IUT Villetaneuse
Université Sorbonne Paris Nord

Suivie par M.M'lik

SAE21-Construire un Réseau Informatique pour une petite structure

Réaliser par :
Rafael MAIA - Jason LAMVU
- Rayan RIGOULET - Andon
PENDICHEV - Erwin JEAN

2024/2025

Tables des matières :

Introduction :

Dans le cadre de la SAE 21, nous avons eu pour mission de concevoir et déployer un réseau informatique complet pour une entreprise possédant deux sites distants : **Lyon** et **Grenoble**.

Le projet a été divisé en plusieurs rôles bien définis pour reproduire une organisation professionnelle :

Architecte réseau : responsable de la conception de l'architecture réseau, il a élaboré le plan d'adressage IP, mis en place les VLANs pour segmenter les différents services (direction, production, serveurs).

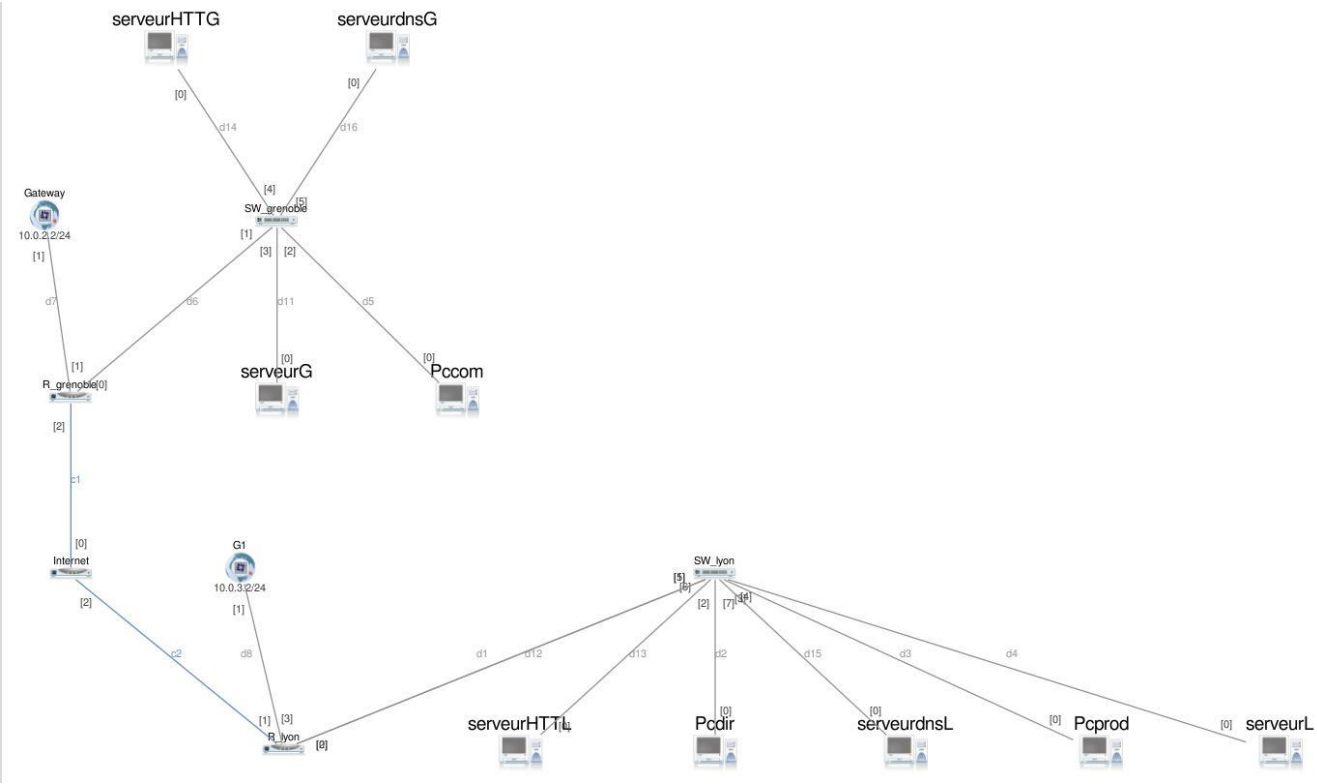
Administrateur systèmes : chargé d'installer et de configurer les services réseaux essentiels sur les serveurs. Cela comprenait la mise en place d'un serveur DHCP pour l'attribution dynamique des adresses IP, un serveur DNS pour la résolution des noms internes, et un serveur HTTP pour héberger un site web accessible dans l'entreprise.

Technicien sécurité : responsable de la sécurisation du réseau, il a configuré les ACL pour filtrer les communications entre les VLANs, mis en œuvre le NAT pour permettre aux machines internes d'accéder à Internet tout en restant protégées, et s'est assuré que la segmentation entre les services était bien respectée.

Testeur/Qualité : chargé de vérifier l'ensemble de l'architecture réseau. Son rôle était de réaliser des tests de connectivité (ping, traceroute), de valider le bon fonctionnement des services (DNS, HTTP, DHCP), et d'assurer que les communications entre les deux sites étaient opérationnelles.

Schéma réseau :

Voici le schéma du réseau que nous avons décidé de réaliser :



Plan d'adressage IP :

Le plan d'adressage est le suivant : (il a été réaliser avec le DHCP en routage dynamique)

Le site de Grenoble a pour adresse réseau 10.0.10.X/24 comme on le voit ici :

```
# IP statique
ip addr add 10.0.10.2/24 dev eth0
ip link set eth0 up
ip route add default via 10.0.10.254
echo "nameserver 10.0.10.3" > /etc/resolv.conf

# Interface DHCP
echo 'INTERFACESv4="eth0"' > /etc/default/isc-dhcp-server

# DHCP config
cat > /etc/dhcp/dhcpd.conf <<EOF
default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 10.0.10.0 netmask 255.255.255.0 {
  range 10.0.10.100 10.0.10.200;
  option routers 10.0.10.254;
  option domain-name-servers 10.0.10.2;
}
EOF

# Démarrage
/etc/init.d/isc-dhcp-server restart
```

