

**VIETNAM NATIONAL UNIVERSITY HOCHIMINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORKS AND
COMMUNICATIONS**

DOAN HAI DANG – TRAN THI MY HUYEN – PHAN THI HONG NHUNG

**NETWORKS AND SYSTEMS
ADMINISTRATION PROJECT**

**NAT + DHCP ON LINUX SERVER
FILE SERVER: NFS, SMB, FTP ON LINUX**

CLASS: NT132.O11.ATCL

HO CHI MINH CITY, 2023

**NATIONAL UNIVERSITY HOCHIMINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORKS AND
COMMUNICATIONS**

**DOAN HAI DANG – 21520679
TRAN THI MY HUYEN – 21520269
PHAN THI HONG NHUNG - 21521250**

**NETWORKS AND SYSTEMS
ADMINISTRATION PROJECT**

**NAT + DHCP ON LINUX SERVER
FILE SERVER: NFS, SMB, FTP ON LINUX**

CLASS: NT132.O11.ATCL

**PROJECT ADVISOR
MSc TRAN THI DUNG**

HO CHI MINH CITY, 2023

ACKNOWLEDGMENTS

First and foremost, we would like to express our gratitude to our advisor, MSc Tran Thi Dung, for her guidance and consistent supervision, as well as for providing crucial project information and assisting us in completing the research.

Our gratitude and appreciation also extend to our colleagues and lecturers who assisted us in the development of the project, as well as to those who have volunteered their time and skills to assist us.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	i
TABLE OF CONTENTS.....	ii
LIST OF FIGURES	iv
LIST OF TABLES	vii
Chapter 1 Introduction.....	1
1.1 General information	1
1.1.1 Linux Server	1
1.1.2 NAT and DHCP network services.....	1
1.1.3 File Server on Linux.....	2
1.1.4 Ansible	3
1.2 Component	3
1.2.1 NAT and DHCP	3
1.2.2 File-sharing protocols: NFS, SMB, FTP	5
1.2.3 Ansible	6
1.3 Operation	7
1.3.1 NAT	7
1.3.2 DHCP.....	7
1.3.3 File-sharing protocols: NFS, SMB, FTP	9
1.3.4 Ansible	11
Chapter 2 Implementation	12
2.1 Topology	12
2.2 Installation.....	12
2.2.1 Server side.....	12
2.2.2 Client side	14
2.2.3 Install packages on Server using Ansible	15
2.3 Configuration	16

2.3.1 DHCP	16
2.3.2 NAT	18
2.3.3 NFS	20
2.3.4 SMB	22
2.3.5 FTP	24
2.3.6 Ansible	27
Chapter 3 Result and Conclusion	29
3.1 Result	29
3.1.1 DHCP	29
3.1.2 NAT	29
3.1.3 NFS	30
3.1.4 SMB	31
3.1.5 FTP	34
3.2 Conclusion	35
REFERENCES	36
APPENDIX	38
Task Assignment	38
Self Assessment	38

LIST OF FIGURES

Figure 1: Static NAT	3
Figure 2: Dynamic NAT	4
Figure 3: Port Address Translation (PAT)	4
Figure 4: DHCP Components	5
Figure 5: Ansible components	6
Figure 6: NAT operation	7
Figure 7: DHCP operation	7
Figure 8: NFS operation	9
Figure 9: SMB operation	9
Figure 10: FTP Active mode	11
Figure 11: FTP Passive mode	11
Figure 12: Ansible operation	11
Figure 13: Project topology	12
Figure 14: Install NAT: iptables	12
Figure 15: Install NAT: iptables-persistent [1]	13
Figure 16: Install NAT: iptables-persistent [2]	13
Figure 17: Install isc-dhcp-server	13
Figure 18: Install nfs-kernel-server	13
Figure 19: Install samba on server	14
Figure 20: Install vsftpd on server	14
Figure 21: Install nfs-common on client	14
Figure 22: Install cifs-utils on client	14
Figure 23: Install filezilla on client	15
Figure 24: Install server's packages using Ansible	15
Figure 25: Ubuntu server netplan	16
Figure 26: Ubuntu server network interfaces	16
Figure 27: Define DHCP interfaces	17
Figure 28: DHCP configuration	17
Figure 29: DHCP server status	18
Figure 30: Enable forwarding in sysctl.conf	18
Figure 31: Enables the changes	18
Figure 32: Iptables policies	19
Figure 33: Network interfaces	19

Figure 34: Save iptables policies.....	20
Figure 35: NFS Server status.....	20
Figure 36: Create root NFS directory.....	20
Figure 37: Set NFS folder permissions	21
Figure 38: Define access for 192.168.23.11	21
Figure 39: NFS Server status after restarting	22
Figure 40: Create Group and User in SMB Server.....	22
Figure 41: SMB Server status	22
Figure 42: Create root SMB directory and change permissions	22
Figure 43: Edit smb.conf file [1]	23
Figure 44: Edit smb.conf file [2]	23
Figure 45: SMB Server status after restarting.....	23
Figure 46: FTP Server status	24
Figure 47: FTP Server configuration Firewall.....	24
Figure 48: FTP Server create User	25
Figure 49: Set up OpenSSL protocol	25
Figure 50: Create FTP folder and set permissions.....	25
Figure 51: Edit vsftpd.conf file [1].....	26
Figure 52: Edit vsftpd.conf file [2].....	26
Figure 53: Edit vsftpd.conf file [3].....	26
Figure 54: Create SSH keys in control node.....	27
Figure 55: Distribute public key to managed nodes	27
Figure 56: Edit inventory file [1].....	27
Figure 57: Edit inventory file [2].....	28
Figure 58: Create Ansible playbook.....	28
Figure 59: Running playbook.....	28
Figure 60: Client side IP	29
Figure 61: Test DHCP	29
Figure 62: Ip route list on client	29
Figure 63: Test NAT.....	29
Figure 64: Create a NFS test file in server	30
Figure 65: Create a NFS local directory	30
Figure 66: Running NFS mount command.....	30
Figure 67: Client edits the NFS test file	30

Figure 68: Changes in NFS test file	30
Figure 69: Create a SMB test file in server.....	31
Figure 70: Connect to SMB share folder	31
Figure 71: Sign in with user created.....	31
Figure 72: Find the SMB test file.....	32
Figure 73: Edit SMB test file.....	32
Figure 74: Changes in SMB test file	33
Figure 75: The packet has been encrypted.....	33
Figure 76: Server signing test.....	33
Figure 77: Log in to FileZilla	34
Figure 78: Create new folder on client.....	34
Figure 79: Move to the new folder.....	34
Figure 80: Test with Wireshark.....	35

LIST OF TABLES

Table 1: Topology details	12
Table 2: NFS permission table	21
Table 3: Task assignment	38
Table 4: Self assessment.....	38

Chapter 1

Introduction

1.1 General information

1.1.1 Linux Server

A Linux server is a powerful server that operates using a Linux-based open-source operating system (OS). It can serve a wide range of functions, including hosting websites and applications, managing network services, and providing file storage.

1.1.2 NAT and DHCP network services

1.1.2.1 NAT

- Network Address Translation (NAT) is a service that enables private IP networks to use the internet and cloud. It's a method to map multiple private addresses inside a local network to a public IP address before transferring the information onto the internet.
- NAT offers the ability to access the internet with more security and privacy by hiding the device IP address from the public network, even when sending and receiving traffic.
- NAT inside and outside addresses:
 - **Inside Local Address:** An IP address within the local network, typically private, used for hosts within the network itself.
 - **Inside Global Address:** An IP address that represents one or more inside local addresses to the outside world, showing how the inside host is seen externally.
 - **Outside Local Address:** The real IP address of a destination host within the local network after translation.
 - **Outside Global Address:** The external IP address of a destination host as seen from the outside network, before any translation occurs.

1.1.2.2 DHCP

- Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns IP addresses to devices on a network, enabling them to communicate using IP. It automates and centralizes this process, eliminating the need for manual IP address assignments by network administrators. DHCP is suitable for both small local networks and large enterprise networks.
- DHCP is a valuable tool in modern networking because it streamlines IP address management, enhances network efficiency, and reduces the potential for errors. It is especially crucial in large and dynamic network environments where manual IP address assignment would be impractical and error-prone.

1.1.3 File Server on Linux

- **File Transfer Protocol (FTP)** serves as a widely adopted network protocol for transferring files across networks, including the Internet. It adheres to a client-server architecture, where the client requests files from the server and the server delivers them. FTP is platform-agnostic, making it compatible with various operating systems.
- **Network File System (NFS)** is a distributed file system protocol specifically tailored for Unix and Unix-like systems. It allows files and directories to be shared seamlessly across a network. Remote clients can access and mount a file system on their local system, creating the illusion that the files are on their own machine. NFS offers users and applications a consistent and transparent view of the file system, regardless of the physical location of the files.
- **Server Message Block (SMB)** is a network protocol that enables users to communicate with remote computers and servers (e.g., to share resources or files). It's also referred to as the server/client protocol because the server has a resource that it can share with the client.

1.1.4 Ansible

Ansible is an open-source IT automation platform that uses agentless architecture to manage and configure IT infrastructure. Ansible can be used to automate a wide range of IT tasks, including provisioning infrastructure, deploying software, configuring systems, and managing applications.

1.2 Component

1.2.1 NAT and DHCP

1.2.1.1 NAT

There are three ways to configure NAT:

- **Static NAT:** This allows for a one-to-one mapping between local and global addresses. These mappings are configured by the network administrator and remained constant. Static NAT is typically used for servers and other devices that need to be consistently accessible from the public internet.

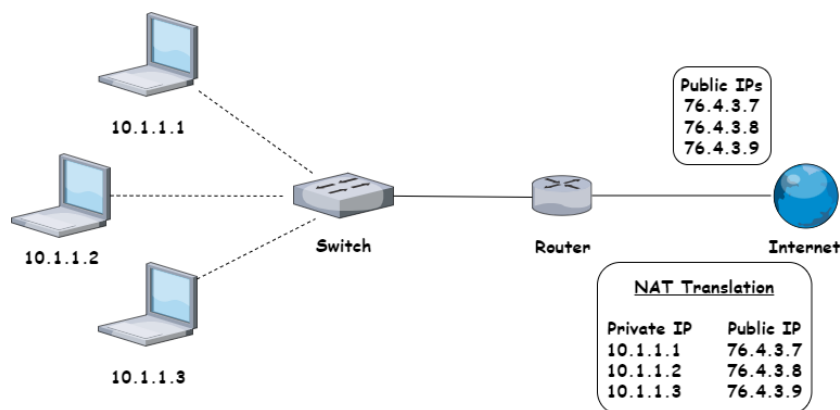


Figure 1: Static NAT

- **Dynamic NAT:** A single public IP address can be shared by multiple private IP addresses. If the pool's IP addresses are all in use, any additional packets will be dropped. Dynamic NAT is typically used for home networks and small businesses, where there are only a few devices that need to access the public Internet.

