

An-Najah National University



Faculty of Engineering and Information Technology

Network Administration Lab Report

Automation Merged

EXP#7

Dr. Bakr Abd Al-Haq

Team Members: Ali Khuffash, Wajd Abd Al-Hadee

Table of Contents

| | |
|--|----|
| 1.Abstract..... | 3 |
| 2.Introduction..... | 3 |
| 3.Objectives | 3 |
| 4.Pre-requisites & Resources | 3 |
| 5.Methodology | 4 |
| 5.1 SSH Setup | 4 |
| 5.2 Version Control..... | 9 |
| 5.3 Ansible [ad-hoc commands]..... | 15 |
| 5.4 Package Management Using Ansible Playbooks | 24 |
| 5.5 Version Control with Git..... | 32 |
| 5.6 The 'when' Conditional | 33 |
| 6.Results and Observations | 37 |
| 7.Conclusion | 37 |
| 8.References..... | 37 |
| 9.Appendices..... | 37 |

1. Abstract

This report outlines the procedures and outcomes of a practical lab focused on automation in network administration. The primary tools utilized include OpenSSH for secure communication, Git for version control, and Ansible for automation. Through this lab, participants configured secure access between multiple virtual machines, collaborated via GitHub repositories, and used Ansible to manage remote servers through ad-hoc commands and playbooks. The implementation aimed to streamline administrative tasks and establish foundational skills in infrastructure automation

2. Introduction

Modern network and system administration heavily relies on automation to ensure scalability, consistency, and efficiency. This lab introduces key automation tools and practices used in real-world IT environments. OpenSSH enables secure remote access, Git facilitates version control and collaboration, and Ansible simplifies configuration management across multiple systems. By integrating these tools, this lab simulates a real-world administrative scenario involving multiple machines and tasks.

3. Objectives

Upon completion of this lab, students are expected to:

- Install and configure OpenSSH using SSH keys.
- Create and configure a GitHub repository and set up Git for version control.
- Install and use Ansible for basic automated tasks.
- Write and run Ansible playbooks to manage packages.
- Utilize conditionals (e.g., when) to tailor Ansible tasks to specific systems

4. Pre-requisites & Resources

- Knowledge Requirements:

Basic command line interface usage (Linux terminal).
Understanding of SSH and public/private key infrastructure.
Familiarity with Git and GitHub.
Introductory knowledge of Ansible and YAML syntax.

- Hardware/Software Requirements:

Six virtual machines configured as follows:

| PC | VM | OS | Hostname | RAM |
|-----|--------------|----------------|--------------|------|
| PC1 | Workstation1 | Ubuntu Desktop | Workstation1 | 4 GB |
| | SRVR01 | Ubuntu Server | SRVR01 | 1 GB |
| | SRVR02 | Ubuntu Server | SRVR02 | 1 GB |
| PC2 | Workstation2 | Ubuntu Desktop | Workstation2 | 4 GB |
| | SRVR03 | Ubuntu Server | SRVR03 | 1 GB |
| | SRVR04 | AlmaLinux | SRVR04 | 2 GB |

All VMs use the same credentials:

Username: student

Password: student

5.Methodology

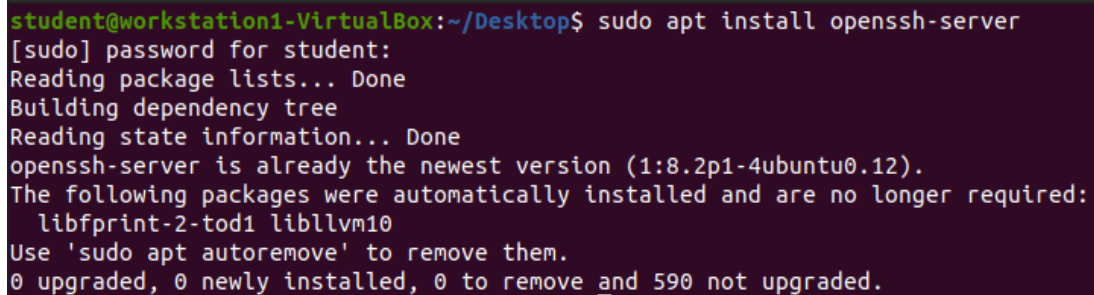
5.1 SSH Setup

SSH (Secure Shell) is a cryptographic network protocol used to securely access and manage devices over an unsecured network. SSH provides a secure channel over an unsecured network by using encryption and authentication mechanisms.

SSH is widely used for remote administration of systems, automation scripts, file transfers (via SCP or SFTP), and secure tunneling. Proper SSH setup ensures both security and ease of access in managing remote systems.

SSH was installed using APT Package:

```
sudo apt install openssh-server
```



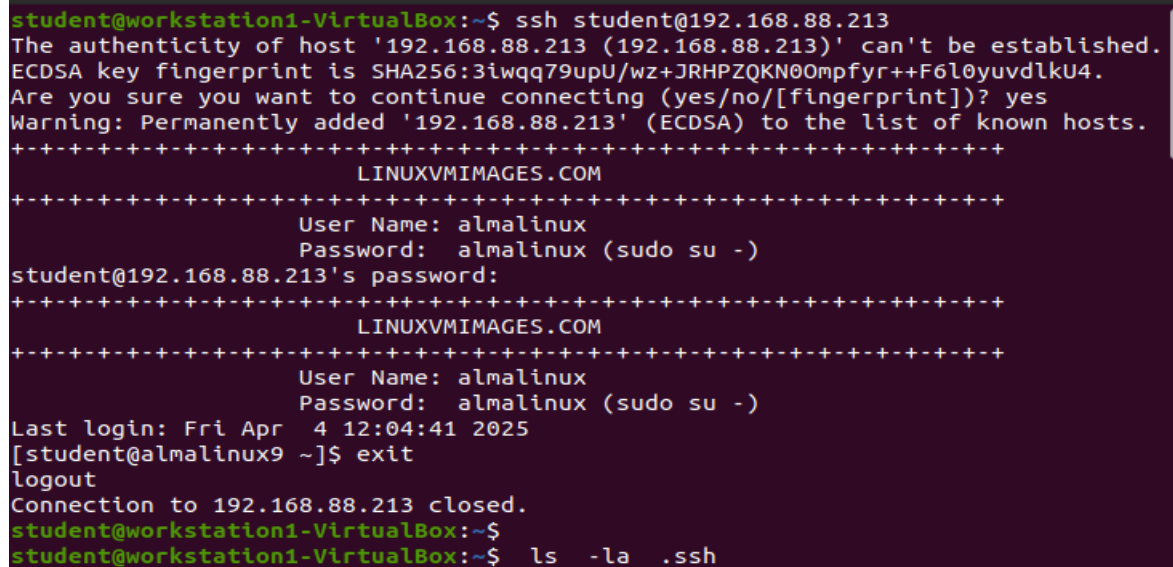
```
student@workstation1-VirtualBox:~/Desktop$ sudo apt install openssh-server
[sudo] password for student:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:8.2p1-4ubuntu0.12).
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm10
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 590 not upgraded.
```

Figure 5.1.1 Installing the SSH.

The SSH service was installed on the servers during the Ubuntu installation process.

SSH connections were initiated from both workstations to each of the Ubuntu servers. This initial connection is essential, as it requires accepting the server's fingerprint in order to add it to the 'known_hosts' file on each workstation. The connection was established using the following command. See figure 5.1.2

```
ssh student@192.168.88.213
```



```
student@workstation1-VirtualBox:~$ ssh student@192.168.88.213
The authenticity of host '192.168.88.213 (192.168.88.213)' can't be established.
ECDSA key fingerprint is SHA256:3iwqq79upU/wz+JRHPZQKN00mpfyr++F6l0yuvdlkU4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.88.213' (ECDSA) to the list of known hosts.
+-----+
LINUXVMIMAGES.COM
+-----+
User Name: almalinux
Password: almalinux (sudo su -)
student@192.168.88.213's password:
+-----+
LINUXVMIMAGES.COM
+-----+
User Name: almalinux
Password: almalinux (sudo su -)
Last login: Fri Apr 4 12:04:41 2025
[student@almalinux9 ~]$ exit
logout
Connection to 192.168.88.213 closed.
student@workstation1-VirtualBox:~$
student@workstation1-VirtualBox:~$ ls -la .ssh
```

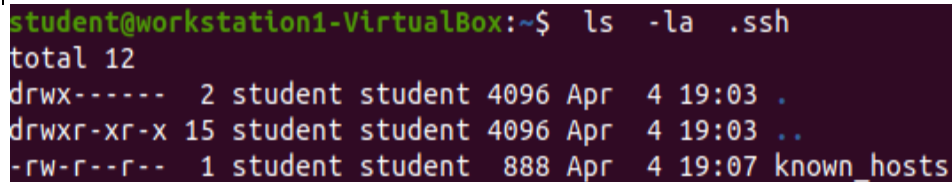
Figure 5.1.2 The fingerprint was accepted for all servers to ensure secure future communications

Creating an SSH key pair enhances the security of our server connections. Although it is not strictly required for using Ansible, it

ensures that connections established by Ansible are strongly encrypted and helps to minimize potential security vulnerabilities.

On Workstation1, the following steps were performed

```
ls -la .ssh
```



```
student@workstation1-VirtualBox:~$ ls -la .ssh
total 12
drwx----- 2 student student 4096 Apr  4 19:03 .
drwxr-xr-x 15 student student 4096 Apr  4 19:03 ..
-rw-r--r--  1 student student  888 Apr  4 19:07 known_hosts
```

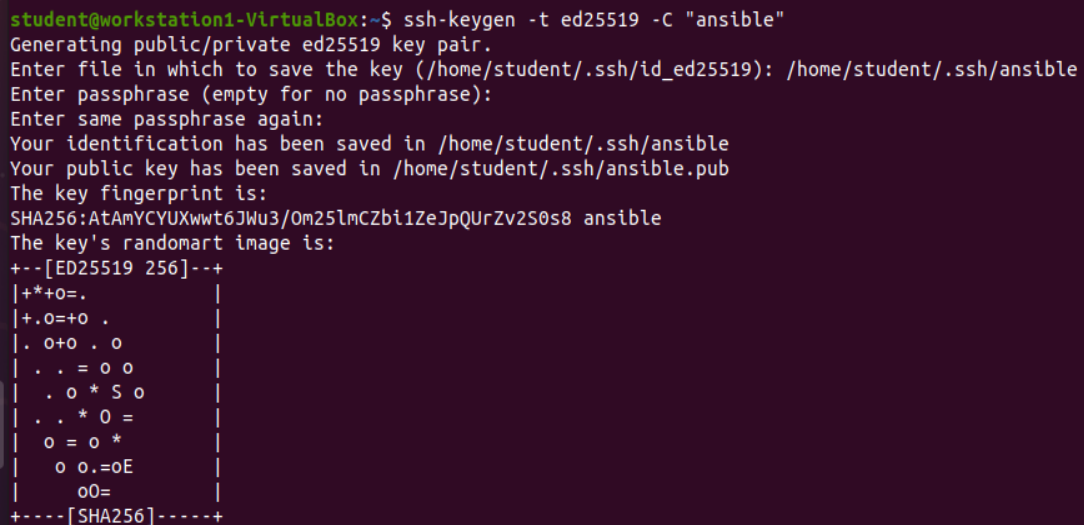
Figure 5.1.3 Lists all files, including hidden ones, in the .ssh directory.

SSH key pair was created on Workstation1 using the command

```
ssh-keygen -t ed25519 -C "ansible"
```

Saving the keys to /home/student/.ssh/ansible. The .ssh directory now contains two files: ansible (private key) and ansible.pub (public key).

See figure 5.1.4



```
student@workstation1-VirtualBox:~$ 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:AtAmYCYUXwWt6JWu3/0m25lmcZbi1ZeJpQURZv2S0s8 ansible
The key's randomart image is:
+--[ED25519 256]--+
|+*+0=.          |
|+.0=+0 .        |
|. 0+0 . 0        |
|. . = 0 0        |
|. 0 * S 0        |
|. . * 0 =        |
| 0 = 0 *         |
| 0 0.=0E         |
| 00=             |
+----[SHA256]-----+
```

Figure 5.1.4 SSH Key Generation Process on Workstation1.