Le site de Lyon a un réseau diviser en 2 car il y a 2 Vlans, d'un côté le réseau 10.0.20.X/24 et aussi 10.0.30.X/24 comme on le voit ici :

```
#!/bin/bash

# Monter les interfaces physiques
ip addr add 10.0.20.2/24 dev eth0   # câble vers VLAN 20
ip addr add 10.0.30.2/24 dev eth1   # câble vers VLAN 30

ip link set eth0 up
ip link set eth1 up
echo "nameserver 10.0.30.3" > /etc/resolv.conf

# Déclarer les interfaces DHCP
echo 'INTERFACESv4="eth0 eth1"' > /etc/default/isc-dhcp-server

# Config DHCP
cat > /etc/dhcp/dhcpd.conf <<EOF
default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 10.0.20.0 netmask 255.255.255.0 {
  range 10.0.20.100 10.0.20.200;
  option routers 10.0.20.254;
  option domain-name-servers 10.0.20.2;
}

subnet 10.0.30.0 netmask 255.255.255.0 {
  range 10.0.30.100 10.0.30.200;
  option routers 10.0.30.254;
  option domain-name-servers 10.0.30.2;
}
EOF

# Démarrer le serveur DHCP
/etc/init.d/isc-dhcp-server restart
```

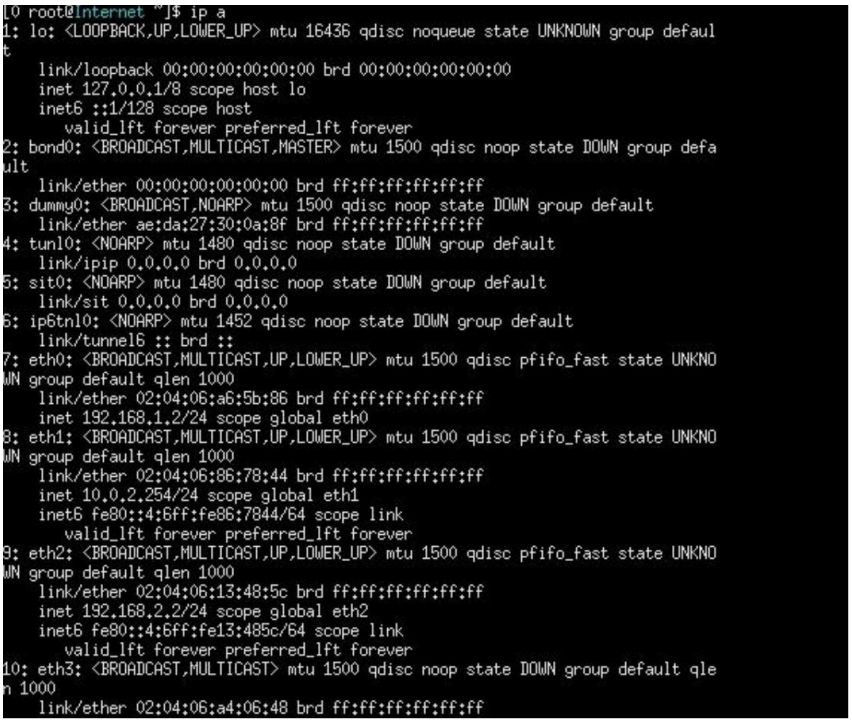
Vérification de la fonctionnalité de l'architecture :

Pour verifier que l'architecture de notre réseau fonctionne correctement nous avons tester ce réseau en transmettant des paquets entre les différents composants du réseau (avec la commande ping).

Nous avons tout d'abord besoin de connaître les différentes adresses IP des équipements ainsi que l'interface à laquelle sont attribué ces adresses.

Rgrenoble:

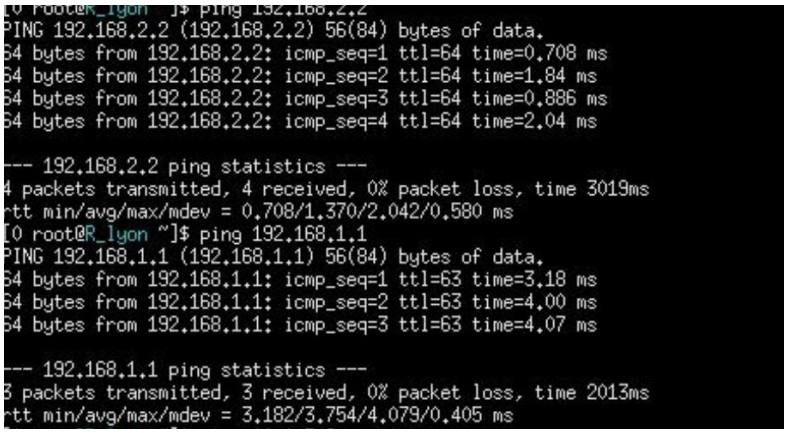
Rinternet:

