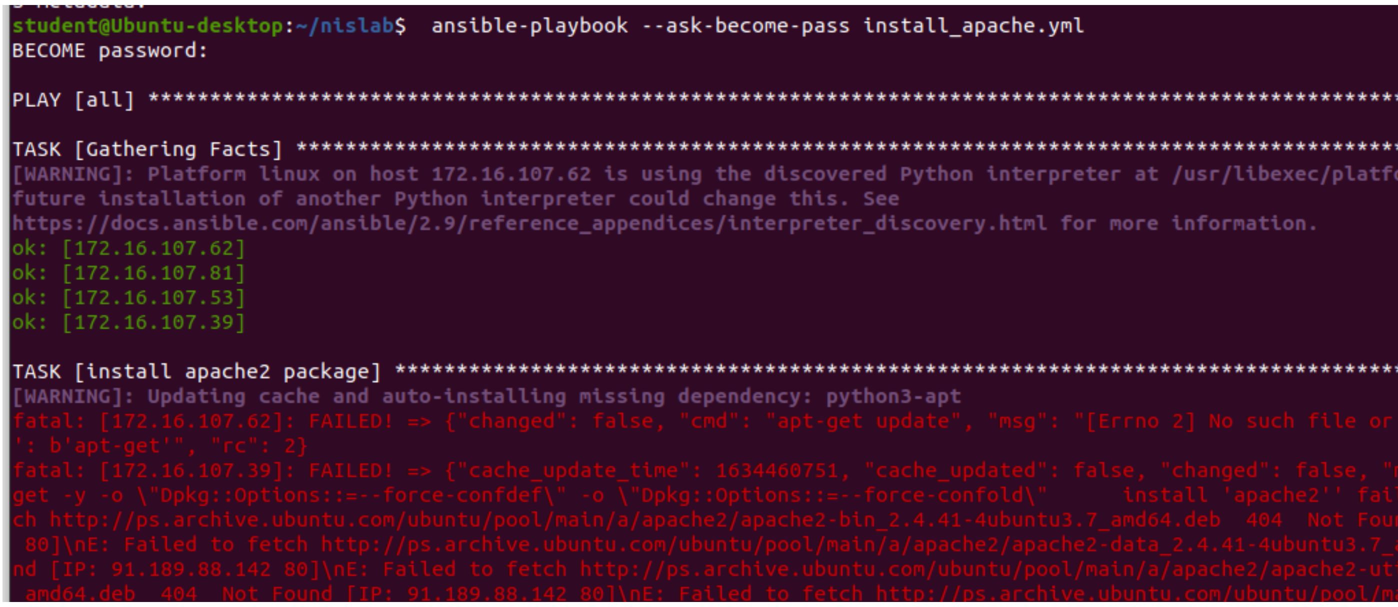
1. At first, we create a file configuration using YAML languge called **install_apache.yml** in the nislab directory on workstation1 as figure 50 show the contents.
   The structure of the file is as follows:
   — : The start of the file.
   - hosts: all : which hosts will be affected by the plays.
   become: true: for sudo.
   tasks: : means after this will be the list of plays to be executed.
   - name:install apache2 package : the name (description) of this play.
   apt: The module to be executed , here it's apt
   name: apache2 : the name of the package we want to install.



```
student@Ubuntu-desktop:~/nislab$ cat install_apache.ym
---

 - hosts: all
   become: true
   tasks:

   - name: install apache2 package
     apt:
       name: apache2
student@Ubuntu-desktop:~/nislab$ 
```

Figure 50: PlayBook File.

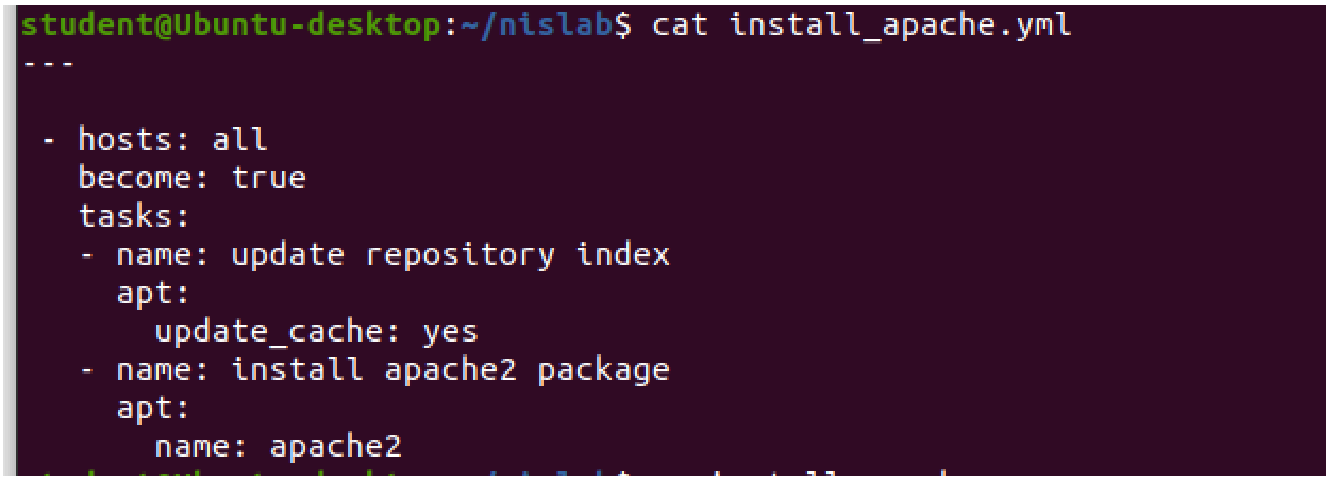
2. Then we run the yml file use the following command: **ansibleplaybook ask-become-pass install_apache.yml** as figure 51 show
   The previous playbook failed depending on the repository index status. On Linux systems, we need to update the repository index before trying to install packages; because we might get an error that ”the packages not found”.



```
student@Ubuntu-desktop:~/nislab$  ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ****************************************************************************
[WARNING]: Platform linux on host 172.16.107.62 is using the discovered Python interpreter at /usr/libexec/platf
future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [172.16.107.62]
ok: [172.16.107.81]
ok: [172.16.107.53]
ok: [172.16.107.39]

TASK [install apache2 package] ********************************************************************
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
fatal: [172.16.107.62]: FAILED! => {"changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or
': b'apt-get'", "rc": 2}
fatal: [172.16.107.39]: FAILED! => {"cache_update_time": 1634460751, "cache_updated": false, "changed": false, "
get -y -o \"Dpkg::Options::=--force-confdef\" -o \"Dpkg::Options::=--force-confold\"      install 'apache2'' fai
ch http://ps.archive.ubuntu.com/ubuntu/pool/main/a/apache2/apache2-bin_2.4.41-4ubuntu3.7_amd64.deb  404  Not Fou
 80]\nE: Failed to fetch http://ps.archive.ubuntu.com/ubuntu/pool/main/a/apache2/apache2-data_2.4.41-4ubuntu3.7_
nd [IP: 91.189.88.142 80]\nE: Failed to fetch http://ps.archive.ubuntu.com/ubuntu/pool/main/a/apache2/apache2-ut
 amd64.deb  404  Not Found [IP: 91.189.88.142 80]\nE: Failed to fetch http://ps.archive.ubuntu.com/ubuntu/pool/m
```

Figure 51: Run yml file.

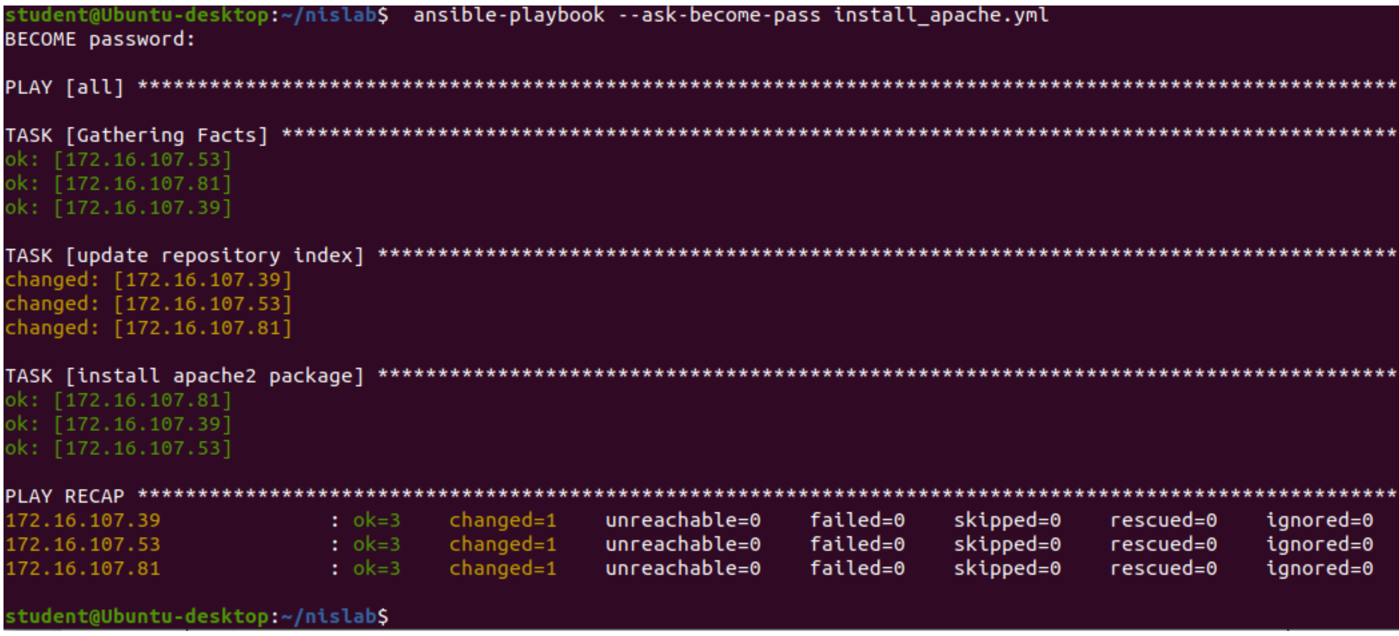
3. After that, we add the task **update_cache: yes** in ansible that equal apt update as figure 52 shown.



```
student@Ubuntu-desktop:~/nislab$ cat install_apache.yml
---

 - hosts: all
   become: true
   tasks:
   - name: update repository index
     apt:
       update_cache: yes
   - name: install apache2 package
     apt:
       name: apache2
```

Figure 52: PlayBook File after add update.

