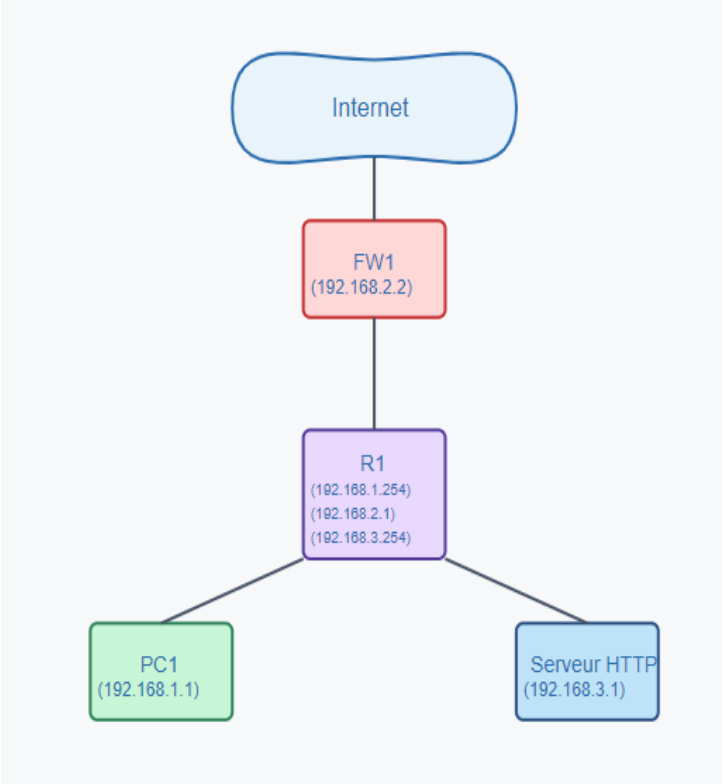
Virtual Network Infrastructure Configuration in VMware

Step 1: Setting up the Test Environment in VMware

To create my test environment, I start by cloning an existing machine (a Kali Linux) in VMware Workstation. Here's how I proceed:

VM Names:

- PC1 (workstation)
- FW1 (firewall)
- R1 (router)
- HTTP Server



Virtual Networks Configuration

To structure the subnets, I add virtual networks in VMware following these steps:

Edit → Virtual Network Editor → Add Network

- vmnet2: for network between PC1 and R1 - 192.168.1.0/24
- vmnet3: for network between R1 and FW1 - 192.168.2.0/24
- vmnet4: for network between R1 and HTTP Server - 192.168.3.0/24

I disable the DHCP service for each of these networks since I'll configure IP addresses manually.

Network Interface Configuration for VMs

Then, I configure the network adapters for each VM according to the appropriate subnet:

PC1:

- Network Adapter → Custom: vmnet2

FW1:

- Network Adapter 1 → NAT (Internet access)
- Add Network Adapter → Custom: vmnet3

R1:

- Network Adapter 1 → Custom: vmnet2
- Add Network Adapter → Custom: vmnet3
- Add Network Adapter → Custom: vmnet4

HTTP Server:

- Network Adapter → Custom: vmnet4

Interface Summary Table

Machine	Interface	IP Address	Gateway
PC1	eth0	192.168.1.1/24	192.168.1.254
FW1	eth1	192.168.2.2/24	-
	eth0	DHCP	-
R1	eth0	192.168.1.254/24	-
	eth1	192.168.2.1/24	-
	eth2	192.168.3.254/24	-
HTTP Server	eth0	192.168.3.1/24	192.168.3.254

Configuring IP Addresses on Interfaces

On each VM, I assign the respective IP addresses:

PC1:

```
sudo ip addr add 192.168.1.1/24 dev eth0
```

FW1:

```
sudo ip addr add 192.168.2.2/24 dev eth1
```

R1:

```
sudo ip addr add 192.168.1.254/24 dev eth0
sudo ip addr add 192.168.2.1/24 dev eth1
sudo ip addr add 192.168.3.254/24 dev eth2
```

HTTP Server:

```
sudo ip addr add 192.168.3.1/24 dev eth0
```

Step 2: Routing Configuration

Enabling Routing

To enable routing between subnets, I activate IPv4 routing on R1 and FW1:

```
sudo sysctl -w net.ipv4.ip_forward=1
echo "net.ipv4.ip_forward=1" | sudo tee /etc/sysctl.d/99-routing.conf
sudo sysctl -p /etc/sysctl.d/99-routing.conf
```

Configuring Routes

PC1:

```
sudo ip route add default via 192.168.1.254
```

HTTP Server:

```
sudo ip route add default via 192.168.3.254
```

R1:

```
sudo ip route add 192.168.1.0/24 dev eth0
sudo ip route add 192.168.2.0/24 dev eth1
sudo ip route add 192.168.3.0/24 dev eth2
# Default Route
sudo ip route add default via 192.168.2.2
```

