

Rapport Systèmes et Réseaux II

Perion Maxence, Pinon Alexandre

Sommaire

1	Introduction	3
2	Installation d'une distribution GNU/Linux sans interface graphique	3
2.1	Installation via le réseau	3
2.2	Partitions du disque	3
2.3	Synthèse des commandes du gestionnaire de paquets	4
2.3.1	Mise à jour du système	4
2.3.2	Les paquets	5
3	Paramétrage du réseau	6
3.1	Répartition des adresse IP (réseau d'Interconnexion entre agences et réseau privé de l'agence)	6
3.2	Les interfaces réseaux du routeur	7
3.3	Les interfaces réseaux du client	9
3.4	Communication basique sans table de routage	10
3.5	Mise en place de la table de routage	10
3.6	Règles iptables pour laisser l'accès a internet a une machine	12
4	Service DHCP	12
4.1	Mise en place du DHCP	12
4.2	Système de logs pour le DHCP	14
5	Sauvegarde automatique	16
5.1	Rsync	16
5.2	Cron	17
6	Interrogation d'un serveur DNS	18
6.1	Les commandes host, dig et nslookup	18
6.2	Fichier de renseignement du serveur DNS	22
6.3	Rôle du fichier /etc/hosts	22
7	Installation d'un DNS	23
7.1	Mise en place du DNS	23
7.1.1	Théorie	23
7.1.2	Pratique	24
7.2	Tests réalisés afin de valider le DNS	28

8	Installation d'un serveur LAMP	29
8.1	Installation et mise du LAMP avec MariaDB	29
8.2	Installation et mise d'un autre LAMP avec PostgreSQL	34
8.3	Installation et mise en place d'un PDO	36
9	Script de routage et matrice de filtrage	37
9.1	Scripts de routage au démarrage	38
9.2	Sauvegarde des bases MySQL et PostgreSQL	39
9.3	Les sudoers	39
10	Mise en place d'un domaine Samba/LDAP	39
10.1	Définition du DIT	39
10.2	Mise en place de LDAP	39
10.3	Mise en place de Samba	40
10.4	Sauvegarde automatique de l'annuaire	40
11	Conclusion	40
12	Annexe	40

1 Introduction

L'objectif de ces Travaux Pratiques a été de réaliser un serveur complet pour une agence fictive. Pour cela, nous étions en possession de deux machines: un serveur/routeur ainsi qu'un client, une machine simulant la connexion d'un appareil au réseau de l'agence. Nous sommes partis de zéro et dans un premier temps nous avons installé Debian sur notre serveur, une distribution GNU/Linux, sans interface graphique via le réseau IEM pour avoir un outil de travail. Nous avons premièrement défini et paramétré notre adressage et routage, c'est à dire la façon d'attribuer les adresses IP et comment communiquer sur les différents réseaux. Afin de s'affranchir des adresses IP et pour s'approcher d'un cadre plus réaliste, nous avons deuxièmement mis en place sur le serveur un service de DNS, permettant d'utiliser des noms de domaines agissant comme des alias pour des adresses IP. Troisièmement, nous avons mis en place ce qui pourrait servir pour héberger un site web pour notre agence grâce à une suite d'outils appelée un LAMP. Pour finir, nous avons mis en place un service d'authentification centralisé LDAP ainsi qu'un service de partage de fichiers Samba.

2 Installation d'une distribution GNU/Linux sans interface graphique

2.1 Installation via le réseau

Pour commencer à travailler, la première étape a été d'installer un système d'exploitation: une distribution GNU/Linux sans interface graphique, à savoir Debian dans sa version "Bullseye" (Debian 11). Sans celui-ci, il nous serait impossible de réaliser la moindre tâche avec notre serveur. Le fait qu'il s'agisse d'une distribution Linux va nous permettre de pouvoir manipuler les différents services autant que nous le souhaitons, avec un système de paquets très pratiques. Afin d'installer cette distribution, les administrateurs Systèmes et Réseaux nous ont mis à disposition un boot via le réseau. En temps normal pour installer un système d'exploitation, quel qu'il soit, les utilisateurs utilisent des clés USB ou un disque dur qui permette de démarrer sur celui-ci et d'installer le système d'exploitation à partir de là. Le principe est le même dans notre cas sauf que cette installation se fait par le réseau IEM qui est le réseau de l'université de Bourgogne. Pour lancer l'installation nous devons brancher notre serveur sur le réseau et le démarrer depuis le bios en sélectionnant le bouton "PXE IEM". Une fois l'installation démarrée nous devons suivre les étapes d'installation, telles que définir le nom et mot de passe du root, choisir/créer des partitions du disque dur, donner un nom à la machine, ...

2.2 Partitions du disque

Lors de l'installation du système d'exploitation il est nécessaire de réaliser des partitions du disque dur pour scinder son espace de mémoire afin d'être utilisé pour différentes choses en simultané et sans risque de collisions ou de chevauchements. Notre disque après l'installation est désormais composé de 3

partitions et il est possible de les afficher ainsi que des détails supplémentaires avec la commande:

```
$ sudo fdisk -l
```

```
root@routerGroupeA2PP:/etc/network# root@routerGroupeA2PP:/etc/network# sudo fdisk -l
Disque /dev/sda : 238,47 GiB, 256060514304 octets, 500118192 secteurs
Modèle de disque : SAMSUNG SSD SM84
Unités : secteur de 1 x 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x82be967b

Périphérique Amorçage Début Fin Secteurs Taille Id Type
/dev/sda1 * 2048 498116607 498114560 237,5G 83 Linux
/dev/sda2 498118654 500117503 1998850 976M 5 Étendue
/dev/sda5 498118656 500117503 1998848 976M 82 partition d'échange Linux / Solaris
root@routerGroupeA2PP:/etc/network#
```

Figure 1: Les 3 partitions d'un disque

Sur la figure 1, nous pouvons voir distinctement les différentes partitions, dont la partition générale sda1 de 237.5 Go qui nous permet de stocker différents fichiers tels que des paquets pour l'utilisation de Debian ou encore différents fichiers de configuration par exemple. De plus, nous pouvons voir la partition qui permet le mécanisme de swap. Il s'agit d'un mécanisme du système d'exploitation qui va se servir du disque dur pour stocker des informations normalement stockées dans la RAM, lorsque celle-ci est saturée. Il y a également un partition étendue qui est un conteneur de partitions logiques: sda2. Maintenant que Debian est installé sur notre routeur, il est prêt à être configuré et utilisé pour différentes tâches.

2.3 Synthèse des commandes du gestionnaire de paquets

Sous Debian il existe de multiple gestionnaires de paquets, tels que aptitude, apt et dpkg. Dans notre cas, nous avons utilisé "apt-get". Il est important de comprendre que les outils de gestion des paquets Debian de plus haut niveau comme aptitude, apt-get ou synaptic reposent sur apt qui, lui-même, utilise dpkg pour la gestion des paquets sur le système.

2.3.1 Mise à jour du système

Il est possible de mettre à jour le système avec le gestionnaire de paquets "apt-get", en étant connecté en tant que root ou en préfixant les commandes avec sudo. Auparavant, on utilise la commande "update" qui permet d'actualiser la liste des paquets disponibles pour le système, à partir du dépôt de paquets. Le dépôt de paquet est un service sur un serveur distant, contenant les fichiers des paquets à mettre à jour ou à installer. On lance la commande avant d'installer un nouveau paquet ou avant d'installer les mises à jour du système. Elle ne modifie pas le système, elle demande simplement s'il existe de nouveaux paquets ou des nouvelles versions de paquets. On utilise cette commande de cette façon en tant que root:

```
$ apt-get update
```

Ou pour un utilisateur qui n'a pas tous les droits:

```
$ sudo apt-get update
```

Cette étape n'est pas nécessaire mais c'est un bon réflexe à avoir avant de mettre à jour tous les paquets.

S'il y a des paquets à mettre à jour, alors on peut utiliser la commande "upgrade" pour les installer à partir des fichiers stockés sur les sources énumérées. De nouveaux paquets seront installés si cela est nécessaire, mais les paquets installés ne seront jamais supprimés.

```
$ apt-get upgrade
```

ou

```
$ sudo apt-get upgrade
```

Afin de mettre à jour l'ensemble du système, il est possible d'utiliser ceci:

```
$ apt-get dist-upgrade
```

Il est recommandé d'installer les dernières versions de paquets disponibles pour le système utilisé que ce soit Linux, Windows ou autres. Cela permet de corriger des bugs et d'installer des correctifs de sécurité.

2.3.2 Les paquets

Afin d'administrer correctement un système GNU/Linux Debian, voici une liste non exhaustive des commandes de gestion de paquets de Debian à connaître.

- ```
$ apt-cache search <expression rationnelle>
```

Pour rechercher un paquet il est possible d'utiliser la commande "search", cette dernière recherche un paquet à partir d'une expression rationnelle. La recherche porte sur le nom et la description du paquet.

- ```
$ apt-cache show <paquet>
```

Pour afficher des informations détaillées concernant un paquet.

- ```
$ apt-cache policy <paquet>
```

Pour afficher les versions disponibles d'un paquet.

- ```
$ apt-get install <paquet>
```

L'installation d'un paquet disponible sur les serveurs, qui sera couramment utilisé durant nos installations.

- ```
$ apt-get remove <paquet>
```

Si nous avons besoin de désinstaller un paquet, on peut utiliser ceci. De plus, la désinstallation d'un paquet avec cette commande ne supprime pas les fichiers de configuration éventuel, ce qui permet de ne pas perdre sa configuration si on est amené à le réinstaller plus tard.

- `$ apt-get purge <paquet>`

A contrario ici le paquet est désinstallé avec tous les fichiers de configuration.

- `$ apt-get autoremove`

Supprimer les paquets installés automatiquement lorsqu'ils ne sont plus nécessaires.

- `$ apt-get clean`

Nettoyer complètement le dépôt local des fichiers de paquets récupérés.

- `$ apt-get autoclean`

Nettoyer le dépôt local des fichiers des paquets périmés.

Et il y a évidemment les différentes mises à jour vues précédemment.

- `$ apt-get update`  
`$ apt-get upgrade`  
`$ apt-get dist-upgrade`

Avec toutes ces commandes, nous sommes fin prêt à mettre en place tout un tas de service divers et variés tel qu'un DNS ou encore un serveur LAMP par exemple.

### 3 Paramétrage du réseau

Pour pouvoir commencer à communiquer entre différentes machines, nous avons ensuite défini la répartition des adresses IP pour chaque groupe (chaque agence) et paramétré en conséquences les différentes interfaces réseaux sur le routeur puis le client. Pour cela, nous nous sommes connectés au réseau IEM via la carte réseau intégré à la carte mère sur routeur, la première carte réseau externe a été branchée au switch afin de permettre la communication avec les autres groupes, donc le réseau d'interconnexion et dernièrement la deuxième carte réseau externe est utilisée pour le réseau privé mais puisque nous n'avons qu'un client, nous l'avons directement connecté. Toutes les connexions ont été effectuées via des câbles Ethernet standard. Ici le but de la manipulation est de mettre en place un réseau d'entreprises et d'une infrastructure de services.