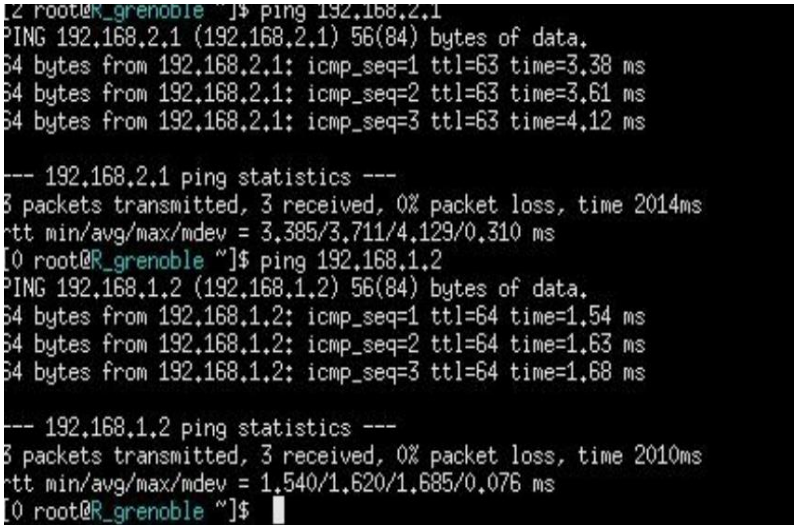

[0 root@internet ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN group defaul
t
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: bond0: <BROADCAST,MULTICAST,MASTER> mtu 1500 qdisc noop state DOWN group defa
ult
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default
link/ether ae:da:27:30:0a:8f brd ff:ff:ff:ff:ff:ff
4: tunl0: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
5: sit0: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/sit 0.0.0.0 brd 0.0.0.0
6: ip6tnl0: <NOARP> mtu 1452 qdisc noop state DOWN group default
link/tunnel6 :: brd ::
7: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
link/ether 02:04:06:a6:5b:86 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.2/24 scope global eth0
8: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
link/ether 02:04:06:86:78:44 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.254/24 scope global eth1
inet6 fe80::4:6ff:fe86:7844/64 scope link
valid_lft forever preferred_lft forever
9: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN group default qlen 1000
link/ether 02:04:06:13:48:5c brd ff:ff:ff:ff:ff:ff
inet 192.168.2.2/24 scope global eth2
inet6 fe80::4:6ff:fe13:485c/64 scope link
valid_lft forever preferred_lft forever
10: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qle
n 1000
link/ether 02:04:06:a4:06:48 brd ff:ff:ff:ff:ff:ff

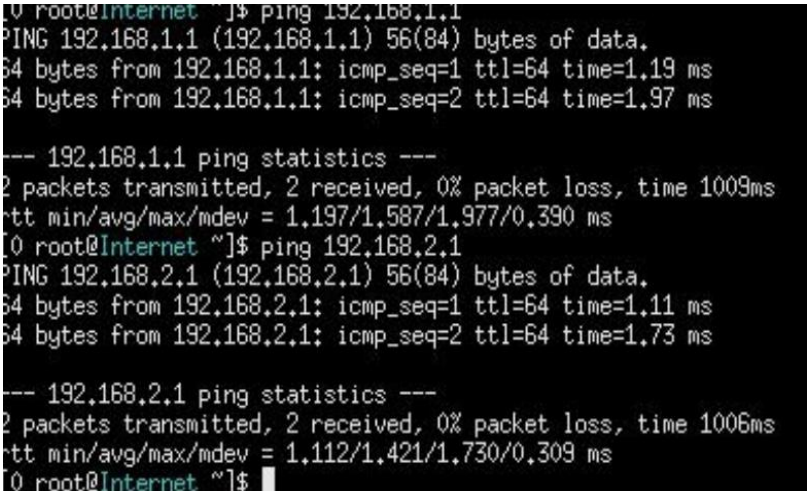
Rlyon:


root@R_lyon ~]$ ip a
lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN group defaul
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
bond0: <BROADCAST,MULTICAST,MASTER> mtu 1500 qdisc noop state DOWN group defa
lt
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default
link/ether 66:c0:45:d1:e9:5a brd ff:ff:ff:ff:ff:ff
tunl0: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
sit0: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/sit 0.0.0.0 brd 0.0.0.0
ip6tnl0: <NOARP> mtu 1452 qdisc noop state DOWN group default
link/tunnel6 :: brd ::
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
group default qlen 1000
link/ether 02:04:06:2d:a2:39 brd ff:ff:ff:ff:ff:ff
inet 10.0.30.254/24 scope global eth0
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
group default qlen 1000
link/ether 02:04:06:2b:c7:e7 brd ff:ff:ff:ff:ff:ff
inet 192.168.2.1/24 scope global eth1
inet6 fe80::4:6ff:fe2b:c7e7/64 scope link
valid_lft forever preferred_lft forever
eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
group default qlen 1000
link/ether 02:04:06:e6:bb:fa brd ff:ff:ff:ff:ff:ff
inet 10.0.20.254/24 scope global eth2
inet6 fe80::4:6ff:fee6:bbfa/64 scope link
valid_lft forever preferred_lft forever
: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qle
1000
link/ether 02:04:06:03:32:f2 brd ff:ff:ff:ff:ff:ff

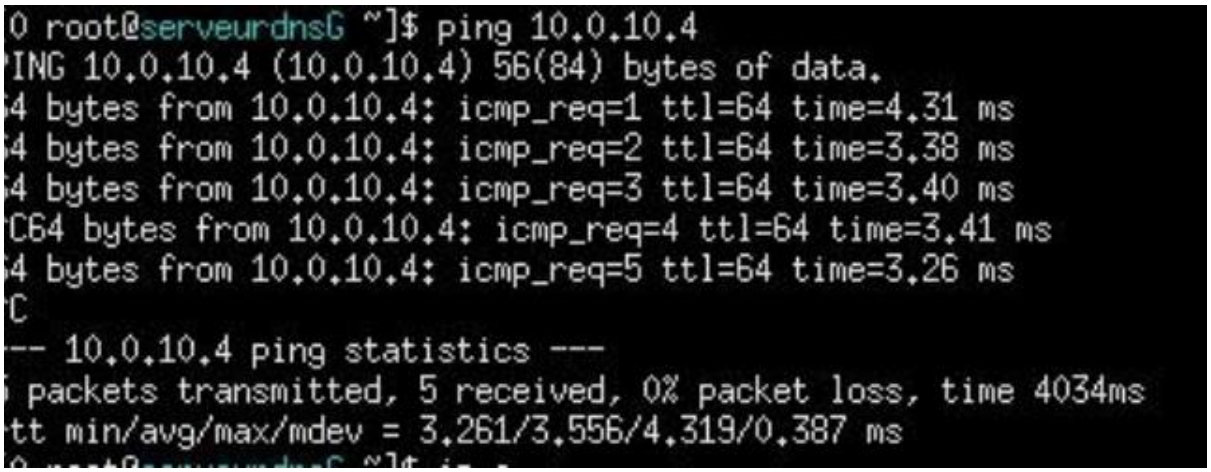
Voici maintenant les pings entre les différents routeurs :