The public key is copied to each of the three servers using the ssh-copy-id command. The command adds the public key to the server's ~/.ssh/authorized_keys file, enabling password-less SSH login. This process was repeated for each server by replacing [SRVR_IP] with the respective server IP address. See figure 5.1.5

```
~/ssh$ ssh-copy-id -i ~/.ssh/ansible.pub [SRVR_IP]
```

```
student@workstation1-VirtualBox:~/ssh$ ssh-copy-id -i ~/.ssh/ansible.pub student@192.168.88.209
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/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@192.168.88.209's password:

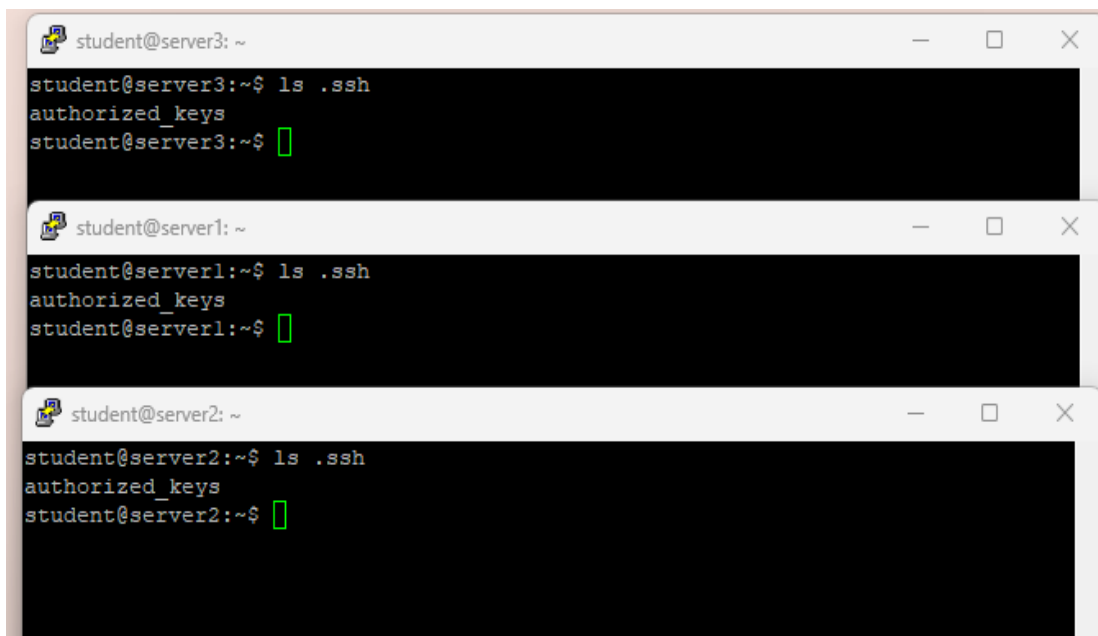
Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'student@192.168.88.209'"
and check to make sure that only the key(s) you wanted were added.

student@workstation1-VirtualBox:~/ssh$
```

Figure 5.1.5 Copying the Public Key to Servers.

Now, testing the servers by check the .ssh directory. See figure 5.1.6



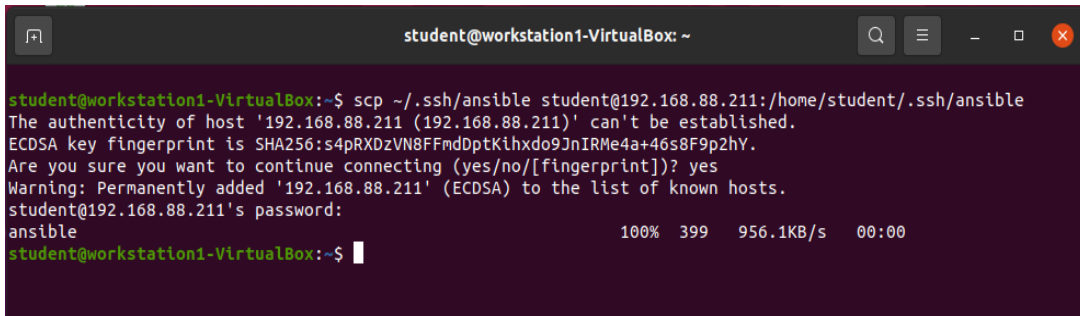
The image shows three overlapping terminal windows. Each window is titled 'student@serverX: ~' where X is 1, 2, or 3. In each window, the user has entered the command 'ls .ssh' and the output is 'authorized_keys'. The prompt in each window is 'student@serverX:~\$'.

Figure 5.1.6 List the ssh directory of the servers.

The public and private SSH keys are copied from Workstation1 to Workstation2 to ensure that both workstations use identical keys for the same user ID, which is "ansible." This ensures seamless SSH authentication across both workstations. The following scp commands were used to transfer the keys. See figure 5.1.7 and figure 5.1.8

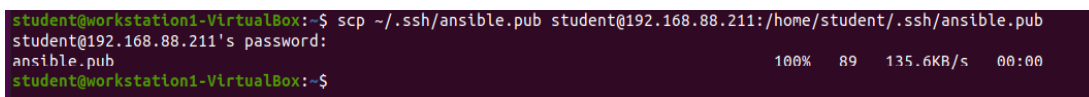
```
Workstation1$ scp ~/.ssh/ansible
student@[workstation2_IP]:/home/student/.ssh/ansible
```

```
Workstation1$ scp ~/.ssh/ansible.pub  
student@[workstation2_IP]:/home/student/.ssh/ansible.pub
```



```
student@workstation1-VirtualBox: ~  
student@workstation1-VirtualBox:~$ scp ~/.ssh/ansible student@192.168.88.211:/home/student/.ssh/ansible  
The authenticity of host '192.168.88.211 (192.168.88.211)' can't be established.  
ECDSA key fingerprint is SHA256:s4pRXDzVN8FFmdDptKihxdo9JnIRMe4a+46s8F9p2hY.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.88.211' (ECDSA) to the list of known hosts.  
student@192.168.88.211's password:  
ansible 100% 399 956.1KB/s 00:00  
student@workstation1-VirtualBox:~$
```

Figure 5.1.7 Copying SSH private key from Workstation1 to Workstation2.



```
student@workstation1-VirtualBox:~$ scp ~/.ssh/ansible.pub student@192.168.88.211:/home/student/.ssh/ansible.pub  
student@192.168.88.211's password:  
ansible.pub 100% 89 135.6KB/s 00:00  
student@workstation1-VirtualBox:~$
```

Figure 5.1.8 Copying SSH public key from Workstation1 to Workstation2.

To test the environment, SSH connections were established to all servers from both workstations using the Ansible SSH key. The following commands were executed on both workstations, and the connection was successful without prompting for a password, indicating that the SSH key authentication was properly configured. See figure 5.1.9

```
Workstation1$ ssh [SRVR_IP]  
Workstation2$ ssh [SRVR_IP]
```

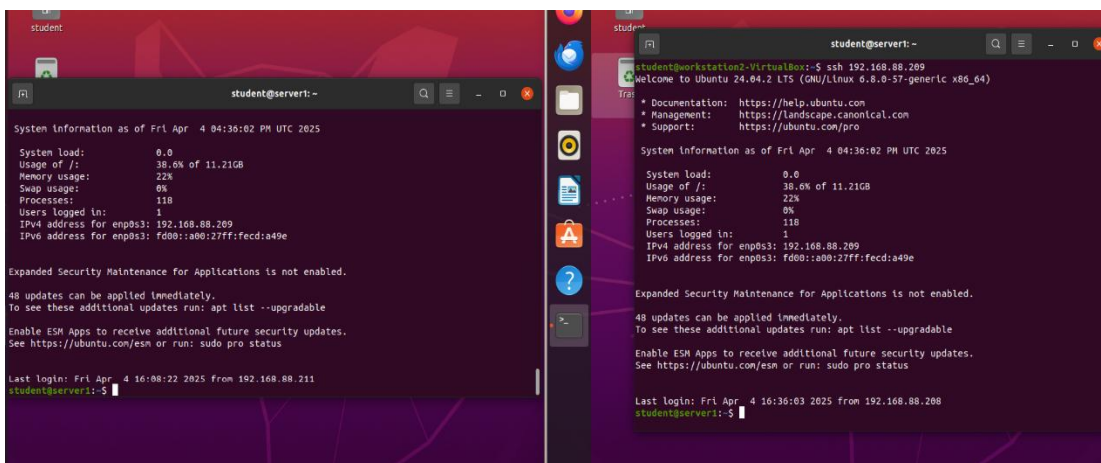


Figure 5.1.9 Testing SSH Connection Using Ansible Key from Both Workstations.

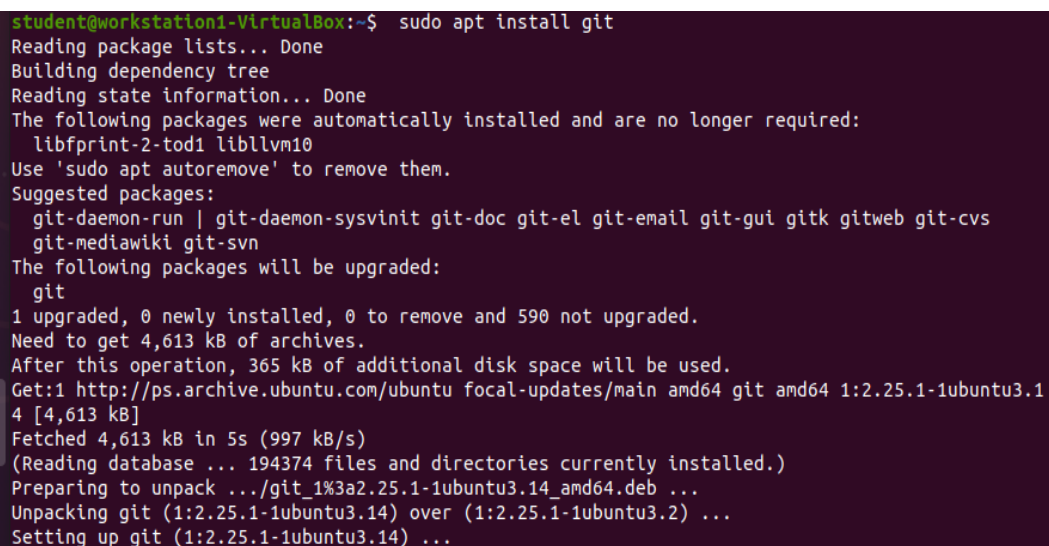
5.2 Version Control

Version control is a critical aspect of system administration in large enterprises, as it allows multiple administrators to collaborate effectively. It ensures transparency by enabling admins to track changes made to configuration files and Ansible playbooks. GitHub plays a key role in this process by providing a platform to share, sync, and collaborate on these files.

In this section of the lab, we will create a GitHub account and use repositories to share configuration files between the two workstations used for administration. GitHub repositories can be cloned to local machines and kept synchronized with the online version, allowing admins to push and pull changes as needed. This enables collaboration and ensures that all administrators are aware of modifications made to the infrastructure.

The following commands are used on both workstations to install Git. See figure 5.2.1

```
sudo apt update  
sudo apt install git
```

A terminal window with a dark background and light text. The prompt is 'student@workstation1-VirtualBox:~\$'. The command 'sudo apt install git' has been executed. The output shows the package lists being read, the dependency tree being built, and state information being read. It lists packages to be removed (libfprint-2-tod1, libllvm10) and suggested packages (git-daemon-run, git-daemon-sysvinit, git-doc, git-el, git-email, git-gui, gitk, gitweb, git-cvs, git-mediawiki, git-svn). It then shows the packages to be upgraded (git) and the disk space requirements (4,613 kB). The installation progress is shown, including fetching the package and unpacking it.

```
student@workstation1-VirtualBox:~$ 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.  
Suggested packages:  
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs  
  git-mediawiki git-svn  
The following packages will be upgraded:  
  git  
1 upgraded, 0 newly installed, 0 to remove and 590 not upgraded.  
Need to get 4,613 kB of archives.  
After this operation, 365 kB of additional disk space will be used.  
Get:1 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-1ubuntu3.1  
4 [4,613 kB]  
Fetched 4,613 kB in 5s (997 kB/s)  
(Reading database ... 194374 files and directories currently installed.)  
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.14_amd64.deb ...  
Unpacking git (1:2.25.1-1ubuntu3.14) over (1:2.25.1-1ubuntu3.2) ...  
Setting up git (1:2.25.1-1ubuntu3.14) ...
```

Figure 5.2.1 Installing Git on Both Workstations.

To begin using GitHub for version control, a GitHub account must be created. On one of the workstations, navigate to www.github.com and sign up for an account. This account will be used to manage repositories and share configuration files between the workstations.