#### 3.1 Répartition des adresse IP (réseau d'Interconnexion entre agences et réseau privé de l'agence)

Chaque binôme du groupe de TP s'est vu attribué une adresse IP de classe C et un masque qui sera utilisé pour le réseau d'interconnexion. Puisqu'il y a 5 binômes dans notre groupe de TP, nous avons pris une adresse dans la plage allant de 192.168.1.1 à 192.168.1.5. Pour la même raison, nous avons défini le masque du réseau d'Interconnexion à 255.255.255.248, résultat que nous avons

obtenu en ajoutant 3 bits à 1 le plus à gauche possible au masque standard de classe C (255.255.255.0). Ces 3 bits supplémentaires nous permettent de coder  $2^3=8$  sous réseaux, ce qui suffit pour nos 5 groupes.

| Groupe | Réseau IEM    | Réseau Interconnexion | Réseau privé  |
|--------|---------------|-----------------------|---------------|
| 1      | 172.31.20.111 | 192.168.1.1           | 10.1.1.0      |
|        | 255.255.255.0 | 255.255.255.248       | 255.255.255.0 |
| 2      | 172.31.20.112 | 192.168.1.2           | 10.1.2.0      |
|        | 255.255.255.0 | 255.255.255.248       | 255.255.255.0 |
| 3      | 172.31.20.113 | 192.168.1.3           | 10.1.3.0      |
|        | 255.255.255.0 | 255.255.255.248       | 255.255.255.0 |
| 4      | 172.31.20.114 | 192.168.1.4           | 10.1.4.0      |
|        | 255.255.255.0 | 255.255.255.248       | 255.255.255.0 |
| 5      | 172.31.30.115 | 192.168.1.5           | 10.1.5.0      |
|        | 255.255.255.0 | 255.255.255.248       | 255.255.255.0 |

Figure 2: Tableau des adresses IP de chaque groupe

Le tableau de la figure 2 récapitule les adresses IP allouées pour chaque réseau et pour chaque groupe. Pour notre cas nous sommes dans le groupe 2 ce qui signifie que notre adresse sur le réseau d'interconnexion est 192.168.1.2 et notre adresse de réseau privé est 10.1.2.0.

Après avoir défini ces adresses et ces réseaux sur le papier, il est désormais temps de configurer les machines afin de les implémenter concrètement.

### 3.2 Les interfaces réseaux du routeur

Les adresses que nous venons de répartir entre chaque groupe et réseaux doivent être utilisées par le routeur. Nous allons donc maintenant expliquer comment nous avons paramétré les interfaces réseaux sur le routeur. Pour cela, nous avons travaillé avec le fichier `/etc/network/interfaces` qui régit le fonctionnement des interfaces réseaux de la machine au niveau du noyau du système d'exploitation. Une interface réseau correspond à une carte réseau (en général) et il est possible à partir de son de la configurer (adresse statique, adresse du réseau, masque, gateway, ...). La gateway d'une telle configuration est la machine qui doit être utilisé pour passer d'un réseau à un autre. En d'autres termes, il s'agit de la "porte d'accès" à utiliser. Par exemple si une machine du réseau privé veut communiquer avec une autre machine située sur le réseau IEM, elle devra utiliser le serveur pour router ses paquets. Nous avons modifié le fichier pour obtenir le résultat de la figure 3.

Il est possible d'afficher les interfaces réseaux disponibles sur la machine dans le shell avec la commande:

```
$ ip a
```

```

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

The loopback network interface
auto lo
iface lo inet loopback

The primary network interface
auto eno1
iface eno1 inet static
 address 172.31.20.112
 netmask 255.255.255.0
 network 172.31.20.1
 broadcast 172.31.20.255
 gateway 172.31.20.1
 dns-nameservers 172.31.21.35 193.50.50.6
pre-up /etc/firewall-rules.sh

#reseau prive
auto enp3s0
iface enp3s0 inet static
 address 10.1.2.1
 netmask 255.255.255.0
 network 10.1.2.0

#interconnexion
auto enp1s0
iface enp1s0 inet static
 address 192.168.1.2
 netmask 255.255.255.248
 network 192.168.1.0

```

Figure 3: Fichier de configuration des interfaces réseaux du routeur

Cette commande a pour intérêt de nous faire connaître rapidement les adresses MAC et/ou IP de chaque interface. Elle nous a permis de connaître le nom des interfaces à utiliser dans notre fichier de configuration mais elle permet également de savoir si une interface est active "UP" ou pas "DOWN". En effet, durant les phases de tests il était important de savoir quelles interfaces étaient allumées ou éteintes. Il est possible qu'après une mauvaise configuration ou un problème, une interface soit éteinte. Il existe des commandes utilitaires pour les manipuler, l'une allumant l'interface nommée "interface" et l'autre l'éteignant.

```

$ ifup interface
$ ifdown interface

```

Pour le fichier de configuration (figure 3 ci-dessus) , nous pouvons voir que nous avons configuré les 3 interfaces (donc les 3 ports Ethernet que possède notre routeur), ainsi que l'interface "lo" qui représente le loopback. Le loopback est la "boucle" locale: c'est une interface virtuelle qui permet de définir à une machine "elle même", notamment lors de l'utilisation de l'adresse "localhost". Pour chacune des interfaces, nous avons donc défini leur adresse (address), leur masque (netmask) et leur réseau (network). Pour l'interface "eno1" qui est relié au réseau IEM, nous avons plusieurs informations supplémentaire: l'adresse de



broadcast (adresse utilisée pour transmettre un message à toutes les autres machines d'un réseau), la passerelle (gateway), les dns, ... Nous pouvons voir que l'interface "eno1" est relié au réseau IEM car pour le réseau nous avons indiqué l'adresse 172.31.20.1 qui est l'adresse du routeur du réseau IEM fournie par les intervenants des travaux pratiques. Le raisonnement est le même pour les interfaces "enp3s0" et "enp1s0", pour lesquels nous avons indiqué respectivement pour le réseau les adresses 10.1.2.0 et 192.158.1.0. Il s'agit donc des réseaux d'interconnexion et privé. Les entrées "dns-nameservers" et "pre-up" seront expliquées plus en détail ultérieurement. Une fois le fichier de configuration des interfaces réseaux édité à notre souhait, ses informations ne sont pas prises en compte immédiatement: il est nécessaire de redémarrer le service réseau associé, en tant que root ou alors avec sudo:

```
$ service networking restart
ou
$ /etc/init.d/networking restart
ou
$ systemctl restart networking
```

Ces 3 commandes permettent de façon similaire de redémarrer le service réseau.

Cependant, ces configurations sur le routeur ne sont pas les seules nécessaires pour que tous les réseaux fonctionnent: il est également nécessaire de configurer la machine client qui est connectée au réseau privé, notamment pour qu'elle connaisse comment communiquer avec une machine du réseau d'interconnexion ou du réseau IEM.

### 3.3 Les interfaces réseaux du client

Nous allons maintenant expliquer comment nous avons paramétré les interfaces réseaux sur le client. De la même manière que pour le routeur, le fichier de configuration se trouve dans /etc/network/interfaces et sa forme est identique à celui du routeur. Cependant, cette fois il n'est nécessaire de configurer qu'une seule interface, la seule qui est utilisée et qui est connectée au routeur par le réseau privé. Les informations que nous retrouvons dans ce fichier sont les suivantes. On peut y retrouver l'adresse de la machine, le masque et le réseau, ainsi que la gateway.

```
auto enp11s0
iface enp11s0 inet static
 address 10.1.2.2
 netmask 255.255.255.0
 network 10.1.2.0
 gateway 10.1.2.1
```

On voit ici que l'IP renseigné pour le réseau de l'interface "enp11s0" est 10.1.2.1, qui est l'adresse de notre sous réseau. L'adresse IP de notre client est désormais fixée à 10.1.2.2. Comme pour le routeur, afin d'appliquer les changements il faut redémarrer le service avec l'une des trois commandes évoquées précédemment. Par la suite, toutes ces configurations vont permettre au client et au routeur de communiquer avec le reste des machines des autres groupes.

### 3.4 Communication basique sans table de routage

Nous allons maintenant aborder la communication entre différentes machines, car nous avons pour but de simuler un réseau en entreprise, et par conséquent il faut que nos machines puissent communiquer. Il est également nécessaire que le routeur soit en capacité de rediriger correctement les paquets des machines la où ils doivent être conduits. Par défaut, le client et le routeur peuvent communiquer car ils sont branchés physiquement entre eux. Il en est de même pour tous les routeurs qui sont branchés sur le réseau IEM car ils sont tous sur le même réseau. Afin de tester les communication entre différentes machines, que ce soit routeurs ou clients, nous pouvons essayer de ping la machine, à condition que le protocole soit activé sur la cible, avec cette commande:

```
$ ping ip_machine
```

Pour montrer le bon fonctionnement jusqu'ici, nous allons avec notre routeur, essayer de ping le routeur d'un autre groupe via le réseau IEM:

```
root@routerGroupeA2PP:~# ping 172.31.20.111
PING 172.31.20.111 (172.31.20.111) 56(84) bytes of data:
64 bytes from 172.31.20.111: icmp_seq=1 ttl=64 time=1.94 ms
64 bytes from 172.31.20.111: icmp_seq=2 ttl=64 time=0.689 ms
64 bytes from 172.31.20.111: icmp_seq=3 ttl=64 time=0.721 ms
64 bytes from 172.31.20.111: icmp_seq=4 ttl=64 time=0.686 ms
64 bytes from 172.31.20.111: icmp_seq=5 ttl=64 time=0.734 ms
64 bytes from 172.31.20.111: icmp_seq=6 ttl=64 time=0.703 ms
^C
--- 172.31.20.111 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5095ms
rtt min/avg/max/mdev = 0.686/0.912/1.941/0.460 ms
```

Figure 4: Ping de notre routeur à un autre

Nous pouvons voir sur la figure 4 que les paquets sont bien envoyés et reçus par notre correspondant et tout s'est bien déroulé. Cependant si notre routeur ou notre client veut ping ou plus globalement communiquer avec un client d'un autre groupe qui est donc sur un autre réseau privé, nous allons devoir lui indiquer le chemin pour aller jusqu'à celui-ci, ce qui est réaliser avec un table de routage.

### 3.5 Mise en place de la table de routage

L'étape suivante est la création de routes afin de permettre la communication entre les différentes machines de la simulation: le client et le routeur, mais également entre chaque routeurs (les autres agences) et entre différents clients de différentes agences.

Dans la configuration actuelle, il nous est impossible de ping le client qui est connecté "derrière" un autre routeur que le notre, car notre routeur ne connaîtra pas de chemin (route) par lequel acheminer les paquets vers cette cible. Pour arriver à faire ceci, il nous faut mettre en place une table de routage. Il s'agit d'une table qui permet d'indiquer les routes possibles pour atteindre certains réseaux et avec une adresse de redirection dite de "default" qui indique vers qui envoyer le paquet si on ne connaît pas de chemin. Si aucun chemin n'est trouvé, l'erreur "Network Unreachable" sera renvoyé à l'émetteur du paquet.

Il est possible d'afficher les routes de notre routeur dans le shell avec la commande:

```
$ ip route
```

```

root@routerGroupeA2PP:~# ip route
default via 172.31.20.1 dev eno1 onlink
10.1.2.0/24 dev enp3s0 proto kernel scope link src 10.1.2.1 linkdown
172.31.20.0/24 dev eno1 proto kernel scope link src 172.31.20.112
192.168.1.0/29 dev enp1s0 proto kernel scope link src 192.168.1.2 linkdown

```

Figure 5: Route de notre routeur

Nous pouvons voir sur la figure 5 que la route par défaut à emprunter pour tous les paquets qui arrivent, si aucune des routes décrites ne convient, est la route par l'adresse 172.31.20.1. Il s'agit de l'adresse du routeur du réseau IEM, qui va à son tour regarder dans ses tables (ou non si le destinataire est directement accessible) une route correspondante ou utiliser sa route par défaut et ainsi de suite. Nous pouvons également voir que les paquets à destination du réseau 10.1.2.0/24 sont dirigés vers l'adresse 10.1.2.1. Il s'agit de l'adresse de notre serveur mais sur le réseau privé: le paquet est donc routé par une autre interface afin d'atteindre le sous réseau privé de destination. Le principe est le même pour le réseau d'interconnexion, et le réseau IEM qui sont également redirigés vers d'autres interfaces sur le serveur mais qui sont toutes différentes. Pour finir, nous pouvons voir que le trafic n'ayant pas trouvé de route sera envoyé vers l'adresse du routeur du réseau IEM (route par défaut).