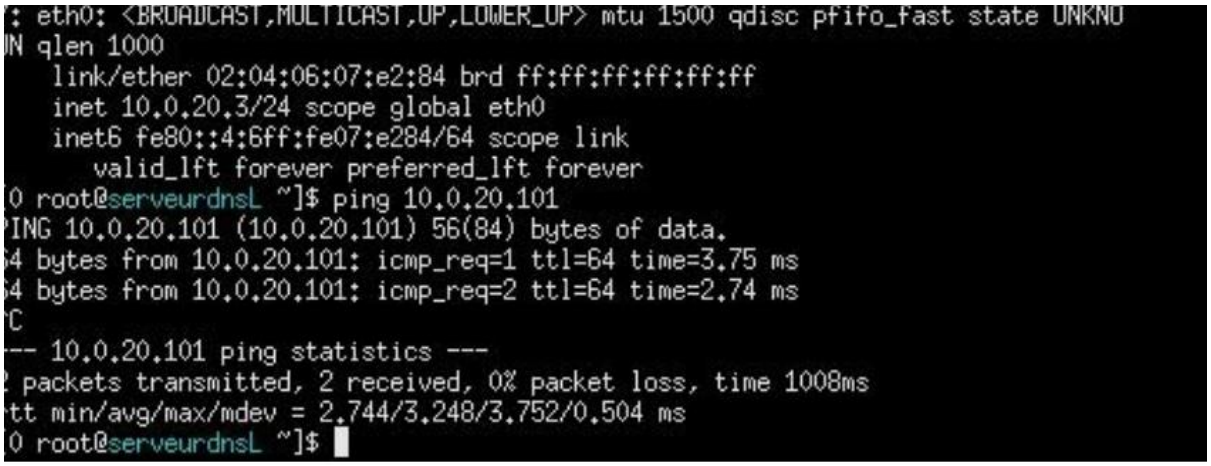


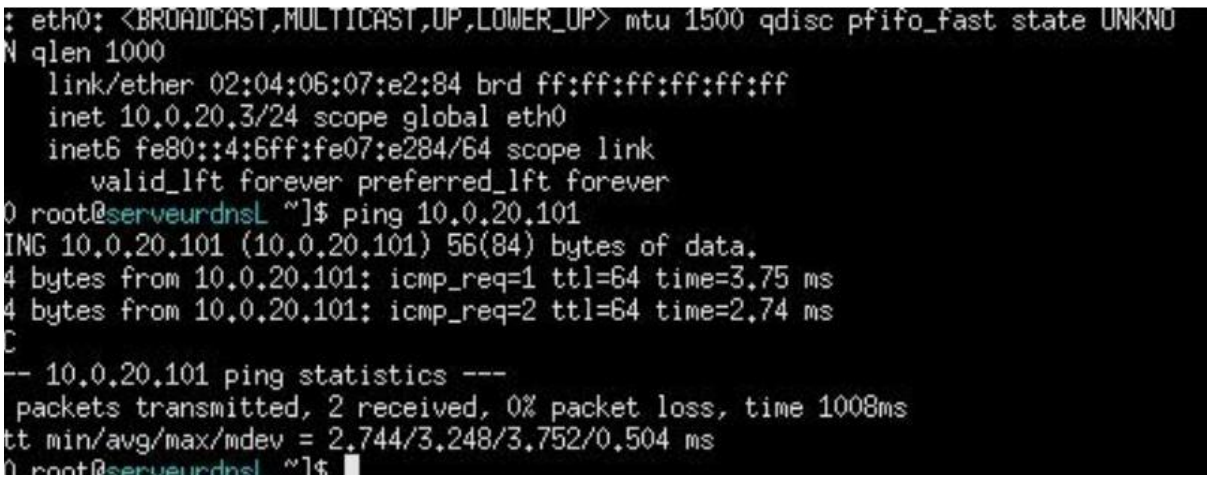




Encore quelques tests pour démontrer que les machines du notre réseau arrivent à communiquer entre eux







Configuration des équipements :

Pour les routeurs :

Nous devions configurer également les routes des 3 routeurs pour permettre les communications entre les 2 sites donc voici comment nous l'avons fait :

Tout d'abord nous avons configurer les switches de Grenoble et Lyon vers leurs réseaux :

Routeur de Grenoble :

```
18
19 # Routes
20 ip route add 10.0.20.0/24 via 192.168.1.2 dev eth2
21 ip route add 10.0.30.0/24 via 192.168.1.2 dev eth2
22 ip route add 192.168.2.0/24 via 192.168.1.2 dev eth2
23 ip route add default via 10.0.2.2 dev eth1
24
25 #Routage Ip
26 echo 1 > /proc/sys/net/ipv4/ip_forward
27
28 # Accés Internet
29 iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
30
31 #ACL
32 iptables -A FORWARD -s 10.0.10.0/24 -o eth2 -j DROP
33
34 #Autorisation du reste
35 iptables -A FORWARD -j ACCEPT
```

Routeur de Lyon :

```
19 # Routes
20 ip route add 10.0.10.0/24 via 192.168.2.2 dev eth1
21 ip route add 192.168.1.0/24 via 192.168.2.2 dev eth1
22 ip route add default via 192.168.2.2 dev eth1
23
24 # Routage IP
25 echo 1 > /proc/sys/net/ipv4/ip_forward
26
27 #Accés Internet
28
29 iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
30
31 # ACL
32 iptables -A FORWARD -s 10.0.20.0/24 -d 10.0.10.0/24 -j DROP
33
34
35 # Autorisation du reste
36 iptables -A FORWARD -j ACCEPT
```

Ensuite nous avons configurer le ***routeur internet*** qui fais le lien entre les 2 sites :

```
19 # Activer le routage IP
20 echo 1 > /proc/sys/net/ipv4/ip_forward
21
22 # Routes
23 ip route add 10.0.10.0/24 via 192.168.1.1 dev eth0
24 ip route add 10.0.20.0/24 via 192.168.2.1 dev eth2
25 ip route add 10.0.30.0/24 via 192.168.2.1 dev eth2
26
```

Pour le Vlans :

Nous avons pris la décision de crée des **Vlans** pour pouvoir faire en sorte que chaque service ai sont réseau : donc pour Lyon par exemple nous avons fait les VLans suivant :

SW_lyon fichier de configuration

```
# VLAN 20 (PROD)
vlan/create 20
vlan/addport 20 5      # trunk → vers R_lyon
vlan/addport 20 3      # pcprod
vlan/addport 20 4      # serveurL

# VLAN 30 (DIR)
vlan/create 30
vlan/addport 30 1      # trunk → vers R_lyon
vlan/addport 30 2      # pcdir
vlan/addport 30 6      # serveurHTTP_L
vlan/addport 30 7      # serveurDNS_L

```

Nous avons donc décidé de faire ces VLans pour que les 2 services de Lyon puissent avoir 2 réseaux différents tout en étant sur le mème switch.