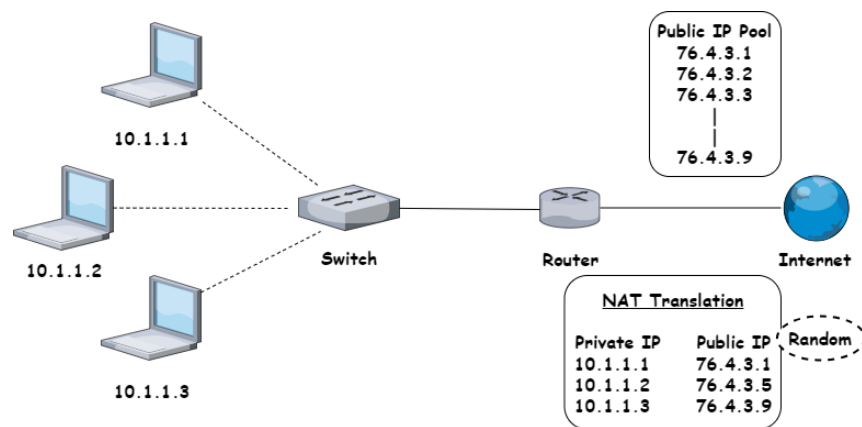


Figure 2: Dynamic NAT

- **Port Address Translation (PAT)** – Also known as NAT overload: One public IP address is used for all internal devices, but a different port is assigned to each private IP address. PAT is the most commonly used method as it is cost-effective.

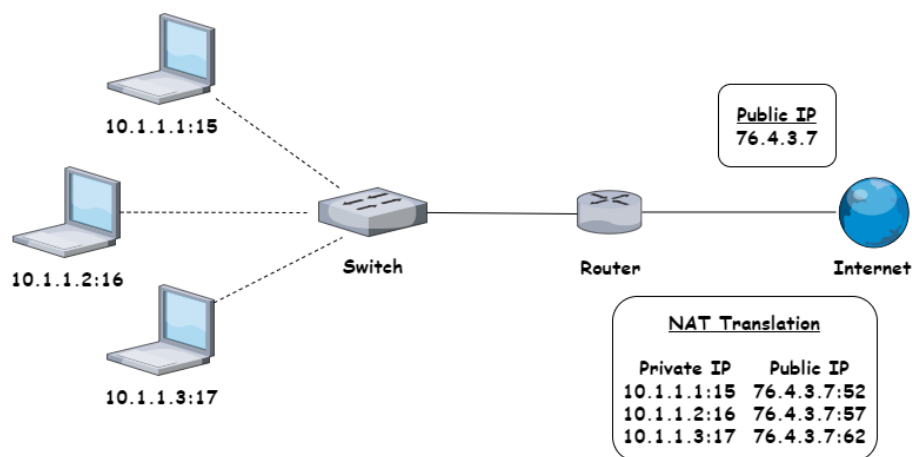


Figure 3: Port Address Translation (PAT)

1.2.1.2 DHCP

- **DHCP Server:** The DHCP server is a central component responsible for managing and distributing IP addresses and network configuration settings to clients. It maintains a pool of available IP addresses to allocate to clients.
- **DHCP Client:** The DHCP client is a device that requests and obtains IP addresses and other network configuration parameters from a DHCP server.

- **DHCP Relay:** The DHCP relay agent is an optional device that forwards DHCP messages between DHCP clients and DHCP servers. It is often used to extend the reach of a DHCP server beyond a single subnet.
- Others: IP address pool, Subnets, DNS Server, Lease time,

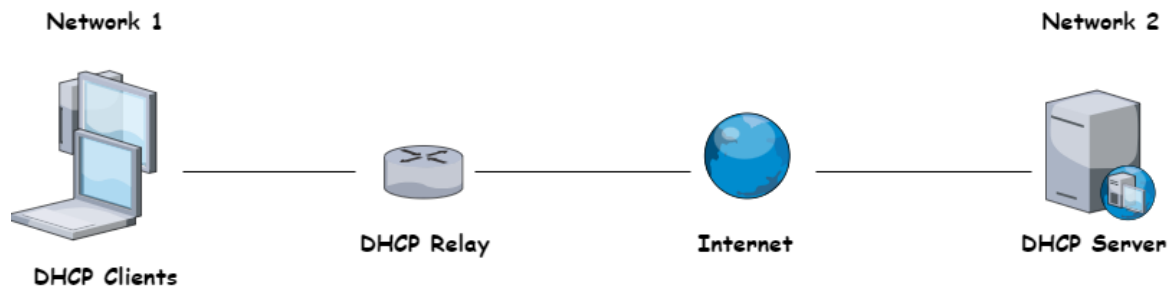


Figure 4: DHCP Components

1.2.2 File-sharing protocols: NFS, SMB, FTP

1.2.2.1 NFS

The main components of NFS are:

- **NFS server:** responsible for exporting file systems to NFS clients.
- **NFS client:** responsible for mounting NFS file systems from NFS servers.

1.2.2.2 SMB

The main components of SMB are:

- **SMB server:** individual or multiple network servers that store SMB resources and grant or deny client access.
- **SMB client:** main device that accesses files and folders on an SMB server.
- **SMB port:** the system on a server or other device used by SMB servers to communicate; modern server variations use port 445, and older variations use port 139.
- **SMB share:** any resource housed on an SMB server; can also be referred to as SMB file shares.

1.2.2.3 FTP

The main components of FTP are:

- **FTP server:** Manages files and folders, authenticates and authorizes users.
- **FTP client:** Browses and transfers files to and from the server.
- **Control connection:** Negotiates transfer mode, authenticates and authorizes users, lists files and folders.
- **Data connection:** Transfers files between the client and server.

1.2.3 Ansible

The main components of Ansible are:

- **Control node:** The controller is the machine that runs Ansible and manages the nodes.
- **Managed nodes:** The managed nodes are the machines that Ansible controls.
- **Inventory:** The inventory is a list of the managed nodes.
- **Playbooks:** Playbooks are YAML files that describe the desired state of the managed nodes.
- **Modules:** Modules are Python code that Ansible uses to perform tasks on the managed nodes.

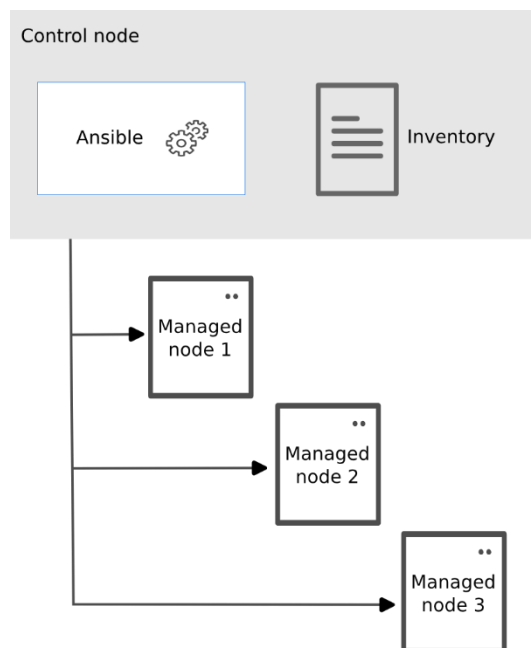


Figure 5: Ansible components

1.3 Operation

1.3.1 NAT

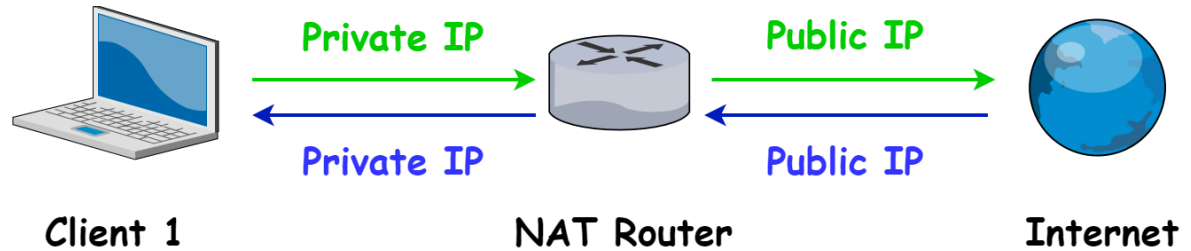


Figure 6: NAT operation

- NAT works on a router or gateway and interconnects two networks with each other. If a packet traverses outside the local (inside) network, NAT works by translating the private addresses into the public addresses before the data being transmitted to the global (outside) network. Vice versa when packet enters the local network from the global network.
- If NAT runs out of addresses, i.e., no address is left in the pool configured, incoming packets will be discarded, and the destination will receive an Internet Control Message Protocol (ICMP) host unreachable packet as a response.

1.3.2 DHCP

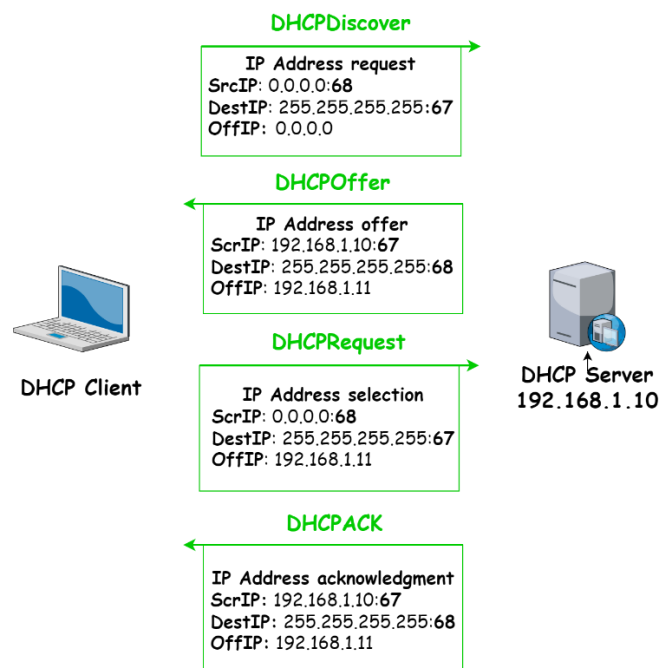


Figure 7: DHCP operation

Step 1: DHCP Discover

- When connecting a new device to a network, it still does not have an IP address. It will ask for an IP address. It calls over the network for a DHCP server. This request will arrive to all of the devices (broadcast), and the DHCP server will also get it.

Step 2: DHCP Offer

- The DHCP hears the call, and answers with an IP address, which it offers to the newly connected device.

Step 3: DHCP Request

- The IP address arrives at the device. The device will accept it and will send a request to use it.