Les routes statiques sont ajoutées dans le noyau de notre serveur par l'intermédiaire de la commande `ip route`, utilisée en tant que `root` ou avec `sudo` dans le cas contraire:

```
$ ip route add {network} via {IP}
```

Ces routes fonctionnent pour notre réseau privé, le réseau d'interconnexion ou le réseau IEM. Cependant, elles ne nous permettent toujours pas de communiquer avec un router connecté derrière le serveur d'un autre groupe. Pour cela, il va nous falloir ajouter encore une route et de la même façon il nous faudra une route pour chaque groupe avec lequel nous souhaitons communiquer. Le groupe avec qui nous effectuons les tests ont pour adresse (leur serveur) sur le réseau d'interconnexion l'adresse IP 192.168.1.1 et pour adresse de leur réseau privé 10.1.1.0. L'adresse IP de leur client dans leur réseau privé est 10.1.1.1. Pour pouvoir communiquer avec une machine sur leur réseau privé, nous devons donc appliquer sur notre routeur la commande :

```
$ ip route add 10.1.1.0/24 via 192.168.1.1
```

Cette commande permet de dire que tout ce qui est à destination de leur réseau privé, d'adresse IP 10.1.1.0, doit être redirigé vers la machine ayant l'adresse 192.168.1.1, c'est à dire leur routeur. Leur routeur sera alors atteint en empruntant le réseau d'interconnexion. Nous pouvons désormais, avec notre routeur, taper la commande qui permet de ping leur client:

```
$ ping 10.1.1.1
```

Le ping du client de leur réseau privé fonctionne désormais: nos paquets sont bien envoyés et réceptionnés.

### 3.6 Règles iptables pour laisser l'accès à internet à une machine

L'architecture de notre réseau commence à prendre forme, mais un client sur notre réseau privé ne peut toujours pas accéder à internet: pour le moment seul le routeur est connecté à internet à travers le réseau IEM. Pour réaliser cette manoeuvre appelée une translation d'adresse, nous allons utiliser les règles "iptables":

```
$ iptables -t nat -A POSTROUTING -s 10.1.2.0/24
-o eno1 -j MASQUERADE
```

Cette commande permet d'indiquer au routeur qu'il doit retransmettre les paquets reçus du réseau privé (source 10.1.2.0/24) sur le réseau IEM (interface de sortie "eno1"). L'option masquerade permet d'indiquer que le paquet sera retransmis par translation d'adresse, c'est à dire que le routeur va stocker dans son noyau les informations sur l'émetteur de ce paquet (adresse, port, ...) puis va l'émettre à nouveau en son nom. Lorsque le routeur recevra une réponse, il pourra consulter la table de translation dans son noyau pour trouver à qui il doit transférer le paquet.

Afin de vérifier si notre routeur était capable d'utiliser internet, nous avons installé "links2" qui a été auparavant installé via le gestionnaire de paquet "apt". Il s'agit d'une commande qui permet de consulter des pages web avec un affichage non graphique, que nous utilisons avec la commande:

```
$ links2 google.com
```

Et en effet, notre routeur était capable de surfer sur le web. Nous avons également appliqué cette procédure pour notre client afin de vérifier si il était connecté à internet.

L'inconvénient de cette nouvelle règle iptable est que n'importe quel client qui arrive à se connecter au réseau privé peut avoir accès à toutes les ressources et se connecter au réseau IEM ou d'interconnexion. Nous pouvons restreindre l'accès à notre réseau privé en n'attribuant une adresse IP qu'à certaines adresses MAC connues, via le service DHCP.

## 4 Service DHCP

Maintenant partons du principe que nous souhaitons connecter plusieurs clients sur notre réseau privé, il est alors important d'approfondir le paramétrage de notre réseau avec la mise en place d'un service DHCP afin de définir les adresses IP à attribuer pour les machines (clients) se connectant sur notre réseau privé.

### 4.1 Mise en place du DHCP

Avant toute chose, nous avons cherché et installé le bon paquet pour l'utilisation du DHCP:

```
$ apt-get install isc-dhcp-server
```

Une fois l'installation effectuée, nous nous sommes rendu dans le fichier `/etc/default/isc-dhcp-server` qui a été installé avec le paquet via la commande décrite précédemment. Dans ce fichier de configuration, il faut renseigner avec quelle interface réseau nous voulons travailler pour la suite de la mise du DHCP.

```
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="enp3s0"
INTERFACESv6=""
```

Figure 6: Interface du DHCP

Puisque nous voulons allouer automatiquement l'adresse IP de chaque client connecté à notre réseau privé nous avons utilisé l'interface "enp3s0", qui est celle utilisée pour notre réseau privé. Nous travaillons uniquement en IPv4 donc nous avons laissé vide les interfaces pour IPv6.

Ceci étant fait, nous pouvons désormais passer au paramétrage en détails du serveur DHCP, comme par exemple la plage d'adresse IP qui va être subnetée, ou encore fixer des adresses IP pour des postes ayant des adresses MAC définies. Cette configuration du serveur DHCP se fait sur le fichier `"/etc/dhcp/dhcpd.conf"`.

```
#logs separates
deny unknown-clients;
log-facility local7;

subnet 10.1.2.0 netmask 255.255.255.0 {
 option routers 10.1.2.1;
 option domain-name "agence.atlantide";
 option domain-name-servers 192.168.1.2;
}

host client1 {
 hardware ethernet f0:4d:a2:a0:e1:af;
 fixed-address 10.1.2.2;
}

host client2 {
 hardware ethernet f0:4d:a2:a0:e2:7d;
 fixed-address 10.1.2.2;
}
```

Figure 7: Configuration du serveur DHCP

Sur la figure 7 nous pouvons voir que le réseau subneté est notre réseau privé ayant comme adresse 10.1.2.0 et le masque 255.255.255.0. Nous attribuons une adresse fixe à plusieurs clients car durant nos tests il est arrivé que nous ayons des clients différents et sans cela, il aurait été impossible pour le client de se connecter au réseau privé. Afin d'attribuer une adresse fixe, il suffit d'indiquer l'adresse qu'on lui fixe avec comme entrée "fixed-address" ainsi que son adresse MAC

“hardware ethernet”. Nous avons également renseigné différentes informations concernant le DNS mais nous aborderons le sujet plus tard.

De la même façon que pour les interfaces réseaux, quand une modification est apportée il faut redémarrer le service correspondant afin de prendre en compte les changements effectués, avec la commande:

```
$ service isc-dhcp-server restart
```

Il est également possible de démarrer le service, si il est arrêté, avec:

```
$ service isc-dhcp-server start
```

Ou a contrario nous pouvons le stopper si besoin avec:

```
$ service isc-dhcp-server stop
```

Il est également possible d’afficher les erreurs en cas de soucis avec:

```
$ cat /var/log/syslog
```

Et finalement on peut afficher l’interface d’écoute du démon par l’intermédiaire de la commande:

```
$ ps ax | grep dhcpd
```

En cas de problème ou pour simplement monitorer ce qu’il se passe sur notre serveur, il peut être intéressant de garder une trace de ce qu’il se passe avec le service. Pour cela, nous avons besoin de mettre en place un système de logs dans notre DHCP.

## 4.2 Système de logs pour le DHCP

En réalité, notre service DHCP écrit déjà des logs et ce sont que nous n’ayons rien à configurer. Cependant, ces logs sont écrits avec ceux d’autres services dans les logs du système: syslog. Pour les obtenir dans un fichier spécialement dédié, nous devons modifier, ou plutôt ajouter une configuration à syslog.

```

#
First some standard log files. Log by facility.
#
auth,authpriv.* /var/log/auth.log
.;auth,authpriv.none -/var/log/syslog
#cron.* /var/log/cron.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
lpr.* -/var/log/lpr.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
local7.* -/var/log/dhcpd.log
local4.* -/var/log/ldap.log

```

Figure 8: Fichier de configuration de syslog

Pour cela, nous avons modifié le fichier de configuration `/etc/rsyslog.conf` pour y ajouter `"local7.* -/var/log/dhcpd.log"` comme nous le montre la figure 8. Le tiret avant le chemin du fichier n'est pas là par hasard: il sert à indiquer à syslog d'effectuer une écriture asynchrone: mettre en pause le processus à chaque fois que nous devons effectuer une écriture pour les logs peut se révéler très gourmand sur les performances de la machine et le tiret indique que ce n'est pas nécessaire.

Nous pouvons voir sur la figure 7 que lors de la configuration initiale du serveur DHCP, nous avons ajouté `"log-facility local7;"` afin de définir les logs sur l'entrée 7, qui est par défaut pour les logs du réseau. C'est grâce à cela que nous avons pu le mettre en correspondance dans syslog, qui lui définit le fichier dans lequel écrire.

Pour finir cette configuration, nous avons créé le fichier dans lequel nous souhaitons écrire nos logs, en tant que `"/var/log/dhcpd.log"`. Nous lui avons donné le propriétaire ainsi que les droits nécessaires avec les commandes :

```

$ sudo touch /var/log/dhcpd.log
$ sudo chown syslog:adm /var/log/dhcpd.log
$ sudo chmod 0640 /var/log/dhcpd.log

```

Il a finalement été nécessaire de vérifier que tout fonctionnait bien en regardant si des informations étaient effectivement écrites dans ce fichier. La figure 9 en montre un petit extrait. On peut d'ailleurs y voir le client tenter de se connecter (DHCPDISCOVER) et que l'adresse 10.1.2.2 lui est proposée (DHCPOFFER).

```

GNU nano 5.4 /var/log/dhcp.log
Feb 14 17:24:07 routerGroupeA2PP dhcpd[593]: Internet Systems Consortium DHCP Server 4.4.1
Feb 14 17:24:07 routerGroupeA2PP dhcpd[593]: Copyright 2004-2018 Internet Systems Consortium.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[593]: All rights reserved.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[593]: For info, please visit https://www.isc.org/software/dhcp/
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Internet Systems Consortium DHCP Server 4.4.1
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Copyright 2004-2018 Internet Systems Consortium.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: All rights reserved.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: For info, please visit https://www.isc.org/software/dhcp/
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Wrote 0 deleted host decls to leases file.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Wrote 0 new dynamic host decls to leases file.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Wrote 0 leases to leases file.
Feb 14 17:24:07 routerGroupeA2PP dhcpd[600]: Server starting service.
Feb 14 17:25:25 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 50.0.0.2 from f0:4d:a2:a0:e1:af via enp3s0: ignored (not a
Feb 14 17:25:27 routerGroupeA2PP dhcpd[600]: DHCPDISCOVER from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:25:27 routerGroupeA2PP dhcpd[600]: DHCPOFFER on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:25:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 (10.1.2.1) from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:25:27 routerGroupeA2PP dhcpd[600]: DHCPACK on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:30:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:30:27 routerGroupeA2PP dhcpd[600]: DHCPACK on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:35:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:35:27 routerGroupeA2PP dhcpd[600]: DHCPACK on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:40:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:40:27 routerGroupeA2PP dhcpd[600]: DHCPACK on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:45:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 from f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:45:27 routerGroupeA2PP dhcpd[600]: DHCPACK on 10.1.2.2 to f0:4d:a2:a0:e1:af via enp3s0
Feb 14 17:50:27 routerGroupeA2PP dhcpd[600]: DHCPREQUEST for 10.1.2.2 from f0:4d:a2:a0:e1:af via enp3s0