Nous avons donc fait la configuration de plusieurs comme **le DHCP** montrer avant, le **service DNS** ou encore **le serveur HTTP** :

Pour le serveur DHCP :

Pour configurer **le serveur DHCP** nous avons configurer les fichiers /etc/dhcp/dhcpd.conf et dans celui si nous avons mis le contenue suivant (dans ce cas pour le réseau de Grenoble) :

```
# IP statique
ip addr add 10.0.10.2/24 dev eth0
ip link set eth0 up
ip route add default via 10.0.10.254
echo "nameserver 10.0.10.3" > /etc/resolv.conf

# Interface DHCP
echo 'INTERFACESv4="eth0"' > /etc/default/isc-dhcp-server

# DHCP config
cat > /etc/dhcp/dhcpd.conf <<EOF
default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 10.0.10.0 netmask 255.255.255.0 {
  range 10.0.10.100 10.0.10.200;
  option routers 10.0.10.254;
  option domain-name-servers 10.0.10.2;
}
EOF

# Démarrage
/etc/init.d/isc-dhcp-server restart
```

Ce script configure une machine avec une IP statique et met en place un serveur DHCP. Le serveur distribue des adresses IP entre 10.0.10.100 et 10.0.10.200 aux clients du réseau. Il indique aussi aux clients la passerelle (10.0.10.254) et le serveur DNS (10.0.10.2). Enfin, le service DHCP est redémarré pour appliquer cette configuration.

Pour le DNS et Filtrage :

Nous avons configuré un service DNS local sur chacun des deux sites : Lyon et Grenoble. L'objectif était de permettre la résolution de noms au sein de notre réseau. Nous avons choisi d'utiliser le fichier /etc/hosts présent sur chaque machine pour associer les noms des machines à leurs adresses IP. Ce fichier agit comme un mini serveur DNS local et permet de résoudre les noms sans infrastructure supplémentaire.

Chaque machine possède ainsi un /etc/hosts mis à jour, répertoriant toutes les IP et noms des équipements présents sur le réseau.

Des tests ont été réalisés à Lyon et à Grenoble pour vérifier le bon fonctionnement de la résolution DNS locale : nous avons réalisé des pings en utilisant les noms des machines au lieu de leurs adresses IP. Les résultats ont confirmé que la résolution de noms était fonctionnelle.

(Exemple avec celui de lyon depuis Pcprod vers pcdir)

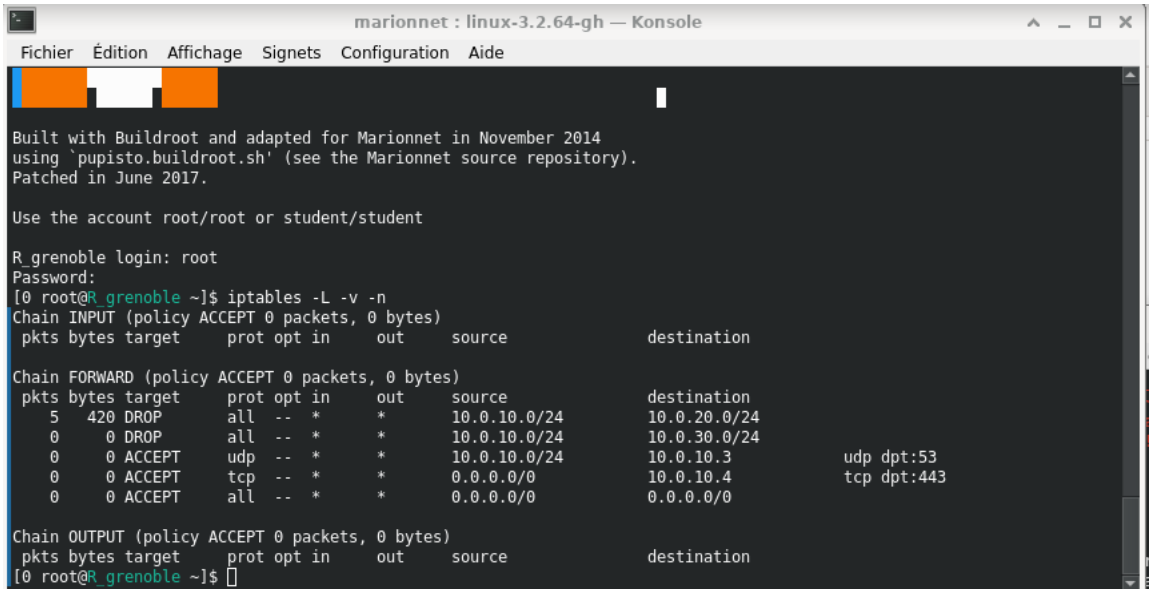
```
[1 root@Pcprod ~]$ ping pcdir.lyon.local
PING pcdir.lyon.local (10.0.20.101) 56(84) bytes of data.
64 bytes from 10.0.20.101: icmp_req=1 ttl=64 time=2.16 ms
64 bytes from 10.0.20.101: icmp_req=2 ttl=64 time=0.697 ms
64 bytes from 10.0.20.101: icmp_req=3 ttl=64 time=0.787 ms
64 bytes from 10.0.20.101: icmp_req=4 ttl=64 time=0.420 ms
^C64 bytes from 10.0.20.101: icmp_req=5 ttl=64 time=0.441 ms

--- pcdir.lyon.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 30034ms
rtt min/avg/max/mdev = 0.420/0.901/2.164/0.648 ms
[0 root@Pcprod ~]$
```

Ensuite pour assurer la sécurité nous avons fait du filtrage, à Grenoble, des règles iptables ont été mises en place pour bloquer l'accès aux VLANs de Lyon tout en autorisant les requêtes DNS locales. Cela garantit que seules les communications DNS autorisées sont permises : (exemple avec le ping vers le serveurg qui est interdit)

```
[0 root@Pcprod ~]$ ping 10.0.10.2
PING 10.0.10.2 (10.0.10.2) 56(84) bytes of data.
^C
--- 10.0.10.2 ping statistics ---
176 packets transmitted, 0 received, 100% packet loss, time 175719ms
```

Donc on voit que le ping a bien été bloqué.