After signing into GitHub, a new repository is created by clicking the "New" button. The repository is named nislalab, and the option to initialize it with a README file is selected. Once these steps are completed, the repository is created. This repository will be used for storing and sharing configuration files between the workstations. See figure 5.2.2

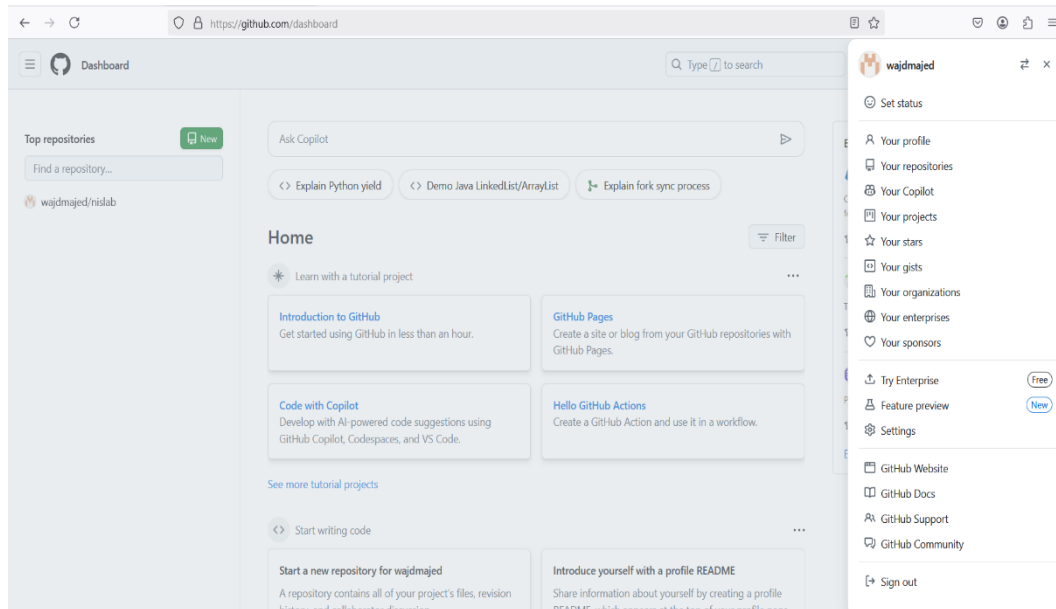


Figure 5.2.2 Signing Up for a GitHub Account & Creating a GitHub Repository.

Enabling SSH access to GitHub, go to "Settings" > "SSH and GPG keys," delete any existing keys, and click "New SSH Key." Title it "ansible," paste the contents of the ansible.pub public key, and click "Add SSH Key. See figure 5.2.3

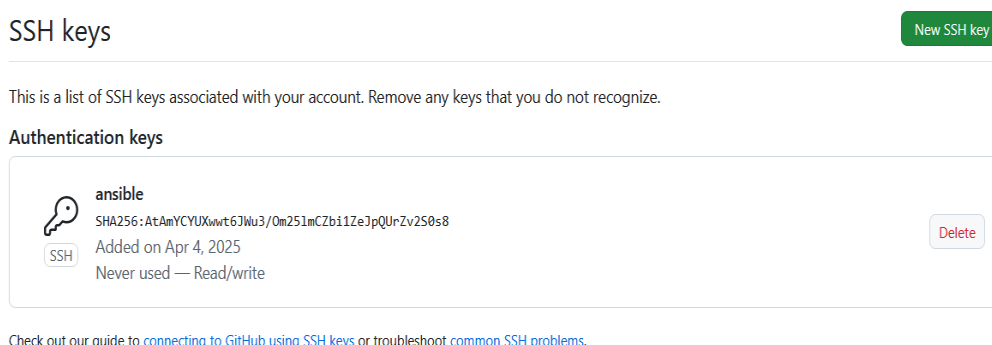


Figure 5.2.3 Adding an SSH Key to GitHub Account.

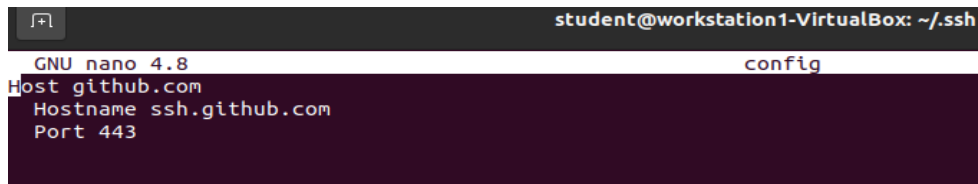
To configure the GitHub SSH host, a new file named config is created inside the .ssh directory on both workstations. The following contents are added to the file, see figure 5.2.4

```

nano config
inside the file write

Host github.com
Hostname ssh.github.com
Port 443

```



```

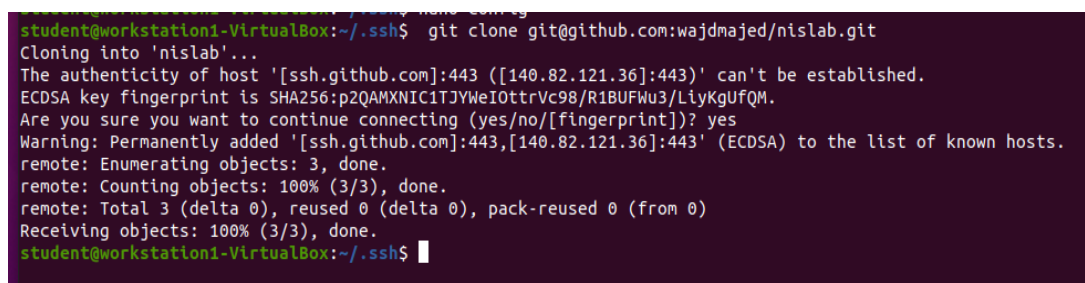
student@workstation1-VirtualBox: ~/.ssh
GNU nano 4.8 config
Host github.com
  Hostname ssh.github.com
  Port 443

```

Figure 5.2.4 Configuring the GitHub SSH Host on Workstations.

Clone the GitHub repository, open GitHub in a browser on both workstations, navigate to the repository, click the green "Code" button, and copy the SSH link. Then, use the following command to clone the repository into the home directory of both workstations. See figure 5.2.5

```
git clone [url copied]
```



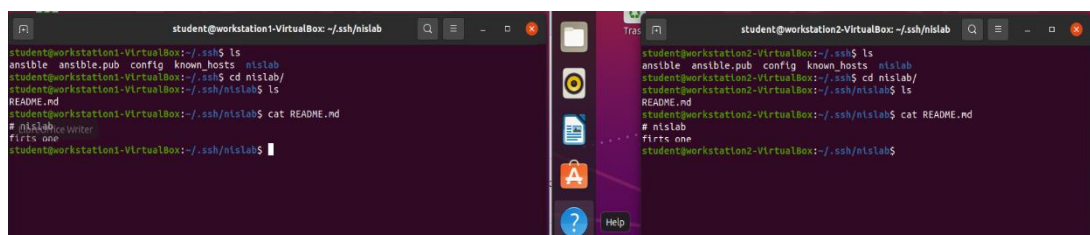
```

student@workstation1-VirtualBox: ~/.ssh$ git clone git@github.com:wajdmajed/nislab.git
Cloning into 'nislab'...
The authenticity of host '[ssh.github.com]:443 ([140.82.121.36]:443)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOtrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[ssh.github.com]:443,[140.82.121.36]:443' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
student@workstation1-VirtualBox: ~/.ssh$

```

Figure 5.2.5 Cloning the GitHub Repository to Workstations.

List the home directory contents on both workstations, nislab directory should be present on both of them. See figure 5.2.6



```

student@workstation1-VirtualBox: ~/.ssh/nislab$ ls
ansible  ansible.pub  config  known_hosts  nislab
student@workstation1-VirtualBox: ~/.ssh/nislab$ cd nislab/
student@workstation1-VirtualBox: ~/.ssh/nislab$ ls
README.md
student@workstation1-VirtualBox: ~/.ssh/nislab$ cat README.md
# nislab
# nislabe writer
first one
student@workstation1-VirtualBox: ~/.ssh/nislab$

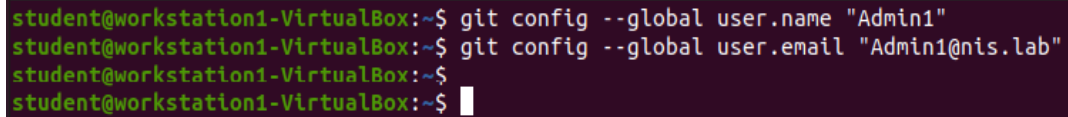
student@workstation2-VirtualBox: ~/.ssh/nislab$ ls
ansible  ansible.pub  config  known_hosts  nislab
student@workstation2-VirtualBox: ~/.ssh/nislab$ cd nislab/
student@workstation2-VirtualBox: ~/.ssh/nislab$ ls
README.md
student@workstation2-VirtualBox: ~/.ssh/nislab$ cat README.md
# nislab
# nislabe writer
first one
student@workstation2-VirtualBox: ~/.ssh/nislab$

```

Figure 5.2.6 Listing and Viewing the README.MD File in the "nislab" Directory.

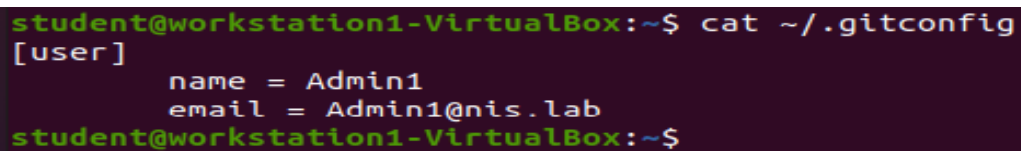
configuration the user identity for Git on both workstations, the following commands are executed. See figure 5.2.7

```
Workstation1$ git config --global user.name "Admin1"  
Workstation1$ git config --global user.email Admin1@nis.lab  
Workstation1$ cat ~/.gitconfig
```



```
student@workstation1-VirtualBox:~$ git config --global user.name "Admin1"  
student@workstation1-VirtualBox:~$ git config --global user.email "Admin1@nis.lab"  
student@workstation1-VirtualBox:~$  
student@workstation1-VirtualBox:~$
```

Figure 5.2.7 Configuring User ID for Git on Both Workstations.

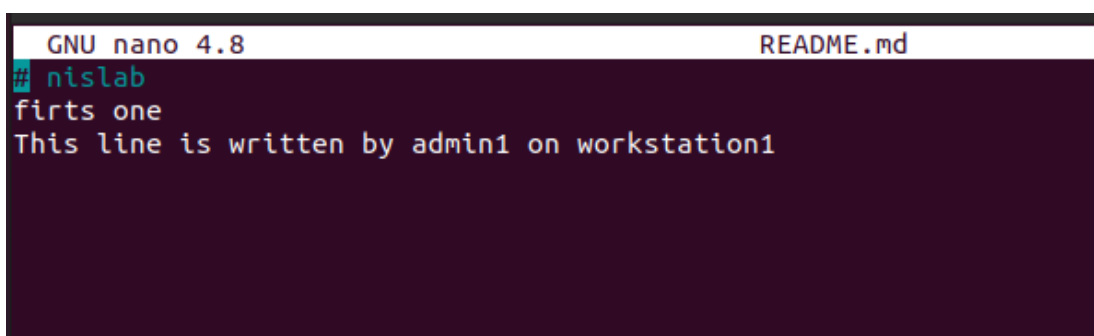


```
student@workstation1-VirtualBox:~$ cat ~/.gitconfig  
[user]  
    name = Admin1  
    email = Admin1@nis.lab  
student@workstation1-VirtualBox:~$
```

Figure 5.2.7 View the configuring User ID for Git on Workstations.

To test version control, the README.MD file is edited on Workstation1. A line is added to the end of the file, such as "This line is written by admin1 on workstation1." After saving and closing the file, the following Git commands are used. See figure 5.2.8

```
nano README.md  
inside the file:
```



```
GNU nano 4.8                                README.md  
# nislal  
firts one  
This line is written by admin1 on workstation1
```

Figure 5.2.8 Editing the README.md file.

This shows that a change has been made. See figure 5.2.9

```
git status
git diff
```

```
student@workstation1-VirtualBox: ~/.ssh/nislab$ nano README.md
student@workstation1-VirtualBox: ~/.ssh/nislab$
student@workstation1-VirtualBox: ~/.ssh/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@workstation1-VirtualBox: ~/.ssh/nislab$
student@workstation1-VirtualBox: ~/.ssh/nislab$ git diff
diff --git a/README.md b/README.md
index 6335b7e..655f87d 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
 # nislab
 firts one
+This line is written by admin1 on workstation1
student@workstation1-VirtualBox: ~/.ssh/nislab$
```

Figure 5.2.9 View change has been made before add the file.

The file is added to the staging area and Commit the changes with a message. See figure 5.2.10

```
Workstation1 ~/.nislab$ git add README.MD
Workstation1 ~/.nislab$ git commit -m "Admin1 edited readme
file"
```

```
student@workstation1-VirtualBox: ~/.ssh/nislab$ git add README.md
student@workstation1-VirtualBox: ~/.ssh/nislab$
student@workstation1-VirtualBox: ~/.ssh/nislab$ git status
On branch main
Your branch is up to date with 'origin/main'.

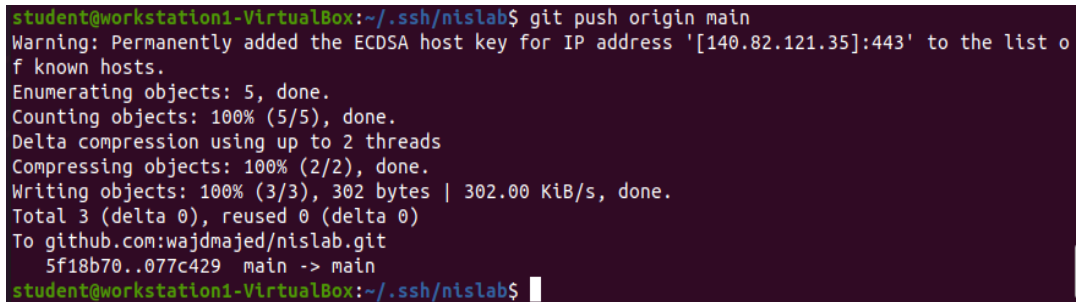
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   README.md

student@workstation1-VirtualBox: ~/.ssh/nislab$
student@workstation1-VirtualBox: ~/.ssh/nislab$ git commit -m "Admin1 edited readme f
[main 077c429] Admin1 edited readme file
1 file changed, 1 insertion(+)
student@workstation1-VirtualBox: ~/.ssh/nislab$
```

Figure 5.2.10 Add and commit the README.md file.