```

Figure 9: Fichier de configuration de syslog

## 5 Sauvegarde automatique

### 5.1 Rsync

Les fichiers importants tels que les logs ou les données sont souvent sauvegardés afin de pouvoir les récupérer en cas de problèmes. Pour mettre en place une telle sauvegarde, nous avons utilisé un paquet nommé “rsync” qui permet la synchronisation de fichiers. De plus, il est utilisé pour mettre en place des systèmes de sauvegardes distantes ou de points de restauration du système, ce qui est parfaitement ce que nous cherchons pour notre serveur. La commande de base pour utiliser rsync se fait par :

```
$ rsync source / destination /
```

Cette commande va sauvegarder tous les fichiers d’un répertoire source dans un autre répertoire destination. Cependant, dans notre cas nous allons pousser un peu plus loin et utiliser SSH afin de synchroniser nos fichiers à distance sur une autre machine. Le protocole SSH permet d’établir une communication chiffrée entre une machine locale (le client) et une machine distante (le serveur). Pour utiliser SSH, il est nécessaire de l’installer via le gestionnaire de paquets, et le paramétrer via les commandes :

```

$ sudo apt install openssh-server
$ ssh-keygen
$ ssh-copy-id -i ~/.ssh/id_rsa.pub etudiant@10.1.2.2

```

Comme expliqué précédemment, une fois l’installation de SSH terminée, il nous faut générer une paire de clé public et privée d’authentification (ssh-keygen) qui vont être enregistrés dans les fichiers “/.ssh/id\_rsa” pour la clé privée et “/.ssh/id\_rsa.pub” pour la clé public. Une fois les clés générées avec succès, nous allons les utiliser pour se connecter en SSH à notre client. La commande



ssh-copy-id facilite la connexion par clé SSH, ce qui supprime le besoin d'un mot de passe pour chaque connexion. Une fois la connexion au client correctement configuré, on peut l'utiliser avec rsync afin de sauvegarder les fichiers que nous voulons sur le client. Voici la manipulation de base à effectuer:

```
$ rsync -avz -e ssh chemin/source/
user@ip:"/chemin_de_destination/"
```

Et dans notre cas nous utilisons précisément:

```
$ rsync -avz -e ssh /etc
etudiant@10.1.2.2:/home/etudiant/Bureau/backup/
```

Le serveur est capable dorénavant de sauvegarder l'ensemble de son répertoire "/etc" dans le répertoire "/home/etudiant/Bureau/backup/" de notre client, simplement via l'utilisation de rsync par SSH. Mais il est encore plus intéressant de se demander comment automatiser totalement cette sauvegarde.

## 5.2 Cron

Dans le monde de l'informatique, l'automatisation de tâches est très souvent recherchée pour gagner du temps et ne plus s'occuper des tâches automatisées. C'est pour cette raison que nous allons utiliser en plus de rsync et SSH un programme nommé "cron". Cron est un programme qui permet d'exécuter automatiquement des scripts, des commandes ou des services à une date et une heure spécifiée précise, ou selon un cycle défini à l'avance. Cron est parfois appelé "gestionnaire de tâches planifiées" ou "planificateur de tâches". Chaque utilisateur à un fichier crontab, lui permettant d'indiquer les actions à exécuter et leur fréquence. Dans notre cas nous allons utiliser le fichier crontab de l'utilisateur "root". La commande :

```
$ crontab -e
```

est utilisée afin d'éditer les actions grâce au fichier crontab, qui s'ouvre après avoir défini un éditeur de texte à utiliser la première fois qu'on lance cette commande. Dans le fichier crontab, nous allons renseigner cette ligne qui reprend ce qui a été expliqué précédemment avec rsync et SSH :

```
0 15 * * * rsync -avz -e ssh /etc
etudiant@10.1.2.2:/home/etudiant/Bureau/backup/
```

Mais nous y ajoutons cette fois "0 15 \* \* \*", qui indique que nous allons appliquer le principe de rsync et SSH tous les jours de tous les mois à 15h et 0 minute. En effet, le premier paramètre indique les minutes comprises entre 1 et 60, le second indique les heures de 1 à 24, le troisième est pour les jours dans le mois donc de 1 à 31. Et les deux derniers sont pour les mois et jours de la semaines compris entre 1 à 12 pour les mois et 1 (lundi) et 7 (dimanche) pour les jours.

Pour examiner les tâches planifiées de l'utilisateur courant (le contenu du fichier crontab), on peut utiliser:

```
$ crontab -l
```

Ce qui nous affiche : La mise en place de notre réseau d'entreprise et d'infrastructure est terminée. Pour rappel, nous avons installé une distribution Linux, puis

```

Edit this file to introduce tasks to be run by cron.
#
Each task to run has to be defined through a single line
indicating with different fields when the task will be run
and what command to run for the task
#
To define the time you can provide concrete values for
minute (m), hour (h), day of month (dom), month (mon),
and day of week (dow) or use '*' in these fields (for 'any').
#
Notice that tasks will be started based on the cron's system
daemon's notion of time and timezones.
#
Output of the crontab jobs (including errors) is sent through
email to the user the crontab file belongs to (unless redirected).
#
For example, you can run a backup of all your user accounts
at 5 a.m every week with:
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
For more information see the manual pages of crontab(5) and cron(8)
#
m h dom mon dow command
0 15 * * * rsync -avz -e ssh /etc etudiant@10.1.2.2:/home/etudiant/Bureau/backup/

```

Figure 10: Contenu du fichier crontab.

paramétré les bases de notre réseau. Ensuite, nous avons mis en place un service DHCP et finalement une sauvegarde automatique. Nous allons désormais étendre notre réseau et mettre en place un serveur de nom de domaine (DNS). Mais avant de mettre en place le DNS, nous allons regarder comment il fonctionne.

## 6 Interrogation d'un serveur DNS

Avant toute installation, nous avons commencé par comprendre le fonctionnement du DNS. En effet, le DNS (pour Domain Name Serveur en anglais) est un service informatique distribué. Il est utilisé pour traduire les noms de domaine sur Internet avec leurs adresse IP ou autres enregistrements. Pour faire simple, le serveur DNS est un service qui permet d'associer à un site web (ou un ordinateur connecté ou un serveur) une adresse IP. Cette traduction peut se faire dans les 2 sens: trouver une IP à partir d'un nom de domaine mais également de retrouver le nom de domaine à partir de l'IP. Pour la suite de notre installation le DNS va être un composant important de notre réseau. Il est important de rappeler que tous les équipements connectés à un réseau IP, comme Internet, possèdent une adresse IP qui les identifie sur le réseau.

### 6.1 Les commandes host, dig et nslookup

Les serveurs DNS que nous allons interroger ont pour but de montrer les informations qu'il contiennent et ainsi comprendre les bases. Les serveurs DNS que l'on a interrogé durant nos tests, sont interrogés via les commandes : host, dig et nslookup.

La commande host renvoie l'adresse IP et l'adresse MAC associées au nom de domaine, on l'utilise comme ceci

```
$ host {nom}
```

Nous l'avons testé avec:

```
$ host www.debian.org
```

Qui nous a retourné ceci:

```
root@routerGroupeA2PP:~# host www.debian.org
www.debian.org has address 130.89.148.77
www.debian.org has IPv6 address 2001:67c:2564:a119::77
```

Figure 11: Exemple d'interrogation d'un DNS avec host.

Sur l'image 11, 2 informations importantes sont données: l'adresse IP du site sur la première ligne avec comme valeur: "130.89.148.77", et l'adresse MAC sur la seconde: "2001:67c:2564:a119::77". Voilà ce que nous ressort la commande host, voyons maintenant avec dig.

Dig est également un utilitaire en ligne de commande qui effectue une recherche DNS en interrogeant des serveurs de noms, et on l'utilise de cette façon:

```
$ dig {nom}
```

Nous l'avons testé avec:

```
$ dig www.debian.org
```

Qui nous a indiqué:

```
root@routerGroupeA2PP:~# dig www.debian.org

;; <<>> DiG 9.16.22-Debian <<>> www.debian.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8082
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; COOKIE: 02f2a76e6d552d7801000000624345ca4c418f8953a60a04 (good)
;; QUESTION SECTION:
;www.debian.org. IN A

;; ANSWER SECTION:
www.debian.org. 230 IN A 130.89.148.77

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Mar 29 19:45:46 CEST 2022
;; MSG SIZE rcvd: 87
```

Figure 12: Exemple d'interrogation d'un DNS avec dig.

Cette fois sur l'image 12, nous obtenons d'avantages d'informations sur le DNS interrogé. Les lignes commençant par ";;" sont des commentaires. La

première ligne nous indique la version de la commande dig (9.16.22). Dans l'en-tête nous pouvons voir si une réponse a été reçue (ANSWER: 1), afin de savoir si le DNS a répondu ou non. Ensuite la section "QUESTION", nous indique simplement la requête, qui dans ce cas est une requête pour l'enregistrement "A" de www.debian.org. IN signifie qu'il s'agit d'une recherche Internet. La section "ANSWER" nous indique que www.debian.org a l'adresse IP "130.89.148.77". Enfin, il y a quelques statistiques supplémentaires sur la requête, comme la date ou la taille du paquets qui contenait le message.

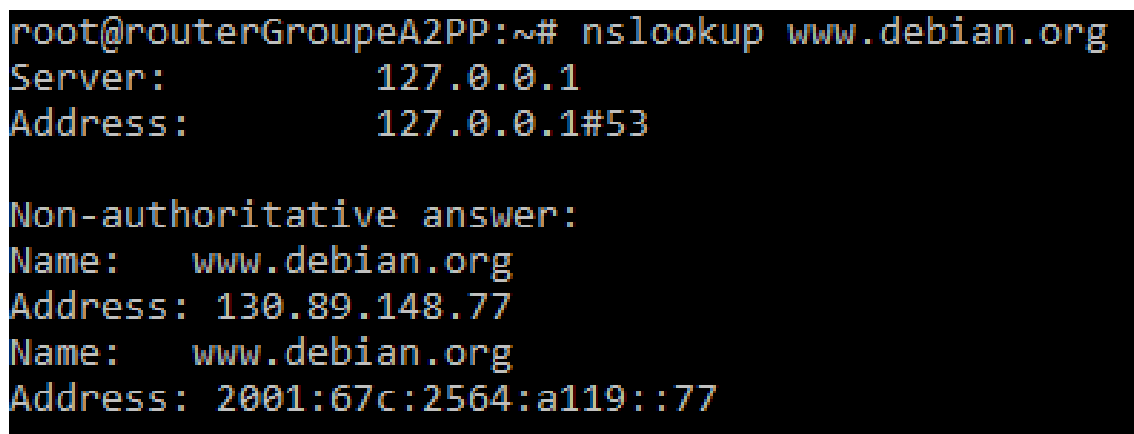
L'utilitaire nslookup (Name System Look Up) est, tout comme les deux autres, un outil qui permet d'interroger directement un serveur de noms et d'en obtenir les informations concernant un domaine. On peut l'utiliser comme ceci:

```
$ nslookup {nom}
```

Nous l'avons testé avec:

```
$ nslookup www.debian.org
```

Qui nous a donc retourné:



```
root@routerGroupeA2PP:~# nslookup www.debian.org
Server: 127.0.0.1
Address: 127.0.0.1#53

Non-authoritative answer:
Name: www.debian.org
Address: 130.89.148.77
Name: www.debian.org
Address: 2001:67c:2564:a119::77
```

Figure 13: Exemple d'interrogation d'un DNS avec nslookup.