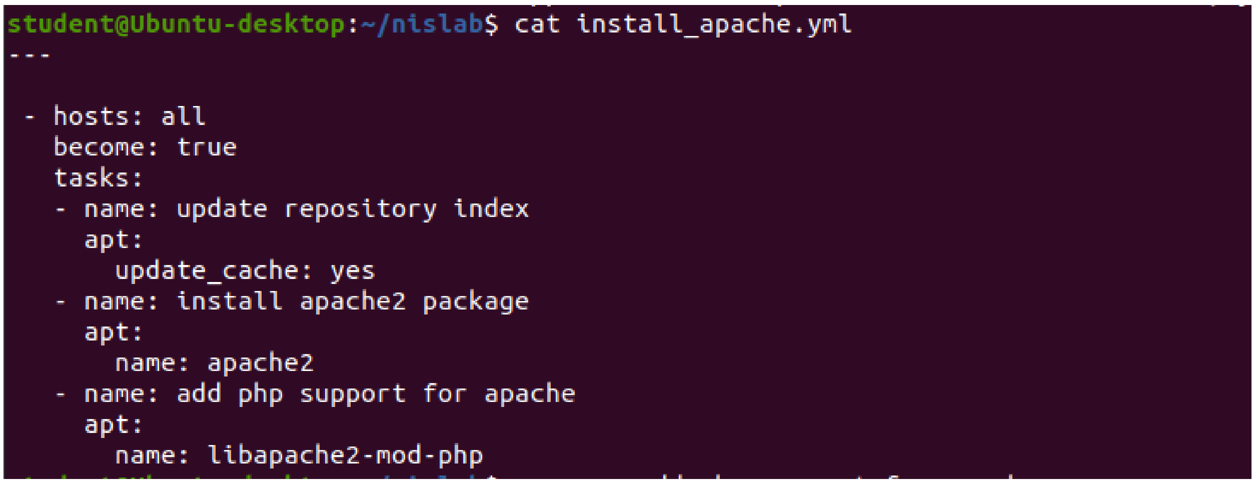
4. Then we run the playbook and notice the tasks that are executed successfully. **ok=3** which are the gathering fact,update repository, and install apache2 package tasks as figure 53 shown.



```
student@Ubuntu-desktop:~/nislab$  ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

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

TASK [Gathering Facts] ************************************************************************************
ok: [172.16.107.53]
ok: [172.16.107.81]
ok: [172.16.107.39]

TASK [update repository index] ****************************************************************************
changed: [172.16.107.39]
changed: [172.16.107.53]
changed: [172.16.107.81]

TASK [install apache2 package] ****************************************************************************
ok: [172.16.107.81]
ok: [172.16.107.39]
ok: [172.16.107.53]

PLAY RECAP ************************************************************************************************
172.16.107.39              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.53              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.81              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

student@Ubuntu-desktop:~/nislab$
```

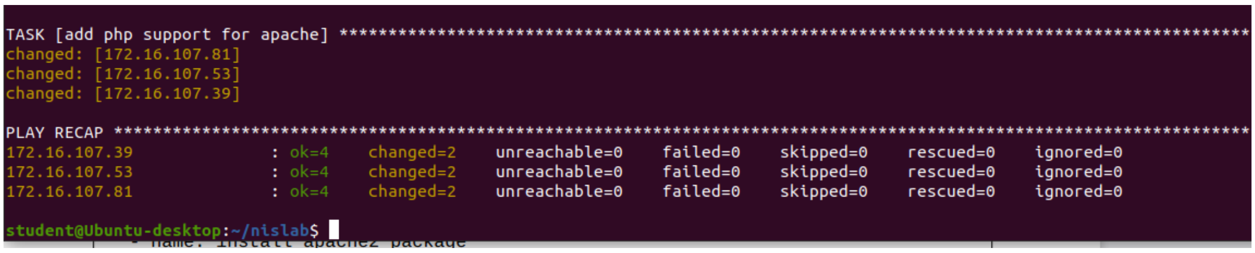
Figure 53: Run yml file.

5. We added php support to the Apache server as figure 54 show then we run the playbook to make changes as figure 55 show. This playbook will install apache2 and libapache2mod-php packages if they are not installed but it won't update them if there are updates available. To make the playbook capable of updating packages we need to use the **state** parameter as figure56 state: latest will make sure the package is always the latest one available.



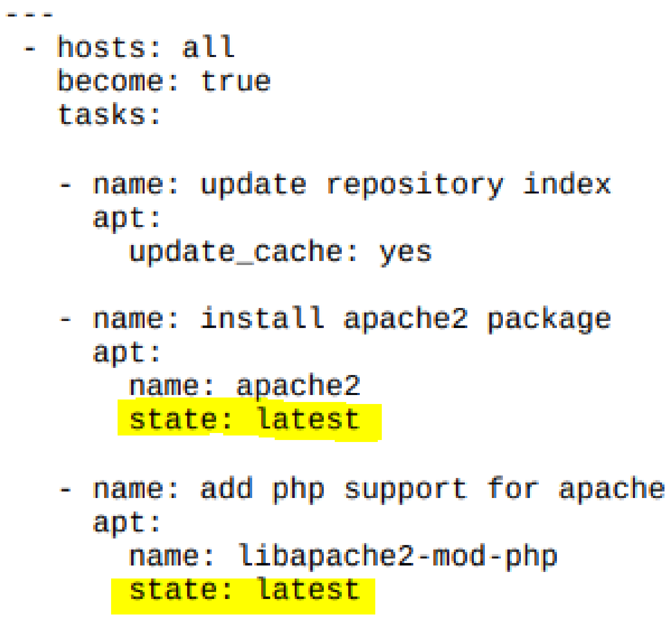
```
student@Ubuntu-desktop:~/nislab$ cat install_apache.yml
---

 - 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
```

Figure 54: Add php support to playbook file.



```
TASK [add php support for apache] **************************************************************************************
changed: [172.16.107.81]
changed: [172.16.107.53]
changed: [172.16.107.39]

PLAY RECAP *************************************************************************************************************
172.16.107.39              : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.53              : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.81              : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

student@Ubuntu-desktop:~/nislab$
```

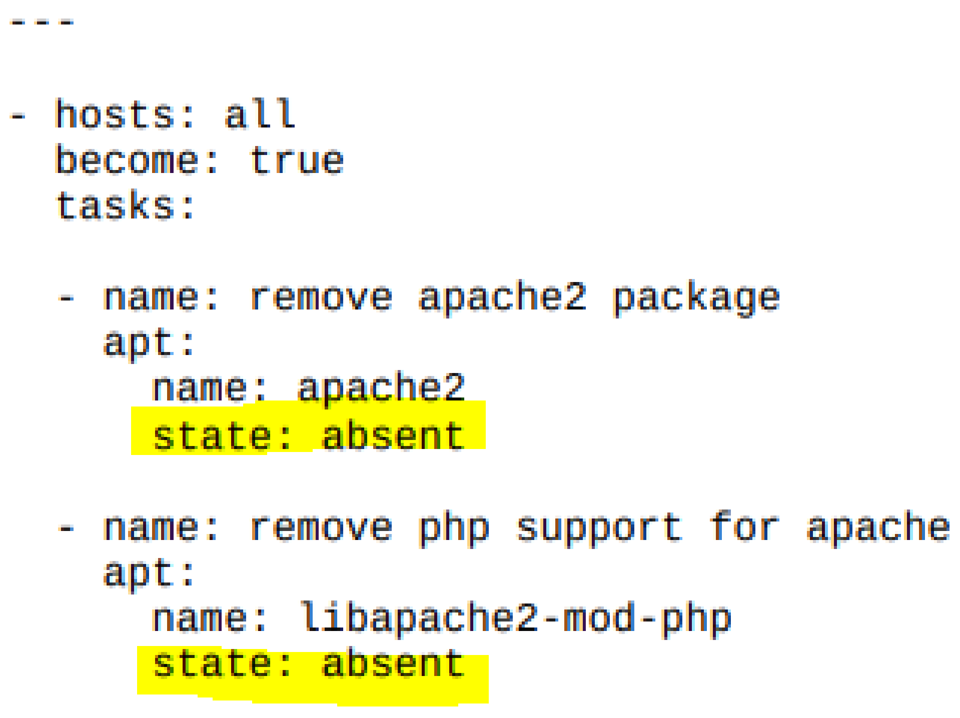
Figure 55: Run yml file.