Push the changes to the remote repository. See figure 5.2.11

Workstation1 ~/nislalab\$ git push origin main

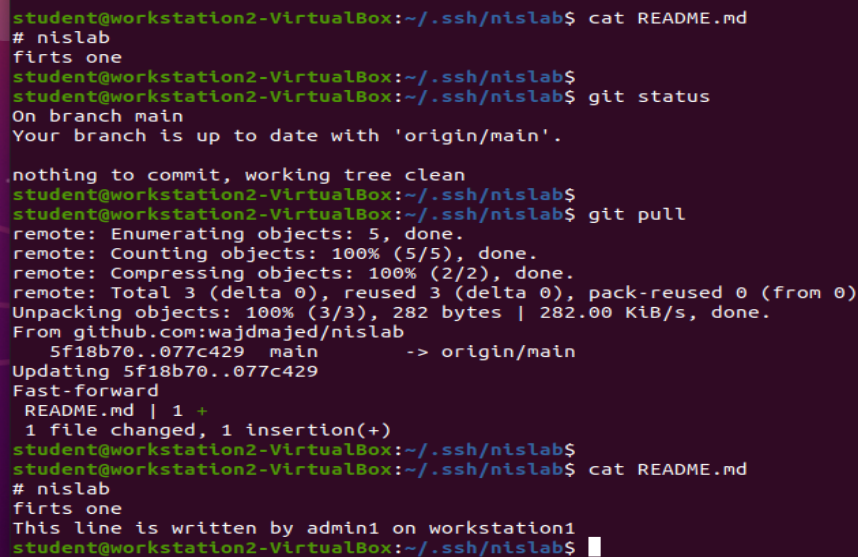


```
student@workstation1-VirtualBox:~/.ssh/nislalab$ git push origin main
Warning: Permanently added the ECDSA host key for IP address '[140.82.121.35]:443' to the list of known hosts.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:wajdmajed/nislalab.git
 5f18b70..077c429  main -> main
student@workstation1-VirtualBox:~/.ssh/nislalab$
```

Figure 5.2.11 Push the README.md file to GitHub repository.

After pushing the changes from Workstation1, the repository is updated online. However, Workstation2 is still out of sync. To confirm, the contents of the README.MD file are checked on Workstation2 using the following commands. See figure 5.2.12

Workstation2 ~/nislalab\$ git pull



```
student@workstation2-VirtualBox:~/.ssh/nislalab$ cat README.md
# nislalab
firts one
student@workstation2-VirtualBox:~/.ssh/nislalab$
student@workstation2-VirtualBox:~/.ssh/nislalab$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
student@workstation2-VirtualBox:~/.ssh/nislalab$
student@workstation2-VirtualBox:~/.ssh/nislalab$ 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 (from 0)
Unpacking objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
From github.com:wajdmajed/nislalab
 5f18b70..077c429  main      -> origin/main
Updating 5f18b70..077c429
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
student@workstation2-VirtualBox:~/.ssh/nislalab$
student@workstation2-VirtualBox:~/.ssh/nislalab$ cat README.md
# nislalab
firts one
This line is written by admin1 on workstation1
student@workstation2-VirtualBox:~/.ssh/nislalab$
```

Figure 5.2.12 Synchronizing Workstation2 with Remote Repository Changes.

The previous steps should be repeated, but with the adjustment from Workstation1 to Workstation2. See figure 5.2.13 and figure 5.2.14

```
student@workstation2-VirtualBox:~/.ssh/nislab$
student@workstation2-VirtualBox:~/.ssh/nislab$ nano README.md
student@workstation2-VirtualBox:~/.ssh/nislab$ git add README.md
student@workstation2-VirtualBox:~/.ssh/nislab$ git commit -m "Admin2 edited readme file"
[main 8a526bc] Admin2 edited readme file
 1 file changed, 2 insertions(+)
student@workstation2-VirtualBox:~/.ssh/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), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:wajdmajed/nislab.git
   077c429..8a526bc  main -> main
student@workstation2-VirtualBox:~/.ssh/nislab$
```

Figure 5.2.13 Editing README.md file on Workstation2 and push the file.

```
student@workstation1-VirtualBox:~/.ssh/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 (from 0)
Unpacking objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
From github.com:wajdmajed/nislab
   077c429..8a526bc  main      -> origin/main
Updating 077c429..8a526bc
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
student@workstation1-VirtualBox:~/.ssh/nislab$ cat README.md
# nislab
firts one
This line is written by admin1 on workstation1

Admin2 sayes hi
student@workstation1-VirtualBox:~/.ssh/nislab$
```

Figure 5.2.14 Synchronizing Workstation1 with Remote Repository Changes.

5.3 Ansible [ad-hoc commands]

Ansible ad-hoc commands allow for quick, one-time execution of tasks on remote servers without the need for writing a full playbook. These commands are particularly useful for running simple tasks such as installing packages, checking system information, or copying files across multiple systems. The commands are executed directly from the command line using ansible followed by the target hosts and the module to be used.

To install Ansible on both workstations, the following commands are executed:

```
sudo apt update
sudo apt install ansible
```


see figure 5.3.1

```
student@workstation2-VirtualBox:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libbftm2-tod1 liblvm10 linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ieee-data python3-argcomplete python3-crypto python3-distutils python3-dnspython
  python3-jinja2 python3-jmespath python3-kerberos python3-libcloud python3-netaddr
  python3-ntlm-auth python3-requests-kerberos python3-requests-ntlm python3-selinux
  python3-winrm python3-xmltodict
Suggested packages:
  cowsay sshpass python-jinja2-doc ipython3 python-netaddr-docs
The following NEW packages will be installed:
  ansible ieee-data python3-argcomplete python3-crypto python3-distutils
  python3-dnspython python3-jinja2 python3-jmespath python3-kerberos python3-libcloud
  python3-netaddr python3-ntlm-auth python3-requests-kerberos python3-requests-ntlm
  python3-selinux python3-winrm python3-xmltodict
0 upgraded, 17 newly installed, 0 to remove and 141 not upgraded.
Need to get 9,867 kB of archives.
After this operation, 92.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-jinja2 all 2.10.
1-2ubuntu0.6 [96.3 kB]
Get:2 http://ps.archive.ubuntu.com/ubuntu focal/main amd64 python3-crypto amd64 2.6.1-13ubu
ntu2 [237 kB]
Get:3 http://ps.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-distutils all 3.
```

Figure 5.3.1 Installing Ansible on Both Workstations.

On Workstation1, navigate to the nislalab directory and create an inventory file using the nano text editor. Add the IP addresses of SRVR01 and SRVR02, each on a separate line. See figure 5.3.2

```
student@workstation1-VirtualBox: ~/.ssh/nislalab
GNU nano 4.8 inventory
192.168.88.209
192.168.88.210
```

Figure 5.3.2 Creating and adding the ip's of servers to the Inventory file.

Add and push the file to GitHub and then pull it from Workstation2 to edit the file and add the SRVR03. See figure 5.3.3, figure 5.3.4 and figure 5.3.5

```
student@workstation2-VirtualBox:~/.ssh/nislalab$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
From github.com:wajdmajed/nislalab
  8a526bc..d6b6219  main      -> origin/main
Updating 8a526bc..d6b6219
Fast-forward
 inventory | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 inventory
student@workstation2-VirtualBox:~/.ssh/nislalab$ ls
inventory README.md
student@workstation2-VirtualBox:~/.ssh/nislalab$
```

Figure 5.3.3 Pull the changes from the repository.


```

student@workstation2-VirtualBox:~/.ssh/nislab$ nano inventory
student@workstation2-VirtualBox:~/.ssh/nislab$
student@workstation2-VirtualBox:~/.ssh/nislab$ git add inventory
student@workstation2-VirtualBox:~/.ssh/nislab$ git commit -m "Admin2 modified the inventory
and added SRVR03"
[main e62d8e4] Admin2 modified the inventory and added SRVR03
1 file changed, 1 insertion(+)
student@workstation2-VirtualBox:~/.ssh/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), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:wajdmajed/nislab.git
d6b6219..e62d8e4  main -> main
student@workstation2-VirtualBox:~/.ssh/nislab$

```

Figure 5.3.4 Save and exit the editor, then add and push the file to GitHub.

```

student@workstation1-VirtualBox:~/.ssh/nislab$ cat inventory
192.168.88.209
192.168.88.210
student@workstation1-VirtualBox:~/.ssh/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 (from 0)
Unpacking objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
From github.com:wajdmajed/nislab
d6b6219..e62d8e4  main -> origin/main
Updating d6b6219..e62d8e4
Fast-forward
 inventory | 1 +
1 file changed, 1 insertion(+)
student@workstation1-VirtualBox:~/.ssh/nislab$
student@workstation1-VirtualBox:~/.ssh/nislab$
student@workstation1-VirtualBox:~/.ssh/nislab$ cat inventory
192.168.88.209
192.168.88.210
192.168.88.212
student@workstation1-VirtualBox:~/.ssh/nislab$

```

Figure 5.3.5 Syncing Inventory File Between Workstations.

To run a command on all servers, the following Ansible command is executed on Workstation1:

```

Workstation1~/.nislab$ ansible all --key-file ~/.ssh/ansible -i
inventory -m ping

```

Upon successful connection to all servers, three success messages will appear, confirming that Ansible was able to connect to each server. See figure 5.3.6

```

student@workstation1-VirtualBox: ~/.nislab
student@workstation1-VirtualBox:~/.nislab$ ansible all --key-file ~/.ssh/ansible
-i inventory -m ping
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.88.212 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.88.210 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
student@workstation1-VirtualBox:~/.nislab$

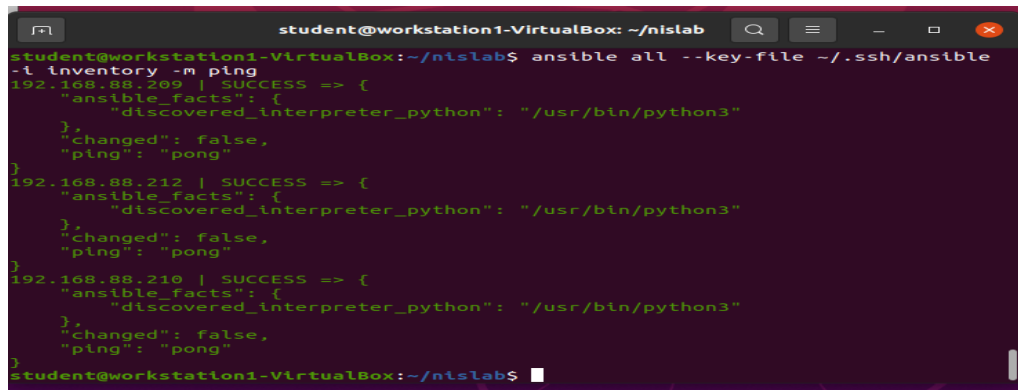
```

Figure 5.3.6 Running Ansible Ping Command to Test SSH Connection to All Servers.

To run a command on all servers, the following Ansible command is executed on Workstation1:

```
Workstation1 ~/nislalab$ ansible all --key-file ~/.ssh/ansible -i  
inventory -m ping
```

Upon successful connection to all servers, three success messages will appear, confirming that Ansible was able to connect to each server. See figure 5.3.6

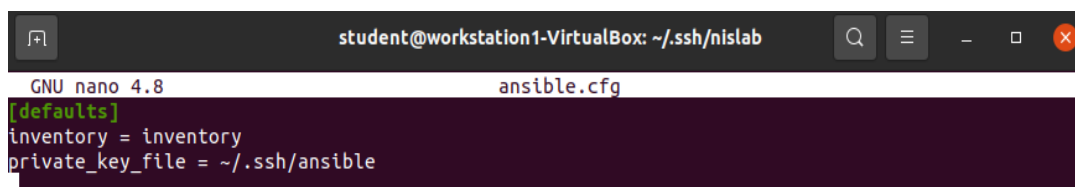


```
student@workstation1-VirtualBox: ~/nislalab  
student@workstation1-VirtualBox:~/nislalab$ ansible all --key-file ~/.ssh/ansible  
-i inventory -m ping  
192.168.88.209 | SUCCESS => {  
  "ansible_facts": {  
    "discovered_interpreter_python": "/usr/bin/python3"  
  },  
  "changed": false,  
  "ping": "pong"  
}  
192.168.88.212 | SUCCESS => {  
  "ansible_facts": {  
    "discovered_interpreter_python": "/usr/bin/python3"  
  },  
  "changed": false,  
  "ping": "pong"  
}  
192.168.88.210 | SUCCESS => {  
  "ansible_facts": {  
    "discovered_interpreter_python": "/usr/bin/python3"  
  },  
  "changed": false,  
  "ping": "pong"  
}  
student@workstation1-VirtualBox:~/nislalab$
```

Figure 5.3.6 Running Ansible Ping Command to Test SSH Connection to All Servers.