Finalement on observe sur l'image 13, que nslookup est à mis chemin entre dig et host. En effet, il nous affiche le serveur qui a interrogé le DNS, et la réponse du DNS en question avec son nom, son adresse IP et son adresse MAC, comme dig mais en plus synthétique. En règle générale avec nslookup nous avons toutes les informations nécessaire.

Que se soit pour host, dig ou encore nslookup, l'indication "nom" indique l'adresse IP ou le nom d'hôte du serveur à interroger. Cependant, ils sont utilisables avec d'autres paramètres, comme nous allons le voir juste après.

L'option "ns" est pour "nameserver", cela permet d'indiquer où aller (à qui demander) pour trouver l'adresse IP d'un domaine. En effet, le mécanisme consistant à trouver l'adresse IP correspondant au nom d'un hôte est appelé "résolution de nom de domaine". L'application permettant de réaliser cette opération est appelée en anglais "resolver". Lorsqu'une application souhaite se

connecter à un hôte connu par son nom de domaine (par exemple "www.debian.org"), celle-ci va interroger un serveur de noms défini dans sa configuration réseau.

Voici les différentes requêtes que nous avons essayé:

```
$ host -t ns www.debian.org
$ dig www.debian.org ns
$ dig debian.org ns
$ dig org ns
$ dig . ns
```

Voyons en détails les réponse renvoyé par la ligne 1 et 2.

```
root@routerGroupeA2PP:~# host -t ns www.debian.org
www.debian.org name server geo1.debian.org.
www.debian.org name server geo2.debian.org.
www.debian.org name server geo3.debian.org.
```

Figure 14: Requête de nameserver pour un DNS avec host.

```
root@routerGroupeA2PP:~# dig www.debian.org ns

;; <<>> DiG 9.16.22-Debian <<>> www.debian.org ns
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 50442
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 9adef0501114c16301000000624618a0625a2fee2ad7e56f (good)
;; QUESTION SECTION:
;www.debian.org. IN NS

;; ANSWER SECTION:
www.debian.org. 657 IN NS geo1.debian.org.
www.debian.org. 657 IN NS geo2.debian.org.
www.debian.org. 657 IN NS geo3.debian.org.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Mar 31 23:09:52 CEST 2022
;; MSG SIZE rcvd: 128
```

Figure 15: Requête de nameserver pour un DNS avec dig.

Nous pouvons immédiatement constater que nous avons interrogé le même DNS afin de comparer les réponses. Comme expliqué précédemment, dig ren-

voient beaucoup plus de détails sur la requête que `host`. On observe principalement que la réponse des 2 commandes est identique, mais pour une requête qui aurait besoin de passer par beaucoup plus de nom de domaine, la commande `dig` serait plus intéressante. En effet, avec `dig` il est possible d'avoir le nombre de réponses obtenues; ici "ANSWER: 3" pour "geo1.debian.org", "geo2.debian.org" et "geo3.debian.org".

Pour conclure, une requête est ainsi envoyée au premier serveur de noms (appelé "serveur de nom primaire"). Si celui-ci possède l'enregistrement dans son cache, il l'envoie à l'application. Dans le cas contraire, il interroge un serveur racine. Le serveur de nom racine renvoie une liste de serveurs de noms faisant autorité sur le domaine. Le serveur de noms primaire faisant autorité sur le domaine va alors être interrogé et retourner l'enregistrement correspondant à l'hôte sur le domaine: dans le cas présent les serveurs "geo1.debian.org", "geo2.debian.org" et "geo3.debian.org".

L'utilitaire `dig` avec l'option "+trace" nous donne toutes les redirection, c'est-à-dire les routes par lesquels il est passé. Ce qui nous permet de savoir que 16 serveurs gèrent le ".". Et grâce au site "<https://root-servers.org/>" on peut savoir que 254 serveurs gèrent le "e".

Finalement, nous pouvons conclure que `dig` et `host` renvoie les informations nécessaires identiques, `host` reste tout de même limité en informations tandis que `dig` est davantage détaillé. Les 2 commandes peuvent servir, tout dépend du niveau de détails dont on a besoin.

## 6.2 Fichier de renseignement du serveur DNS

Sur notre serveur, le serveur DNS à utiliser doit être indiqué dans le fichier système: "/etc/network/interfaces". C'est le même fichier qui renseigne les interfaces réseau. Ce fichier a déjà été évoqué précédemment, mais pas dans sa totalité. En effet, il est temps d'expliquer la ligne suivante:

```
$ dns-nameserver 172.31.21.35 193.50.50.6
```

Cette ligne indique les 2 serveurs de noms de domaine que le système doit utiliser. En réalité le DNS avec l'adresse "172.31.21.35" est le serveur DNS primaire du réseau IEM, et le DNS avec l'adresse "193.50.50.5" est le serveur DNS secondaire du réseau IEM. Ces informations sont traitées par le "resolver" qui a été évoqué dans la partie précédente.

## 6.3 Rôle du fichier /etc/hosts

Ce fichier, existant depuis le début des années 80, a pour rôle d'associer une adresse IP à un nom d'hôte (un nom de domaine n'étant qu'une forme structurée de nom d'hôte). Il est donc présent dans le répertoire "/etc" et est consulté par le système à chaque fois qu'il accède à un nom d'hôte spécifique. Notons qu'il est consulté avant qu'une requête soit envoyée à un DNS. Aucune autre machine de notre réseau (même local) ne prendra en compte ce qui y est écrit dans celui-ci. Son utilisation est restreinte à de petits réseaux locaux ainsi qu'à quelques autres cas particuliers.

## 7 Installation d'un DNS

Pour la suite des explications, il est important de préciser que nous avons choisi comme nom de domaine: agence.atlantide.

### 7.1 Mise en place du DNS

#### 7.1.1 Théorie

Cette partie du rapport est consacré à la mise en œuvre d'un serveur DNS en utilisant ISC BIND (Berkeley Internet Name Domain). Un DNS est une base de données répartie dont la structure est similaire à une arborescence inversée. Le nœud racine est représenté par le ".". Une branche complète du type mail.yahoo.fr est appelée FQDN ( Fully Qualified Domain Name) pour nom totalement qualifié. La partie la plus à droite est appelé TLD ( Top Level Domain) et celui le plus à gauche représente l'hôte. Au milieu c'est le domaine. Un serveur de noms définit une zone avec des enregistrements appelés RR pour Ressources Records. Elles constituent les informations fondamentales du DNS. Mais il existe plusieurs types d'enregistrements, chacun possédant sa propre fonction:

- Enregistrement SOA

L'enregistrement SOA (Start Of Authority) retourne des informations officielles sur la zone DNS, incluant la référence du serveur DNS principal (primaire), l'e-mail de l'administrateur, un numéro de série ainsi que divers timers définissant la fréquence de renouvellement et la durée de validité pour certains éléments. Le service de gestion DNS ne propose qu'un seul enregistrement SOA.

- Enregistrement NS

Les enregistrements NS indiquent quels sont les serveurs qui vont gérer la zone du serveur DNS. Lorsque l'on crée un nom de domaine, pour le relier aux différents services pour lesquels nous allons l'utiliser, nous devons indiquer quels seront les serveurs DNS qui vont héberger les paramètres de notre zone DNS. Généralement, on utilise 2 ou 3 noms de serveurs DNS : le serveur primaire, mais également un ou deux serveurs secondaires qui peuvent prendre le relais en cas d'indisponibilité du serveur DNS primaire.

- Enregistrement A

Les enregistrements DNS de type A (également appelés enregistrements d'hôte) permettent de relier un nom de domaine ou un sous-domaine à l'adresse IP d'un serveur.

- Enregistrement PTR

Les enregistrements de type PTR sont quasiment le contraire des enregistrements A. Au lieu d'assigner une adresse IP à un nom de domaine, ces enregistrements font le contraire. Les enregistrements PTR permettent ainsi un DNS inversé.

- Enregistrement MX

MX est l'abréviation de Mail Exchange. Cet enregistrement DNS est différent des autres. L'enregistrement MX est utilisé pour diriger les emails envoyés aux adresses personnalisées associées à un nom de domaine.

- Enregistrement CNAME

Les enregistrements DNS de type CNAME, appelés aussi enregistrements de noms canoniques, ne résolvent que les domaines et les sous-domaines. Contrairement aux enregistrements A, ils ne peuvent pas être nus (c'est-à-dire qu'il doit y avoir `www.` devant eux pour que l'URL résolve correctement).

### 7.1.2 Pratique

Passons à la mise en pratique avec l'installation de notre serveur DNS. Dans un premier temps, il faut installer les paquets `"bind9"` et `"dnsutils"` toujours via la même commande et le même gestionnaire de paquets (`apt-get`).

BIND9 peut être utilisé de différentes façons, les plus fréquentes sont :

Un serveur cache, dans cette configuration, BIND9 va effectuer les requêtes DNS et se rappeler de la réponse pour la prochaine requête. Cette méthode peut être utile pour une connexion internet lente. En mettant les réponses DNS en cache, on diminue l'utilisation de la bande passante et (encore plus important) on réduit également le temps de latence.

Un serveur primaire qui est utilisé pour contenir les enregistrements DNS d'un nom de domaine enregistré. Un ensemble d'enregistrements DNS pour un nom de domaine est appelé une "zone". (Le nom de domaine peut être imaginaire si on est dans le cas d'un réseau local fermé). Il s'agit du cas d'utilisations que nous allons implémenter.

Et finalement un serveur secondaire, qui est un serveur utilisé en complément d'un serveur primaire, en servant de copie à la ou les zones configurées sur le serveur primaire. Les serveurs secondaires sont recommandés sur des gros réseaux. Ceux-ci assurent la disponibilité de la zone DNS, même si le serveur primaire est hors ligne.

La première étape consiste à créer 2 zones, une zone dite "normal" et une zone dite "inversée" dans le fichier de configuration `"/etc/bind/named.conf.local"`. Il est possible d'utiliser le fichier `"/etc/bind/named.conf.default-zones"` comme base de travail, que nous copions en notre propre configuration.



```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

// prime the server with knowledge of the root servers

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "2.1.10.in-addr.arpa" {
 type master;
 file "/etc/bind/db.agence.atlantide.inv";
};

zone "agence.atlantide" {
 type master;
 file "/etc/bind/db.agence.atlantide";
};
```

Figure 16: Contenu du fichier de configuration named.conf.local.

Sur l'image 16, les 2 zones sont identifiables par l'identificateur "zone". La première zone décrite est la zone "inverse". Elle a pour nom l'adresse ip "10.1.2." inversée accompagnée de ".in-addr.arpa". L'inversion de l'adresse IP est le standard pour décrire une zone "inverse". La seconde zone est la zone "normale" et que l'on a décrit par le nom de domaine, à savoir "agence.atlantide". Que ce soit pour la zone "normale" ou la zone "inverse" nous indiquons 2 informations: le "type" pour le type de zone et le fichier de configuration associé avec le mot clé "file".