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

   - name: update repository index
     apt:
       update_cache: yes

   - name: install apache2 package
     apt:
       name: apache2
       state: latest

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

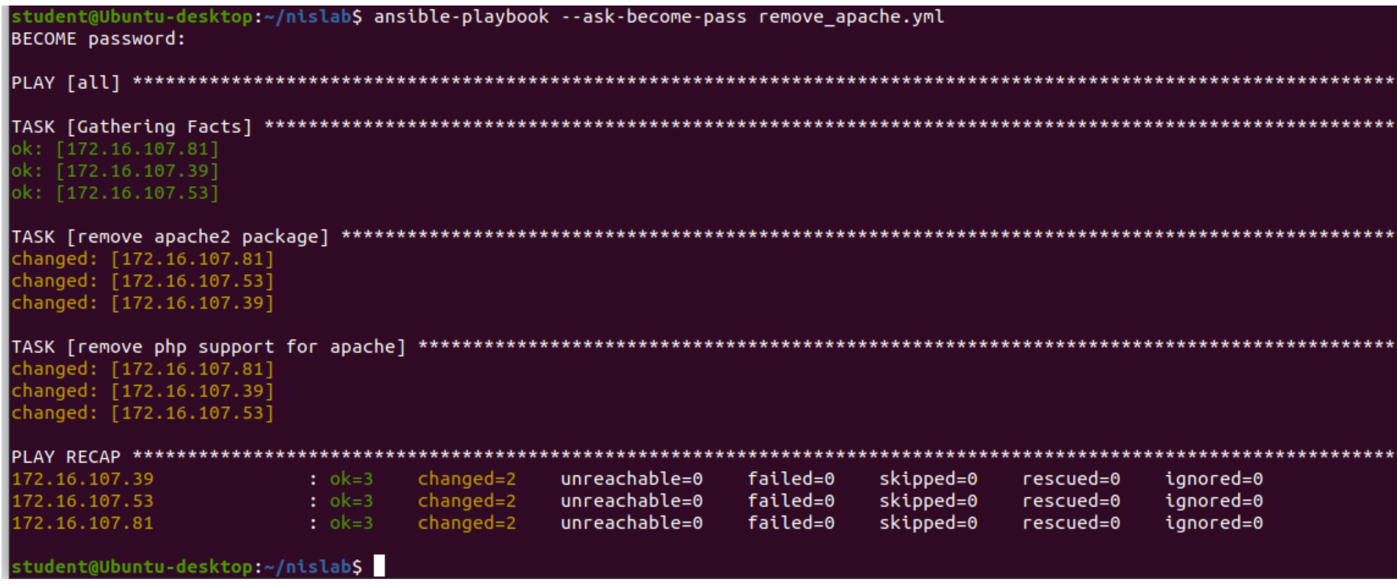
Figure 56: Add state to playbook file.

6. Also, we create another playbook that removes these packages the file called **remove_apache.yml** the **state: absent** parameter value means removing the package if present as figure 57 shown then run the playbook as figure 58 show.



```
---

- hosts: all
  become: true
  tasks:

  - name: remove apache2 package
    apt:
      name: apache2
      state: absent

  - name: remove php support for apache
    apt:
      name: libapache2-mod-php
      state: absent
```

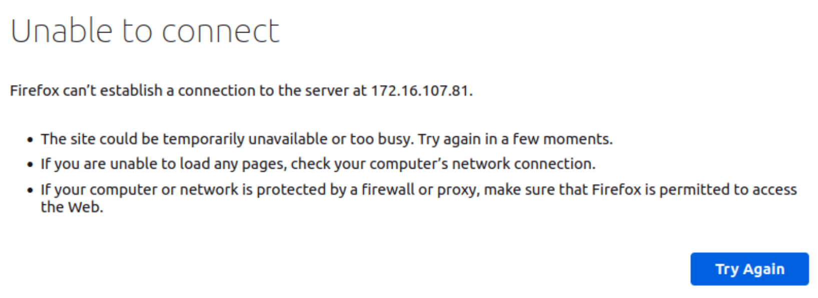
Figure 57: Remove package.



```
student@Ubuntu-desktop:~/nislab$ ansible-playbook --ask-become-pass remove_apache.yml
BECOME password:

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

TASK [Gathering Facts] ************************************************************************************************
ok: [172.16.107.81]
ok: [172.16.107.39]
ok: [172.16.107.53]

TASK [remove apache2 package] *****************************************************************************************
changed: [172.16.107.81]
changed: [172.16.107.53]
changed: [172.16.107.39]

TASK [remove php support for apache] **********************************************************************************
changed: [172.16.107.81]
changed: [172.16.107.39]
changed: [172.16.107.53]

PLAY RECAP ************************************************************************************************************
172.16.107.39              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.53              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.81              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

student@Ubuntu-desktop:~/nislab$ 
```

Figure 58: Run yml file.

7. We try opening the site on one of our servers as figure 59 show can't open the apache server because the package was removed from all servers.
   Then we run the install playbook file and return open the site it was accessed and opened because the package downloaded.



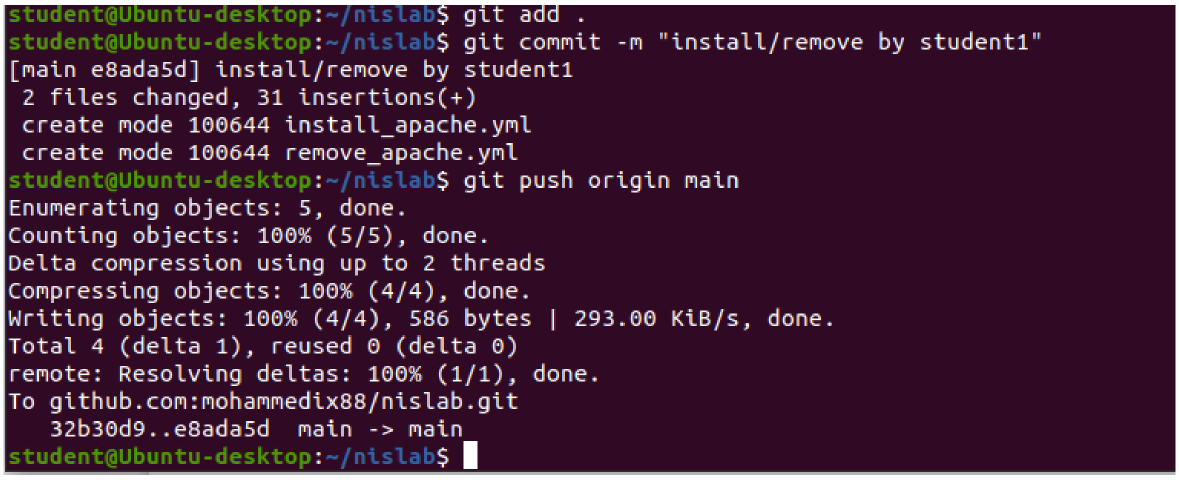
Unable to connect

Firefox can't establish a connection to the server at 172.16.107.81.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Figure 59: Try to open apache server.

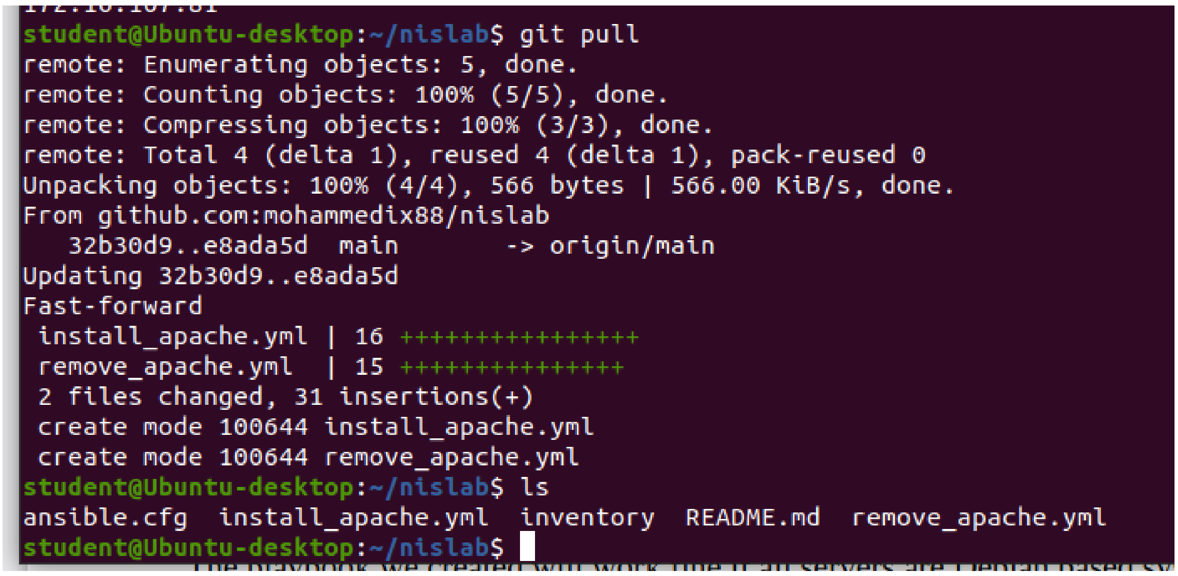
8. We added two files to our nislab directory which is connected to a git repository so we add both these files to gihub as figure 60 show .



```
student@Ubuntu-desktop:~/nislab$ git add .
student@Ubuntu-desktop:~/nislab$ git commit -m "install/remove by student1"
[main e8ada5d] install/remove by student1
 2 files changed, 31 insertions(+)
 create mode 100644 install_apache.yml
 create mode 100644 remove_apache.yml
student@Ubuntu-desktop:~/nislab$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 586 bytes | 293.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:mohammedix88/nislab.git
   32b30d9..e8ada5d  main -> main
student@Ubuntu-desktop:~/nislab$
```

Figure 60: Push the fils to github.