Step 4: DHCP Ack

- Upon receiving the acknowledgment from the device, the server assigns an IP address along with the subnet mask and DNS server details to the device.
- It then logs the connected device's information, typically including the MAC address, the assigned IP address, and the IP address's expiration date.
- The DHCP allocates IP addresses for a specific duration. Once this period elapses, the IP address returns to the pool of available addresses and can be assigned to a different device.
- Typically, client communication occurs through UDP port 68, while servers use port 67. There might be slight variations based on network equipment vendors, but this is the general operational pattern.

1.3.3 File-sharing protocols: NFS, SMB, FTP

1.3.3.1 NFS

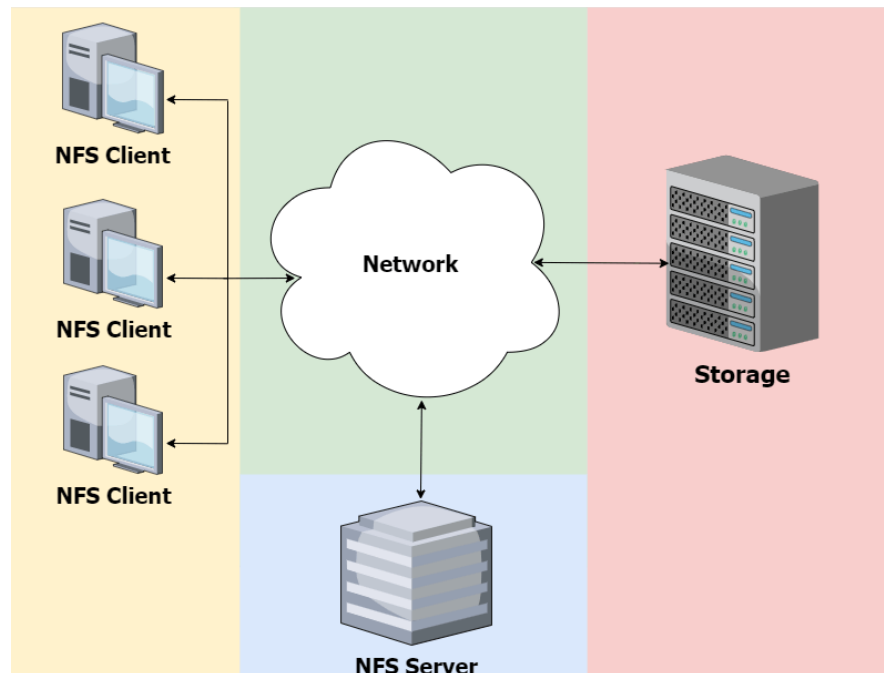


Figure 8: NFS operation

- NFS operates on a client-server model where one computer serves as the server, and others act as clients. Clients send data requests, including read and write requests, to the server, which retrieves the data from storage and sends it back to the clients.
- However, if multiple computers or threads attempt to access the same file simultaneously, shared file access functionality might not function properly.

1.3.3.2 SMB

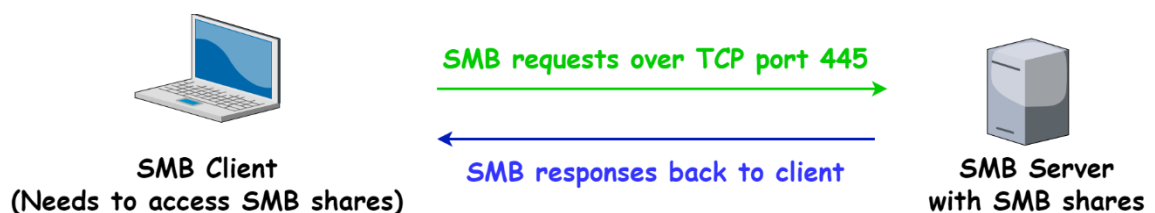


Figure 9: SMB operation

- SMB protocol works by sending and receiving messages from a user to another device or file.
- A user, or SMB client, must use an SMB port for making requests. The server evaluates the request and can either grant or deny access. If access is permitted, the client gains entry to SMB shares.
- After gaining access to a SMB share, clients can modify, print, work together on, remove, and distribute files across a network without having to download the files to their individual devices

1.3.3.3 FTP

- Normally, users are required to log in to an FTP server, although certain servers allow access to their content without authentication, a method called anonymous FTP.
- When an FTP connection is set up, it creates two communication channels: the command channel and the data channel. The command channel facilitates the transfer of commands and responses between the client and server in both directions.
- Control connection communication occurs through port number 21. On the other hand, the data channel is responsible for transferring the actual data between the client and server and operates through port number 20.
- FTP enables clients to perform various actions like uploading, downloading, deleting, renaming, moving, and copying files on the server.

FTP sessions work in active or passive modes:

- In **active mode**, when a client starts a session through a command channel request, the server establishes a data connection back to the client and starts transferring data.

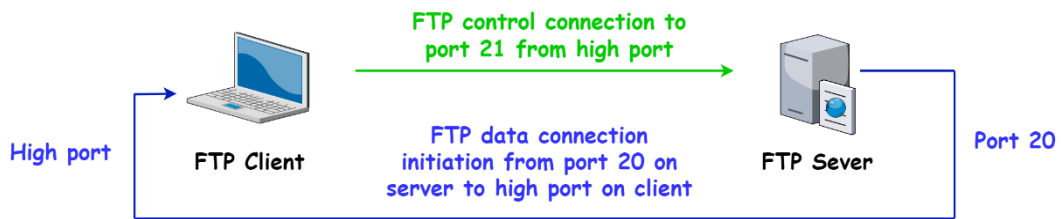


Figure 10: FTP Active mode

- In **passive mode**, the server provides the client with the details necessary to establish a data channel through the command channel. This method is effective across firewalls and network address translation gateways because it allows the client to initiate all connections.

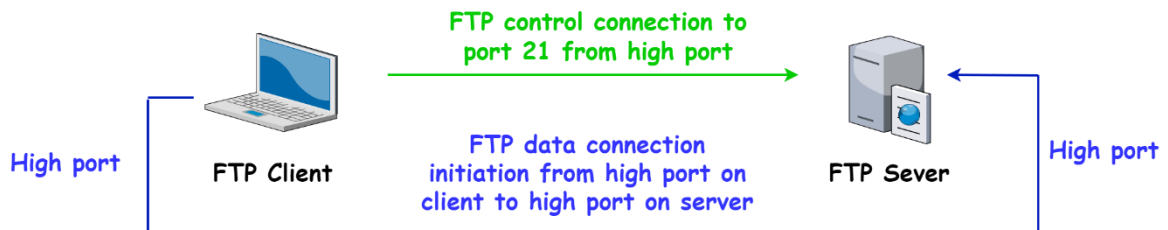


Figure 11: FTP Passive mode

1.3.4 Ansible

- Ansible reads information about which servers to be managed from inventory file.
- Ansible uses SSH protocol to connect to servers and run tasks.
- Once it has connected, Ansible transfers playbook to the remote machine(s) for execution.

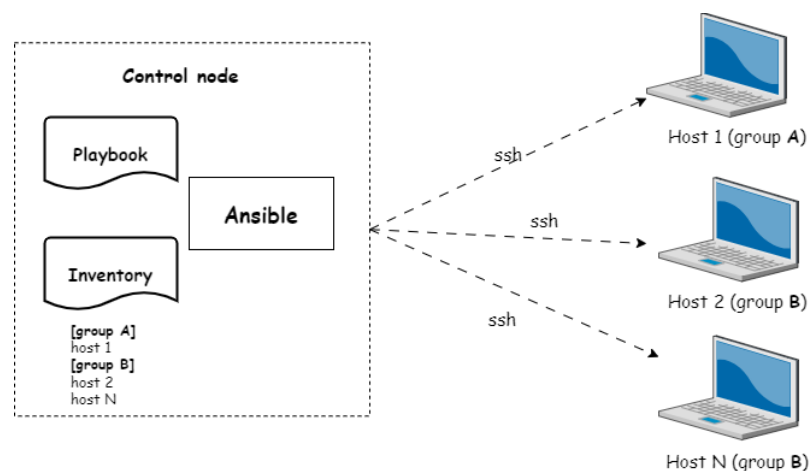


Figure 12: Ansible operation

Chapter 2 Implementation

2.1 Topology



Figure 13: Project topology

Device	IP address	Installed software, service
Ubuntu Server 22.04	192.168.123.10 192.168.23.10	NAT: iptables, iptables-persistent DHCP: isc-dhcp-server nfs-kernel-server, samba, vsftpd
Ubuntu 22.04	192.168.23.11	nfs-common, cifs-utils, FileZilla

Table 1: Topology details

2.2 Installation

Install NAT, DHCP and file-sharing protocols in Ubuntu Server using following commands:

2.2.1 Server side

2.2.1.1 NAT

+ “sudo apt-get install iptables”

```

server@server:~$ sudo apt-get install iptables
[sudo] password for server:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables is already the newest version (1.8.7-1ubuntu5.1).
iptables set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
server@server:~$
  
```

Figure 14: Install NAT: iptables

+ “sudo apt-get install iptables-persistent”

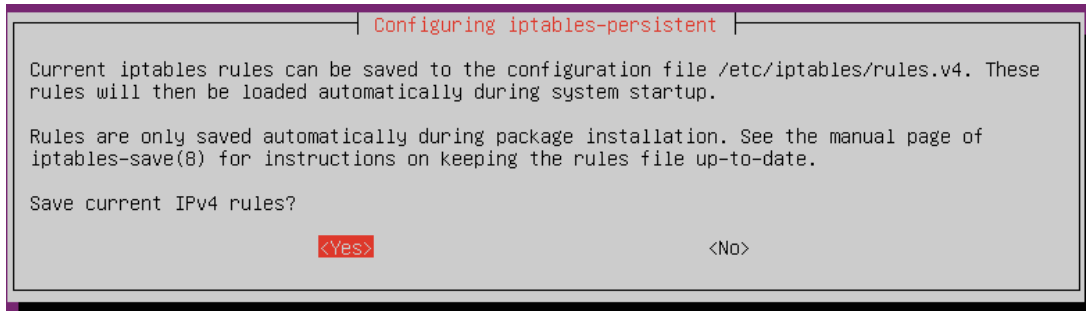


Figure 15: Install NAT: iptables-persistent [1]

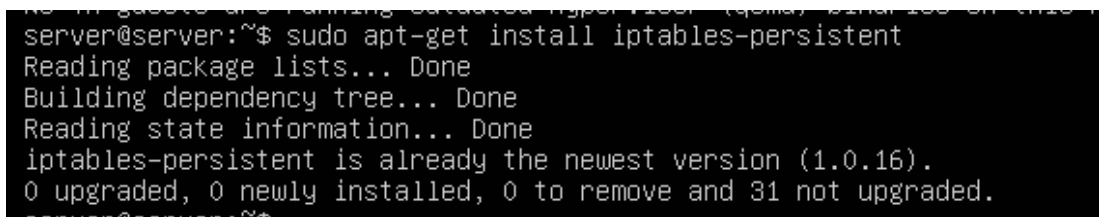


Figure 16: Install NAT: iptables-persistent [2]