Voici la liste des règles de filtrage :



Pour sécuriser le réseau, nous avons mis en place des règles de filtrage. Tout le trafic de Grenoble vers les VLANs de Lyon (VLAN 20 et VLAN 30) est bloqué pour empêcher les communications non autorisées. En revanche, les requêtes DNS locales vers le

serveur 10.0.10.3 et l'accès HTTPS local vers 10.0.10.4 sont autorisés. Enfin, tout le reste du trafic est accepté par une règle générale.

Pour le serveur HTTP :

Nous avons installé un serveur HTTP (Apache) sur les serveurs des deux sites. Le serveur de Lyon écoute sur le port 443 pour proposer un accès sécurisé en HTTPS, tandis que le serveur de Grenoble est accessible en HTTP classique.

Les serveurs web ont été configurés pour répondre aux requêtes locales, en publiant une page d'accueil simple permettant de vérifier la bonne connexion au service.

Des tests ont été réalisés à Lyon et à Grenoble pour vérifier l'accès aux serveurs web : (ici l'exemple est effectué avec le serveur HTTP de grenoble)

```
[0 root@Pccom ~]$ curl -k https://serveurhttps.grenoble.local
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
[0 root@Pccom ~]$
```

On voit donc bien avec cette capture d'écran que ça fonctionne bien.

Pour le SSH :

Pour configurer les communications sur SSH il faut tout d'abord démarrer le service SSH sur chaque machine pour pouvoir l'utiliser comme ceci :

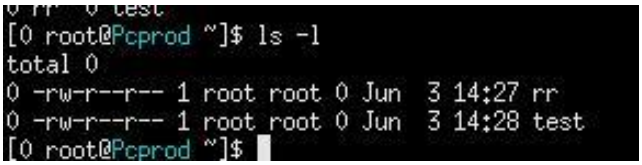
```
[0 root@Pcprod ~]$ service ssh start
[ ok ] Starting OpenBSD Secure Shell server: sshd.
[0 root@Pcprod ~]$
```

Maintenant on va essayer de lancer une connexion depuis un serveurs (dans l'exemple suivant serveurdnsL) vers une autre machine (ici Pcprod) :

```
[130 root@serveurdnsL ~]$ ssh 10.0.20.100
root@10.0.20.100's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Tue Jun  3 14:28:00 2025
```

Pour vérifier que la connexion SSH est bien configurée nous allons créer un fichier test avec le serveurdns qui est connecté à pcprod :

Ensuite nous vérifions que sur pc prod le fichier est bien allé récupérer :



Ensuite pour sécuriser la connexion via SSH nous avons mis en place une authentification par clés. Le principe consiste à générer une paire de clés (publique et privée) sur le client :

Avec ***ssh-keygen.***

La clé publique est copiée sur le serveur dans le fichier :

~/.ssh/authorized_keys.

Ainsi, la connexion SSH se fait sans mot de passe, uniquement avec la clé privée, ce qui renforce la sécurité.

Routage inter-Vlan :

Dans le cadre de la segmentation du réseau en fonctions des services, nous avons mis en place un routage inter-Vlan afin de permettre et restreindre la communication entre les différents Vlan de chaque site. Le routage fonctionne pour tous les Vlan, mais nous avons placé un filtre contrôlé via ACL qui va faire en sorte que les services Http et DNS soient autorisé, et que le Vlan Production ne puissent pas accéder à la Direction.

```
# Routage IP
echo 1 > /proc/sys/net/ipv4/ip_forward

#Accés Internet

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

# ACL
iptables -A FORWARD -s 10.0.20.0/24 -d 10.0.10.0/24 -j DROP


# Autorisation du reste
iptables -A FORWARD -j ACCEPT
```

Mise en place du NAT :

Pour permettre aux machines des Vlan internes d'accéder à Internet tout en restant protégée, nous avons mis en place un NAT (masquerading) sur les deux différents routeurs.

Justifications techniques :

Pour les routeurs :

Pour la communication entre les 2 sites nous avons décidé de crée un routeur Internet qui joue le rôle comme son nom l'indique d'internet car sur marionet le lien entre 2 Gateway ne peut pas se faire directement donc la communication entre les 2 sites aurais été impossible. On préfère utiliser un routeur internet qui jouera ce rôle et pour faire en sorte que les 2 sites peuvent communiquer ensemble.

Pour le serveur DHCP :

Nous avons décidé d'utiliser 2 serveurs DHCP pour simplifier le routage et donc pour éviter de faire un routage statique qui serait long à effectuer et pas simple, alors que le routage statique avec les 2 serveurs DHCP que nous avons configurer comme fais dans les TPs.

Pour le serveur DNS :

Nous avons choisi d'utiliser le fichier /etc/hosts pour la résolution DNS car il s'agit d'une solution simple et rapide à mettre en place dans un réseau de petite taille comme celui de notre projet. Cela nous a permis de fournir un service DNS fonctionnel.

L'utilisation d'iptables pour filtrer le trafic DNS garantit une meilleure sécurité du réseau en limitant les communications aux seules requêtes nécessaires. De plus, en configurant un forwarder vers un serveur DNS public, nous avons permis aux machines du réseau de résoudre également des noms externes tout en conservant un contrôle local sur la résolution des noms internes.

Pour le serveur HTTP :

Nous avons choisi de déployer des serveurs HTTP locaux afin de faciliter l'accès aux ressources internes de l'entreprise sans passer par Internet. Le choix d'utiliser HTTPS à Lyon permet de sécuriser les échanges sensibles. Grâce au filtrage réseau, nous limitons l'accès uniquement aux services nécessaires et renforçons la sécurité globale du réseau en empêchant les communications non autorisées entre VLANs.

Pour le serveur SSH :

Pour la connexion SSH nous avons décidé de lancer le service SSH de base marionnet car il est déjà configuré de base.

On a donc effectué les connexions de base et la connexion fonctionne parfaitement et permettent de se connecter a d'autre machines à distance et même de crée des fichiers etc.

Ensuite pour sécuriser la connexion pour éviter que des personnes non autorisées puissent se connecter au serveur. Utiliser une clé au lieu d'un mot de passe rend la connexion beaucoup plus sûre, car une clé est beaucoup plus dure à deviner ou à craquer.