9. Then we pull it in workstation2 as figure 61 show .



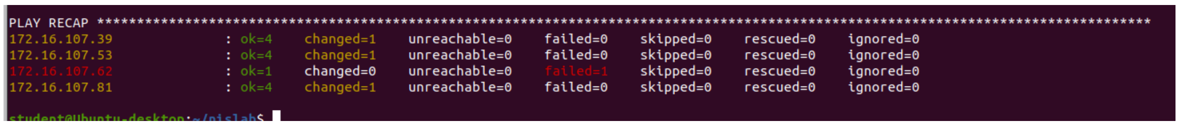
```
student@Ubuntu-desktop:~/nislab$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), 566 bytes | 566.00 KiB/s, done.
From github.com:mohammedix88/nislab
   32b30d9..e8ada5d  main       -> origin/main
Updating 32b30d9..e8ada5d
Fast-forward
 install_apache.yml | 16 ++++++++++++++++
 remove_apache.yml  | 15 +++++++++++++++
 2 files changed, 31 insertions(+)
 create mode 100644 install_apache.yml
 create mode 100644 remove_apache.yml
student@Ubuntu-desktop:~/nislab$ ls
ansible.cfg  install_apache.yml  inventory  README.md  remove_apache.yml
student@Ubuntu-desktop:~/nislab$
```

Figure 61: Pull the fils.

5.2 The 'when' Conditional

The playbook we created will work fine if all servers are Debian-based systems; because we used the apt module. if some of the servers have a base other than Debian then the playbook will fail when used on them.

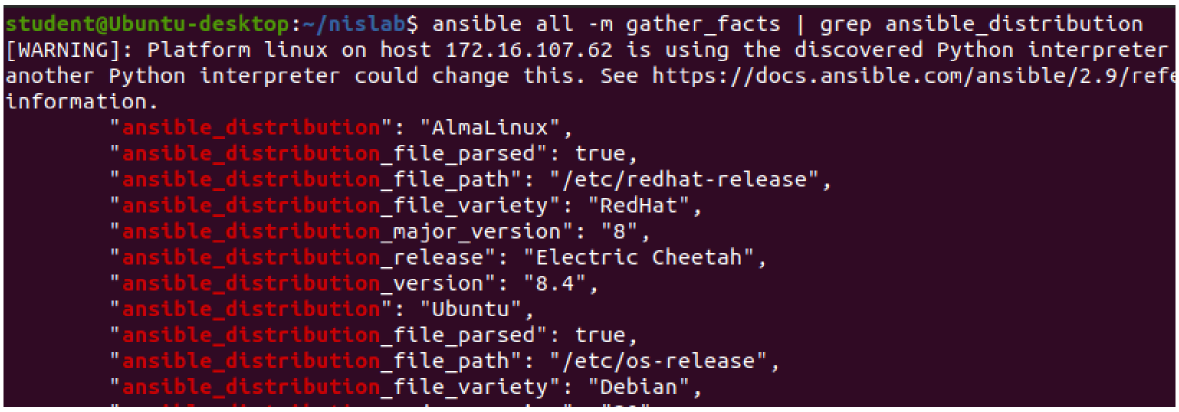
(a) We modify the inventory file by adding the AlmaLinux server IP address. Then, running the playbook file that is just with module **apt** and noticed that the output for the AlmaLinux server was failed as figure 62 show.



```
PLAY RECAP *****************************************************************************************************************
172.16.107.39              : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.53              : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.16.107.62              : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
172.16.107.81              : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

student@Ubuntu-desktop:~/nislab$
```

Figure 62: Failed to install in AlmaLinux.

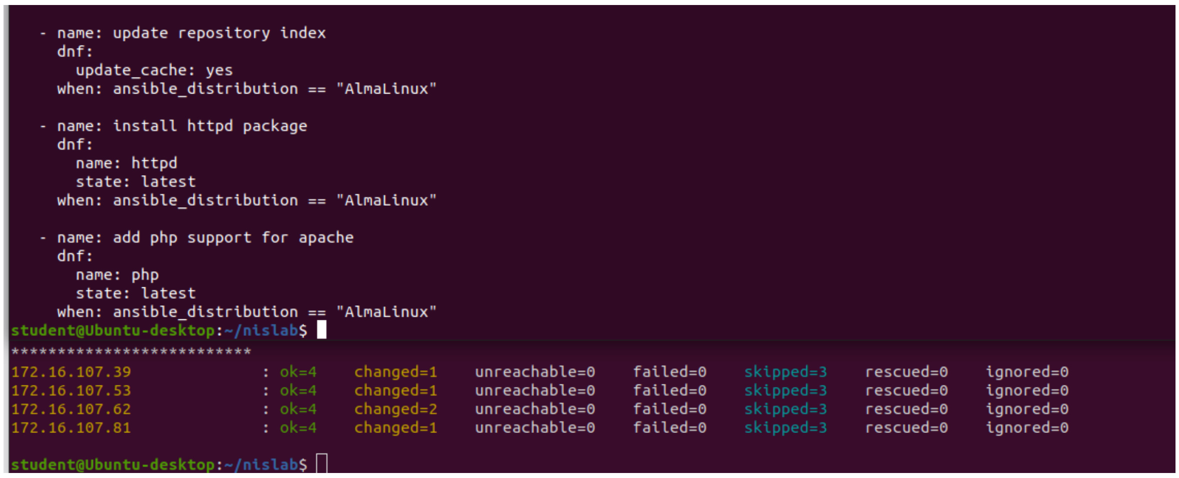
(b) Then we gather information to know the distributions of our servers by running the gather_facts module as figure 63 show that AlmaLinux is a RedHat distribution.



```
student@Ubuntu-desktop:~/nislab$ ansible all -m gather_facts | grep ansible_distribution
[WARNING]: Platform linux on host 172.16.107.62 is using the discovered Python interpreter
another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/refe
information.
        "ansible_distribution": "AlmaLinux",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/redhat-release",
        "ansible_distribution_file_variety": "RedHat",
        "ansible_distribution_major_version": "8",
        "ansible_distribution_release": "Electric Cheetah",
        "ansible_distribution_version": "8.4",
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/os-release",
        "ansible_distribution_file_variety": "Debian",
```

Figure 63: Gather information.

(c) So we modify the playbook that uses when condition to suitable for AlmaLinux then run the file and noticed that no failed as figure 64 show.

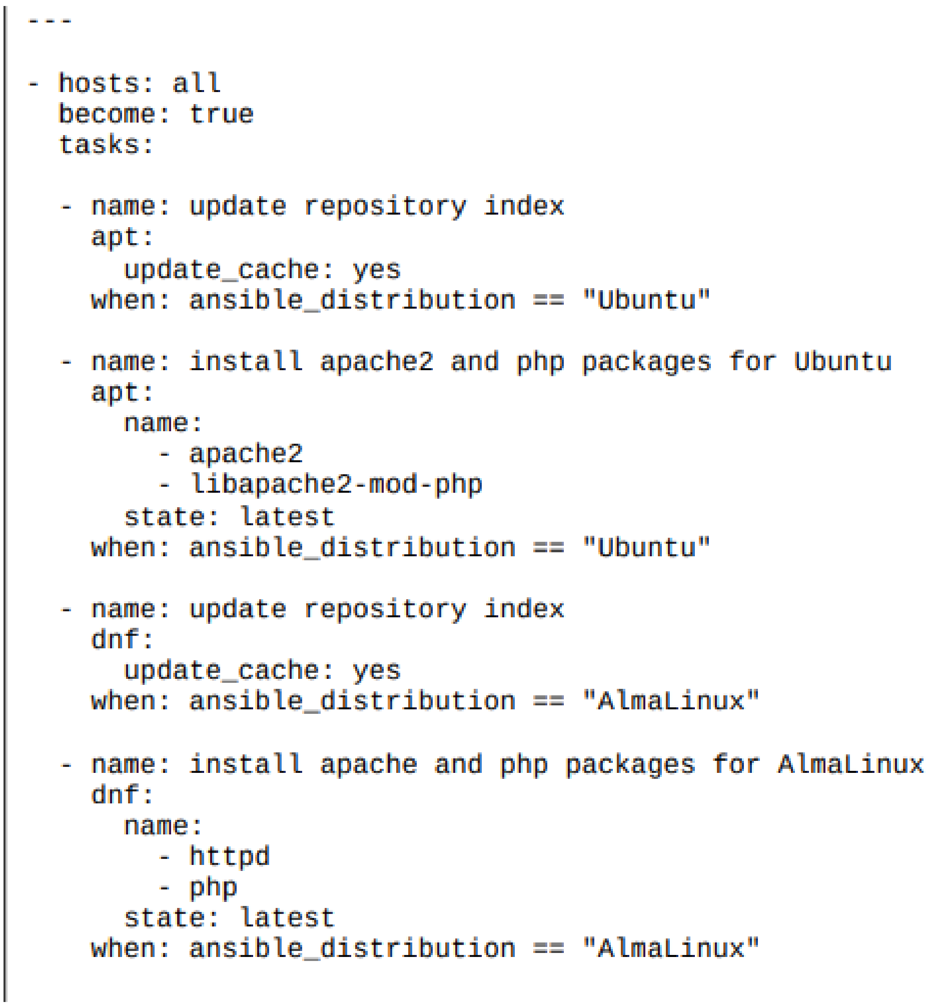


```
  - name: update repository index
    dnf:
      update_cache: yes
    when: ansible_distribution == "AlmaLinux"

  - name: install httpd package
    dnf:
      name: httpd
      state: latest
    when: ansible_distribution == "AlmaLinux"

  - name: add php support for apache
    dnf:
      name: php
      state: latest
    when: ansible_distribution == "AlmaLinux"
student@Ubuntu-desktop:~/nislab$
*************************
172.16.107.39              : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
172.16.107.53              : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
172.16.107.62              : ok=4    changed=2    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
172.16.107.81              : ok=4    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0

student@Ubuntu-desktop:~/nislab$
```

