Livrable 5

Amberny Peran, Barnouin Clement, Burellier Loucas,
Krainik-Saul Vladimir, Schicke Samuel,

2025-04-02

Contents

1 Introduction

MiniCoffee est un groupe français spécialiste de l'univers du café, connu notamment pour ses machines à café en libre service. Pour l'année 2025, l'entreprise souhaite mettre à jour son infrastructure réseau interne en ajoutant :

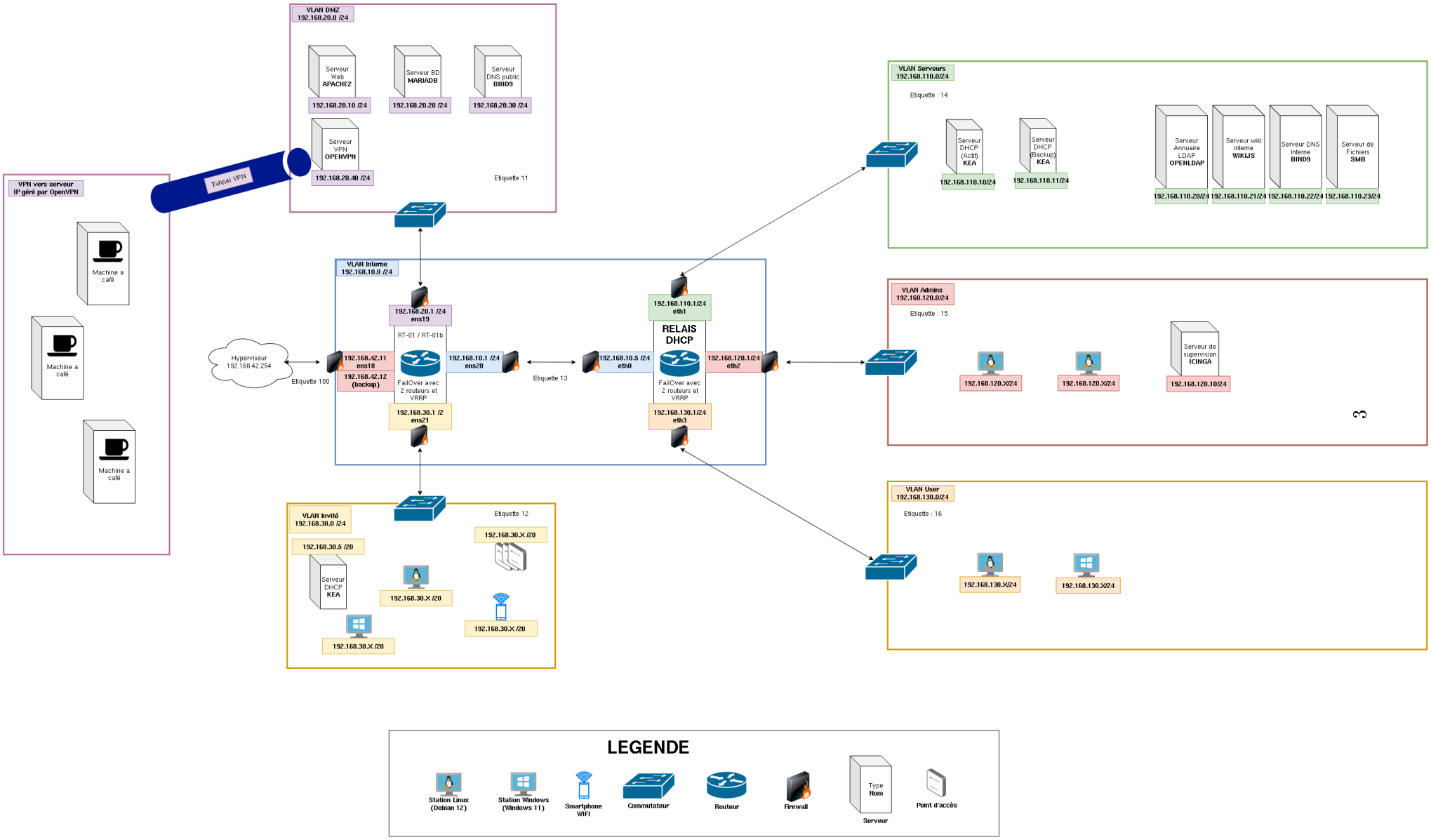
- Divers serveurs d'utilité interne pour les employés et l'équipe informatique;
- Un réseau invité pour permettre à ses fournisseurs d'utiliser du matériel informatique sur place;
- Divers serveurs accessibles en ligne (site Web, serveur DNS public);
- Une meilleure communication entre ses machines à café et son infrastructure, qui a été un des points faibles de l'entreprise ces dernières années.