FW1:

```
sudo ip route add 192.168.1.0/24 via 192.168.2.1
sudo ip route add 192.168.3.0/24 via 192.168.2.1
```

Step 3: NAT and Firewall Configuration

NAT Configuration and Filtering Rules

Install and configure iptables on FW1 to enable NAT and redirect traffic to the HTTP server:

```
#installation iptables
```

```
sudo apt update
sudo apt install iptables iptables-persistent
```

```
#>>>>SCRIPT IPTABLES BLOCKED ICMP
#nettoyage/flush les rules existantes
sudo iptables -t nat -F
sudo iptables -F
#politiques par défaut
sudo iptables -P FORWARD DROP
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
#bloquer ICMP (ping)
sudo iptables -A FORWARD -p icmp -j DROP  # Bloque tous les ICMP qui traversent le pare-feu
sudo iptables -A INPUT -p icmp -j DROP    # Bloque les pings vers le pare-feu lui-même
#rules forwarding
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
#allow forwarding depuis tous les réseaux internes vers internet
sudo iptables -A FORWARD -i eth1 -o eth0 -p tcp -j ACCEPT  # TCP
sudo iptables -A FORWARD -i eth1 -o eth0 -p udp -j ACCEPT  # UDP
sudo iptables -A FORWARD -s 192.168.3.0/24 -o eth0 -p tcp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.3.0/24 -o eth0 -p udp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.1.0/24 -o eth0 -p tcp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.1.0/24 -o eth0 -p udp -j ACCEPT
#configuration NAT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
#sauvegarder les rules et les lister
sudo netfilter-persistent save
sudo netfilter-persistent reload
sudo iptables -L
```

Connectivity Tests and Verification

To verify connectivity between machines, first ensure ICMP protocol is unblocked:

```
#>>>>SCRIPT IPTABLES ALLOWED ICMP
#nettoyage/flush les rules existantes
sudo iptables -t nat -F
```

```
sudo iptables -F
#politiques par défaut
sudo iptables -P FORWARD DROP
sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
#rules forwarding
sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
#allow ICMP (ping)
sudo iptables -A FORWARD -p icmp -j ACCEPT
sudo iptables -A INPUT -p icmp -j ACCEPT
#allow forwarding vers internet depuis sous-réseaux
sudo iptables -A FORWARD -i eth1 -o eth0 -p tcp -j ACCEPT
sudo iptables -A FORWARD -i eth1 -o eth0 -p udp -j ACCEPT
sudo iptables -A FORWARD -i eth1 -o eth0 -p icmp -j ACCEPT # ICMP
sudo iptables -A FORWARD -s 192.168.3.0/24 -o eth0 -p tcp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.3.0/24 -o eth0 -p udp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.3.0/24 -o eth0 -p icmp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.1.0/24 -o eth0 -p tcp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.1.0/24 -o eth0 -p udp -j ACCEPT
sudo iptables -A FORWARD -s 192.168.1.0/24 -o eth0 -p icmp -j ACCEPT
#configuration nat
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
#sauvegarder et lister les rules
sudo netfilter-persistent save
sudo netfilter-persistent reload
sudo iptables -L
```

Ping tests:

From PC1:

```
ping 192.168.1.254
ping 192.168.3.1
ping 192.168.2.2
```

From R1:

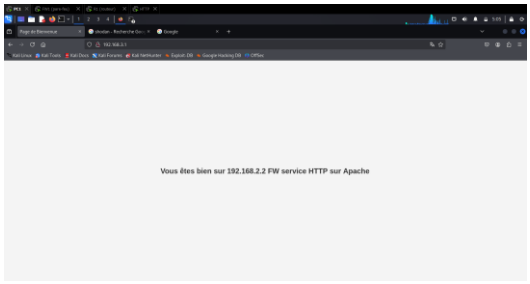
```
ping 192.168.1.1
ping 192.168.3.1
ping 192.168.2.2
```

Internet Access Tests:

```
ping 8.8.8.8
traceroute 8.8.8.8
```

HTTP Server Test:

```
sudo apt install apache2
```



Summary

This configuration enables:

- Communication between all internal networks
- Internet access from all internal networks via NAT
- Correct routing between all network segments
- ICMP (ping) is blocked
- TCP and UDP are allowed for Internet access
- NAT works for all internal networks

Useful Commands

- Interface status: ip addr show
- Routing tables: ip route show
- NAT rules: sudo iptables -t nat -L -v -n
- System logs: sudo tail -f /var/log/syslog
- Testing connectivity: wget google.com, ping, curl, cewl http://192.168.3.1