To shorten the Ansible command, we can store the inventory file name and SSH key path in a configuration file. This makes the command more concise and easier to run. Create and edit the ansible.cfg file. See figure 5.3.7 and figure 5.3.8

```
nano ansible.cfg  
inside the file write:  
[defaults]  
inventory = inventory  
private_key_file = ~/.ssh/ansible
```

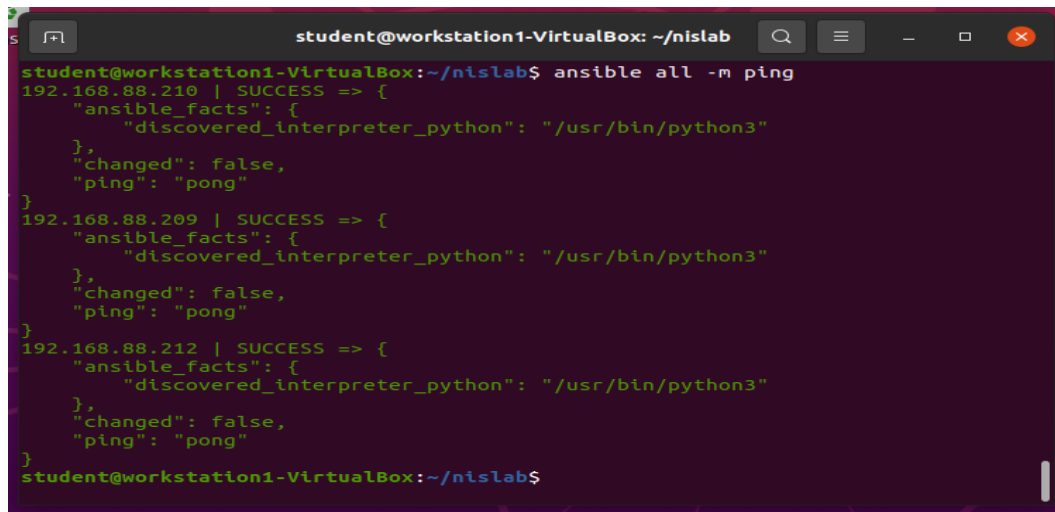


```
student@workstation1-VirtualBox: ~/.ssh/nislalab  
GNU nano 4.8 ansible.cfg  
[defaults]  
inventory = inventory  
private_key_file = ~/.ssh/ansible
```

Figure 5.3.7 ansible.cfg configuration.

Note: If Ansible commands are run outside of the nislalab directory, the default configuration files located in /etc/ansible will be used. To verify this, list the contents of that directory. However, since we are running commands within the nislalab directory, the ansible.cfg file in the

current directory will take priority, making it easier to manage different sites with different inventories and keys.

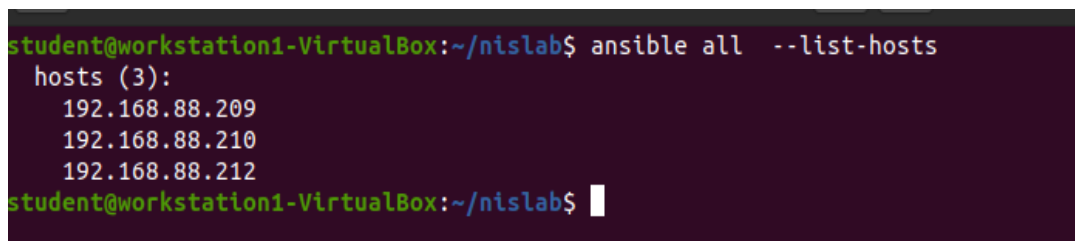
A terminal window titled 'student@workstation1-VirtualBox: ~/nislalab' showing the output of the command 'ansible all -m ping'. The output shows three hosts (192.168.88.210, 192.168.88.209, and 192.168.88.212) all successfully responding with 'pong'. Each host's output includes 'ansible_facts' and 'discovered_interpreter_python' details.

```
student@workstation1-VirtualBox: ~/nislalab$ ansible all -m ping
192.168.88.210 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.88.212 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
student@workstation1-VirtualBox:~/nislalab$
```

Figure 5.3.8 Shortening Ansible Commands with Configuration File.

The `ansible all --list-hosts` command lists all the hosts from the inventory file that Ansible is configured to manage. This command allows you to verify which servers are included in your inventory file and ensures that all the target machines are correctly defined for use in Ansible tasks. See figure 5.3.9

| `ansible all --list-hosts`

A terminal window titled 'student@workstation1-VirtualBox: ~/nislalab' showing the output of the command 'ansible all --list-hosts'. The output lists three hosts: 192.168.88.209, 192.168.88.210, and 192.168.88.212.

```
student@workstation1-VirtualBox:~/nislalab$ ansible all --list-hosts
hosts (3):
  192.168.88.209
  192.168.88.210
  192.168.88.212
student@workstation1-VirtualBox:~/nislalab$
```

Figure 5.3.9 Listing Hosts in Ansible Inventory.

The `ansible all -m gather_facts` command is used to gather and display system information (facts) from all the hosts in the inventory. Ansible uses this module to collect details such as the operating system, CPU architecture, memory, and network interfaces of the remote servers. This information is useful for making decisions in more complex playbooks.

| `ansible all -m gather_facts`

see figure 5.3.10, figure 5.3.11, and figure 5.3.12

```
student@workstation1-VirtualBox:~/nislabs$ ansible all -m gather_facts
192.168.88.210 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "192.168.88.210"
    ],
    "ansible_all_ipv6_addresses": [
      "fd00::a00:27ff:fe58:4b1d",
      "fe80::a00:27ff:fe58:4b1d"
    ],
    "ansible_apparmor": {
      "status": "enabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "12/01/2006",
    "ansible_bios_version": "VirtualBox",
    "ansible_cmdline": {
      "BOOT_IMAGE": "/vmlinuz-5.15.0-25-generic",
      "ro": true,
      "root": "/dev/mapper/ubuntu--vg-ubuntu--lv"
    },
    "ansible_date_time": {
      "date": "2025-04-09",
      "day": "09",
```

Figure 5.3.10 Gathering System Facts from All Hosts with Ansible.

```
student@workstation1-VirtualBox:~/nislabs$ ansible all -m gather_facts --limit 1
192.168.88.209
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "192.168.88.209"
    ],
    "ansible_all_ipv6_addresses": [
      "fd00::a00:27ff:fe8c:f62a",
      "fe80::a00:27ff:fe8c:f62a"
    ],
    "ansible_apparmor": {
      "status": "enabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "12/01/2006",
    "ansible_bios_version": "VirtualBox",
    "ansible_cmdline": {
      "BOOT_IMAGE": "/vmlinuz-5.15.0-25-generic",
      "ro": true,
      "root": "/dev/mapper/ubuntu--vg-ubuntu--lv"
    },
    "ansible_date_time": {
      "date": "2025-04-09",
```

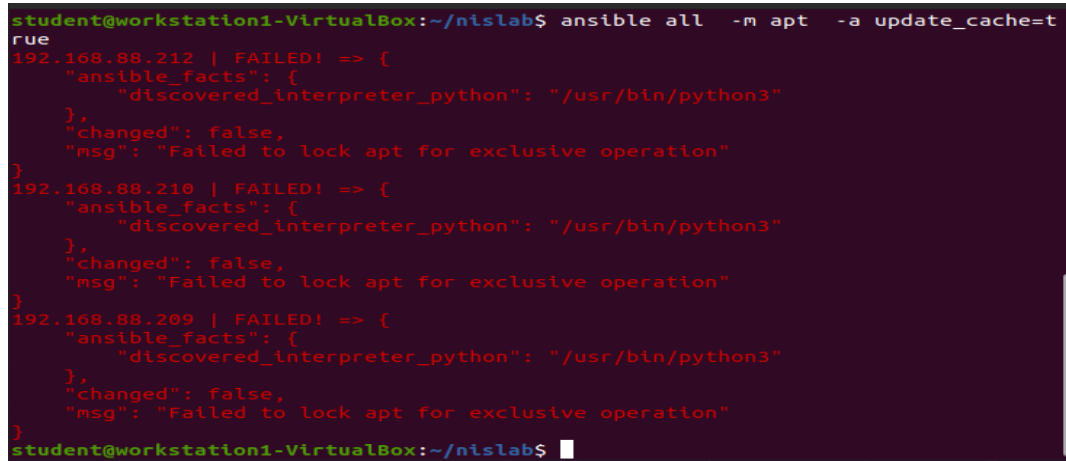
Figure 5.3.11 Gathering System Facts from a Specific Host Using Ansible.

```
student@workstation1-VirtualBox:~/nislabs$ ansible all -m gather_facts --limit 1
192.168.88.209 | 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": "22",
  "ansible_distribution_release": "jammy",
  "ansible_distribution_version": "22.04",
student@workstation1-VirtualBox:~/nislabs$
```

Figure 5.3.12 Filtering System Facts to Display OS Distribution on a Specific Host.

When running the `ansible all -m apt -a update_cache=true` command, it attempts to update the package cache on all hosts, but it will fail because the apt update command requires elevated privileges (i.e., sudo). See figure 5.3.13

```
Workstation1~/nislalab$ ansible all -m apt -a update_cache=true
```

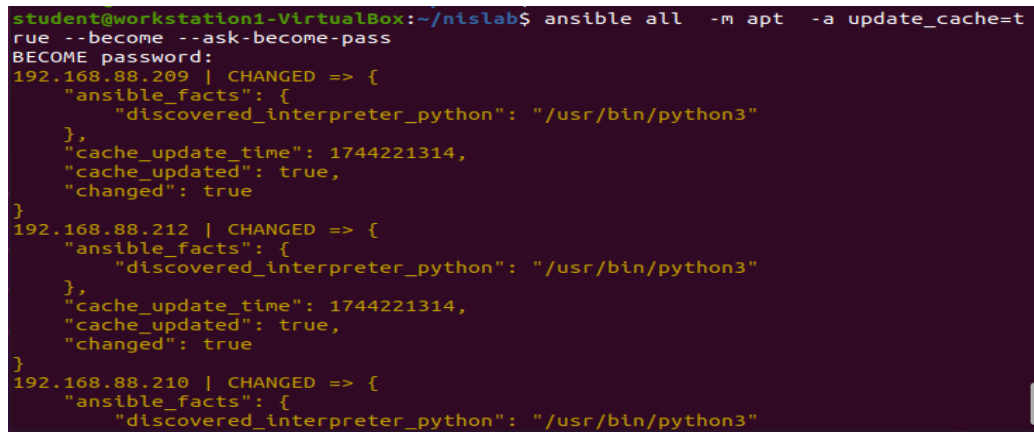
A terminal window with a dark purple background. The prompt is 'student@workstation1-VirtualBox:~/nislalab\$'. The command 'ansible all -m apt -a update_cache=true' has been executed. The output shows three failed attempts to update the apt cache on IP addresses 192.168.88.212, 192.168.88.210, and 192.168.88.209. Each failure message includes 'ansible_facts' with 'discovered_interpreter_python' and a 'msg' indicating 'Failed to lock apt for exclusive operation'.

```
student@workstation1-VirtualBox:~/nislalab$ ansible all -m apt -a update_cache=true
192.168.88.212 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation"
}
192.168.88.210 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation"
}
192.168.88.209 | FAILED! => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation"
}
student@workstation1-VirtualBox:~/nislalab$
```

Figure 5.3.13 Initial Command (Without Privileges).

To address this, Ansible needs to elevate the privileges for this command by using the `--become` option, which mimics `sudo`. Additionally, the `--ask-become-pass` option prompts for the `sudo` password. See figure 5.3.14

```
Workstation1~/nislalab$ ansible all -m apt -a update_cache=true --become --ask-become-pass
```

A terminal window with a dark purple background. The prompt is 'student@workstation1-VirtualBox:~/nislalab\$'. The command 'ansible all -m apt -a update_cache=true --become --ask-become-pass' has been executed. The output shows three successful updates of the apt cache on IP addresses 192.168.88.209, 192.168.88.212, and 192.168.88.210. Each success message includes 'ansible_facts' with 'discovered_interpreter_python', 'cache_update_time', 'cache_updated', and 'changed' set to true. A 'BECOME password:' prompt is visible before the first success message.

```
student@workstation1-VirtualBox:~/nislalab$ ansible all -m apt -a update_cache=true --become --ask-become-pass
BECOME password:
192.168.88.209 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744221314,
  "cache_updated": true,
  "changed": true
}
192.168.88.212 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744221314,
  "cache_updated": true,
  "changed": true
}
192.168.88.210 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744221314,
  "cache_updated": true,
  "changed": true
}
```

Figure 5.3.14 Running apt Command with Elevated Privileges in Ansible.

