**Library Server**

**Step 1: Setting Up a VirtualBox Network Configuration**

On VirtualBox we have created two Virtual Machines:

- an Ubuntu Server

- a Linux workstation

Both Virtual Machines have been configured with two network adapters:

Adapter 1: Set as NAT for internet access.

Adapter 2: Set as Internal Network

The server has been configured with the following services:

DHCP, aiming to serve the local internal network

DNS, to resolve internal resources along with a redirector for external resources

Web Server, hosting a local webpage.

**Step 2: DHCP Server Configuration**

Installation of the server package has been done with the following commands:

sudo apt update

sudo apt install isc-dhcp-server

The DHCP server configuration file was edited to define the DHCP range for clients:

sudo nano /etc/dhcp/dhcpd.conf

The following lines have been modified to set up a basic subnet:

subnet 192.168.56.0 netmask 255.255.255.0 {

range 192.168.56.10 192.168.56.50;

option routers 192.168.56.1;

option domain-name-servers 8.8.8.8, 8.8.4.4;

}

The INTERFACESv4 in the DHCP server file is configured to ensure it listens only on the internal network adapter (in our case the interface named "enp0s8"):

sudo nano /etc/default/isc-dhcp-server

Modification:

INTERFACESv4="enp0s8"

With the necessary modifications done, we restarted and enabled the DHCP service:

sudo systemctl restart isc-dhcp-server

sudo systemctl enable isc-dhcp-server

After this, we set up a static IP on the internal network by editing the netplan configuration on the internal adapter:

sudo nano /etc/netplan/00-installer-config.yaml

The configuration was like this:

network:

ethernets:

enp0s3:

dhcp4: true

enp0s8:

dhcp4: false

addresses:

- 192.168.56.1/24

version: 2

We applied the changes with:

sudo netplan apply

Both the server and workstation now should have internet access via Adapter 1 (NAT). We tested this by pinging an external site:

ping google.com

We also tested it by pinging from the Linux workstation to the server:

ping 192.168.56.1

**Step 3: DNS Server Configuration**

To set up a DNS server on our Ubuntu server we used BIND9 and configured it to resolve internal resources and use a forwarder (redirector) for external DNS queries.

For the installation we used:

sudo apt install bind9

We configured the main "bind" configuration file in order to forward external queries:

sudo nano /etc/bind/named.conf.options

The modification is as follows:

options {

directory "/var/cache/bind";

// Forward queries for external resources

forwarders {

8.8.8.8;

8.8.4.4;

};

dnssec-validation auto;

auth-nxdomain no;

listen-on-v6 { any; };

};

The Google DNS was used as a public DNS server.

We configured the local zone file for internal resources by using the following command:

sudo nano /etc/bind/named.conf.local

We added the following zone configuration:

zone "example.local" {

type master;

file "/etc/bind/db.example.local";

};

In this case "example.local" is our internal domain.

We copied the default db.local template to create our custom zone file:

sudo cp /etc/bind/db.local /etc/bind/db.example.local

Then we configured this new file:

sudo nano /etc/bind/db.example.local

Modifications were as follows:

;

; BIND data file for example.local

;

$TTL 604800

@ IN SOA ns.example.local. admin.example.local. (

2 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

;

@ IN NS ns.example.local.

ns IN A 192.168.56.1

host1 IN A 192.168.56.10

We made sure to remodify the "named.conf.options" file, so that "bind" listens on the internal network interface:

sudo nano /etc/bind/named.conf.options

The final modification of this file was as follows:

options {

directory "/var/cache/bind";

listen-on { 192.168.56.1; 127.0.0.1; };

listen-on-v6 { none; };

forwarders {

8.8.8.8;

8.8.4.4;

};

dnssec-validation auto;

auth-nxdomain no;

};

We restarted "bind" so that the modifications were taken in consideration:

sudo systemctl restart bind9

At this point, we tested our configuration using our server IP address and running the following command to see if the DNS server can resolve an internal resource:

nslookup host1.example.local 192.168.56.1

The response we received was the IP address set for host1 in our bind configuration:

Server: 192.168.56.1

Address: 192.168.56.1#53

Name: host1.example.local

Address: 192.168.56.10

We also tested it for external resource forwarding, to ensure that forwarding works for external queries. We used the Google domain as follows:

nslookup google.com 192.168.56.1

The response was as follows:

Server: 192.168.56.1

Address: 192.168.56.1#53

Non-authoritative answer:

Name: google.com

Address: 172.217.23.206

Name: google.com

Address: 2a00:1450:400e:805:200e

**Step 4: Local Web Server Configuration**

To set up a web server on our Ubuntu server to host a local webpage, we used Nginx.

First, we proceeded with the installation:

sudo apt install nginx -y

Then we started and enabled the service:

sudo systemctl start nginx

sudo systemctl enable nginx

By default, Nginx serves files from /var/www/html, so we placed an "index.html" file here to act as our local webpage.

We used the following command to add the file and insert a welcome header inside it:

echo "<h1>Welcome to our Library Local Webpage</h1>" | sudo tee /var/www/html/index.html

We used the following command to test the configuration for syntax errors:

sudo nginx -t

With all configurations done, we restarted the service and checked its status:

sudo systemctl restart nginx

sudo systemctl status nginx

To confirm our modifications, we opened a web browser in out Linux workstation and entered the following:

http://192.168.56.1

We were greeted by the message: "Welcome to our Library Local Webpage".

**Step 5: Weekly backup of the configuration files for each service**

To set up a weekly backup of configuration files into a single compressed archive, we used cron to automate the backup process.

The paths to the configuration files for each service that we needed to back up are as follows:

Nginx: /etc/nginx/

BIND (DNS): /etc/bind/

DHCP: /etc/dhcp/

We started the configuration by creating a script that archives the configuration files into a compressed file:

sudo nano /usr/local/bin/backup\_configs.sh

We added the following script content to compress the directories:

#!/bin/bash

BACKUP\_DIR="/backups"

TIMESTAMP=$(date +"%Y-%m-%d")

BACKUP\_FILE="$BACKUP\_DIR/config\_backup\_$TIMESTAMP.tar.gz"

# To create the backup directory if it doesn't exist

mkdir -p "$BACKUP\_DIR"

# To archive and compress configuration files

tar -czf "$BACKUP\_FILE" /etc/nginx /etc/bind /etc/dhcp

echo "Backup completed: $BACKUP\_FILE"

Next, we made the script executable:

sudo chmod +x /usr/local/bin/backup\_configs.sh

With 'nano', we opened the cron jobs for editing:

sudo crontab -e

In the file, we added a cron job to run the script weekly every Sunday at 2 a.m. with the following line:

0 2 \* \* 0 /usr/local/bin/backup\_configs.sh

Having done the modifications, we verified the cron job with the following:

sudo /usr/local/bin/backup\_configs.sh

It was successful, and the backup file appeared in the /backups directory with a filename corresponding to the time when the command was used.

**Step 6: SSH Configuration**

We enabled remote management of owr server using SSH.

First, we proceeded with the installation of the OpenSSH packet:

sudo apt install openssh-server -y

We started and enabled the service:

sudo systemctl start ssh

sudo systemctl enable ssh

We configured the sshd\_config file using the following command:

sudo nano /etc/ssh/sshd\_config

In this file we added the following to disable root login, because it’s a good practice to disable SSH login for the root user and use a regular user with sudo privileges:

PermitRootLogin no

In the end we restarted the SSH service:

sudo systemctl restart ssh

From our Linux workstation we opened the terminal and tried to access the server remotely to verify that everything is working correctly:

ssh elvis@192.168.56.1

The password was inserted and the connection was successful.