2.2.1.2 DHCP

“sudo apt install isc-dhcp-server”

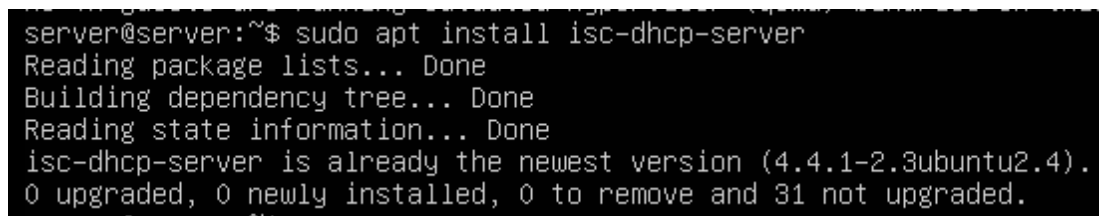


Figure 17: Install isc-dhcp-server

2.2.1.3 NFS

“sudo apt install nfs-kernel-server”

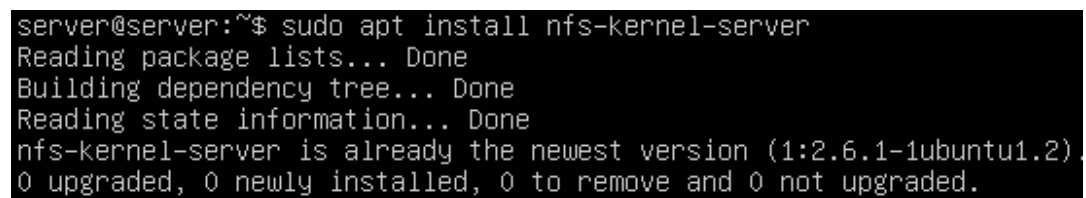


Figure 18: Install nfs-kernel-server

2.2.1.4 SMB

“sudo apt install samba”

```
server@server:~$ sudo apt install samba
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
samba is already the newest version (2:4.15.13+dfsg-0ubuntu1.5).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Figure 19: Install samba on server

2.2.1.5 *FTP*

“sudo apt install vsftpd”

```
server@server:~$ sudo apt install vsftpd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
vsftpd is already the newest version (3.0.5-0ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Figure 20: Install vsftpd on server

2.2.2 Client side

Install file-sharing protocols in Ubuntu Client using following commands:

2.2.2.1 *NFS*

“sudo apt install nfs-common”

```
dazel@ubuntu22:~/Desktop$ sudo apt-get install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-common is already the newest version (1:2.6.1-1ubuntu1.2).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

Figure 21: Install nfs-common on client

2.2.2.2 *SMB*

“sudo apt install cifs-utils”

```
dazel@ubuntu22:~/Desktop/qtm$ sudo apt install cifs-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cifs-utils is already the newest version (2:6.14-1ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

Figure 22: Install cifs-utils on client

2.2.2.3 FTP

“sudo apt install filezilla”

```
dazel@ubuntu22:~/Desktop$ sudo apt install filezilla
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
filezilla is already the newest version (3.58.0-1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

Figure 23: Install filezilla on client

2.2.3 Install packages on Server using Ansible

```
GNU nano 7.2                                setup_server.yaml
- name: Setup Ubuntu Server
  hosts: server
  become: yes

tasks:
  - name: Update apt package cache
    apt:
      update_cache: yes

  - name: Install iptables and iptables-persistent
    apt:
      name:
        - iptables
        - iptables-persistent
      state: present

  - name: Install isc-dhcp-server
    apt:
      name: isc-dhcp-server
      state: present

  - name: Install nfs-kernel-server
    apt:
      name: nfs-kernel-server
      state: present

  - name: Install samba
    apt:
      name: samba
      state: present

  - name: Install vsftpd
    apt:
      name: vsftpd
      state: present

$ ansible-playbook -i hosts setup_server.yaml
PLAY [Setup Ubuntu Server] *****
TASK [Gathering Facts] *****
ok: [192.168.23.15]

TASK [Update apt package cache] *****
changed: [192.168.23.15]

TASK [Install iptables and iptables-persistent] *****
changed: [192.168.23.15]

TASK [Install isc-dhcp-server] *****
changed: [192.168.23.15]

TASK [Install nfs-kernel-server] *****
changed: [192.168.23.15]

TASK [Install samba] *****
changed: [192.168.23.15]

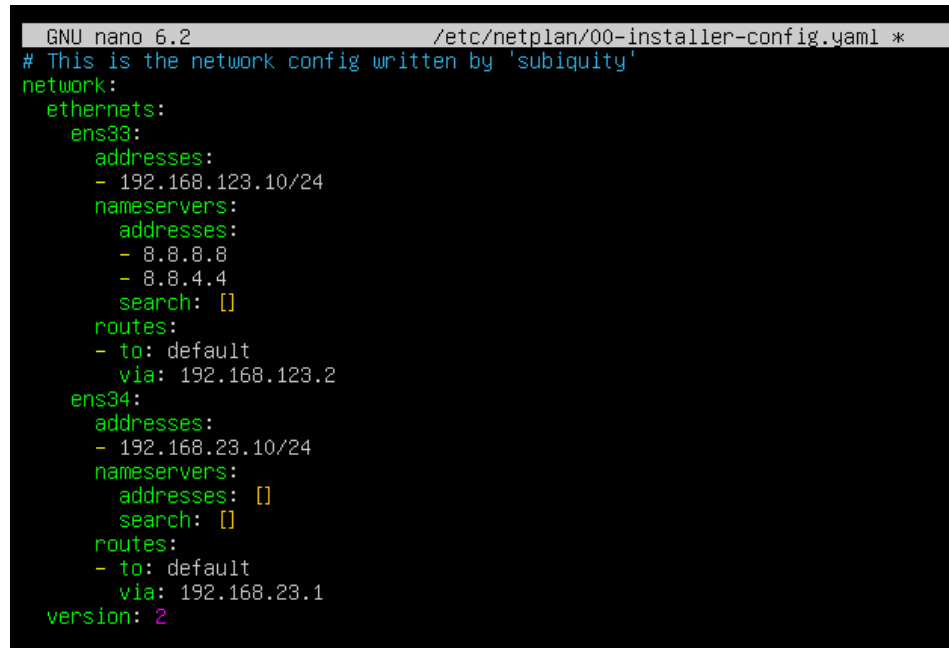
TASK [Install vsftpd] *****
changed: [192.168.23.15]

PLAY RECAP *****
192.168.23.15 : ok=7 changed=6 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

Figure 24: Install server's packages using Ansible

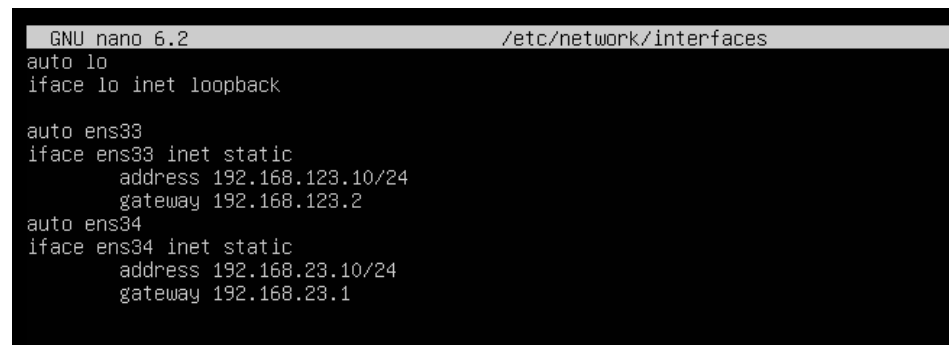
2.3 Configuration

Set static IP for 2 Network Interface Card use for DHCP Server and NAT Server using netplan:

A terminal window showing the netplan configuration file. The title bar indicates 'GNU nano 6.2' and the file path is '/etc/netplan/00-installer-config.yaml *'. The content is a YAML configuration for two network interfaces, ens33 and ens34. ens33 is configured with a static IP address of 192.168.123.10/24, nameservers 8.8.8.8 and 8.8.4.4, and a default route via 192.168.123.2. ens34 is configured with a static IP address of 192.168.23.10/24 and a default route via 192.168.23.1. The version is set to 2.

```
GNU nano 6.2 /etc/netplan/00-installer-config.yaml *
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens33:
      addresses:
        - 192.168.123.10/24
      nameservers:
        addresses:
          - 8.8.8.8
          - 8.8.4.4
        search: []
      routes:
        - to: default
          via: 192.168.123.2
    ens34:
      addresses:
        - 192.168.23.10/24
      nameservers:
        addresses: []
        search: []
      routes:
        - to: default
          via: 192.168.23.1
  version: 2
```

Figure 25: Ubuntu server netplan

A terminal window showing the network interfaces configuration file. The title bar indicates 'GNU nano 6.2' and the file path is '/etc/network/interfaces'. The content defines three interfaces: lo (loopback), ens33, and ens34. ens33 and ens34 are configured as static interfaces with their respective IP addresses and gateways.

```
GNU nano 6.2 /etc/network/interfaces
auto lo
iface lo inet loopback

auto ens33
iface ens33 inet static
    address 192.168.123.10/24
    gateway 192.168.123.2
auto ens34
iface ens34 inet static
    address 192.168.23.10/24
    gateway 192.168.23.1
```

Figure 26: Ubuntu server network interfaces

2.3.1 DHCP

- Step 1: Define which interface the DHCP server should listen to in the “etc/default/isc-dhcp-server” file.

```

GNU nano 6.2 /etc/default/isc-dhcp-server *
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
# Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="ens34"
INTERFACESv6=""

```

Figure 27: Define DHCP interfaces

- Step 2: Assign IP address pool, domain name, default gateway, lease time, ... in the “/etc/dhcp/dhcpd.conf” file.

```

GNU nano 6.2 /etc/dhcp/dhcpd.conf *

#subnet 10.152.187.0 netmask 255.255.255.0 {
#}

# This is a very basic subnet declaration.

#subnet 10.254.239.0 netmask 255.255.255.224 {
# range 10.254.239.10 10.254.239.20;
# option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
#}

# This declaration allows BOOTP clients to get dynamic addresses,
# which we don't really recommend.

#subnet 10.254.239.32 netmask 255.255.255.224 {
# range dynamic-bootp 10.254.239.40 10.254.239.60;
# option broadcast-address 10.254.239.31;
# option routers rtr-239-32-1.example.org;
#}

# A slightly different configuration for an internal subnet.
subnet 192.168.23.0 netmask 255.255.255.0 {
    range 192.168.23.11 192.168.23.254;
    option domain-name-servers server.group2.org;
    option domain-name "internal.group2.org";
    option subnet-mask 255.255.255.0;
    option routers 192.168.23.10;
    option broadcast-address 192.168.23.255;
    default-lease-time 600;
    max-lease-time 7200;
}

```

Figure 28: DHCP configuration

- Step 3: Restart the DHCP service with the following command: “sudo systemctl restart isc-dhcp-server.service”.