Figure 64: Playbook file the run it.

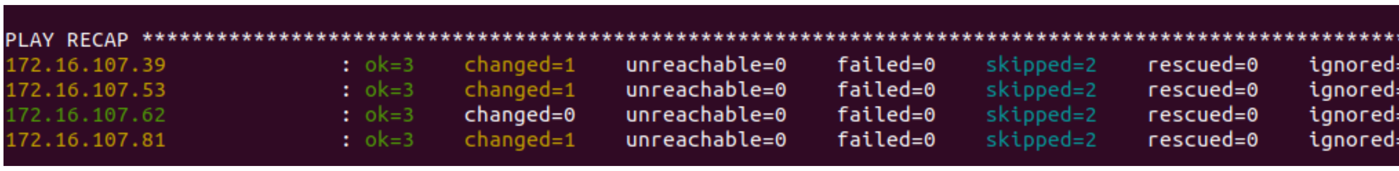
6 Improving The Playbook

1. Our playbook includes many unnecessary lines that we can omit. So we did add multiple packages to the apt module to be installed on the system. This way, we need only one task or play to install all needed packages. So we edit the playbook on workstation 2, as shown in figure 65, after that we run our play as figure 66 shown to make sure we have no syntax errors in the file.



```
---

- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install apache2 and php packages for Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: update repository index
    dnf:
      update_cache: yes
    when: ansible_distribution == "AlmaLinux"

  - name: install apache and php packages for AlmaLinux
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "AlmaLinux"
```

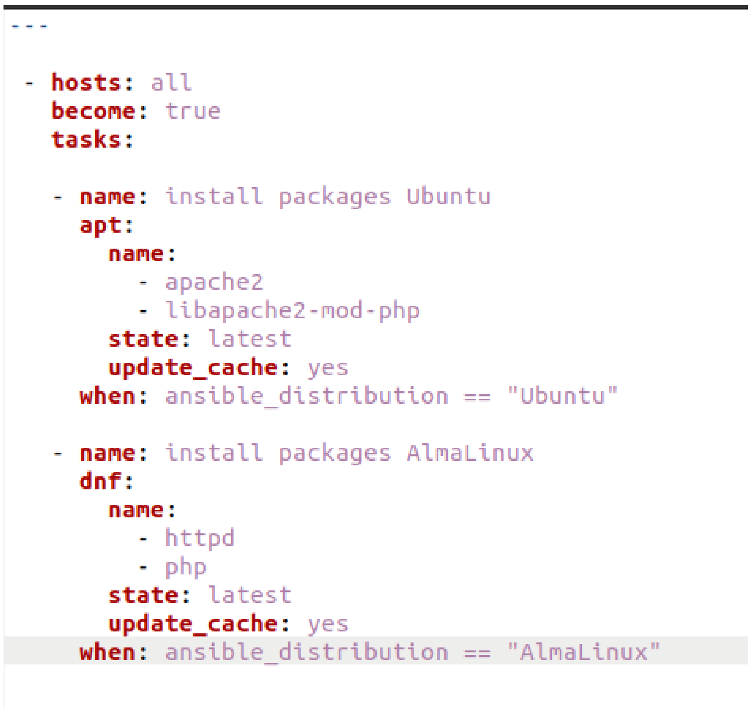
Figure 65: Edit playbook file.



```
PLAY RECAP ****************************************************************************************
172.16.107.39              : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored
172.16.107.53              : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored
172.16.107.62              : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored
172.16.107.81              : ok=3    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored
```

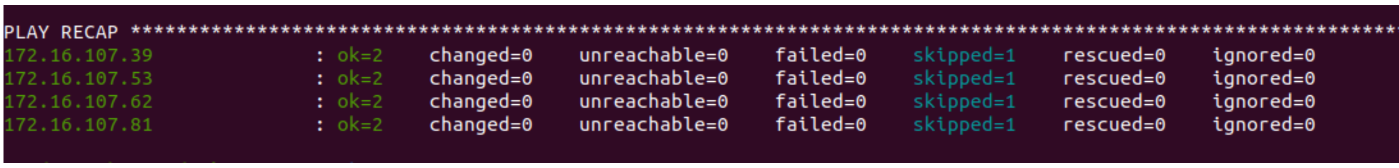
Figure 66: Run playbook file.

2. Since the update cache is a parameter of the apt module, we can also eliminate that task as shown in figure 67, now our playbook is down to 2 plays only, after that we run our playbook as figure 68 shown to make sure we have no syntax errors in the file.



```
---

- hosts: all
  become: true
  tasks:

  - name: install packages Ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: install packages AlmaLinux
    dnf:
      name:
        - httpd
        - php
      state: latest
      update_cache: yes
    when: ansible_distribution == "AlmaLinux"
```

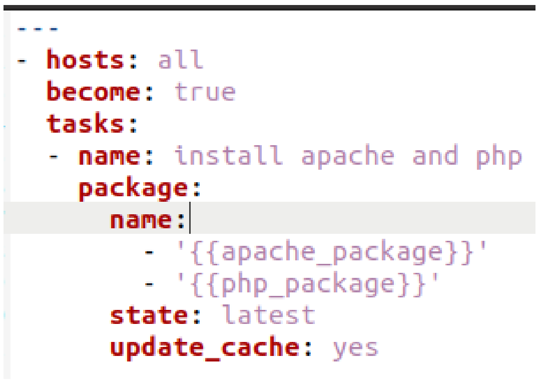
Figure 67: Improving playbook file.



```
PLAY RECAP ****************************************************************************************************
172.16.107.39              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.53              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.62              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.81              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

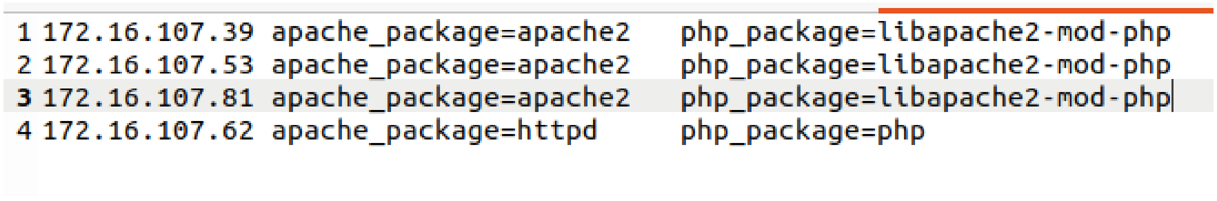
Figure 68: Run playbook file.

3. Now we can even get our playbook down to one play using variables as shown in figures 69 and 70 package module is a generic package manager, which means it will use the default package manager of each distribution. the {{apache_package}} and {{php_package}} are variables we created and we can use any name we want. Now for this to work, we need to edit the inventory file to include the package names. Then we run our playbook as figure 71 shown to make sure we have no syntax errors in the file.



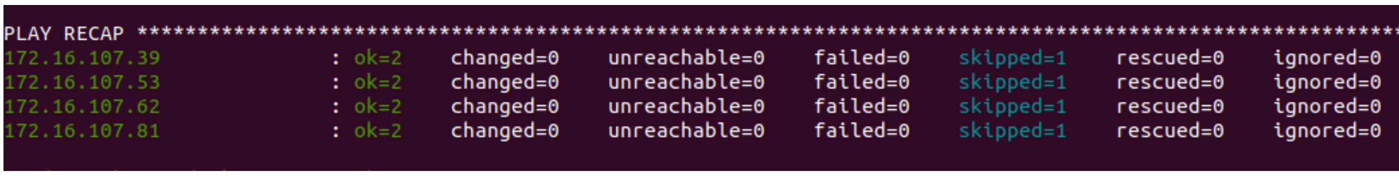
```yaml
---
- hosts: all
  become: true
  tasks:
  - name: install apache and php
    package:
      name:
        - '{{apache_package}}'
        - '{{php_package}}'
      state: latest
      update_cache: yes
```

Figure 69: Run playbook file.