To install a package on all servers, we use the `ansible all -m apt -a "name=apache2" --become --ask-become-pass` command, which installs the Apache web server on all 3 servers. The command elevates privileges to run the installation using `--become` and prompts for the `sudo` password with `--ask-become-pass`.

```
Workstation1~/nislalab$ ansible all -m apt -a "name=apache2" --become --ask-become-pass
```

After executing this command, the Apache web server will be installed on all 3 servers. See figure 5.3.15 and figure 5.3.16

```
student@workstation1-VirtualBox:~/nislalab$ ansible all -m apt -a "name=apache2"
--become --ask-become-pass
BECOME password:
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
192.168.88.212 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
192.168.88.210 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
```

Figure 5.3.15 Installing Apache Web Server on All Servers Using Ansible.

```
student@server2:~$ curl http://127.0.0.1
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.d
td">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
      * {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }

      body, html {
        padding: 3px 3px 3px 3px;
      }
    </style>
  </head>
  <div style="text-align: center;">
    <img alt="Ubuntu logo" data-bbox="309 418 351 448" style="vertical-align: middle;"/>
    <br/>
    <div style="display: inline-block; width: 45%; vertical-align: middle;">
      <h1>Ubuntu</h1>
      <h2>Apache2</h2>
    </div>
    <div style="display: inline-block; width: 45%; vertical-align: middle;">
      <h1>It works</h1>
    </div>
  </div>
</html>
```

Figure 5.3.16 Test if the apache2 was installed by curl command on one of the servers.

To specifically manage the snapd package, we can use Ansible to ensure it is installed and updated to the latest version across all servers. See figure 5.3.17

```
Workstation1~/nislalab$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
```

```
student@workstation1-VirtualBox:~/nislalab$ ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass
BECOME password:
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
192.168.88.212 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
192.168.88.210 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1744220478,
  "cache_updated": false,
  "changed": false
}
```

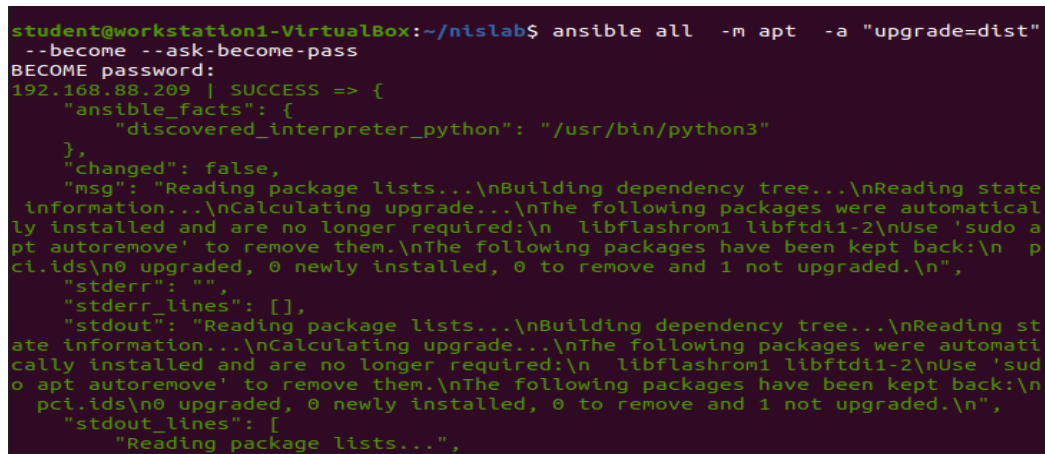
Figure 5.3.17 Installing and Updating the snapd Package Using Ansible.

This command will install snapd if it is not present and upgrade it to the latest version if it is already installed.

Now, to update all packages on all servers, we use the following Ansible command:

```
Workstation1 ~/nislalab$ ansible all -m apt -a "upgrade=dist" --become --ask-become-pass
```

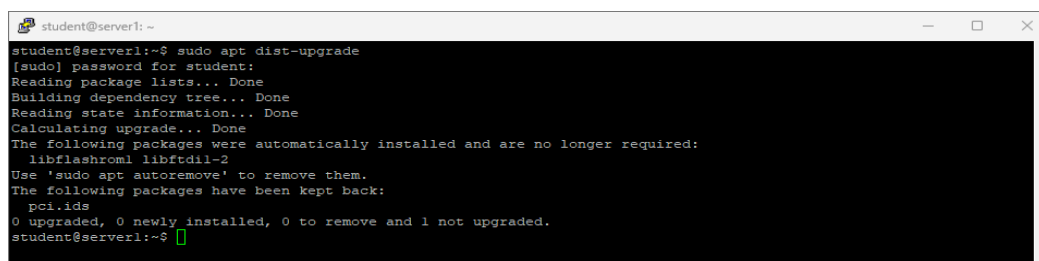
See figure 5.3.18, figure 5.3.19, and figure 5.3.20

A terminal window showing the execution of an Ansible command. The prompt is 'student@workstation1-VirtualBox:~/nislalab\$'. The command is 'ansible all -m apt -a "upgrade=dist" --become --ask-become-pass'. The output shows the Ansible process running on host 192.168.88.209, displaying facts, package lists, and dependency trees. It indicates that no packages were upgraded, newly installed, or removed, and one package was not upgraded. The output is color-coded with green for success and yellow for warnings.

```
student@workstation1-VirtualBox:~/nislalab$ ansible all -m apt -a "upgrade=dist"
--become --ask-become-pass
BECOME password:
192.168.88.209 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "msg": "Reading package lists...\nBuilding dependency tree...\nReading state
information...\nCalculating upgrade...\nThe following packages were automatical
ly installed and are no longer required:\n libflashrom1 libftdi1-2\nUse 'sudo a
pt autoremove' to remove them.\nThe following packages have been kept back:\n p
ci.ids\n0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.\n",
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nCalculating upgrade...\nThe following packages were automati
cally installed and are no longer required:\n libFlashrom1 libftdi1-2\nUse 'sud
o apt autoremove' to remove them.\nThe following packages have been kept back:\n
pci.ids\n0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.\n",
  "stdout_lines": [
    "Reading package lists...",

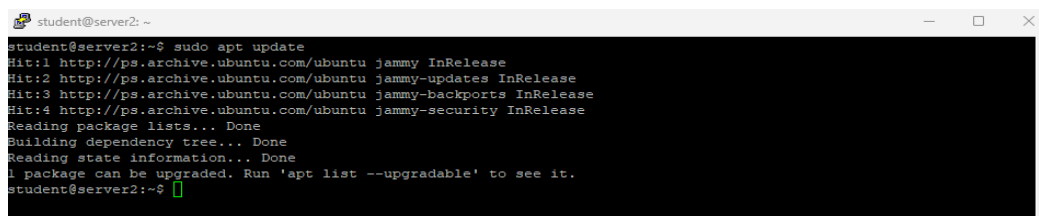
```

Figure 5.3.18 Updating All Packages on Servers Using Ansible.

A terminal window showing the execution of 'sudo apt dist-upgrade' on a server. The prompt is 'student@server1:~\$'. The output shows the process of reading package lists, building dependency trees, and calculating upgrades. It indicates that no packages were upgraded, newly installed, or removed, and one package was not upgraded. The output is color-coded with green for success and yellow for warnings.

```
student@server1:~$ sudo apt dist-upgrade
[sudo] password for student:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  pci.ids
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
student@server1:~$
```

Figure 5.3.19 sudo apt dist-upgrade on server to check if there are updates or not.

A terminal window showing the execution of 'sudo apt update' on a server. The prompt is 'student@server2:~\$'. The output shows the process of reading package lists, building dependency trees, and calculating upgrades. It indicates that one package can be upgraded. The output is color-coded with green for success and yellow for warnings.

```
student@server2:~$ sudo apt update
Hit:1 http://ps.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ps.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ps.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://ps.archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
student@server2:~$
```

Figure 5.3.20 sudo apt update on server to check if there are updates or not.

5.4 Package Management Using Ansible Playbooks

A task that we want to perform on our hosts using Ansible is called a play. A playbook contains one or more tasks that we want to execute. Playbooks are written in the YAML language, which is a human-readable data serialization format. YAML is commonly used for configuration files.

The real strength of Ansible comes from playbooks. When writing playbooks, we define the desired state of our servers and the specific tasks Ansible will perform to bring the servers to that state.

To create an Ansible playbook for installing Apache2, follow these steps:

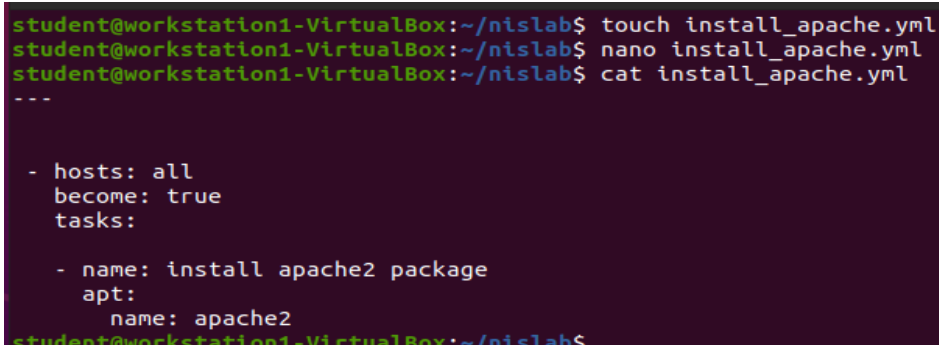
Navigate to the nislabs directory on Workstation1.

Create a file named install_apache.yml with the following contents

```
---

- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: apache2
```

see figure 5.4.1

A terminal window with a dark background and light green text. The prompt is 'student@workstation1-VirtualBox:~/nislabs\$'. The user enters three commands: 'touch install_apache.yml', 'nano install_apache.yml', and 'cat install_apache.yml'. The output of the 'cat' command is displayed, showing the YAML content of the playbook: '---', a blank line, '- hosts: all', ' become: true', ' tasks:', a blank line, '- name: install apache2 package', ' apt:', ' name: apache2', and a final blank line.

```
student@workstation1-VirtualBox:~/nislabs$ touch install_apache.yml
student@workstation1-VirtualBox:~/nislabs$ nano install_apache.yml
student@workstation1-VirtualBox:~/nislabs$ cat install_apache.yml
---

- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: apache2

```

Figure 5.4.1 Creating an Ansible Playbook to Install Apache2.

To run the playbook file, use the following command:

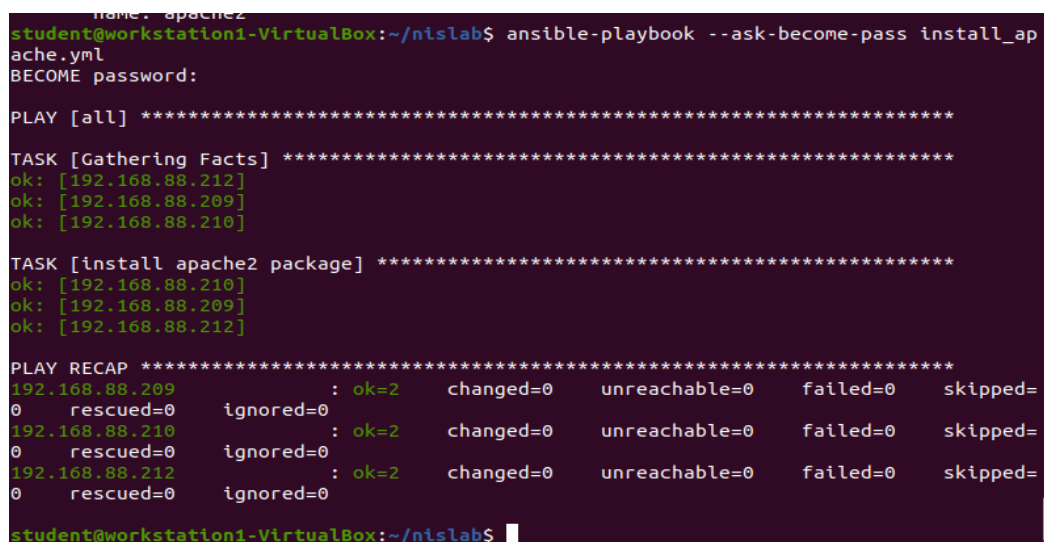
```
| ansible-playbook --ask-become-pass install_apache.yml
```

Potential Issues: The playbook may either succeed or fail depending on the repository index status. Before installing packages, it is important to update the repository index on Linux systems. If the repository index is outdated, you might receive an error indicating that the package was not found. This typically occurs because URLs for the packages are frequently updated and require the latest links to successfully download the packages.