- Step 4: Check whether the server is running with: “sudo systemctl status isc-dhcp-server.service”.

```
server@server:~$ sudo systemctl restart isc-dhcp-server.service
server@server:~$ sudo systemctl status isc-dhcp-server.service
● isc-dhcp-server.service - ISC DHCP IPv4 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-11-02 06:50:13 UTC; 2s ago
     Docs: man:dhcpd(8)
    Main PID: 1254 (dhcpd)
      Tasks: 4 (limit: 2178)
     Memory: 4.6M
        CPU: 14ms
    CGroup: /system.slice/isc-dhcp-server.service
            └─1254 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhcpd.pid -cf /etc/dh

Nov 02 06:50:13 server dhcpd[1254]: Database file: /var/lib/dhcp/dhcpd.leases
Nov 02 06:50:13 server dhcpd[1254]: PID file: /run/dhcp-server/dhcpd.pid
Nov 02 06:50:13 server dhcpd[1254]: Wrote 0 leases to leases file.
Nov 02 06:50:13 server dhcpd[1254]: Listening on LPF/ens34/00:0c:29:1e:78:f1/192.168.23.0/24
Nov 02 06:50:13 server sh[1254]: Listening on LPF/ens34/00:0c:29:1e:78:f1/192.168.23.0/24
Nov 02 06:50:13 server sh[1254]: Sending on LPF/ens34/00:0c:29:1e:78:f1/192.168.23.0/24
Nov 02 06:50:13 server sh[1254]: Sending on Socket/fallback/fallback-net
Nov 02 06:50:13 server dhcpd[1254]: Sending on LPF/ens34/00:0c:29:1e:78:f1/192.168.23.0/24
Nov 02 06:50:13 server dhcpd[1254]: Sending on Socket/fallback/fallback-net
Nov 02 06:50:13 server dhcpd[1254]: Server starting service.
lines 1-21/21 (END)
```

Figure 29: DHCP server status

2.3.2 NAT

Port Address Translation (PAT) configuration

- Step 1: Open the “etc/sysctl.conf” file and set the “net.ipv4.ip_forward” parameter to 1 by uncommenting it.

```
GNU nano 6.2 /etc/sysctl.conf
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

Figure 30: Enable forwarding in sysctl.conf

- Step 2: Enable the changes to above file using “sudo sysctl -p”.

```
server@server:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
```

Figure 31: Enables the changes

- Step 3: List the already configured iptables policies by using the command “sudo iptables -L”.

```
server@server:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 32: Iptables policies

- Step 4: Check network interface cards.

```
server@server:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.123.10  netmask 255.255.255.0  broadcast 192.168.123.255
    inet6 fe80::20c:29ff:fead:df6a  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:ad:df:6a  txqueuelen 1000  (Ethernet)
    RX packets 28484  bytes 42405219 (42.4 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 4720  bytes 324725 (324.7 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens34: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.23.10  netmask 255.255.255.0  broadcast 192.168.23.255
    inet6 fe80::20c:29ff:fead:df74  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:ad:df:74  txqueuelen 1000  (Ethernet)
    RX packets 102  bytes 27200 (27.2 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 96  bytes 14303 (14.3 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 33: Network interfaces

- Step 5: Configure NAT.

On server, add these three instructions:

1. iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
2. iptables -A FORWARD -i ens33 -o ens34 -m state --state RELATED,ESTABLISHED -j ACCEPT
3. iptables -A FORWARD -i ens34 -o ens33 -j ACCEPT

- where ens33 is the interface to the internet (NAT)

- where ens34 is the interface to LAN, also DHCP-server interface (host-only)

line1: enable NAT on interface ens33, use ens33 for outgoing packets

line2: forward IP-packets from ens33 to ens34 where there is an established initial request

line3: forward packages from ens34 to ens33

- Step 6: Save the config into /etc/iptables/rules.v4 .

```
server@server:~$ sudo iptables-save | sudo tee /etc/iptables/rules.v4
# Generated by iptables-save v1.8.7 on Wed Nov  1 15:12:22 2023
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -i ens33 -o ens34 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i ens34 -o ens33 -j ACCEPT
COMMIT
# Completed on Wed Nov  1 15:12:22 2023
# Generated by iptables-save v1.8.7 on Wed Nov  1 15:12:22 2023
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o ens33 -j MASQUERADE
COMMIT
# Completed on Wed Nov  1 15:12:22 2023
```

Figure 34: Save iptables policies

2.3.3 NFS

- Step 1: Check the NFS server status.

```
server@server:~$ sudo systemctl status nfs-server
● nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Thu 2023-11-02 06:46:27 UTC; 9min ago
     Process: 848 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
     Process: 868 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
    Main PID: 868 (code=exited, status=0/SUCCESS)
       CPU: 8ms

Nov 02 06:46:27 server systemd[1]: Starting NFS server and services...
Nov 02 06:46:27 server systemd[1]: Finished NFS server and services.
```

Figure 35: NFS Server status

- Step 2: Create root NFS directory, also known as an export folder.

```
server@server:~$ sudo mkdir /mnt/mysharedir
```

Figure 36: Create root NFS directory

- Step 3: Set permissions so that any user on the client machine can access the folder.

```
server@server:~$ sudo chown nobody:nogroup /mnt/mysharedir
server@server:~$ sudo chmod 777 /mnt/mysharedir
```

Figure 37: Set NFS folder permissions

- Step 4: Define access for NFS Clients in “etc/exports” file.

To enable access to a single client	/mnt/mysharedir {clientIP}(rw,sync,no_subtree_check)
To enable access to several clients	/mnt/mysharedir {clientIP- 1}(rw,sync,no_subtree_check) {clientIP-2}(...) {clientIP-3}(...)
To enable access to an entire subnet	/mnt/mysharedir {subnetIP}/{subnetMask}(rw,sync,no_subtree_check)

Table 2: NFS permission table

Define access for IP 192.168.23.11:

```
GNU nano 6.2 /etc/exports *
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/mnt/mysharedir 192.168.23.11(rw,sync,no_subtree_check)
```

Figure 38: Define access for 192.168.23.11

- Step 5: Make the NFS share available to Clients.

Make the shared directory available to clients using the ”exportfs” command. Then restart NFS Kernel to apply changes.

```

server@server:~$ sudo exportfs -a
server@server:~$ sudo systemctl restart nfs-kernel-server
server@server:~$ sudo systemctl status nfs-kernel-server
• nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Thu 2023-11-02 07:14:53 UTC; 10s ago
     Process: 1327 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
     Process: 1329 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
    Main PID: 1329 (code=exited, status=0/SUCCESS)
       CPU: 7ms

Nov 02 07:14:52 server systemd[1]: Starting NFS server and services...
Nov 02 07:14:53 server systemd[1]: Finished NFS server and services.

```

Figure 39: NFS Server status after restarting

2.3.4 SMB

- Step 1: In order to add a folder that is password protected, create a new group “group2” and a new user “shares” with password.

```

server@server:/mnt/smb_share$ sudo addgroup group2
Adding group `group2' (GID 1001) ...
Done.
server@server:/mnt/smb_share$ sudo useradd shares -G group2
server@server:/mnt/smb_share$ sudo smbpasswd -a shares
New SMB password:
Retype new SMB password:
Added user shares.

```

Figure 40: Create Group and User in SMB Server

- Step 2: Check the status of SMB server.

```

server@server:/mnt/myshareddir$ sudo systemctl status smbd
• smbd.service - Samba SMB Daemon
   Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-11-02 06:46:27 UTC; 59min ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
     Process: 940 ExecStartPre=/usr/share/samba/update-apparmor-samba-profile (code=exited, status=0)
    Main PID: 952 (smbd)
   Status: "smbd: ready to serve connections..."
      Tasks: 4 (limit: 2178)
     Memory: 18.4M
        CPU: 180ms
     CGroup: /system.slice/smbd.service
            └─952 /usr/sbin/smbd --foreground --no-process-group
              └─955 /usr/sbin/smbd --foreground --no-process-group
                └─956 /usr/sbin/smbd --foreground --no-process-group
                  └─957 /usr/lib/x86_64-linux-gnu/samba/samba-bgqd --ready-signal-fd=45 --parent-watch-fd=46

Nov 02 06:46:27 server systemd[1]: Starting Samba SMB Daemon...
Nov 02 06:46:27 server systemd[1]: Started Samba SMB Daemon.

```

Figure 41: SMB Server status

- Step 3: Create a SMB share directory and change permissions.

```

server@server:/mnt/smb_share$ sudo mkdir -p /mnt/smb_share_secured
server@server:/mnt/smb_share$ sudo chmod -R 0770 /mnt/smb_share_secured

```

Figure 42: Create root SMB directory and change permissions

- Step 4: Edit the “/etc/samba/smb.conf” file.

```
GNU nano 6.2 /etc/samba/smb.conf *
[share-folder]
    path = /mnt/smb_share_secured
    browseable = yes
    read only = yes
    writeable = yes
    valid user = group2
```

Figure 43: Edit smb.conf file [1]

Set “security = user” for user-level authentication.

Server signing: ensures the integrity and authenticity of transmitted data.

“smb encryption = on”: encryption during transmission.

```
GNU nano 6.2 /etc/samba/smb.conf
# server string is the equivalent of the NT Description field
server string = %h server (Samba, Ubuntu)

#### Networking ####

# The specific set of interfaces / networks to bind to
# This can be either the interface name or an IP address/netmask;
# interface names are normally preferred
; interfaces = 192.168.23.0/24 ens34

# Only bind to the named interfaces and/or networks; you must use the
# 'interfaces' option above to use this.
# It is recommended that you enable this feature if your Samba machine is
# not protected by a firewall or is a firewall itself. However, this
# option cannot handle dynamic or non-broadcast interfaces correctly.
; bind interfaces only = yes
security = user
server signing = mandatory
smb encrypt = on
```

Figure 44: Edit smb.conf file [2]

- Step 5: Restart Samba to apply changes.