Routage inter-Vlan :

Le routage inter-Vlan a été mis en place pour permettre une contrôlé entre les Vlan lorsqu'il est nécessaire. Au niveau des routeurs nous avons attribué des Ip spécifique en fonction des interfaces :

Pour garantir une sécurité entre les services, nous avons mis en place des règles de filtrages en employant la commande iptables, le but était d'empêcher les communications entre les Vlans, tout en permettant les communications nécessaires comme le DNS/ HTTP

Pour exemple nous avons bloqué l'accès du Vlans Production vers le Vlan Direction

Mise en place du Nat :

Afin de permettre aux machines d'accéder à internet, nous avons configuré un mécanisme de traduction d'adresse sur les routeurs ; Pour cela à partir du fichier de configuration de chaque routeur nous avons utilisé la commande "iptables"

La commande permet de substituer l'adresse Privée des machines par l'adresse publique du routeur, par cette configuration les machines des Vlans Production, Commercial et Direction peuvent accéder à Internet sans être directement exposées

Conclusion :

Ce projet nous a permis de découvrir et de mettre en pratique les étapes nécessaires pour construire un réseau complet dans un projet professionnel. Nous avons appris à concevoir une architecture réseau adaptée à une entreprise qui contient plusieurs sites, à segmenter le réseau avec des VLANs pour mieux organiser et sécuriser les échanges, et à mettre en place des services essentiels comme le DHCP, le DNS et un serveur web.

En travaillant en équipe, chacun avec un rôle bien précis, nous avons aussi développé des compétences en gestion de projet et en collaboration. Ce travail nous a aidés à mieux comprendre l'importance de la planification, de la sécurité réseau et du bon fonctionnement des communications entre différents sites d'une même entreprise.

Grâce à cette SAE, nous avons renforcé nos compétences techniques et professionnelles, et nous sommes mieux préparés à répondre aux besoins réels des entreprises dans le domaine des réseaux informatiques.

Annexes :

Ping qui montre le bon fonctionnement du réseau :

```
[1 root@serveurdnsG ~]$ ping 10.0.20.2
PING 10.0.20.2 (10.0.20.2) 56(84) bytes of data.
^C
--- 10.0.20.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms

[1 root@serveurdnsG ~]$
```

```
[0 root@serveurG ~]$ ping 10.0.2.2
ING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
4 bytes from 10.0.2.2: icmp_req=1 ttl=254 time=4.19 ms
4 bytes from 10.0.2.2: icmp_req=2 ttl=254 time=4.39 ms
4 bytes from 10.0.2.2: icmp_req=3 ttl=254 time=5.10 ms
C
-- 10.0.2.2 ping statistics ---
 packets transmitted, 3 received, 0% packet loss, time 2017ms
tt min/avg/max/mdev = 4.193/4.566/5.109/0.400 ms
```

```
[0 root@R_grenoble ~]$ ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=255 time=2.48 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=255 time=1.58 ms

--- 10.0.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.588/2.034/2.480/0.446 ms
[0 root@R_grenoble ~]$
```

Voici les tests du DNS de Lyon :

```
[0 root@Pcdir ~]$ nslookup pcdir.lyon.local
Server:         10.0.20.3
Address:        10.0.20.3#53

Name:   pcdir.lyon.local
Address: 10.0.30.101
```

```
[0 root@Pcprod ~]$ nslookup pcprod.lyon.local
Server:         10.0.20.3
Address:        10.0.20.3#53

Name:   pcprod.lyon.local
Address: 10.0.20.100

[0 root@Pcprod ~]$
```

```
[0 root@serveurHTTL ~]$ nslookup serveurhttps.lyon.local
Server:         10.0.20.3
Address:        10.0.20.3#53

Name:   serveurhttps.lyon.local
Address: 10.0.30.4

[0 root@serveurHTTL ~]$
```

Voici les tests du DNS de Grenoble :

```
[0 root@serveurG ~]$ nslookup serveurg.grenoble.local
Server:         10.0.10.3
Address:        10.0.10.3#53

Name:   serveurg.grenoble.local
Address: 10.0.10.2

[0 root@serveurG ~]$
```

```
    inet 10.0.10.4/24 scope global eth0
[0 root@serveurHTTG ~]$ nslookup serveurhttps.grenoble.local
Server:         10.0.10.3
Address:        10.0.10.3#53

Name:   serveurhttps.grenoble.local
Address: 10.0.10.4
```

```
       valid_lft forever preferred_lft forever
[0 root@Pccom ~]$ nslookup pccom.grenoble.local
Server:         10.0.10.3
Address:        10.0.10.3#53

Name:   pccom.grenoble.local
Address: 10.0.10.100

[0 root@Pccom ~]$
```

```
       valid_lft forever preferred_lft forever
[0 root@Pccom ~]$ nslookup pccom.grenoble.local
Server:         10.0.10.3
Address:        10.0.10.3#53

Name:   pccom.grenoble.local
Address: 10.0.10.100

[0 root@Pccom ~]$
```

Voici d'autre ping avec le DNS :

Partie Lyon :

```
[1 root@Pcprod ~]$ ping pcdir.lyon.local
PING pcdir.lyon.local (10.0.20.101) 56(84) bytes of data.
64 bytes from 10.0.20.101: icmp_req=1 ttl=64 time=2.16 ms
64 bytes from 10.0.20.101: icmp_req=2 ttl=64 time=0.697 ms
64 bytes from 10.0.20.101: icmp_req=3 ttl=64 time=0.787 ms
64 bytes from 10.0.20.101: icmp_req=4 ttl=64 time=0.420 ms
^C64 bytes from 10.0.20.101: icmp_req=5 ttl=64 time=0.441 ms

--- pcdir.lyon.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 30034ms
rtt min/avg/max/mdev = 0.420/0.901/2.164/0.648 ms
[0 root@Pcprod ~]$
```

```
[1 root@Pcdir ~]$ ping pcprod.lyon.local
PING pcprod.lyon.local (10.0.20.100) 56(84) bytes of data.
64 bytes from 10.0.20.100: icmp_req=1 ttl=64 time=0.425 ms
64 bytes from 10.0.20.100: icmp_req=2 ttl=64 time=0.886 ms
```

```
[0 root@Pcprod ~]$ ping serveurhttps.lyon.local
PING serveurhttps.lyon.local (10.0.30.4) 56(84) bytes of data.
64 bytes from 10.0.30.4: icmp_req=1 ttl=63 time=3.54 ms
```

```
[127 root@Pcdir ~]$ ping serveurhttps.lyon.local
PING serveurhttps.lyon.local (10.0.30.4) 56(84) bytes of data.
64 bytes from 10.0.30.4: icmp_req=1 ttl=63 time=4.34 ms
64 bytes from 10.0.30.4: icmp_req=2 ttl=63 time=1.23 ms
64 bytes from 10.0.30.4: icmp_req=3 ttl=63 time=2.03 ms
^Z
[1]+  Stopped                 ping serveurhttps.lyon.local
```