Output Details: When running the playbook on each host, you'll encounter several key pieces of information:

- ok: The number of tasks that completed successfully without any issues.
- changed: The number of tasks that made changes on the host.
- unreachable: Indicates that the host is unreachable or offline.
- failed: The number of tasks that failed to execute on the host.
- skipped: The number of tasks skipped because the host did not meet the necessary conditions to run the task.
- rescued: The number of tasks that ran as a rescue after other tasks failed.
- ignored: The number of tasks that were ignored during execution

see figure 5.4.2



```
name: apache2
student@workstation1-VirtualBox:~/nislabs$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.212]
ok: [192.168.88.209]
ok: [192.168.88.210]

TASK [install apache2 package] *****
ok: [192.168.88.210]
ok: [192.168.88.209]
ok: [192.168.88.212]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
student@workstation1-VirtualBox:~/nislabs$
```

Figure 5.4.2 Running Ansible Playbook and Interpreting the Output.

To update the repository index in Ansible, modify the playbook to include the following task:

```
| ---
```

```

- hosts: all
  become: true
  tasks:

- name: update repository index
  apt:
    update_cache: yes

- name: install apache2 package
  apt:
    name: apache2

```

see figure 5.4.3 and figure 5.4.4

```

student@workstation1-VirtualBox:~/nislalab$ nano install_apache.yml
student@workstation1-VirtualBox:~/nislalab$ cat install_apache.yml
---

- hosts: all
  become: true
  tasks:

- name: update repository index
  apt:
    update_cache: yes

- name: install apache2 package
  apt:
    name: apache2
student@workstation1-VirtualBox:~/nislalab$

```

Figure 5.4.3 Modified Ansible Playbook for Repository Update and Package Installation.

```

student@workstation1-VirtualBox:~/nislalab$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.209]
ok: [192.168.88.210]
ok: [192.168.88.212]

TASK [update repository index] *****
changed: [192.168.88.209]
changed: [192.168.88.212]
changed: [192.168.88.210]

TASK [install apache2 package] *****
ok: [192.168.88.209]
ok: [192.168.88.212]
ok: [192.168.88.210]

PLAY RECAP *****
192.168.88.209      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Figure 5.4.4 Output of Ansible Playbook Execution Showing Tasks and Changes.

To install multiple packages in a single task, you can modify the playbook as follows:

```
---
```

```

- hosts: all
  become: true
  tasks:

- name: install Apache2 and PHP packages
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes

```

See figure 5.4.5 and figure 5.4.6

```

student@workstation1-VirtualBox:~/nislabs$ nano install_apache.yml
student@workstation1-VirtualBox:~/nislabs$ cat install_apache.yml
---

- hosts: all
  become: true
  tasks:

- name: install Apache2 and PHP packages
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes
student@workstation1-VirtualBox:~/nislabs$

```

Figure 5.4.5 Ansible Playbook for Installing Multiple Packages Without Updating.

```

student@workstation1-VirtualBox:~/nislabs$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]

TASK [install Apache2 and PHP packages] *****
changed: [192.168.88.209]
changed: [192.168.88.212]
changed: [192.168.88.210]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Figure 5.4.6 Executing Ansible Playbook to Install Apache2 and PHP Packages.

What Happens When the Playbook is Run:

- The playbook will update the repository cache (apt update), then attempt to install the apache2 and libapache2-mod-php packages.
- If these packages are not already installed, they will be installed. However, if they are already present on the servers, the playbook will not update them.

To modify the playbook so that it installs and updates packages to the latest version, we add the state: latest parameter. This ensures that the packages are always the most up-to-date versions available.

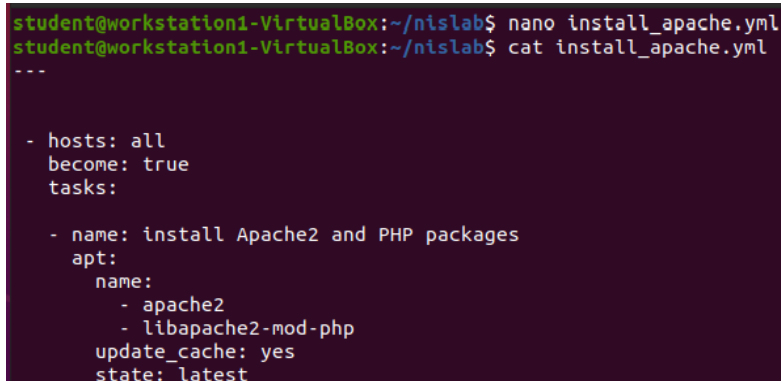
Here is the updated playbook:

```
---

- hosts: all
  become: true
  tasks:

  - name: install Apache2 and PHP packages
    apt:
      name:
        - apache2
        - libapache2-mod-php
      update_cache: yes
      state: latest
```

See figure 5.4.7, figure 5.4.8, and figure 5.4.9



```
student@workstation1-VirtualBox:~/nislabs$ nano install_apache.yml
student@workstation1-VirtualBox:~/nislabs$ cat install_apache.yml
---

- hosts: all
  become: true
  tasks:

  - name: install Apache2 and PHP packages
    apt:
      name:
        - apache2
        - libapache2-mod-php
      update_cache: yes
      state: latest
```

Figure 5.4.7 Ansible Playbook with State Parameter to Ensure Latest Package Versions.

What Happens When the Playbook is Run:

- The playbook will ensure that the latest versions of the apache2 and libapache2-mod-php packages are installed or updated on the target servers.
- After execution, you can verify that Apache is running by accessing the default Apache start page through a web browser, using the IP address of one of the servers

```
student@workstation1-VirtualBox:~/nislabs$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.212]
ok: [192.168.88.210]
ok: [192.168.88.209]

TASK [Install Apache2 and PHP packages] *****
ok: [192.168.88.210]
ok: [192.168.88.209]
ok: [192.168.88.212]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

student@workstation1-VirtualBox:~/nislabs$
```

Figure 5.4.8 Executing the Ansible Playbook to Install and Update Apache2 and PHP.

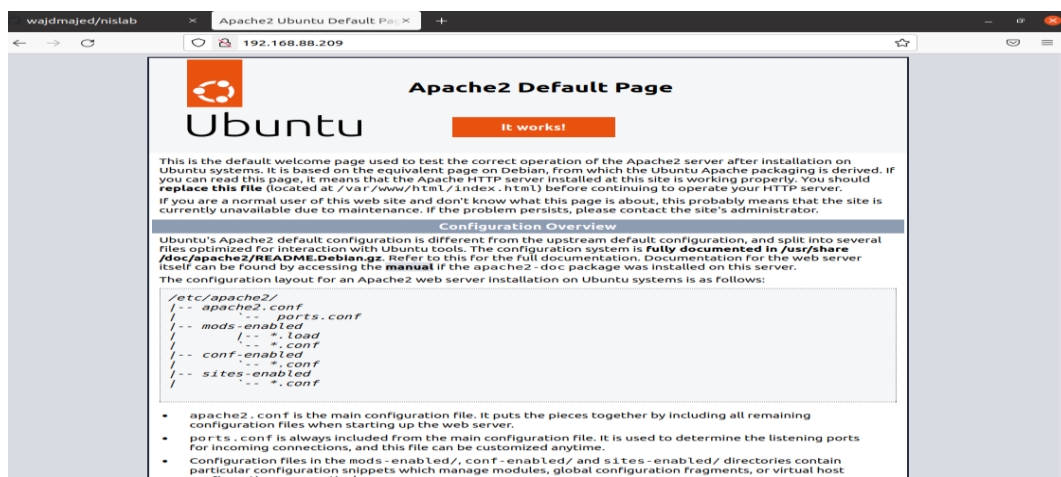


Figure 5.4.9 Check if the apache2 was installed in the servers by http://[ip_server].

To remove the Apache2 and PHP packages from all target servers, we create a new playbook called remove_apache.yml. This playbook will uninstall the apache2 and libapache2-mod-php packages if they are currently installed.

Here is the contents of the remove_apache.yml playbook:

```
---
```

```

- hosts: all
  become: true
  tasks:
    - name: remove Apache2 and PHP packages
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: absent

```

state: absent: This parameter ensures that the specified packages (apache2 and libapache2-mod-php) are removed from the servers if they are currently installed. See figure 5.4.10 and figure 5.4.11

```

student@workstation1-VirtualBox:~/nislalab$ touch remove_apache.yml
student@workstation1-VirtualBox:~/nislalab$ nano remove_apache.yml
student@workstation1-VirtualBox:~/nislalab$ cat remove_apache.yml
---
- hosts: all
  become: true
  tasks:
    - name: remove Apache2 and PHP packages
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: absent
student@workstation1-VirtualBox:~/nislalab$ █

```

Figure 5.4.10 Ansible Playbook to Remove Apache2 and PHP Packages.

What Happens When the Playbook is Run:

- The playbook will attempt to remove both the apache2 and libapache2-mod-php packages from all target servers.
- After running the playbook, the packages will be uninstalled from the servers

```

student@workstation1-VirtualBox:~/nislalab$ ansible-playbook --ask-become-pass remove_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]

TASK [remove Apache2 and PHP packages] *****
changed: [192.168.88.210]
changed: [192.168.88.212]
changed: [192.168.88.209]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
student@workstation1-VirtualBox:~/nislalab$

```

Figure 5.4.11 Executing the Playbook to Install and Update Apache2 and PHP Packages.

Open the browser on the workstation and write [http://\[ip_server\]](http://[ip_server]). See figure 5.4.12

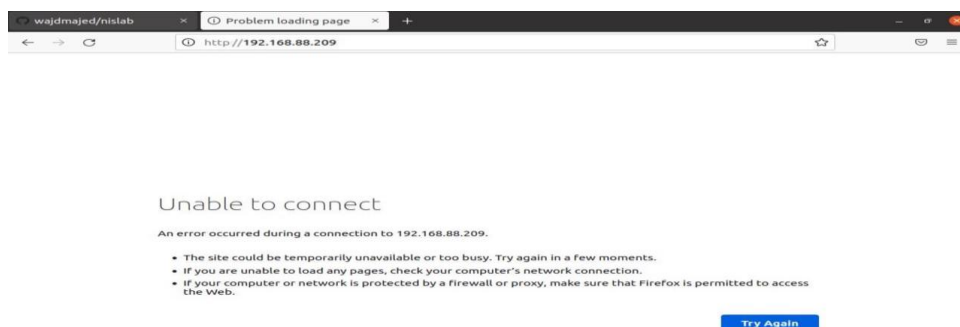


Figure 5.4.12 Check if there is an apache2 on one of the servers or not.

Run the playbook to install it again and test the ip again. See figure 5.4.13 and figure 5.4.14

```
student@workstation1-VirtualBox:~/nislalab$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]

TASK [Install Apache2 and PHP packages] *****
changed: [192.168.88.210]
changed: [192.168.88.212]
changed: [192.168.88.209]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 5.4.13 Reinstall the packages by running the playbook again.

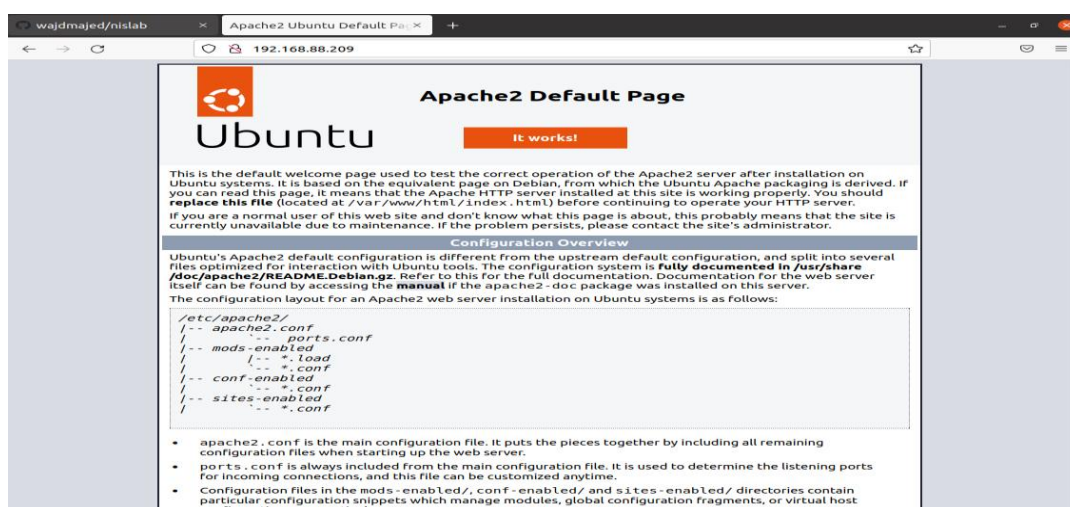


Figure 5.4.14 Check if there is an apache2 on one of the servers after repeat the playbook

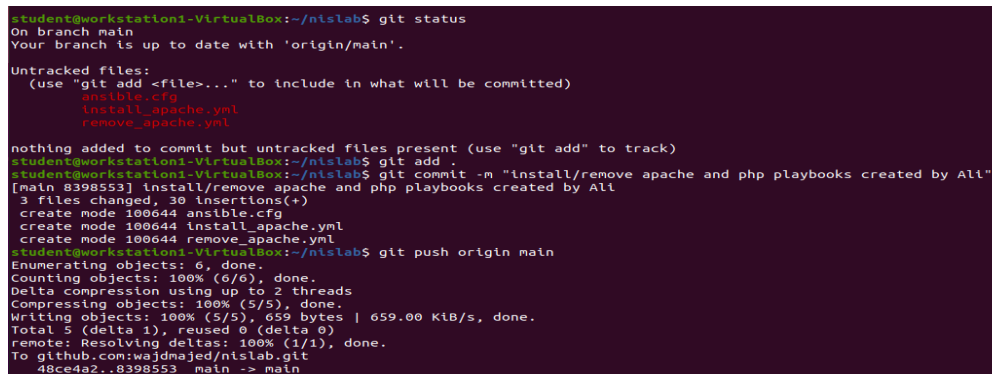
5.5 Version Control with Git

In this step, we add the changes made to our nislabs directory (which is connected to a GitHub repository) to version control. This ensures that the playbooks for installing and removing Apache and PHP packages are tracked and shared between the workstations.

Here are the commands used to commit the changes and push them to the remote GitHub repository