```
server@server:/$ sudo service smbd restart
server@server:/$ sudo service smbd status
• smbd.service - Samba SMB Daemon
  Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2023-11-02 08:08:25 UTC; 8s ago
    Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
  Process: 1517 ExecStartPre=/usr/share/samba/update-apparmor-samba-profile (code=exited, status=0)
 Main PID: 1527 (smbd)
  Status: "smbd: ready to serve connections..."
    Tasks: 4 (limit: 2178)
  Memory: 8.8M
    CPU: 145ms
  CGroup: /system.slice/smbd.service
          └─1527 /usr/sbin/smbd --foreground --no-process-group
            └─1530 /usr/sbin/smbd --foreground --no-process-group
              └─1531 /usr/sbin/smbd --foreground --no-process-group
                └─1532 /usr/lib/x86_64-linux-gnu/samba/samba-bgqd --ready-signal-fd=45 --parent-watch=2

Nov 02 08:08:25 server systemd[1]: Starting Samba SMB Daemon...
Nov 02 08:08:25 server systemd[1]: Started Samba SMB Daemon.
```

Figure 45: SMB Server status after restarting

2.3.5 FTP

Configure FTP server:

- Step 1: Check the status of FTP with “sudo service vsftpd status”.

```
server@server:~$ sudo service vsftpd status
• vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-11-01 17:58:32 UTC; 10h ago
     Main PID: 4423 (vsftpd)
       Tasks: 1 (limit: 4515)
      Memory: 856.0K
         CPU: 11ms
       CGroup: /system.slice/vsftpd.service
               └─4423 /usr/sbin/vsftpd /etc/vsftpd.conf

Nov 01 17:58:32 server systemd[1]: Starting vsftpd FTP server...
Nov 01 17:58:32 server systemd[1]: Started vsftpd FTP server.
```

Figure 46: FTP Server status

- Step 2: Configure Firewall.

FTP uses port 20 for active mode, port 21 for commands, and a range of ports for passive mode.

“sudo ufw allow 20/tcp”

“sudo ufw allow 21/tcp”

“sudo ufw allow 990/tcp” (for TLS: optional)

“sudo ufw allow 5000:10000/tcp”

```
server@server:~$ sudo ufw allow 20/tcp
Rules updated
Rules updated (v6)
server@server:~$ sudo ufw allow 21/tcp
Rules updated
Rules updated (v6)
server@server:~$ sudo ufw allow 990/tcp
Rules updated
Rules updated (v6)
server@server:~$ sudo ufw allow 5000:10000/tcp
Rules updated
Rules updated (v6)
server@server:~$
```

Figure 47: FTP Server configuration Firewall

- Step 3: Create user.

“sudo adduser rosy”

- Step 6: Edit “/etc/vsftpd.conf” file, uncomment and add following lines.

```
GNU nano 6.2 /etc/vsftpd.conf
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (:::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
```

Figure 51: Edit vsftpd.conf file [1]

```
GNU nano 6.2 /etc/vsftpd.conf
dirmessage_enable=YES
#
# If enabled, vsftpd will display directory listings with the time
# in your local time zone. The default is to display GMT. The
# times returned by the MDTM FTP command are also affected by this
# option.
use_localtime=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
```

Figure 52: Edit vsftpd.conf file [2]

```
GNU nano 6.2 /etc/vsftpd.conf *
secure_chroot_dir=/var/run/vsftpd/empty
#
# This string is the name of the PAM service vsftpd will use.
pam_service_name=vsftpd
force_dot_files=YES
#
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=NO
#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES
pasv_min_port=5000
pasv_max_port=10000
user_sub_token=$USER
local_root=/folder
```

Figure 53: Edit vsftpd.conf file [3]

2.3.6 Ansible

- Step 1: Create SSH keys in control node.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dazel/.ssh/id_rsa):
/home/dazel/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dazel/.ssh/id_rsa
Your public key has been saved in /home/dazel/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0bwSH3RpsboNXHvnVzq/IyEXZNo0Z68+wnDQ5CNEVFQ dazel@kali
The key's randomart image is:
+--[RSA 3072]--+
|      o+oo.E      |
|      .+.= 0      |
|      ..**+ + .    |
|      + *00+0 .    |
|      . S .0.. 0..  |
|      o B.o.= ...  |
|      . + .+0.= .  |
|      .   0.0=     |
|      .. 0+       |
+--[SHA256]--+
```

Figure 54: Create SSH keys in control node

- Step 2: Copy the Public Key to managed nodes using “ssh-copy-id”.

```
$ ssh-copy-id dazel@192.168.23.15
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/dazel/.ssh/id_rsa.pub"
The authenticity of host '192.168.23.15 (192.168.23.15)' can't be established.
ED25519 key fingerprint is SHA256:93R/XXpTSr19NkfHeoy9JhSxvmuN1noUo09bo2xHDXy.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already i
nstalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install th
e new keys
dazel@192.168.23.15's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'dazel@192.168.23.15'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 55: Distribute public key to managed nodes

- Step 3: Create Inventory file.

```
GNU nano 7.2 hosts
[server]
192.168.23.15
```

Figure 56: Edit inventory file [1]

```
GNU nano 7.2          ansible.cfg
[defaults]
inventory = hosts
```

Figure 57: Edit inventory file [2]

- Step 4: Create an Ansible playbook.

```
GNU nano 7.2          setup_server.yaml
- name: Setup Ubuntu Server
  hosts: server
  become: yes
  tasks:
    - name: Update apt package cache
      apt:
        update_cache: yes

    - name: Install iptables and iptables-persistent
      apt:
        name:
          - iptables
          - iptables-persistent
        state: present

    - name: Install isc-dhcp-server
      apt:
        name: isc-dhcp-server
        state: present

    - name: Install nfs-kernel-server
      apt:
        name: nfs-kernel-server
        state: present

    - name: Install samba
      apt:
        name: samba
        state: present

    - name: Install vsftpd
      apt:
        name: vsftpd
        state: present
```

Figure 58: Create Ansible playbook

- Step 5: Running the Playbook using the command:
“ansible-playbook -i hosts setup_server.yaml”.

```
$ ansible-playbook -i hosts setup_server.yaml

PLAY [Setup Ubuntu Server] *****
TASK [Gathering Facts] *****
ok: [192.168.23.15]

TASK [Update apt package cache] *****
changed: [192.168.23.15]

TASK [Install iptables and iptables-persistent] *****
changed: [192.168.23.15]

TASK [Install isc-dhcp-server] *****
changed: [192.168.23.15]

TASK [Install nfs-kernel-server] *****
changed: [192.168.23.15]

TASK [Install samba] *****
changed: [192.168.23.15]

TASK [Install vsftpd] *****
changed: [192.168.23.15]

PLAY RECAP *****
192.168.23.15 : ok=7  changed=6  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Figure 59: Running playbook

Chapter 3

Result and Conclusion

3.1 Result

3.1.1 DHCP

Client side:

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.23.11 netmask 255.255.255.0 broadcast 192.168.23.255
    inet6 fe80::91ec:86d5:b199:1340 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:08:02:3f txqueuelen 1000 (Ethernet)
    RX packets 105 bytes 18710 (18.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 245 bytes 30943 (30.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 60: Client side IP

```
[~/Desktop]-[pknova@pknova]
[:(] $ ping 192.168.23.10 -c 4
PING 192.168.23.10 (192.168.23.10) 56(84) bytes of data.
64 bytes from 192.168.23.10: icmp_seq=1 ttl=64 time=0.943 ms
64 bytes from 192.168.23.10: icmp_seq=2 ttl=64 time=0.828 ms
64 bytes from 192.168.23.10: icmp_seq=3 ttl=64 time=1.15 ms
64 bytes from 192.168.23.10: icmp_seq=4 ttl=64 time=1.21 ms
```

Figure 61: Test DHCP

3.1.2 NAT

```
rosy@rosy:~$ ip route
default via 192.168.23.10 dev ens33 proto dhcp metric 20100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.23.0/24 dev ens33 proto kernel scope link src 192.168.23.11 metric 100
```

Figure 62: Ip route list on client

```
rosy@rosy:~$ ping facebook.com
PING facebook.com (157.240.218.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=1 ttl=128 time=28.6 ms
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=2 ttl=128 time=44.6 ms
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=3 ttl=128 time=30.0 ms
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=4 ttl=128 time=53.0 ms
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=5 ttl=128 time=25.5 ms
64 bytes from edge-star-mini-shv-02-xsp1.facebook.com (157.240.218.35): icmp_seq=6 ttl=128 time=25.2 ms
```

Figure 63: Test NAT

3.1.3 NFS

Create a txt file in server to test:

```
server@server:/mnt/myshareddir$ cat shared.txt
This is NFS test file
```

Figure 64: Create a NFS test file in server

In NFS client:

- Create a local directory to mount the NFS folder:

```
dazel@ubuntu22:~/Desktop$ mkdir ./qtm
dazel@ubuntu22:~/Desktop$ ls -a
.  ..  qtm
```

Figure 65: Create a NFS local directory

- Mount the file share by running the mount command:

```
dazel@ubuntu22:~/Desktop$ sudo mount -t nfs 192.168.23.10:/mnt/myshareddir ./qtm
dazel@ubuntu22:~/Desktop$ cd qtm/
dazel@ubuntu22:~/Desktop/qtm$ ls -a
.  ..  shared.txt
dazel@ubuntu22:~/Desktop/qtm$ cat shared.txt
This is NFS test file
```

Figure 66: Running NFS mount command

- Edit the file by client:

```
dazel@ubuntu22:~/Desktop/qtm$ cat shared.txt
This is NFS test file
dazel@ubuntu22:~/Desktop/qtm$ nano shared.txt
dazel@ubuntu22:~/Desktop/qtm$ cat shared.txt
This is NFS test file

Test response from client
```

Figure 67: Client edits the NFS test file

- Server can see the changes in test file:

```
server@server:~$ cd /mnt/myshareddir/
server@server:/mnt/myshareddir$ ls -a
.  ..  shared.txt
server@server:/mnt/myshareddir$ cat shared.txt
This is NFS test file

Test response from client
server@server:/mnt/myshareddir$
```

Figure 68: Changes in NFS test file

3.1.4 SMB

- Create a txt file in server to test:

```
server@server:/mnt/smb_share_secured$ cat shared.txt  
This is SMB file test
```

Figure 69: Create a SMB test file in server

In SMB Client

- Connect to SMB share folder:

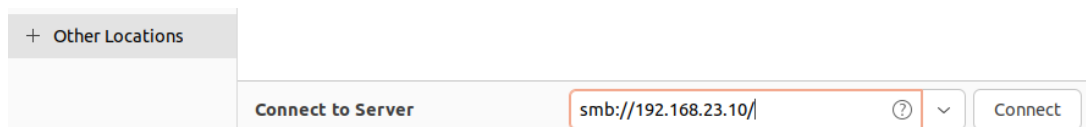


Figure 70: Connect to SMB share folder

- Sign in with user created:

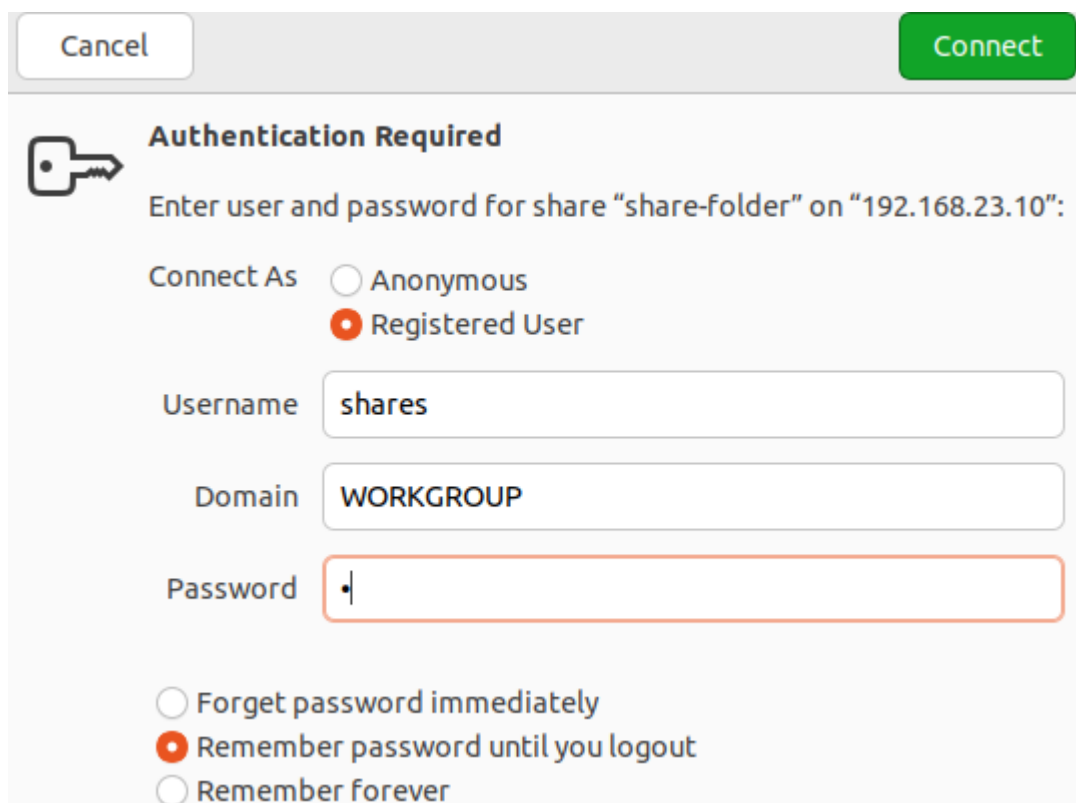


Figure 71: Sign in with user created

- Find the txt file in share-folder:

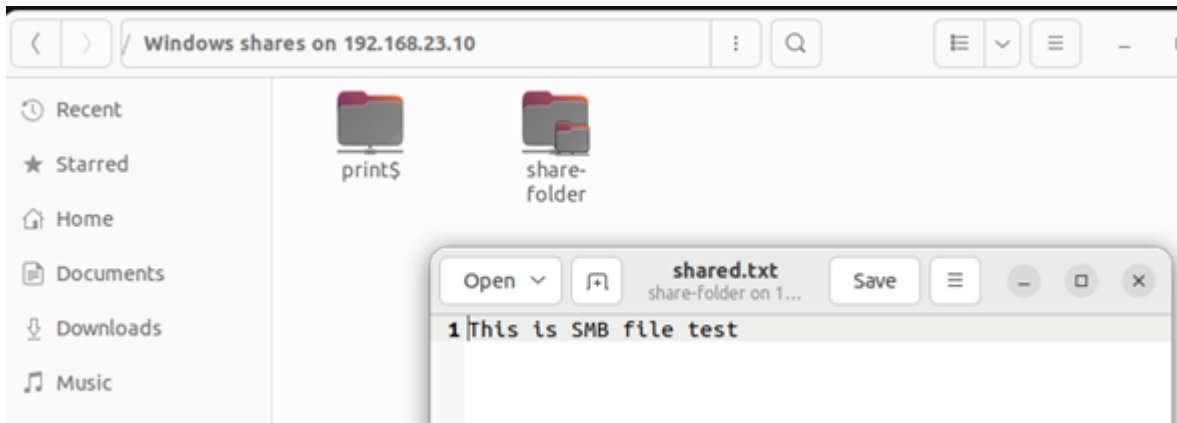


Figure 72: Find the SMB test file

- Edit share file:

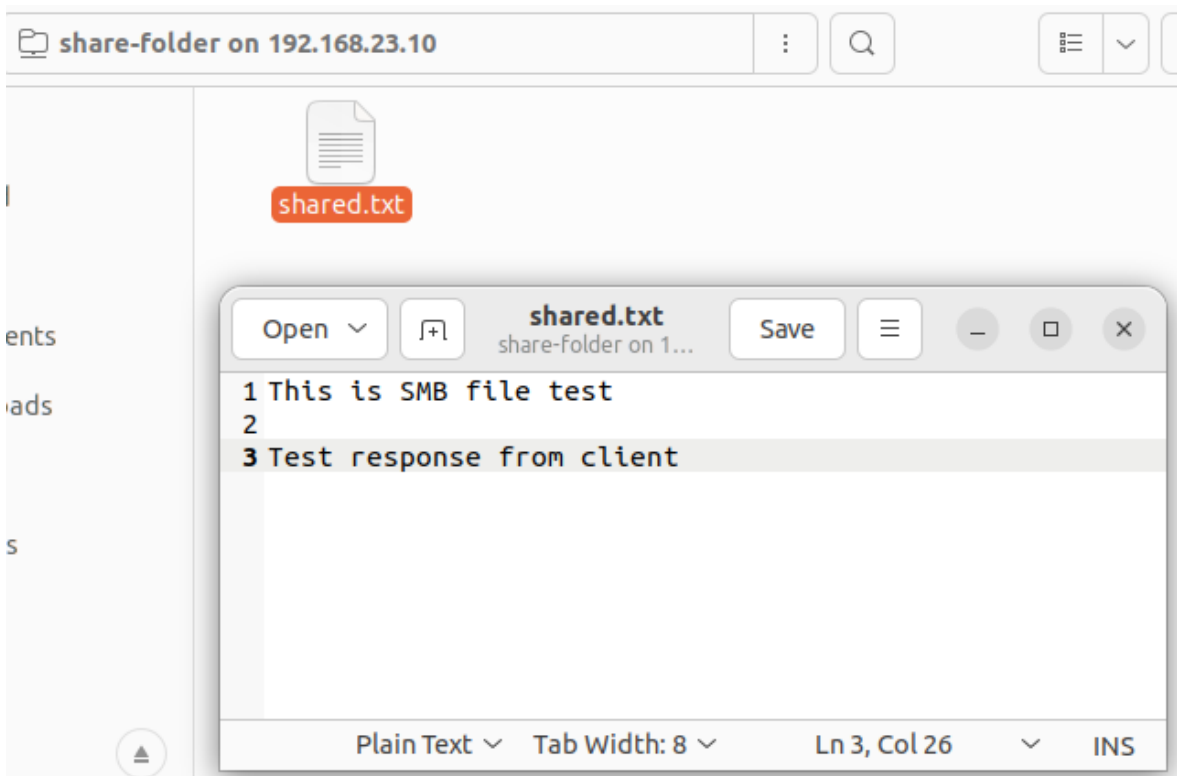


Figure 73: Edit SMB test file

- Server can see the changes in the file:

```

server@server:/mnt/myshareddir$ cd /mnt/smb_share_secured/
server@server:/mnt/smb_share_secured$ ls -la
total 4
drwxr-xr-x 2 root root 4096 Nov 22 03:25 .
drwxr-xr-x 1 root root 4096 Nov 22 03:25 ..
-rw-r--r-- 1 root root   20 Nov 22 03:25 shared.txt
server@server:/mnt/smb_share_secured$ cat shared.txt
This is SMB file test

Test response from client
server@server:/mnt/smb_share_secured$

```

Figure 74: Changes in SMB test file

- Test with wireshark: The packet has been encrypted

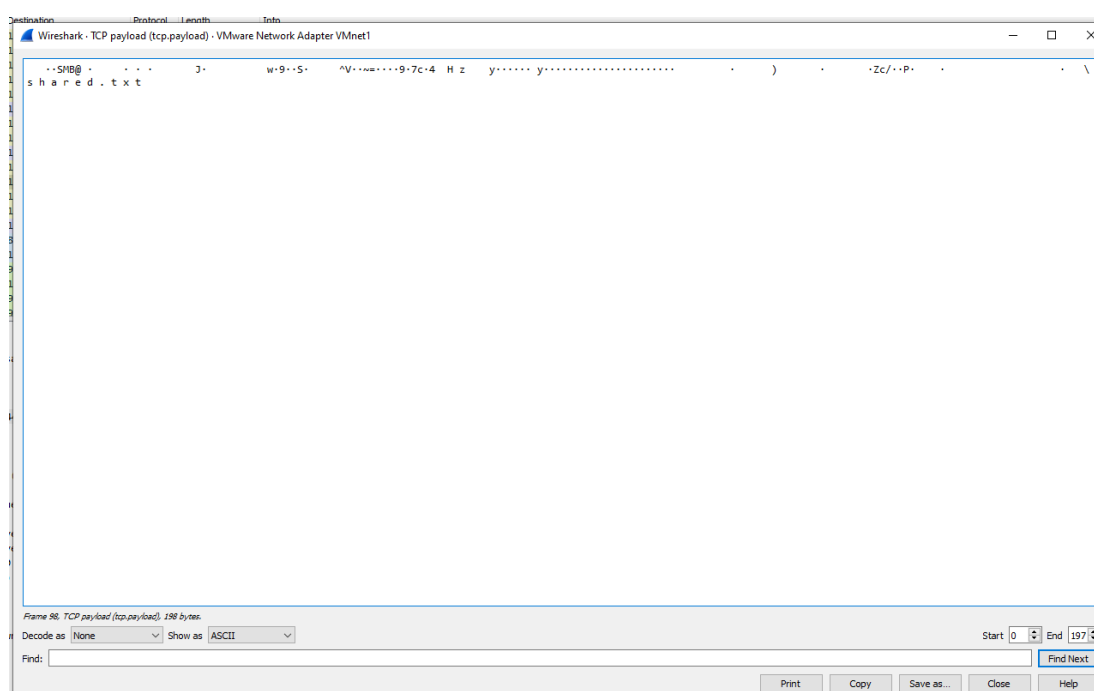


Figure 75: The packet has been encrypted

- Test smb status:

```

server@server:/#$ sudo smbstatus

Samba version 4.15.13-Ubuntu
PID Username Group Machine Protocol Version Encryption
tion Signing
-----
10092 shares shares 192.168.23.11 (ipv4:192.168.23.11:42718) SMB3_11 -
10088 nobody nogroup client (ipv4:192.168.23.11:37688) SMB3_11 -
Service pid Machine Connected at Encryption Signing
-----
share-folder 10092 192.168.23.11 Wed Nov 22 03:25:00 AM 2023 UTC - AES-128-GMAC
IPC$ 10088 client Wed Nov 22 03:23:58 AM 2023 UTC - -