Afin de mettre en place les 2 fichiers de configuration des 2 zones, nous sommes partis du fichier existant "/etc/bind/db.local". Celui-ci nous a fournis toutes les informations nécessaires et nous n'avions plus qu'à remplacer les lignes dont nous avons besoin. Par exemple le "localhost" a été changé en "agence.atlantide", l'adresse IP de loopback (le "127.0.0.1"), a été changé en "10.1.2.1". Nous indiquons ainsi que le nom de domaine agence.atlantide décrit l'adresse "10.1.2.1", donc notre serveur. Pour le fichier de la zone "normale" nous avons utilisé des enregistrements de type "A" afin de relier IP et nom de domaine. Nous avons donc un champ A qui définit que le sous domaine client.agence.atlantide décrit l'adresse "10.1.2.2", donc notre client. Voici ci-dessous le fichier "/etc/bind/db.agence.atlantide" de configuration de la zone "normale":

```

;
; BIND data file for local loopback interface
;
$TTL 604800
@ IN SOA agence.atlantide. root.agence.atlantide. (
 2 ; Serial
 604800 ; Refresh
 86400 ; Retry
 2419200 ; Expire
 604800) ; Negative Cache TTL
;
@ IN NS agence.atlantide.
@ IN A 10.1.2.1
client IN A 10.1.2.2
informatique IN A 10.1.2.1

```

Figure 17: Contenu du fichier de configuration du DNS normal.

Pour la zone inverse il faut utiliser des enregistrements de type “PTR” qui permettent la mise en place du DNS inversé. Et voici ici le fichier “/etc/bind/db.agence.atlantide.inv” de configuration de la zone “inverse”:

```

; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA agence.atlantide. root.agence.atlantide. (
 1 ; Serial
 604800 ; Refresh
 86400 ; Retry
 2419200 ; Expire
 604800) ; Negative Cache TTL
;
@ IN NS agence.atlantide.
1 IN PTR agence.atlantide.
2 IN PTR client.agence.atlantide.

```

Figure 18: Contenu du fichier de configuration du DNS inversé.

Sur les images 17 et 18, la première ligne débutant avec un “@” permet de définir les informations relatives à la zone (champ SOA). Les “@” suivants pourront remplacer des noms de machines ou le dernier octet “host-id” de nos adresses IP.

De plus, 2 autres fichiers ont été modifiés: le fichier “/etc/resolv.conf” et “/etc/bind/named.conf.options”. Pour le fichier “/etc/resolv.conf”, afin que les requêtes de résolution des noms passent par BIND. Ce fichier doit donc contenir:

```
domain iem
search iem
nameserver 127.0.0.1
```

Figure 19: Contenu du fichier de configuration de resolv.conf.

Dans le fichier “/etc/bind/named.conf.options”, nous avons ajouter des forwarders. Dans notre cas, ce sera notre fournisseur DNS (172.31.21.35) du réseau IEM et nous avons changer du paramètre “dnssec-validation” a “no”. Les forwarders sont les adresses d’autres serveurs DNS qui seront consultés si celui local ne connaît pas la réponse. En effet, notre propre serveur DNS ne connaît que notre nom de domaine et n’en connaît aucun autre. Le fichier “/etc/bind/named.conf.options” contient désormais ces informations:

```
options {
 directory "/var/cache/bind";

 // If there is a firewall between you and nameservers you want
 // to talk to, you may need to fix the firewall to allow multiple
 // ports to talk. See http://www.kb.cert.org/vuls/id/800113

 // If your ISP provided one or more IP addresses for stable
 // nameservers, you probably want to use them as forwarders.
 // Uncomment the following block, and insert the addresses replacing
 // the all-0's placeholder.
 recursion yes;

 forwarders {
 172.31.21.35;
 };

 //=====
 // If BIND logs error messages about the root key being expired,
 // you will need to update your keys. See https://www.isc.org/bind-keys
 //=====
 dnssec-validation no;

 listen-on-v6 { any; };
};
```

Figure 20: Contenu du fichier de configuration de named.conf.options.

Après avoir fait toutes ces modifications, il est nécessaire de redémarrer le service BIND, en lançant la commande:

```
$ systemctl restart bind9
```

Pour le DHCP (figure 7), nous avons pris en compte notre nom de domaine avec l’option “option domain-name “agence.atlantide””. Dans ce cas, il sert à faire référence aux ordinateurs du réseau par leurs noms sans ajouter le nom de domaine. Pour faire simple c’est un suffixe DNS. L’option “option domain-name-servers 192.168.1.2” indique un serveur DNS à l’adresse “192.168.1.2” (notre serveur sur le réseau d’interconnexion).

## 7.2 Tests réalisés afin de valider le DNS

L'installation suivie de la configuration global du DNS est effectué, mais avant de l'utiliser sur d'autres machines il est important de vérifier son fonctionnement.

Tout d'abord il faut vérifier la syntaxe des fichiers, en effet avant de chercher quelconque erreur, la syntaxe des fichiers modifiée est la première étape de vérification. Pour vérifier la syntaxe des fichiers de configuration on peut utiliser 2 commandes:

```
$ named-checkzone
$ named-checkconf
```

Les fichiers étant syntaxiquement corrects, le serveur DNS peut être démarré (en root) avec ceci:

```
$ systemctl start bind9
```

Au moment du démarrage si le serveur DNS renvoie une erreur et ne démarre donc pas, il est possible de voir les détails des erreurs dans le fichier "/var/log/syslog".

Et finalement voici une liste de tests effectués afin de tester le DNS: Depuis notre serveur:

```
$ nslookup ip-client
$ nslookup 10.1.2.1
```

Depuis notre serveur:

```
$ nslookup nom-client
$ nslookup client.agence.atlantide
```

Depuis un client connecté en filaire à notre routeur:

```
$ nslookup ip-client
$ nslookup 10.1.2.1
```

Depuis un client connecté en filaire à notre routeur :

```
$ nslookup nom-client
$ nslookup client.agence.atlantide
```

Vers le serveur d'une autre agence :

```
$ nslookup nom-serveur
$ nslookup agence.pekin
```

Vers le client d'une autre agence :

```
$ nslookup nom-client
$ nslookup client.agence.pekin
```

Le serveur DNS est fonctionnel, et donc prêt à être utilisé sur d'autres machines, et autres projets.

TODO Photo test(quand on aura accès au serveur) TODO

## 8 Installation d'un serveur LAMP

LAMP est un acronyme, en effet le "L" désigne Linux: le système d'exploitation, le "A" désigne Apache: le serveur Web, le "M" est pour MySQL ou MariaDB: le serveur de Base de données, et le "P" pour PHP: le langage de script.

Avec LAMP, on peut donc mettre en place un serveur Web, hébergeant un site web dynamique écrit en PHP, tout en allant chercher des données dans une base MySQL ou MariaDB, mais également PostgreSQL.

### 8.1 Installation et mise du LAMP avec MariaDB

Installation du serveur web apache et son module permettant de gérer PHP plus tard:

```
$ apt install apache2 libapache2-mod-php
```

Nous avons également utilisé la commande:

```
$ systemctl enable apache2
```

Qui permet de démarrer le serveur Apache au boot de notre routeur.

Après l'installation donc par défaut, la page web de notre serveur se trouve dans le fichier: `/var/www/html/index.html`. Et par conséquent à ce moment de la configuration nous pouvons visiter la page en allant sur le lien `http://localhost/index.html` sur notre serveur.

Pour la suite nous avons créé un compte pour un utilisateur qui développe des applications web. Cette utilisateur a été créé avec la commande:

```
$ adduser dev1
```

Qui est une commande intégrée nativement à Linux permettant de créer des utilisateurs. Ensuite nous avons créé le dossier où l'utilisateur va pouvoir ajouter ces fichiers. Le dossier se trouve dans `/srv/dev1`. Mais par défaut l'utilisateur "dev1" n'a les bons droits. C'est pour cela que nous avons également changé les droits et changé le propriétaire du dossier via les commandes:

```
$ chown utilisateur nom(s)_de_fichier
$ chown -R dev1 dev1
$ chmod -R 755 dev1
```

Le changement de droit avec `chmod 755` est équivalent au `chmod u=rwx,go=rx`.

Ensuite nous avons fait pointer un alias sur la page `/srv/dev1/www`, pour cela nous avons utilisé le mot clé "Alias" dans le fichier de configuration `/etc/apache2/apache2.conf`.

```

Include list of ports to listen on
Include ports.conf

Sets the default security model of the Apache2 HTTPD server. It does
not allow access to the root filesystem outside of /usr/share and /var/www.
The former is used by web applications packaged in Debian,
the latter may be used for local directories served by the web server. If
your system is serving content from a sub-directory in /srv you must allow
access here, or in any related virtual host.
<Directory />
 Options FollowSymLinks
 AllowOverride None
 Require all denied
</Directory>

<Directory /usr/share>
 AllowOverride None
 Require all granted
</Directory>

Alias "/dev1" "/srv/dev1/www"

<Directory /var/www/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
</Directory>

<Directory /srv/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
</Directory>

AccessFileName: The name of the file to look for in each directory
for additional configuration directives. See also the AllowOverride
directive.
#
AccessFileName .htaccess

#
The following lines prevent .htaccess and .htpasswd files from being
viewed by Web clients.
#
<FilesMatch "\.ht">
 Require all denied
</FilesMatch>

```

Figure 21: Contenu du fichier de configuration de apache.

La ligne qui nous intéresse est la suivante: "Alias "/dev1" "/srv/dev1/www"". Cette ligne permet de dire au navigateur sur laquelle nous allons afficher la page que nous n'avons pas besoin de taper "http://localhost/srv/dev1/www/index.html" mais simplement "http://localhost/dev1".

Pour tester notre alias nous avons écrit un petit "index.html" dans le dossier "/srv/dev1/www". Nous sommes donc passé par c'est différentes commandes:

```

$ touche /srv/dev1/www/index.html
$ nano /srv/dev1/www/index.html

```

Contenue du fichier "/srv/dev1/www/index.html":

```

<html>
 <head>
 <title> DEV 1 WEBSITE </title>
 </head>
 <body>
 <h1>Succes</h1>
 </body>
</html>

```

Cette petite page HTML nous permet simplement d'avoir un affichage en cas de réussite.

Un second utilisateur semblable au premier est créé. Tout comme pour le développeur web, nous avons créé un utilisateur "dev2", ajouté tous les dossiers et fichiers nécessaires et changé les droits. Ce qui nous donne ceci:

```
$ adduser dev2
$ mkdir /srv/dev2
$ mkdir /srv/dev2/www
$ touch /srv/dev2/www/index.html
$ nano /srv/dev2/www/index.html
$ chown -R dev2 dev2
$ chmod -R 755 dev2
```

Contenu du fichier "/srv/dev2/www/index.html":

```
<html>
 <head>
 <title> DEV 2 WEBSITE </title>
 </head>
 <body>
 <h1>Encore un succès</h1>
 </body>
</html>
```

Dans la même dynamique que la première page, elle est destinée à avoir un affichage de succès pour les phases de test. Mais nous avons également trouvé le besoin d'utiliser un VirtualHost pour cette utilisateur, en effet les serveurs virtuels, ou VirtualHost, permettent de faire cohabiter plusieurs serveurs web sur une même machine. Pour créer un VirtualHost il faut le renseigner dans le fichier "/etc/apache2/sites-available/dev2.conf". Voici donc la configuration du VirtualHost de dev2:

```
<VirtualHost *:80>
 # The ServerName directive sets the request scheme, hostname and port that
 # the server uses to identify itself. This is used when creating
 # redirection URLs. In the context of virtual hosts, the ServerName
 # specifies what hostname must appear in the request's Host: header to
 # match this virtual host. For the default virtual host (this file) this
 # value is not decisive as it is used as a last resort host regardless.
 # However, you must set it for any further virtual host explicitly.
 ServerName informatique.agence.atlantide
 #ServerAlias informatique.agence.atlantide

 ServerAdmin webmaster@localhost
 DocumentRoot /srv/dev2/www

 # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
 # error, crit, alert, emerg.
 # It is also possible to configure the loglevel for particular
 # modules, e.g.
 #LogLevel info ssl:warn

 ErrorLog ${APACHE_LOG_DIR}/error-informatique.agence.log
 CustomLog ${APACHE_LOG_DIR}/access-informatique.agence.log combined

 # For most configuration files from conf-available/, which are
 # enabled or disabled at a global level, it is possible to
 # include a line for only one particular virtual host. For example the
 # following line enables the CGI configuration for this host only
 # after it has been globally disabled with "a2disconf".
 #Include conf-available/serve-cgi-bin.conf
 <Directory /srv/dev2/www>
 # Options -Indexes +FollowSymLinks
 # AllowOverride All
 #
 </Directory>
</VirtualHost>
```

Figure 22: Contenu du fichier de configuration de dev2.