Pour cette tâche, MiniCoffee a fait appel à BAV4, notre équipe d'étudiants de l'IUT2 Informatique de Grenoble.

2 Architecture

L'architecture de notre réseau n'a pas énormément changé. Les seules modifications apportées au réseau sont :

- Passage d'un LAN à un VLAN pour une meilleure segmentation du réseau.
- Les adresses IP internes se terminent par 1XX.
- Les adresses IP externes se terminent par XX.


VLAN DMZ
192.168.20.0 /24
Serveur Web APACHE2
192.168.20.10 /24
Serveur BD MARIADB
192.168.20.20 /24
Serveur DNS public BIND9
192.168.20.30 /24
Serveur VPN OPENVPN
192.168.20.40 /24
Etiquette 11
Tunnel VPN
VPN vers serveur IP géré par OpenVPN
Machine a café
Machine a café
Machine a café
VLAN Interne
192.168.10.0 /24
192.168.20.1 /24 ens19
RT-01 / RT-01b
Hyperviseur 192.168.42.254
Etiquette 100
192.168.42.11 ens18
192.168.42.12 (backup)
FailOver avec 2 routeurs et VRRP
192.168.10.1 /24 ens20
Etiquette 13
192.168.10.5 /24 eth0
RELAIS DHCP
FailOver avec 2 routeurs et VRRP
192.168.120.1/24 eth2
192.168.110.1/24 eth1
192.168.30.1 /2 ens21
192.168.130.1/24 eth3
VLAN Invité
192.168.30.0 /24
Etiquette 12
192.168.30.5 /20
Serveur DHCP KEA
192.168.30.X /20
192.168.30.X /20
192.168.30.X /20
192.168.30.X /20
VLAN Serveurs
192.168.110.0/24
Etiquette : 14
Serveur DHCP (Actif) KEA
192.168.110.10/24
Serveur DHCP (Backup) KEA
192.168.110.11/24
Serveur Annuaire LDAP OPENLDAP
192.168.110.20/24
Serveur wiki interne WIKIJS
192.168.110.21/24
Serveur DNS interne BIND9
192.168.110.22/24
Serveur de Fichiers SMB
192.168.110.23/24
VLAN Admins
192.168.120.0/24
Etiquette : 15
192.168.120.X/24
192.168.120.X/24
Serveur de supervision ICINGA
192.168.120.10/24
VLAN User
192.168.130.0/24
Etiquette : 16
192.168.130.X/24
192.168.130.X/24
LEGENDE
Station Linux (Debian 12)
Station Windows (Windows 11)
Smartphone WIFI
Commutateur
Routeur
Firewall
Type Nom
Serveur
Point d'accès

3 Ressources Matérielles utilisées

Actuellement, notre infrastructure réseau comprend un total de 9 machines actives, chacune jouant un rôle spécifique dans notre environnement.

En ce qui concerne le stockage, nous allouons entre 3 et 5 Go d'espace disque par machine, en fonction de leurs besoins en ressources et des tâches qu'elles doivent accomplir. Plus précisément :

- Les machines nécessitant le moins de ressources se voient attribuer 3 Go de stockage, ce qui est suffisant pour assurer leur bon fonctionnement sans surconsommation d'espace.
- Les machines les plus sollicitées, qui traitent des volumes de données plus importants ou exécutent des processus plus intensifs, bénéficient quant à elles de 5 Go de stockage afin de garantir des performances optimales.

En termes de mémoire vive (RAM), chaque machine de notre réseau dispose actuellement de 1 Go. Cette allocation permet de répondre aux besoins de nos applications tout en maintenant un bon équilibre entre performance et consommation de ressources.

Nous surveillons régulièrement l'utilisation de la RAM et du stockage afin d'optimiser notre infrastructure si nécessaire et d'anticiper toute montée en charge.

4 Installation et Configuration des éléments de l'infrastrucure

On va détailler dans cette partie comment nous avons configuré les éléments de notre infrastructure réseau.