```
172.16.107.39 apache_package=apache2   php_package=libapache2-mod-php
172.16.107.53 apache_package=apache2   php_package=libapache2-mod-php
172.16.107.81 apache_package=apache2   php_package=libapache2-mod-php
172.16.107.62 apache_package=httpd     php_package=php
```

Figure 70: Edit Inventory file.

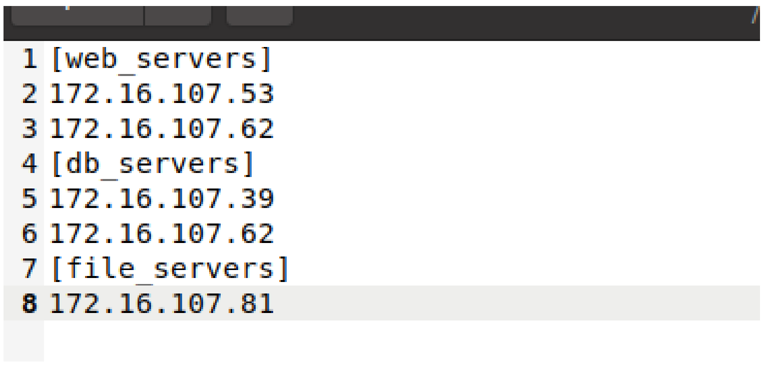


```
PLAY RECAP ************************************************************************************
172.16.107.39              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.53              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.62              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.16.107.81              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Figure 71: Run playbook file.

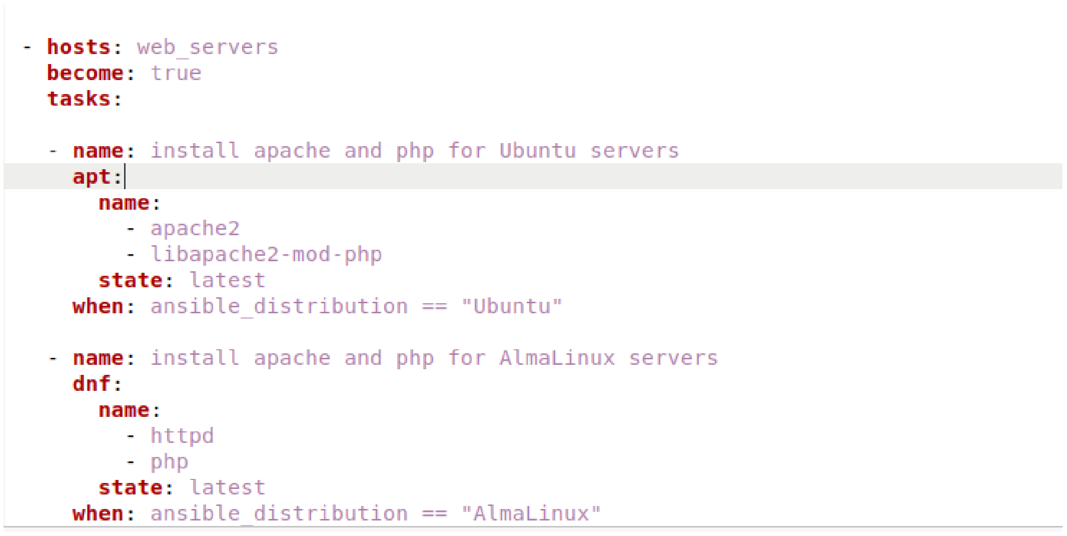
7 Targeting Specific Nodes

1. Now suppose that we have different roles that our servers have. Suppose we have web, database, and file servers. If we want to target the web servers; with certain tasks while doing other tasks for the file and web servers, then we need to categories our servers in the inventory file as shown in figure 72. After that, we edit the playbook file as figure 73 and add a host for every task and, the host will be the variable that we use in the inventory file to categories our servers. Then we run our playbook to made sure that only web server will install, as shown in figure 74.



```
[web_servers]
172.16.107.53
172.16.107.62
[db_servers]
172.16.107.39
172.16.107.62
[file_servers]
172.16.107.81
```

Figure 72: Edit inventory file.



```
- hosts: web_servers
  become: true
  tasks:

  - name: install apache and php for Ubuntu servers
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"

  - name: install apache and php for AlmaLinux servers
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "AlmaLinux"
```

Figure 73: Edit playbook file.

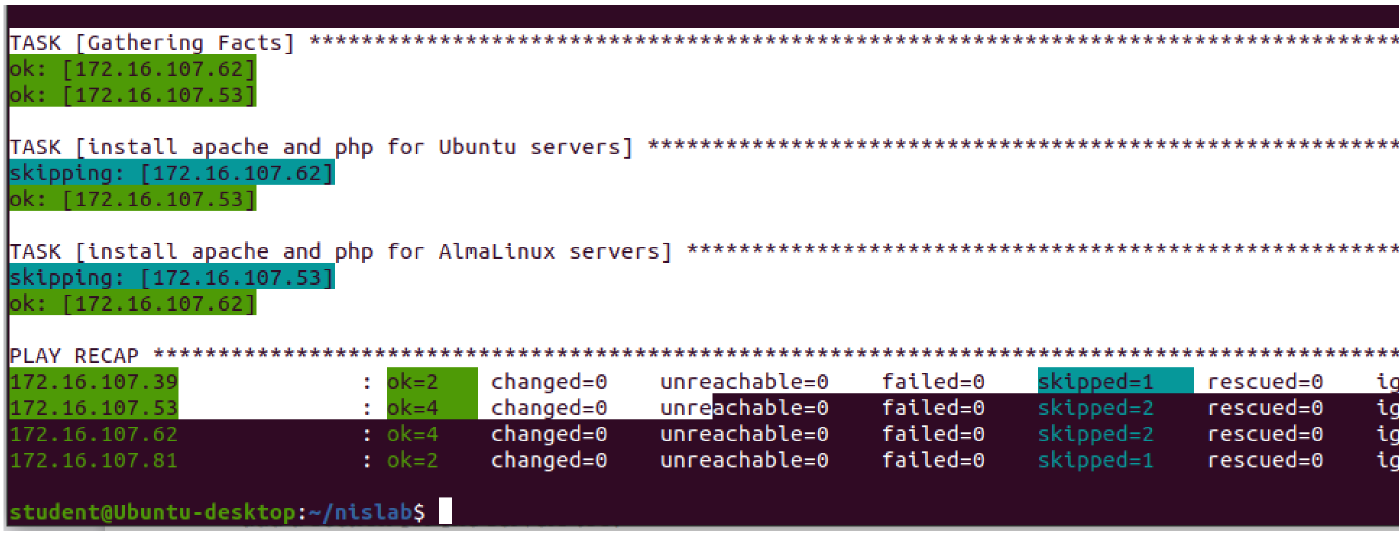

```
TASK [Gathering Facts] ********************************************************
ok: [172.16.107.62]
ok: [172.16.107.53]

TASK [install apache and php for Ubuntu servers] ******************************
skipping: [172.16.107.62]
ok: [172.16.107.53]

TASK [install apache and php for AlmaLinux servers] ***************************
skipping: [172.16.107.53]
ok: [172.16.107.62]

PLAY RECAP ********************************************************************
172.16.107.39              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ig
172.16.107.53              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.62              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.81              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ig

student@Ubuntu-desktop:~/nislab$
```

Figure 74: Run playbook file.

2. After that, we add the database task as shown in figure 75, then we run the playbook to ensure that the webserver and database will install for a particular host as shown in figure 76. Also, we add a task for the fileserver host to install as shown in figure 77, then we run it again to make sure that all the packages install and there is no error as shown in figure 78.

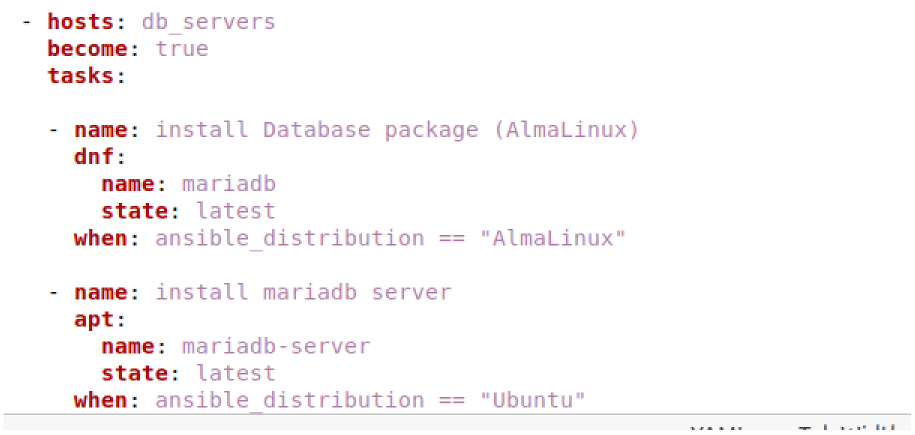

```
- hosts: db_servers
  become: true
  tasks:

   - name: install Database package (AlmaLinux)
     dnf:
       name: mariadb
       state: latest
     when: ansible_distribution == "AlmaLinux"

   - name: install mariadb server
     apt:
       name: mariadb-server
       state: latest
     when: ansible_distribution == "Ubuntu"
```

Figure 75: Edit playbook file & add database task.