Dans ce fichier de configuration nous y retrouvons le chemin du dossier ou l'on va mettre nos pages web, ainsi que le nom du serveur qui est "informatique.agence.dijon".

Pour des utilisateurs occasionnels, nous avons mis en place un public.html afin de publier des pages personnelles. La première étape à été de créer le fichier "/var/www/html/public.html". Nous avons ensuite activé le module "userdir" d'apache, qui permet d'accéder à des répertoires spécifiques à l'utilisateur en utilisant la syntaxe http://example.com/user/. Voici la commande qui a permis d'activer ce module:

```
$ a2enmod userdir
```

Et pour la suite des manipulations, c'est le même principe que les 2 premiers utilisateurs.

```
$ mkdir ~/public_html
$ touch index.html
$ nano index.html
```

Contenu du fichier "/var/www/html/public.html/index.html":

```
<html>
 <head>
 <title> PUBLIC HTML </title>
 </head>
 <body>
 <h1>public html succes</h1>
 </body>
</html>
```

Le DNS configuré dans la section précédente va être utilisé, c'est pour cela que dans le fichier "/etc/bind/db.agence.atlantide" on a rajouté la ligne :

```
$ informatique IN A 10.1.2.1
```

Une fois toutes les installations terminées, avec les utilisateurs, les droits, les VirtualHosts, les alias et le DNS. Alors seulement on peut redémarrer le serveur apache avec la commande:

```
$ systemctl restart bind9
```

Le serveur configuré et fonctionnel, on peut brancher un client à notre routeur et utiliser les 2 pages suivantes sur un navigateur (firefox, chrome, ...): "http://agence.atlantide/dev1" pour visiter la page du premier utilisateur, "http://informatique.agence.atlantide/" pour le deuxième utilisateur, et finalement la page des utilisateurs occasionnels est disponible sous "http://agence.atlantide/dev1/". Pour naviguer entre les différents utilisateurs il faut utiliser cette commande:

```
$ su nom_Utilisateur
```

Et si un mot de passe à été renseigné à la création alors il sera demandé pour se connecter.

Pour la suite il est important de rappeler qu'un serveur LAMP est fait à partir de l'acronyme Linux Apache MariaDB PHP. Il est donc temps de passer à l'installation de MariaDB.



MariaDB est un système de gestion de base de données open source (SGBD). Pour installer MariaDb avec notre gestionnaire de paquet, il suffit d'utiliser:

```
$ apt install mariadb-server
```

Toujours en étant root ou alors en utilisant "sudo".

Nous allons utiliser l'utilisateur root qui possède tous les droits pour créer et gérer la base de données. Afin d'utiliser MariaDB c'est-à-dire d'utiliser l'invite MariaDB depuis votre terminal:

```
$ mysql -u root
```

Une fois dans l'invite de MariaDB on peut alors dans un premier temps créer la base de données, simplement par:

```
mysql> CREATE DATABASE dbtest;
```

Nous voilà en possession d'une base de données MariaDB, avec comme nom "dbtest".

Pour être certain qu'elle a été créée nous avons taper:

```
mysql> SHOW DATABASES;
```

Il est facile de voir notre base de données "dbtest", dans la liste des tables renvoyée par MariaDB. Une fois la base de données établie, il est alors possible de l'utiliser et d'insérer des tables et des données comme une base de données classique. Nous avons essayé ceci:

```
mysql> use dbtest;
mysql> CREATE TABLE test (nom TEXT, prenom TEXT);
mysql> INSERT INTO test (nom, prenom) VALUES ("NomTest", "PrenomTest");
mysql> SELECT * FROM test;
```

La première ligne indique à MariaDB que nous changeons de base de données, si cela est possible il nous indique "Database changed". Pour la suite la table de test est nommée "test" et contient 2 informations, un nom et un prénom. La dernière ligne nous renvoie les lignes de la table si il y en a, dans notre cas oui (ajout d'une ligne à la ligne précédente).

Afin que PHP puisse interagir avec la base de données nous avons créé un utilisateur propre à celui-ci. En effet MariaDB propose la création des utilisateurs au sein même de son environnement. Avec la simple commande:

```
mysql> CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
```

L'utilisateur est créé avec comme nom "user" et mot de passe "password", mais il ne possède pas les bons droits et c'est pour cela que nous lui avons donné tous les droits avec:

```
mysql> GRANT ALL PRIVILEGES ON * . * TO user;
```

Une fois MariaDB installé et fonctionnel, il ne nous reste plus que PHP. PHP (acronyme récursif de PHP: Hypertext Preprocessor) est un langage de script open source à usage général largement utilisé, particulièrement adapté au développement Web et pouvant être intégré au HTML. Toujours dans le même démarche d'installation, il faut cette fois-ci installer PHP:

```
$ apt install php 7.4
```

Une fois l'installation terminée, la page de test est créée dans le répertoire `"/srv/dev2/www/testMariaDb.php"`, cette page est éditée avec l'éditeur de texte nano. Dans `"/srv/dev2/www/testMariaDb.php"` voici ce que l'on retrouve:

```
<?php
echo "<h1>Page de test de la base de données de MariaDB</h1>"

$host_name = "localhost";
$database = "dbtest";
$user_name = "user";
$password = "password";

$connect = new mysqli($host_name, $user_name, $password, $database);

if($connect->connect_error)
{
 echo '<p>connexion au serveur MySQL a choue :_' .
 $connect->connect_error . '</p>';
}
else
{
 $result = $connect->query("SELECT_*_FROM test;");
 if($result->num_rows > 0){
 while($row = $result->fetch_assoc()){
 echo "NOM:_" . $row["nom"] . "PRENOM:_" . $row["prenom"];
 }
 }
 else{
 echo '<p>0 result</p>';
 }
}
?>
```

Ce petit script php permet de se connecter à la base de données MariaDB avec les différents identifiants (Nom de l'hôte, nom de la base de données, nom de l'utilisateur et le mot de passe). Dans le cas où la connexion n'est pas possible, alors une erreur est affichée, et dans le cas contraire, on affiche sur la page le contenu de la base de données. La page web est accessible via `"http://informatique.agence.atlantide/testMariaDb.php"`, si sur cette page le contenu de la base de données MariaDB est affiché, alors le trio Apache MariaDB PHP est fonctionnelle. MariaDB n'est pas la seule alternative, il en existe d'autres comme nous allons le voir prochainement. TODO - changer le script écrit par une image, et ajouter des images des pages web de test - TODO

## 8.2 Installation et mise d'un autre LAMP avec PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle et objet. Ce système est comparable à d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB et Firebird), ou propriétaires (comme Oracle, MySQL, Sybase, DB2, Informix et Microsoft SQL Server). Dans cette partie PostgreSQL prendra la place de MariaDB. L'installation de

PostgreSQL et des paquets PHP compatible avec PostgreSQL est la première étape:

```
$ apt install postgresql postgresql-client
$ apt-get install php-pgsql
```

Pour ce qui est du serveur web Apache pas besoin de le réinstaller ou de changer la configuration, de même pour Linux.

Pour démarrer la base de données au boot du routeur, il existe cette commande:

```
$ systemctl enable --now postgresql
```

Toutes les modifications doivent se faire avec l'utilisateur "postgres" qui est crée avec l'installation de PostgreSQL. On se connecte (depuis root) à cet utilisateur avec la commande:

```
$ su - postgres
```

La base de données PostgreSQL ne possède pas de données à cette étape. C'est à partir de maintenant que nous allons créer la base de données pour nos test, ainsi que l'utilisateur et la table dans la base. Tous ceci est fait à partir des commandes:

```
$ su - postgres
$ psql
postgres=# createuser user
postgres=# createdb dbtest -O user
postgres=# psql dbtest
postgres=# CREATE TABLE test (nom TEXT, prenom TEXT);
postgres=# INSERT INTO test(nom,prenom) VALUES ('Postgre','SQL');
postgres=# SELECT * FROM test;
```

Le principe des commandes précédentes et la suivante: premièrement on passe sur l'utilisateur "postgres", puis on ouvre le terminal de PostgreSQL. Une fois dans l'invite de PostgreSQL on peut alors créer un utilisateur "user", une base de données "dbtest" puis une table "test". Finalement le fonctionnement est similaire à celui de la base de données MariaDB, le seul changement se trouve dans les valeurs de la table "dbtest".

Le script suivant permet d'afficher le contenu dans la base de données de PostgreSQL, ce script PHP se situe dans "/srv/dev2/www/testPostgreSQL.php".

```
<?php
echo "<h1>Page de test de la base de données PostgreSQL</h1>"
$connect = pg_connect("host=localhost dbname=dbtest user=user password=password");

if(pg_result_error())
{
 echo '<p>Echec de connexion au serveur PostgreSQL: ' .
 pg_result_error() . '</p>';
}
else
{
 $result = pgquery('SELECT * FROM test') or die("Echec de requête" . pg_last_error());
}
```

```

while($row = pg_fetch_array($result , null , PGSQLASSOC))
{
 foreach($row as $value)
 echo $value . " ";
 echo "
";
}
pg_free_result($result);
}

pg_close($connect);
?>

```

Avec ceci il est possible de se connecter à la base de données MariaDB avec les différent identifiant (Nom de l'hôte, nom de la base de données, nom de l'utilisateur et le mot de passe). Encore une fois c'est le même principe que la base de données MariaDB mais avec une syntaxe différente. Dans le cas ou la connexion n'est pas possible alors une erreur est affiché, et dans le cas contraire on affiche sur la page le contenu de la base de données. La page web est accessible via "http://informatique.agence.atlantide/testPostgreSQL.php".