```
nislabs$ git status
nislabs$ git add .
nislabs$ git commit -m "install/remove apache and php playbooks created by admin 1"
nislabs$ git push origin main
```

see figure 5.5.1



```
student@workstation1-VirtualBox:~/nislabs$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ansible.cfg
        install_apache.yml
        remove_apache.yml

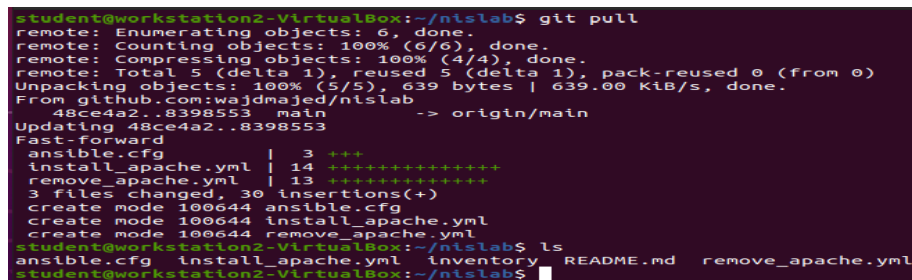
nothing added to commit but untracked files present (use "git add" to track)
student@workstation1-VirtualBox:~/nislabs$ git add .
student@workstation1-VirtualBox:~/nislabs$ git commit -m "install/remove apache and php playbooks created by Ali"
[main 8398553] install/remove apache and php playbooks created by Ali
3 files changed, 30 insertions(+)
create mode 100644 ansible.cfg
create mode 100644 install_apache.yml
create mode 100644 remove_apache.yml
student@workstation1-VirtualBox:~/nislabs$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 639 bytes | 639.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:wajdmajed/nislabs.git
48ce4a2..8398553  main -> main
```

Figure 5.5.1 Pushing Changes to GitHub.

After pushing the changes to GitHub, go to Workstation2 and run the following command to download the changes:

```
git pull
```

see figure 5.5.2



```
student@workstation2-VirtualBox:~/nislabs$ git pull
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 5 (delta 1), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 639 bytes | 639.00 KiB/s, done.
From github.com:wajdmajed/nislabs
48ce4a2..8398553  main      -> origin/main
Updating 48ce4a2..8398553
Fast-forward
 ansible.cfg      | 3 +++
 install_apache.yml | 14 ++++++++
 remove_apache.yml | 13 ++++++++
 3 files changed, 30 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 install_apache.yml
 create mode 100644 remove_apache.yml
student@workstation2-VirtualBox:~/nislabs$ ls
ansible.cfg  install_apache.yml  inventory  README.md  remove_apache.yml
student@workstation2-VirtualBox:~/nislabs$
```

Figure 5.5.2 Pulling changes on Workstation2.

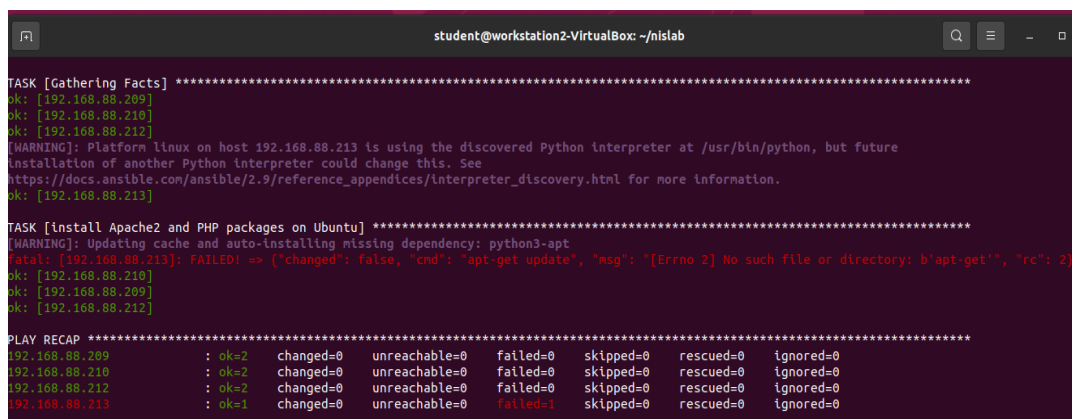
5.6 The 'when' Conditional

The Ansible playbook we created will work properly on Debian-based systems because it uses the apt module to install or remove packages. However, if some of the servers are running non-Debian-based systems (e.g., Red Hat, CentOS), the playbook will fail because the apt module is not applicable to those systems.

To handle such cases, we can use the when conditional in Ansible. The when statement allows us to specify conditions under which specific tasks will be executed. This way, we can ensure that tasks, such as package installation or removal, only run on compatible systems.

For example, we can modify our playbook to use different package management modules (like yum for Red Hat-based systems) based on the distribution of the host

On Workstation2, edit the inventory file and add the IP address of the AlmaLinux server. And run the playbook for installation. See figure 5.6.1



```
student@workstation2-VirtualBox: ~/nislalab

TASK [Gathering Facts] *****
ok: [192.168.88.209]
ok: [192.168.88.210]
ok: [192.168.88.212]
[WARNING]: Platform linux on host 192.168.88.213 is using the discovered Python interpreter at /usr/bin/python, but 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: [192.168.88.213]

TASK [install Apache2 and PHP packages on Ubuntu] *****
[WARNING]: Updating cache and auto-installing missing dependency: python3-apt
Fatal: [192.168.88.213]: FAILED! => ["changed": false, "cmd": "apt-get update", "msg": "[Errno 2] No such file or directory: b'apt-get'", "rc": 2]
ok: [192.168.88.210]
ok: [192.168.88.209]
ok: [192.168.88.212]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.213      : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Figure 5.6.1 Running the Playbook on AlmaLinux and Observing the Output.

To address the issue of applying tasks specific to certain distributions, we modify the playbook to include a conditional statement (when). This ensures that tasks, such as installing Apache and PHP packages, are only executed on Ubuntu servers and skipped on others, like AlmaLinux:

```
---
```

```

- hosts: all
  become: true
  tasks:

- name: install Apache2 and PHP packages
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes
    state: latest
    when: ansible_distribution == "Ubuntu"

```

See figure 5.6.2 and figure 5.6.3

```

student@workstation2-VirtualBox:~/nislalab$ nano install_apache.yml
student@workstation2-VirtualBox:~/nislalab$ cat install_apache.yml
---
- hosts: all
  become: true
  tasks:

- name: install Apache2 and PHP packages on Ubuntu
  apt:
    name:
      - apache2
      - libapache2-mod-php
    update_cache: yes
    state: latest
    when: ansible_distribution == "Ubuntu"
student@workstation2-VirtualBox:~/nislalab$

```

Figure 5.6.2 When conditional was added.

```

student@workstation2-VirtualBox:~/nislalab$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]
[WARNING]: Platform linux on host 192.168.88.213 is using the discovered Python interpreter at /usr/bin/python, but 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: [192.168.88.213]

TASK [install Apache2 and PHP packages on Ubuntu] *****
skipping: [192.168.88.213]
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.88.213      : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

student@workstation2-VirtualBox:~/nislalab$

```

Figure 5.6.3 Running the Ansible Playbook to Install Apache and PHP.

To ensure compatibility with both Ubuntu and AlmaLinux systems, we modify the Ansible playbook to handle the installation of Apache and PHP for each respective distribution. The playbook has two tasks:

- Installing Apache2 and PHP on Ubuntu: This task uses the apt module to install Apache2 (apache2) and PHP (libapache2-mod-php) on Ubuntu-based systems. The task is conditioned to only run on Ubuntu servers by using the when: ansible_distribution == "Ubuntu" condition.
- Installing HTTPD and PHP on AlmaLinux: This task uses the dnf module to install Apache HTTP Server (httpd) and PHP (php) on AlmaLinux servers. Similarly, the task is conditioned to only run on AlmaLinux systems with the when: ansible_distribution == "AlmaLinux" condition.

The Updated Playbook:

```

---

- hosts: all
  become: true
  tasks:
    - name: install Apache2 and PHP packages on Ubuntu
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        update_cache: yes
        when: ansible_distribution == "Ubuntu"
    - name: install httpd and PHP packages on AlmaLinux
      dnf:
        name:
          - httpd
          - php
        state: latest
        update_cache: yes
        when: ansible_distribution == "AlmaLinux"

```

See figure 5.6.4 and figure 5.6.5

```
student@workstation2-VirtualBox:~/nislabs$ cat install_apache.yml
---
- hosts: all
  become: true
  tasks:
    - name: install Apache2 and PHP packages on Ubuntu
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        update_cache: yes
        when: ansible_distribution == "Ubuntu"

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

student@workstation2-VirtualBox:~/nislabs$
```

Figure 5.6.4 Modified Playbook to Install Apache and PHP on Ubuntu and AlmaLinux Servers

```
student@workstation2-VirtualBox:~/nislabs$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.88.210]
ok: [192.168.88.212]
ok: [192.168.88.209]
[WARNING]: Platform linux on host 192.168.88.213 is using the discovered Python interpreter at
/usr/bin/python, but 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: [192.168.88.213]

TASK [install Apache2 and PHP packages on Ubuntu] *****
skipping: [192.168.88.213]
ok: [192.168.88.209]
ok: [192.168.88.210]
ok: [192.168.88.212]

TASK [install httpd and PHP packages on AlmaLinux] *****
skipping: [192.168.88.209]
skipping: [192.168.88.210]
skipping: [192.168.88.212]
changed: [192.168.88.213]

PLAY RECAP *****
192.168.88.209      : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.88.210      : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.88.212      : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.88.213      : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

student@workstation2-VirtualBox:~/nislabs$
```

Figure 5.6.5 Running the Ansible Playbook after the updates on the file.

What Happens When the Playbook is Run:

- For Ubuntu servers: It will install apache2 and libapache2-mod-php.
- For AlmaLinux servers: It will install httpd and php.
- If the when conditions are met, you should see the task labeled as ok or changed for each host.
- For hosts that do not meet the conditions (e.g., an AlmaLinux server running the Ubuntu tasks), you should see that the task is skipped.

6 Results and Observations

- SSH setup was successful, enabling password-less access.
- Git version control facilitated collaborative editing with synchronized updates between workstations.
- Ansible ad-hoc commands executed successfully, confirming correct inventory and SSH configuration.
- Playbooks deployed applications consistently and selectively based on OS type.
- Conditional logic (when) allowed cross-platform compatibility.
- Installing Apache resulted in accessible web servers on Ubuntu machines.
- Removing Apache was reflected immediately on browser refresh

7 Conclusion

This lab successfully demonstrated the integration of OpenSSH, Git, and Ansible to automate system administration tasks. The hands-on experience illustrated the importance of secure communication, collaborative workflows, and efficient configuration management. Moreover, the implementation of conditional logic in playbooks highlighted the flexibility of automation tools in heterogeneous environments. These foundational skills are critical for modern system administrators aiming to maintain reliable and scalable infrastructures.

8 References

- Ansible Documentation. Retrieved from: <https://docs.ansible.com/ansible/latest/modules>
- GitHub Help. Retrieved from: <https://docs.github.com/>
- Ubuntu SSH Guide. Retrieved from: <https://help.ubuntu.com/community/SSH/OpenSSH/Configuring>
- YAML Language Reference. Retrieved from: <https://yaml.org/>

9 Appendices

- List of executed Git and Ansible [ad-hoc] commands.
- Playbook configuration.
- Screenshots (if applicable).