```

Figure 76: Server signing test

3.1.5 FTP

In Client:

- Enter host IP, username, password of the user created:

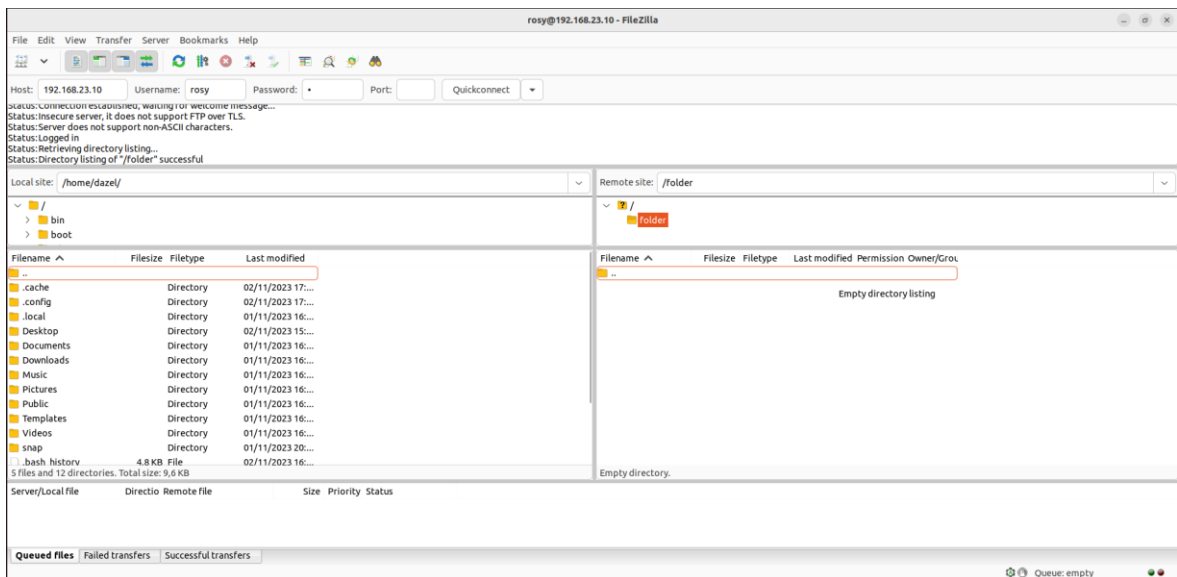


Figure 77: Log in to FileZilla

- Create a new folder on client side:

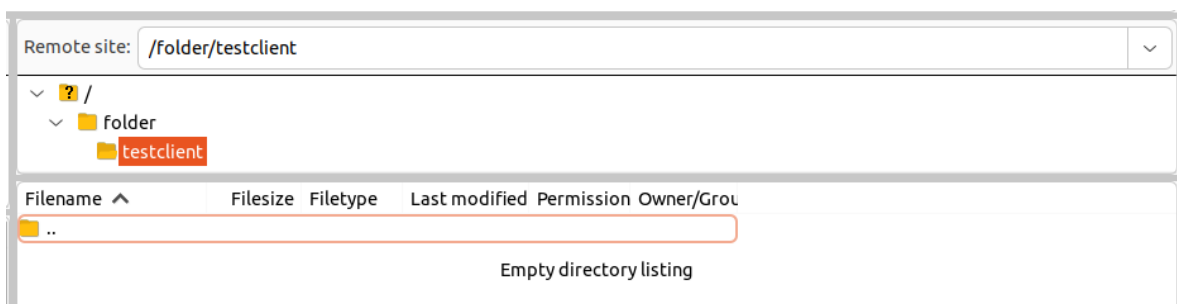


Figure 78: Create new folder on client

- Server can move to the new folder:

```
server@server:/$ cd folder
server@server:/folder$ ls -la
.  ..  testclient
server@server:/folder$ cd testclient/
server@server:/folder/testclient$ _
```

Figure 79: Move to the new folder

- User Dazel can not edit because permission is not granted.

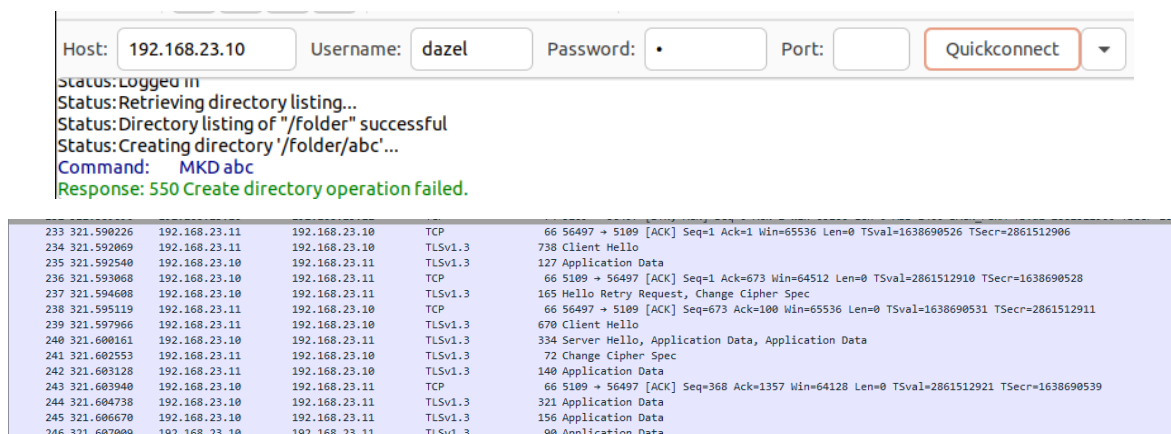


Figure 80: Test with Wireshark

3.2 Conclusion

The integration of Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP) services on a Linux server, along with the implementation of file server protocols, results in the establishment of a resilient and adaptable network environment. This project is particularly valuable for small to medium-sized enterprises, home networks, and any environment where efficient network management, reliable Internet access, and secure file sharing are essential. It provides a foundation that can be further expanded and customized to meet specific requirements, making it a valuable addition to any network infrastructure.

REFERENCES

- [1] W. by: S. Datta, “What is a network file system?,” Baeldung on Computer Science, <https://www.baeldung.com/cs/nfs> (accessed Nov. 4, 2023).
- [2] “What is SMB? what it decision makers need to know,” Visuality Systems, <https://visualitynq.com/resources/articles/what-is-smb-what-it-decision-makers-need-to-know/> (accessed Nov. 4, 2023).
- [3] “Dynamic Host Configuration Protocol (DHCP),” GeeksforGeeks, <https://www.geeksforgeeks.org/dynamic-host-configuration-protocol-dhcp/> (accessed Nov. 4, 2023).
- [4] “Network Address Translation (NAT),” GeeksforGeeks, <https://www.geeksforgeeks.org/network-address-translation-nat/> (accessed Nov. 4, 2023).
- [5] NairaBytes, “Home,” N.A.I.R.A.B.Y.T.E.S, http://www.nairabytes.net/linux/how-to-set-up-a-nat-router-on-ubuntu-server-16-04?fbclid=IwAR03lzfhLjXTypdAzaFZ_POX4Kk3MCtU91XAqsq8Dh32ipOEzo0eHeEkg7g (accessed Nov. 4, 2023).
- [6] “How to setup and configure an FTP server in linux?,” GeeksforGeeks, <https://www.geeksforgeeks.org/how-to-setup-and-configure-an-ftp-server-in-linux-2/> (accessed Nov. 4, 2023).
- [7] S. Zivanov, “How to install samba in ubuntu {+configuring and connecting},” Knowledge Base by phoenixNAP, <https://phoenixnap.com/kb/ubuntu-samba> (accessed Nov. 4, 2023).
- [8] C. D. S. Jeff Whitaker, “Linux NFS server: How to set up server and client,” NetApp BlueXP, [https://bluexp.netapp.com/blog/azure-anf-blg-linux-nfs-server-how-to-set-up-server-and-client#:~:text=Network%20File%20Sharing%20\(NFS\)%20is,have%20access%20to%20the%20folder](https://bluexp.netapp.com/blog/azure-anf-blg-linux-nfs-server-how-to-set-up-server-and-client#:~:text=Network%20File%20Sharing%20(NFS)%20is,have%20access%20to%20the%20folder) (accessed Nov. 4, 2023).
- [9] A. I. Nagori, How to configure Nat on ubuntu, <https://linuxhint.com/configure-nat-on-ubuntu/> (accessed Nov. 4, 2023).
- [10] I. Arora, “A step-by-step guide to set up a DHCP server on ubuntu,” LinuxForDevices, <https://www.linuxfordevices.com/tutorials/ubuntu/dhcp-server-on-ubuntu> (accessed Nov. 4, 2023).
- [11] “Ubuntu documentation,” NetworkConfigurationCommandLine - Community Help Wiki, <https://help.ubuntu.com/community/NetworkConfigurationCommandLine> (accessed Nov. 4, 2023).

- [12] “Introduction#,” Introduction - Netplan documentation, <https://netplan.readthedocs.io/en/stable/examples/> (accessed Nov. 4, 2023).
- [13] H.Maurya, Install and Configure Ansible on Ubuntu 22.04 Linux, <https://linux.how2shout.com/install-and-configure-ansible-on-ubuntu-22-04-linux/> (accessed Nov. 20, 2023).

APPENDIX

Task Assignment

Member	Task name	Completion Percentage
Trần Thị Mỹ Huyền	<ul style="list-style-type: none"> - Operation of NAT, DHCP, NFS, SMB, FTP, Ansible - Configurate NAT and DHCP - Result of NAT, DHCP, Ansible - Presentation and write the project report 	100%
Đoàn Hải Đăng	<ul style="list-style-type: none"> - Component of NAT, DHCP, NFS, SMB, FTP, Ansible - Configurate Ansible - Demo videos - Presentation and write the project report 	100%
Phan Thị Hồng Nhung	<ul style="list-style-type: none"> - General information about NAT, DHCP, NFS, SMB, FTP, Ansible - Configurate NFS, SMB, FTP - Result of NFS, SMB, FTP - Presentation and write the project report 	100%

Table 3: Task assignment

Self Assessment

Criteria	4	3	2	1
Report format (1 point)	X			
Presentation (1 point)	X			
Theory (2 point)	X			
Demonstration (5 point)	X			
Total	9			

Table 4: Self assessment