Partie Grenoble :

```
[0 root@serveurHTTG ~]$ ping pccom.grenoble.local
PING pccom.grenoble.local (10.0.10.100) 56(84) bytes of data.
64 bytes from 10.0.10.100: icmp_req=1 ttl=64 time=1.84 ms
64 bytes from 10.0.10.100: icmp_req=2 ttl=64 time=0.634 ms
64 bytes from 10.0.10.100: icmp_req=3 ttl=64 time=0.743 ms
64 bytes from 10.0.10.100: icmp_req=4 ttl=64 time=0.890 ms
64 bytes from 10.0.10.100: icmp_req=5 ttl=64 time=0.776 ms
64 bytes from 10.0.10.100: icmp_req=6 ttl=64 time=0.517 ms
64 bytes from 10.0.10.100: icmp_req=7 ttl=64 time=0.563 ms
64 bytes from 10.0.10.100: icmp_req=8 ttl=64 time=0.727 ms
^C
```

```
[127 root@Pccom ~]$ ping serveurg.grenoble.local
PING serveurg.grenoble.local (10.0.10.2) 56(84) bytes of data.
64 bytes from 10.0.10.2: icmp_req=1 ttl=64 time=0.539 ms
64 bytes from 10.0.10.2: icmp_req=2 ttl=64 time=0.908 ms
64 bytes from 10.0.10.2: icmp_req=3 ttl=64 time=0.731 ms
^C
--- serveurg.grenoble.local ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2018ms
rtt min/avg/max/mdev = 0.539/0.726/0.908/0.150 ms
[0 root@Pccom ~]$
```

```
[127 root@serveurG ~]$ ping serveurhttps.grenoble.local
PING serveurhttps.grenoble.local (10.0.10.4) 56(84) bytes of data.
64 bytes from 10.0.10.4: icmp_req=1 ttl=64 time=0.507 ms
64 bytes from 10.0.10.4: icmp_req=2 ttl=64 time=0.721 ms
64 bytes from 10.0.10.4: icmp_req=3 ttl=64 time=0.665 ms
64 bytes from 10.0.10.4: icmp_req=4 ttl=64 time=0.733 ms
^C
--- serveurhttps.grenoble.local ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.507/0.656/0.733/0.093 ms
[0 root@serveurG ~]$
```

Règle de filtrage du routeur de Lyon :

```
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      *       10.0.20.0/24         10.0.20.3           udp dpt:53
    0     0 ACCEPT     udp  --  *      *       10.0.30.0/24         10.0.20.3           udp dpt:53
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            10.0.30.4           tcp dpt:443
  189 15876 DROP       all  --  *      *       10.0.20.0/24         10.0.10.0/24
    0     0 DROP       all  --  *      *       10.0.30.0/24         10.0.10.0/24
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
[0 root@R_lyon ~]$
```

Pour sécuriser les accès web, nous avons configuré des règles de filtrage. Les requêtes DNS (port 53) sont autorisées pour les VLANs 20 et 30. L'accès HTTPS (port 443) vers le serveur de Lyon est également permis. Les VLANs de Grenoble peuvent accéder librement à Lyon, sauf le VLAN 30 qui est bloqué. Enfin, une règle générale permet au reste du trafic de passer.

HTTP de Lyon (qui fonctionne comme celui de grenoble vue dans le rapport) :

```
[7 root@Pcdir ~]$ curl -k https://serveurhttps.lyon.local
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
[0 root@Pcdir ~]$
```

```
</body></html>
[0 root@Pccom ~]$ nslookup google.com
Server:         10.0.10.3
Address:        10.0.10.3#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.179.110

[0 root@Pccom ~]$
```

Le test réalisé avec la commande `nslookup google.com` depuis la machine `pccom` a fonctionné, ce qui prouve que le serveur DNS local de Grenoble (`10.0.10.3`) est bien configuré. Il est capable de résoudre des noms de domaine externes en contactant un serveur DNS public (comme `8.8.8.8`) grâce au système

de "forwarders". Cela montre que le routage, le NAT et les règles de pare-feu sont bien en place, et que les machines de Grenoble peuvent accéder à Internet pour les requêtes DNS.

Fichier de configuration de R_Lyon;

```
# Interfaces
ip addr add 10.0.30.254/24 dev eth0     # vers VLAN 30 (SW_lyon)
ip link set eth0 up

ip addr add 192.168.2.1/24 dev eth1     # vers Internet
ip link set eth1 up

ip addr add 10.0.20.254/24 dev eth2     # vers VLAN 20 (SW_lyon)
ip link set eth2 up
```

Par la suite la fonction Routage IP a été activé :

```
# Routage IP
echo 1 > /proc/sys/net/ipv4/ip_forward
```

ACL & filtrage :

Fichier de config de R_Lyon:

```
# Interfaces
ip addr add 10.0.30.254/24 dev eth0     # vers VLAN 30 (SW_lyon)
ip link set eth0 up

ip addr add 192.168.2.1/24 dev eth1     # vers Internet
ip link set eth1 up

ip addr add 10.0.20.254/24 dev eth2     # vers VLAN 20 (SW_lyon)
ip link set eth2 up

# Routes
ip route add 10.0.10.0/24 via 192.168.2.2 dev eth1
ip route add 192.168.1.0/24 via 192.168.2.2 dev eth1
ip route add default via 192.168.2.2 dev eth1

# Routage IP
echo 1 > /proc/sys/net/ipv4/ip_forward

#Accés Internet

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

# ACL
iptables -A FORWARD -s 10.0.20.0/24 -d 10.0.10.0/24 -j DROP


# Autorisation du reste
iptables -A FORWARD -j ACCEPT
```