4.1 Résaux Virtuel

Nous avons créé des VXLAN pour interconnecter les hyperviseurs au sein du cluster, permettant ainsi une communication entre eux. De plus, nous avons mis en place des VNET spécifiques pour chaque hyperviseur afin de segmenter et d'optimiser la gestion du réseau virtuel, garantissant une meilleure performance et une isolation accrue des ressources.

4.2 Routeurs

4.2.1 Configuration des interfaces

Tout d'abord, il faut configurer les interfaces de la machine qui va servir de routeur.

Il faut configurer le fichier **/etc/network/interfaces** pour attribuer les bonnes ip et CIDR pour chaque interface. Voici un explique de chaque paramètre que l'on peut mettre :

Voici un fichier de configuration type pour un routeur avec plusieurs interface :

/etc/network/interfaces

```
1         # Interface WAN (connectee a Internet)
2         auto eth0
3         iface eth0 inet dhcp
4         mtu 1450  # Taille MTU standard
5         # Interface LAN 1
6         (reseau interne 192.168.1.0/24)
7         auto eth1
8         iface eth1 inet static
9             address 192.168.1.1/24
10            gateway 192.168.1.254
11            # Facultatif, utilise uniquement si ce reseau doit
12            sortir par une autre route
13            dns-nameservers 192.168.1.1 8.8.8.8
14            dns-domain lan1.local
15            mtu 1450  # Optimise pour les reseaux
16            locaux rapides
17
18        # Interface LAN 2
19        (reseau interne 10.10.0.0/24)
20        auto eth2
21        iface eth2 inet static
22            address 10.10.0.1/24
23            mtu 1450  # Optimise pour un second
24            reseau local
25
26        # Activation du routage entre les reseaux
27    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
28
29        # Ajout de routes pour permettre aux
30        deux reseaux de communiquer entre eux
31        # Ajout de routes pour permettre aux
32        deux reseaux de communiquer entre eux
33    post-up ip route add 192.168.1.0/24
34    via 192.168.1.1 dev eth1
35    post-up ip route add 10.10.0.0/24
36    via 10.10.0.1 dev eth2
```

- Une première ligne avec la façon d'on l'interface s'active.
  - (L2) **`auto [nomInterface]`** : Pour activer automatiquement au démarrage. Idéal pour les interfaces fixes, comme celles des serveurs.
  - **`allow-hotplug [nomInterface]`** : Pour activer uniquement quand elle est détectée. Idéal pour les interfaces amovibles.
- Une seconde qui définit sa configuration.
  - **`iface eth0 inet manual`** : Interface n'a pas de configuration IP et doit être activé à la main
  - **`iface eth0 inet none`** : Interface active mais sans configuration IP
  - (L3) **`iface [nomInterface] inet dhcp`** : Utilise le DHCP pour l'attribution d'IP, ...
  - (L8) **`iface eth1 inet static`** : Pour faire une configuration avec une IP statique
- Ajout de paramètres pour la configuration de IP si configuration **`static`** :
  - (L9) **`address 192.168.1.1/24`** : L'adresse IP static
  - (L10) **`gateway 192.168.1.254`** : Le gateway
  - (L13) **`dns-nameservers 192.168.1.1 8.8.8.8`** : Les DNS
  - (L14) **`dns-domain lan1.local`** : Le domaine DNS
  - **`metric 10`** : La priorité de l'utilisation de cette interface. Plus la valeur est basse, plus la priorité est haute.

- **up ip addr add 192.168.100.15/24 dev [nomInterface]** : Si tu veux plusieurs adresses IP sur la même interface
- (L4) **mtu 1450** : MTU maximum
- (L33) **post-up ip route add 192.168.1.0/24 via 192.168.100.10** : Si la machine doit accéder à un réseau via une passerelle spécifique. Tout le trafic vers 192.168.200.0/24 passera via 192.168.100.10.