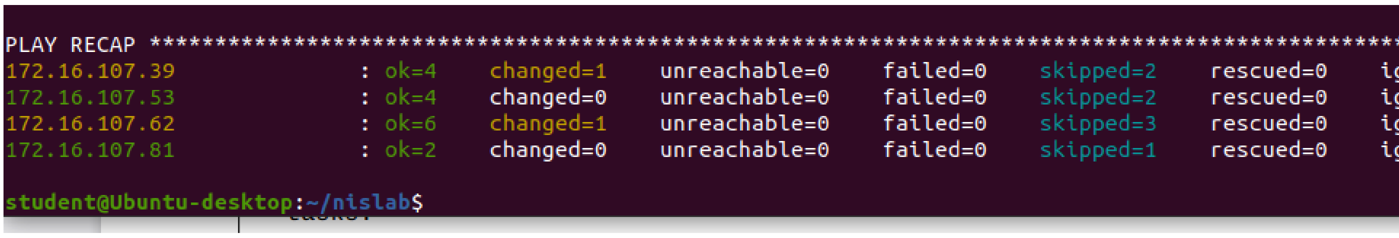
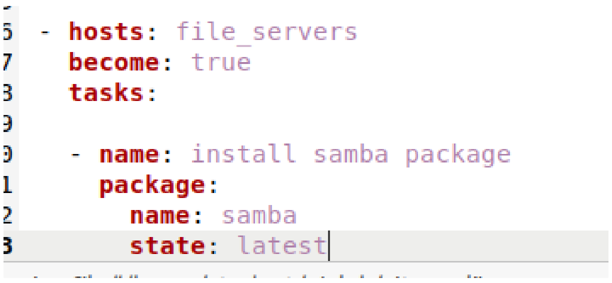
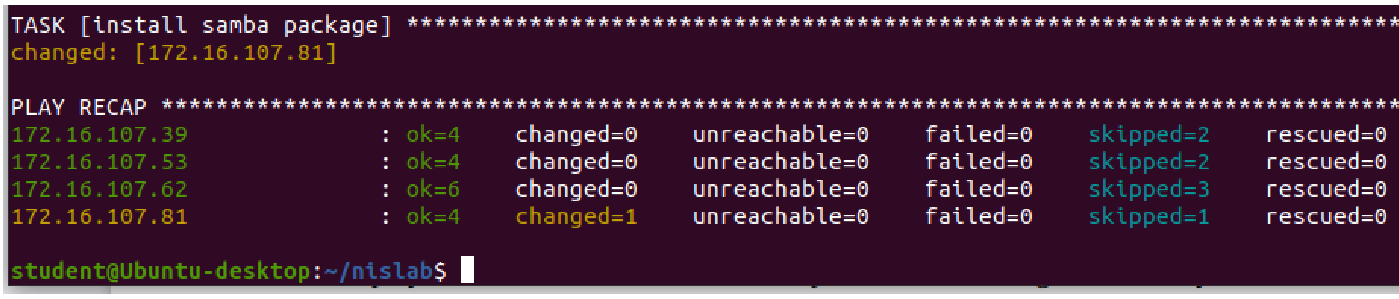

```
PLAY RECAP ********************************************************************
172.16.107.39              : ok=4    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.53              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.62              : ok=6    changed=1    unreachable=0    failed=0    skipped=3    rescued=0    ig
172.16.107.81              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ig

student@Ubuntu-desktop:~/nislab$
```

Figure 76: Run playbook file.



```
- hosts: file_servers
  become: true
  tasks:

  - name: install samba package
    package:
      name: samba
      state: latest
```

Figure 77: Edit playbook file & add file-server task.



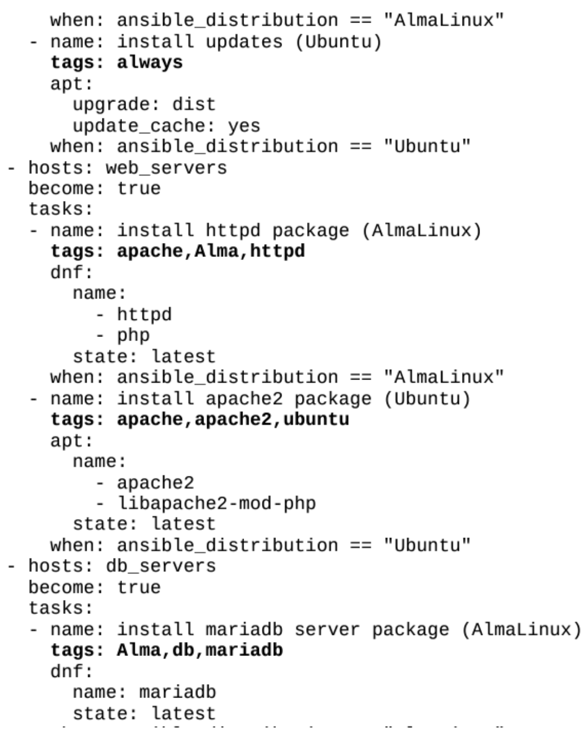
```
TASK [install samba package] ***********************************************************************
changed: [172.16.107.81]

PLAY RECAP *****************************************************************************************
172.16.107.39              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
172.16.107.53              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0
172.16.107.62              : ok=6    changed=0    unreachable=0    failed=0    skipped=3    rescued=0
172.16.107.81              : ok=4    changed=1    unreachable=0    failed=0    skipped=1    rescued=0

student@Ubuntu-desktop:~/nislab$
```

Figure 78: Run playbook file.

8 Using Tags

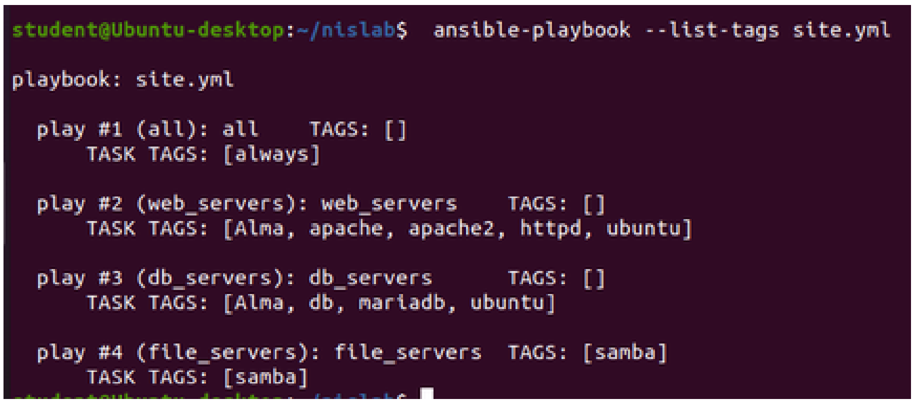
We can add tags to our tasks which is another way to execute plays on certain hosts. Suppose we want to run all plays on Ubuntu servers that have Apache installed. Tags are words we add to each task in the tag line.

1. So we add the tag under the task name as shown in figure 79, as we see that every task have tags, **tag: always** mean that it will always have done when any task run, then we list all available tagging in the playbook file by using list command as shown in figure 80. At first, we run our playbook using DB tag by using **ansible-playbook –tags DB –ask-become-pass site.yml** command, that command will install DB in every server have the DB tags as shown in figure 81. After that, we run our playbook, but now by using the alma tag by running **ansible-playbook –tags Alma –ask-become-pass site.yml** command, that command will run all tasks that contain the alma tag on it as shown in figure 82. In the end, we run the playbook to execute tasks, which will contain apache & DB tags, as the following command: **ansible-playbook –tags "apache, DB" –ask-become-pass site.yml**, figure 83 shows the result for that command.