TODO - Comme pour MariaDB changer le script par une image - TODO

### 8.3 Installation et mise en place d'un PDO

Désormais nous allons installer et mettre en place PDO qui est l'acronyme de PHP Data Objects: c'est une extension PHP permettant d'interagir avec des bases de données via l'utilisation d'objets. L'une de ses forces réside dans le fait qu'il n'est pas strictement lié à une base de données en particulier: son interface offre un moyen commun d'accéder à plusieurs environnements différents, parmi lesquels: MySQL et PostgreSQL. En résumé PDO va nous permettre d'avoir une abstraction des SGBD, et avec ce principe nous avons écrit un script PHP permettant d'afficher le contenu des 2 bases de données précédemment crée. La première étape est l'installation du paquet PDO dans PHP:

```
$ apt-get install php-pdo
```

Et voici donc le script qui est contenu dans "/srv/dev2/www/testPDO.php", permettant de se connecter aux bases de données et afficher les informations contenu dans celle-ci:

```

<?php
echo "<h1>page de test des bases de données PDO</h1>"
echo "<h1>MariaDb</h1>"
try {
 $conn = new PDO("mysql:host=localhost;dbname=dbtest;charset=utf8", "user",
 $query = $conn->prepare("SELECT * FROM test;");
 $query->execute();
 $result=$query->fetchAll();
 foreach($results as $r)
 echo $r["nom"] . " " . $r["prenom"];
} catch (Exception $e) {
 echo "MariaDb error : " . $e->getMessage();
}

```

```

echo "<h1>PostgreSQL</h1>"
try{
 $conn2 = new PDO("psql:host=localhost;dbname=dbtest;charset=utf8", "user",
 $query2 = $conn->prepare("SELECT_*_FROM_test;");
 $query2->execute();
 $result2=$query->fetchAll();
 foreach($results as $r2)
 echo $r2["nom"] . " " . $r2["prenom"];
} catch (Exception $e){
 echo "PostgreSQL_error:_" . $e->getMessage();
}
?>

```

Ce script se connecte à la base de données MariaDB et affiche le contenu de la table test dans un premier temps puis à la base PostgreSQL dans un second temps pour afficher les informations de la table test. Le script se connecte avec les différents identifiants (Nom de l'hôte, nom de la base de données, nom de l'utilisateur et le mot de passe). Encore une fois c'est le même principe que la base de données vue précédemment. Dans le cas où la connexion n'est pas possible alors une erreur est affichée, et dans le cas contraire on affiche sur la page le contenu de la base de données. La page web est accessible via ["http://informatique.agence.atlantide/testPDO.php"](http://informatique.agence.atlantide/testPDO.php).

TODO - Comme pour MariaDB changer le script par une image - TODO

## 9 Script de routage et matrice de filtrage

Maintenant que nous avons différents services, tel qu'un serveur web, un serveur DNS et autres fonctionnalités. Il nous faut réguler le trafic pour sécuriser notre réseau et que seuls les personnes qui le peuvent se connectent aux différents serveurs. Alors les règles de filtrage avec "iptables" sont parfaitement adaptées pour cette tâche. Afin de gérer notre script de filtrage, nous avons créé deux fichiers qui sont: `"/etc/firewall-free.sh"` pour et `"/etc/firewall-rules.sh"`. Le script `"/etc/firewall-free.sh"` permet de supprimer les règles pré-existantes, il contient seulement:

```

#!/bin/sh
iptables -F //supprimer les règles

```

Mais le script `"/etc/firewall-rules.sh"` contient:

```

#!/bin/sh
./etc/firewall-free.sh

//autoriser l'interface loopback
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT

//finir la politique par défaut
iptables -P INPUT DROP

```

```

#####iptables -P OUTPUT DROP
#####iptables -P FORWARD DROP

#####// accepter le ping pour les tests
#####iptables -A INPUT -p icmp -j ACCEPT
#####iptables -A OUTPUT -p icmp -j ACCEPT

#####// règles serveur local
#####iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT -- http_out
#####iptables -A OUTPUT -p tcp --dport 53 -j ACCEPT -- DNS_out , tcp
#####iptables -A OUTPUT -p udp --dport 53 -j ACCEPT -- DNS_out , udp
#####iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT -- https_out
#####iptables -A OUTPUT -p tcp --dport 22 -j ACCEPT -- ssh_out

#####iptables -A INPUT -p tcp --dport 22 -j ACCEPT -- ssh_in
#####iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT -- p
#####iptables -A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT -- p
#####iptables -A OUTPUT -p tcp -m state --state ESTABLISHED/RELATED -j ACCEPT -- /
#####iptables -A OUTPUT -p udp -m state --state ESTABLISHED/RELATED -j ACCEPT -- /

#####// règles r seaux clients
#####iptables -A FORWARD -p tcp --dport 80 -s 10.1.2.0/24 -o eno1 -j ACCEPT -- http
#####iptables -A FORWARD -p tcp --dport 53 -s 10.1.2.0/24 -o eno1 -j ACCEPT -- dns
#####iptables -A FORWARD -p udp --dport 53 -s 10.1.2.0/24 -o eno1 -j ACCEPT -- dns
#####iptables -A FORWARD -p tcp --dport 443 -s 10.1.2.0/24 -o eno1 -j ACCEPT -- http
#####iptables -A FORWARD -p tcp --dport 22 -s 10.1.2.0/24 -o eno1 -j ACCEPT -- ssh
#####iptables -A FORWARD -p tcp -m state --state ESTABLISHED/RELATED -j ACCEPT -- /
#####iptables -A FORWARD -p udp -m state --state ESTABLISHED/RELATED -j ACCEPT -- /

#####// dialogue entre client et serveur
#####iptables -A INPUT -s 10.1.2.0/24 -i enp3s0 -j ACCEPT

#####// masquerade pour aller sur le net
#####iptables -t nat -A POSTROUTING -s 10.1.2.0/24 -o eno1 -j MASQUERADE -- trans

```

Ce scrip appelle le script qui supprime les règles décrit auparavant, mais permet également d'autoriser l'interface loopback, de définir la politique par défaut, de bloquer l'accès d'un autre groupe à notre serveur. De plus on sécurise le serveur en ouvrant les bons ports, le port 80 fait référence au protocole http. Le port 443 fait référence au protocole https, le port 53 pour le DNS et le port 22 pour SSH. On sécurise de la même façon pour le client avec l'ouverture des ports adéquats. Finalement on permet le dialogue entre notre client et le serveur, et on permet au client d'aller sur internet.

TODO - changer le texte par des images - TODO

Dans la configuration actuelle les scripts créés ne sont jamais utilisés, c'est pour cela que dans "/etc/network/interfaces" on ajoute la ligne suivante:

```
pre-up /etc/firewall-rules.sh
```

Cette ligne permet de dire qu'au démarrage des interfaces réseau le fichier "/etc/firewall-rules.sh" est exécuté. Et permet donc d'ajouter nos scripts de filtrage dans les scripts de démarrage.

## 9.1 Sauvegarde des bases MySQL et PostgreSQL

Les bases de données que se soit avec MariaDB ou PostgreSQL peuvent être modifier ou supprimé par des utilisateurs malveillant, c'est pour cette raison qu'une sauvegarde de ces bases de données est un élément important de notre architecture. Pour les sauvegarde de MariaDB donc de MySQL il existe un logiciel qui permet de le faire automatiquement, mais celui-ci est téléchargeable uniquement avec le gestionnaire de paquet "aptitude". Il faut donc l'installer avec:

```
$ apt-get install aptitude
```

Une fois le gestionnaire aptitude installé on peut alors installé le paquet "automysqlbackup":

```
$ aptitude install automysqlbackup
```

TODO - finir d'expliquer la configuration avec les photos - TODO

TODO - ajouter des images pour automysql - TODO

Pour ce qui est de la sauvegarde de la base de données PostgreSQL, nous avons simplement utilisé crontab et zip afin d'économiser de la place sur le disque dur. Voici l'ensemble des commandes que nous avons tapés:

```
$ mkdir -p /backups/postgresql
$ chown postgres /backups/postgresql/
$ su - postgres
postgres=# crontab -e
0 0 * * 0 pg_dumpall -U postgres | gzip > /backups/postgresql/save.gz
```

Pour les sauvegardes nous avons simplement crée le dossier "/backups/postgresql", et donnée les droits de ce dossier a l'utilisateur de PostgreSQL. Finalement nous n'avions plus qu'a crée le fichier crontab sur le compte de l'utilisateur et indiquer dans celui-ci que la sauvegarde sera effectué tous les dimanches à minuit et de compresser la sauvegarde pour l'envoyé dans le fichier "save.gz".

## 9.2 Les sudoers

Sudo est un utilitaire de ligne de commande qui permet aux utilisateurs de confiance d'exécuter des commandes en tant qu'un autre utilisateur, par défaut root. Il faut également savoir que root possède l'entière des droits sur une machine. Dans la continuité de notre serveur nous souhaitons permettre au webmaster de modifier la configuration de apache et qu'il puisse relancer le serveur en cas de panne. Pour cela il faut ajouter cette utilisateurs dans le fichier des sudoers, on peut se rendre dans ce fichier de 2 façons différente: avec la commande "visudo" ou directement dans le fichier en question "/etc/sudoers". Pour rappel l'utilisateur qui représente notre webmaster est "dev1". Dans le fichier des sudoers nous avons rajouté une ligne, qui est la dernière de ce fichier.

```
This file MUST be edited with the 'visudo' command as root.
#
Please consider adding local content in /etc/sudoers.d/ instead of
directly modifying this file.
See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

Host alias specification

User alias specification

Cmnd alias specification

User privilege specification
root ALL=(ALL:ALL) ALL

Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL

See sudoers(5) for more information on "@include" directives:

@include::etc/sudoers.d

dev1 ALL=(ALL:ALL) /usr/bin/nano /etc/apache2/*, /etc/init.d/apache2 restart, /usr/sbin/service apache2 restart
```

Figure 23: Contenu du fichier pour les sudoers.

Cette ligne nous indique que l'utilisateur "dev1" à tous les droits pour les fichiers "/usr/bin/nano" qui est l'éditeur de texte pour modifier le contenu de ces fichiers, mais il possède également les droits sur "/etc/apache2/\*", "etc/init.d/apache2 restart" et "/usr/sbin/service apache2 restart". Dorénavant il peut ajouter/supprimer tout ce qu'il souhaite dans "/etc/apache2/" et peut redémarrer le serveur apache.

## 10 Mise en place d'un domaine Samba/LDAP

### 10.1 Définition du DIT

### 10.2 Mise en place de LDAP

Paramétrage service LDAP, rendre opérationnel fichier LDAP, machine exploite données de l'annuaire.

### 10.3 Mise en place de Samba

groupe, utilisateurs, politique de sécurité, adapter matrice de filtrage.

### 10.4 Sauvegarde automatique de l'annuaire

## 11 Conclusion

## 12 Annexe

### Liste des images

1	Les 3 partitions d'un disque . . . . .	4
2	Tableau des adresses IP de chaque groupe . . . . .	7
3	Fichier de configuration des interfaces réseaux du routeur . . . . .	8



4	Ping de notre routeur à un autre . . . . .	10
5	Route de notre routeur . . . . .	11
6	Interface du DHCP . . . . .	13
7	Configuration du serveur DHCP . . . . .	13
8	Fichier de configuration de syslog . . . . .	15
9	Fichier de configuration de syslog . . . . .	16
10	Contenu du fichier crontab. . . . .	18
11	Exemple d'interrogation d'un DNS avec host. . . . .	19
12	Exemple d'interrogation d'un DNS avec dig. . . . .	19
13	Exemple d'interrogation d'un DNS avec nslookup. . . . .	20
14	Requête de nameserver pour un DNS avec host. . . . .	21
15	Requête de nameserver pour un DNS avec dig. . . . .	21
16	Contenu du fichier de configuration named.conf.local. . . . .	25
17	Contenu du fichier de configuration du DNS normal. . . . .	26
18	Contenu du fichier de configuration du DNS inversé. . . . .	26
19	Contenu du fichier de configuration de resolv.conf. . . . .	27
20	Contenu du fichier de configuration de named.conf.options. . . . .	27
21	Contenu du fichier de configuration de apache. . . . .	30
22	Contenu du fichier de configuration de dev2. . . . .	31
23	Contenu du fichier pour les sudoers. . . . .	39