Une fois le fichier configurer, il redémarrer le service avec

```
$ sudo systemctl restart networking.service
```

4.2.2 Configuration des routes entre les réseaux

Par défaut, une machine Linux ne fait pas passer n'importe quel paquet comme doit le faire un routeur. On doit donc activer cette fonctionnalité qui est sous la forme d'un option dans le fichier **/etc/sysctl.conf**.

```
$ sysctl -p /etc/sysctl.conf
```

Mettre en place le NAT (fichier nft a executer)

</> filtrage-nat

```
1 nft add table filtrage_nat
2 nft 'add chain filtrage_nat prerouting { type nat hook prerouting priority 0 ; }'
3 nft 'add chain filtrage_nat postrouting { type nat hook postrouting priority 0 ; }'
4 nft add rule filtrage_nat postrouting masquerade
```

4.3 Serveur DHCP

4.3.1 Installation

- Creer une VM avec une @IP statique car c'est lui qui donne les IPs
- Installer kea :

```
# apt install kea-dhcp4-server
```

4.3.2 Configuration

- Sauvegarder la configuration pour la réstaurer en cas de problème

```
# mv /etc/kea/kea-dhcp4.conf /etc/kea/kea-dhcp4.conf.bkp
```

- Ajouter les configuration du serveur dans le fichier **/etc/kea/kea-dhcp4.conf**

/etc/kea/kea-dhcp4.conf.bkp

```
{
    "Dhcp4": {
        "interfaces-config": {
            "interfaces": [
                "ens18"
            ]
        },
        "valid-lifetime": 691200,
        "renew-timer": 345600,
        "rebind-timer": 604800,
        "authoritative": true,
        "lease-database": {
            "type": "memfile",
            "persist": true,
            "name": "/var/lib/kea/kea-leases4.csv",
            "lfc-interval": 3600
        },
        "subnet4": [
            {
                "subnet": "192.168.120.0/24",
                "pools": [
                    {
                        "pool": "192.168.120.10
                        - 192.168.120.200"
                    }
                ],
                "option-data": [
                    {
                        "name": "domain-name-servers",
                        "data": "192.168.110.22"
                    },
                    {
                        "name": "domain-search",
                        "data": "bav4.local"
                    },
                    {
                        "name": "routers",
                        "data": "192.168.120.1"
                    }
                ]
            },
            [... Ajouter autant de subnet que de sous reseaux sont
                concernes]
        ]
    }
}
```

- (L2) **"Dhcp4"** : Indique que l'attribution est faite avec des @IPv4
- (L3-5) Indique l'interface qui emettra les DHCP response
- (L8-11) Configuration des differents temps de sauvegarde/renouvellement...
- (L12-17) Configuration de la base de donnée qui contiendras les données DHCP
- (L18) **"subnet4"** : Indique que le subnet spécifié est en IPv4
- (L20) **"subnet": "192.168.14.0/24"** : indique le sous reseau ou le serveur DHCP attribue les adresses.
- (L21-26) Configure les differents intervalles IP attribués
- (L27-L39) Configure les options DHCP (@IP du serveur DNS, adresse du serveur DNS, @IP du routeur)

4.4 Serveur DNS

Marche a suivre :

1. Nous allons créer 2 serveur DNS, un serveur Externe, qui va être accessible depuis l'exterieur, et un serveur DNS Interne, qui va être accessible uniquement depuis l'interieur, ca sera utile pour le wiki.
2. On va commencer par créer le DNS privé puis on le clonera pour en faire un DNS public, et il faudra juste supprimer les alias créés pour le wiki et supprimer l'ACL "lan" et l'option lan dans allow-query (voir suite)

4.4.1 Installation de BIND9

On installe Bind9 avec la commande suivante :

```
# apt install bind9 dnsutils
```

4.4.2 Configuration des options DNS

nous allons d'abord copier la config pour pouvoir la rétablir facilement en cas d'erreur :

```
# cd /etc/bind
```

```
# cp named.conf.options named.conf.options.bkp
```

```
# cp named.conf.local named.conf.local.bkp
```

Nous allons ensuite modifier le fichier de config **`/etc/bind/named.conf.options`** et changer le `0.0.0.0` dans le champ forwarders par le DNS de l'UGA : `152.77.1.22`
Etant donné qu'on créée actuellement le DNS privé, on va déclarer une ACL Lan, pour que seul le reseau local puisse y avoir accès (config a mettre avant le champ "option") :

⚙/etc/bind/named.conf.options

```
1  acl "lan" {
2          192.168.110.0/24;
3          192.168.120.0/24;
4          192.168.130.0/24;
5          localhost;
6          localnets;
7  };
8  options{
9  [...]
10         allow-query { lan; };
11 };
```

- (L1) **`acl "lan"`** : On déclare une ACL appellée "lan", qui permet que seule les @Ip spécifiées auront accès au serveur DNS
- (L2-7) Déclarations des sous-réseaux concernés par l'ACL
- (L10) **`allow-query lan; ;`** indique que seuls les sous-réseaux de l'ACL peuvent query le serveur DNS