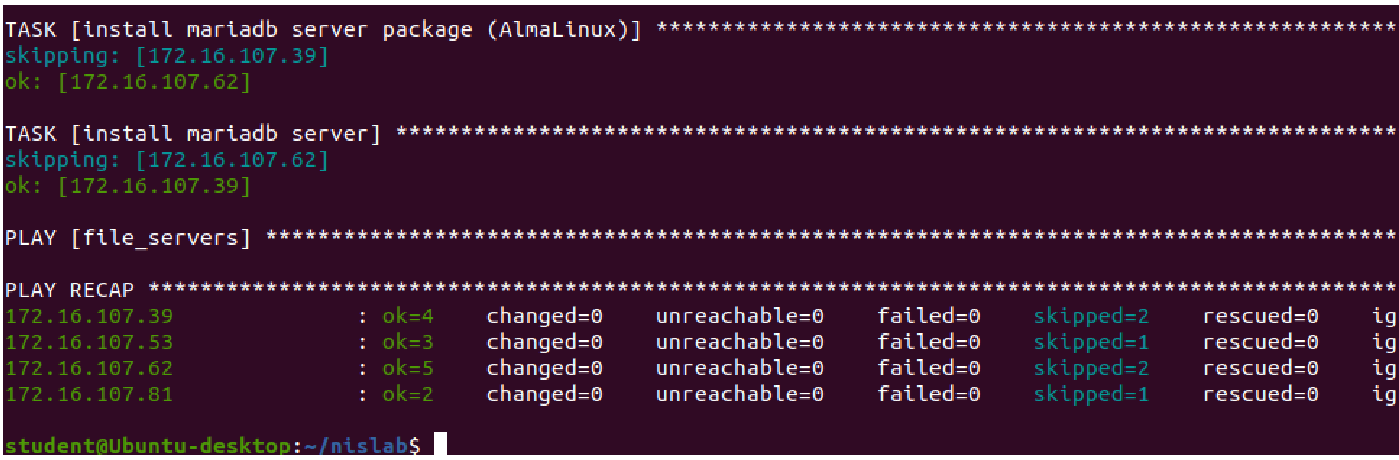
```
    when: ansible_distribution == "AlmaLinux"
  - name: install updates (Ubuntu)
    tags: always
    apt:
      upgrade: dist
      update_cache: yes
    when: ansible_distribution == "Ubuntu"
- hosts: web_servers
  become: true
  tasks:
  - name: install httpd package (AlmaLinux)
    tags: apache,Alma,httpd
    dnf:
      name:
        - httpd
        - php
      state: latest
    when: ansible_distribution == "AlmaLinux"
  - name: install apache2 package (Ubuntu)
    tags: apache,apache2,ubuntu
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
    when: ansible_distribution == "Ubuntu"
- hosts: db_servers
  become: true
  tasks:
  - name: install mariadb server package (AlmaLinux)
    tags: Alma,db,mariadb
    dnf:
      name: mariadb
      state: latest
```

Figure 79: Edit playbook file by adding tags.



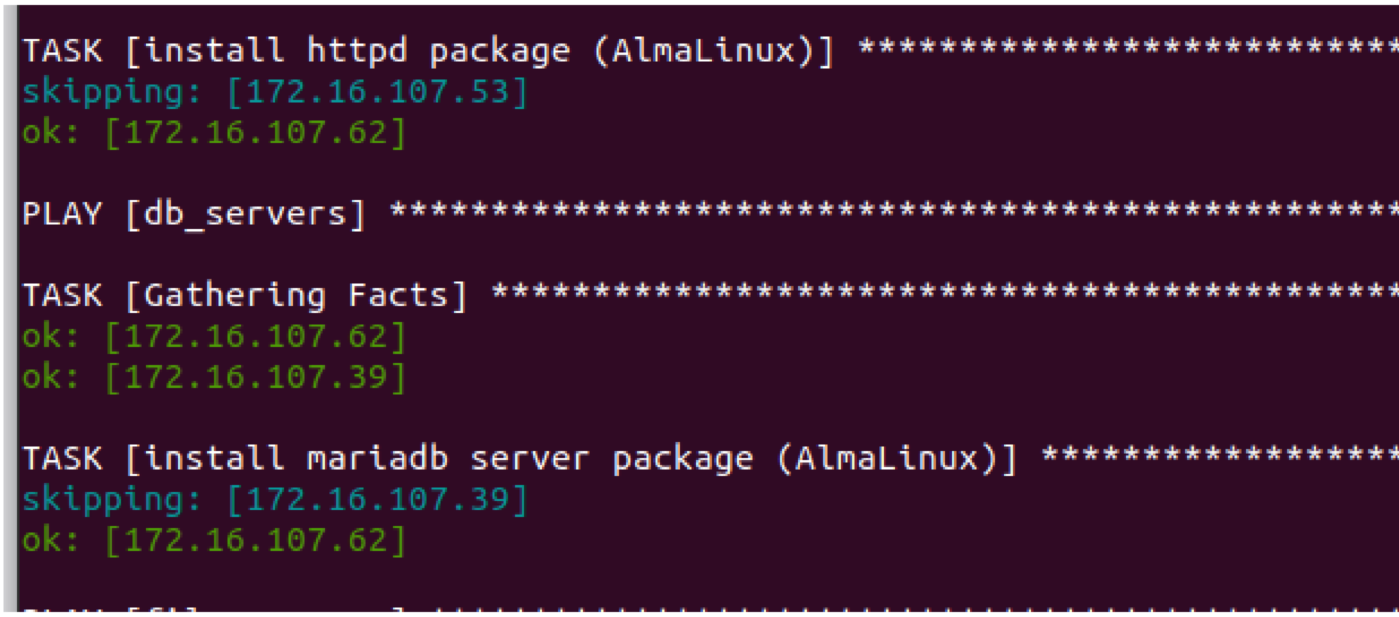
```
student@Ubuntu-desktop:~/nislab$  ansible-playbook --list-tags site.yml

playbook: site.yml

  play #1 (all): all    TAGS: []
      TASK TAGS: [always]

  play #2 (web_servers): web_servers    TAGS: []
      TASK TAGS: [Alma, apache, apache2, httpd, ubuntu]

  play #3 (db_servers): db_servers      TAGS: []
      TASK TAGS: [Alma, db, mariadb, ubuntu]

  play #4 (file_servers): file_servers  TAGS: [samba]
      TASK TAGS: [samba]
```

Figure 80: List all tags.



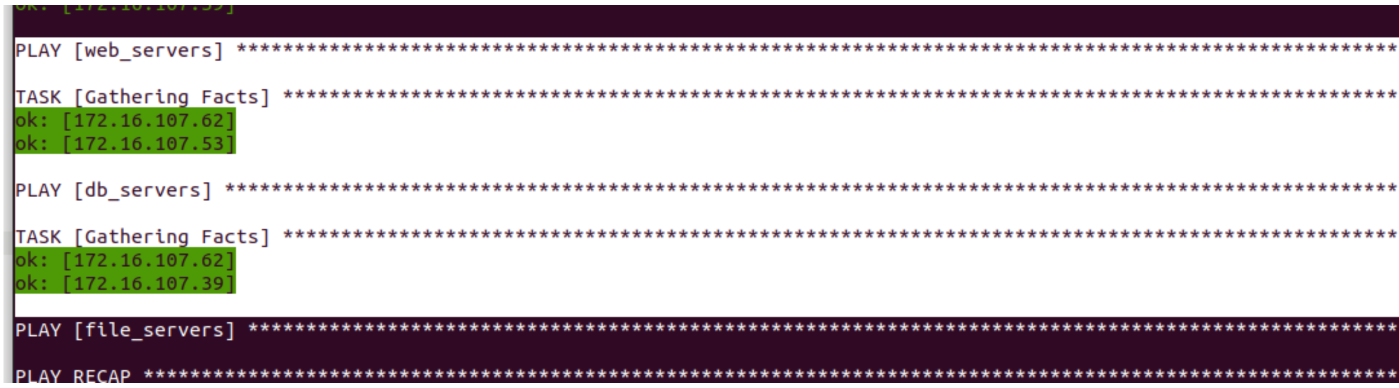
```
TASK [install mariadb server package (AlmaLinux)] *******************************************************
skipping: [172.16.107.39]
ok: [172.16.107.62]

TASK [install mariadb server] ***************************************************************************
skipping: [172.16.107.62]
ok: [172.16.107.39]

PLAY [file_servers] *************************************************************************************

PLAY RECAP **********************************************************************************************
172.16.107.39              : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.53              : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ig
172.16.107.62              : ok=5    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ig
172.16.107.81              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ig

student@Ubuntu-desktop:~/nislab$
```

Figure 81: Run playbook using db tags.



```
TASK [install httpd package (AlmaLinux)] ****************************
skipping: [172.16.107.53]
ok: [172.16.107.62]

PLAY [db_servers] ***************************************************

TASK [Gathering Facts] **********************************************
ok: [172.16.107.62]
ok: [172.16.107.39]

TASK [install mariadb server package (AlmaLinux)] *******************
skipping: [172.16.107.39]
ok: [172.16.107.62]
```

Figure 82: Run playbook using alma tags.



```
PLAY [web_servers] **************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [172.16.107.62]
ok: [172.16.107.53]

PLAY [db_servers] ***************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [172.16.107.62]
ok: [172.16.107.39]

PLAY [file_servers] *************************************************************************************************

PLAY RECAP **********************************************************************************************************
```

Figure 83: Run playbook using apache & db tags.

9 Conclusion

Ansible; is designed to be very simple, reliable, and consistent for configuration management. If our're already in IT, our can get up and running with it very quickly. Ansible configurations are simple data descriptions of infrastructure and are both readable by humans and parsable by machines. All we need, to start managing systems is a password or an SSH (Secure Socket Shell, a network protocol) key. An example of how easy Ansible makes configuration management: If we want to install an updated version, of a specific type of software on all the machines in ourr enterprise, all we have to do is write out all the IP addresses of the nodes (also called remote hosts) in a file called inventory and write an Ansible playbook to install it on all the nodes, then run the playbook from our control machine. We can write the playbook file in many/different ways by using **when condition & tag** and so on.

10 References

https://www.ansible.com/overview/how-ansible-works
https://www.simplilearn.com/tutorials/ansible-tutorial/what-is-ansible: :text=Advantages
https://www.redhat.com/en/technologies/management/ansible