4.4.3 Configuration de zone

Nous allons desormais créer la zone bav4. Dans **`/etc/bind/named.conf.local`** ,on ajoute donc la zone suivante:

/etc/bind/named.conf.local

```
zone "bav4.local" {
    type master;
    file "/etc/bind/db.bav4.local";
    allow-update { none; };
};
```

Puis nous allons dupliquer la db.local en l'appellant db.bav4.local pour pouvoir configurer la zone :

```
# cp /etc/bind/db.local /etc/bind/db.bav4.local
```

Puis, dans db.bav4.local, nous allons mettre en place la config suivante :

/etc/bind/named.conf.local

```
$TTL    604800
@       IN      SOA     srv-dns.bav4.local.  root.bav4.local. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@                IN       NS       srv-dns.bav4.local.
srv-dns          IN       A        192.168.110.22
```

- (L3) Numéro de serie
- (L4) Délai de rafraichissement pour la synchronisation des configurations entre plusieurs serveurs DNS
- (L5) Délai au bout duquel un serveur DNS secondaire devra retenter une synchronisation
- (L6) Temps d'expiration du serveur DNS
- (L7) Durée de conservation dans le cache de l'information "NXDOMAIN"
- (L9-LXX) Création des enregistrement DNS : plusieurs formes possibles :
  1. **Lien nom-@IP :** <nom-de-l'hote> IN A <@IP>
  2. **Lien alias-nom :** <nom-de-l'alias> IN CNAME <nom-de-l'enregistrement-de-référence>

Une fois cela fait, on restart bind9 et notre serveur DNS est opérationnel :

```
# systemctl restart bind9
```

```
# systemctl enable named.service
```

4.4.4 Configuration de DoT (DNS over TLS

Afin de rajouter une couche de sécurité dans les requetes DNS, nous allons utiliser DoT (DNS over TLS) qui permet de chiffrer dnos requetes DNS :

Niveau Serveur **`bind9 v9.18.33`** supporte DoT sans besoin d'un logiciel tiers de type proxy, nous allons donc le mettre en place directement dans bind9.

1. On crée et se place dans un nouveau dossier **`ssl`**

   ```
   # cd /etc/bind/ssl
   ```

2. On genere une cle et un certificat :

```
# openssl req -x509 -newkey rsa:2048 -nodes -keyout /etc/bind/ssl/cleDNS.key
-out /etc/bind/ssl/certDNS.crt -days 365 -subj "/CN=bav4.local"
```

3. On donne ensuite l'ownership de la clé a **bind**

```
# chown bind:bind /etc/bind/ssl/cleDNS.key
```

4. Finalement, on modifie le fichier **/etc/bind/named.conf.options** :

⚙/etc/bind/named.conf.options

```
1  tls servertls {
2          cert-file "/etc/bind/ssl/certDNS.crt";
3          key-file "/etc/bind/ssl/cleDNS.key";
4  };
5  options {
6  [...]
7
8          listen-on { any; };
9          listen-on-v6 { any; };
10         listen-on tls servertls { any; };
11         allow-query { lan; };
12
13 }
```

- (L1) On déclare un bloc tls "**servertls**"
- (L2-3) On indique le chemin vers le certificat et la clé
- (L10) on indique qu'on écoute les requêtes TLS liées au blocs **servertls** sur toutes les adresses

Niveau Client

Pour utiliser DoT sur les machines client, nous allons utiliser le package **systemd-resolved** :

1. On installe systemd-resolved :

```
# apt install systemd-resolved
```

2. On va ensuite modifier la configuration de ce dernier :

⚙/etc/systemd/resolved.conf

```
1  [...]
2
3  [Resolve]
4  DNS=192.168.110.24#bav4.local
5  DNSOverTLS=yes
6
7  [...]
```

- (L4) On déclare le DNS qui va être utilisé (`192.168.110.24`) et son hostname (bav4.local), spécifié lors de la création du certificat vu plus haut dans ce document
- (L5) On active le mode DNSOverTLS

3. Ensuite on copie le certificat DNS **certDNS.crt** présent sur le serveur DNS :

```
# scp SERVDNS:/etc/bind/ssl/certDNS.crt
/usr/local/share/ca-certificates/certDNS.crt
```

4. Puis on reload les certificats avec :

```
# update-ca-certificates
```

5. Finalement, dans **/etc/nsswitch.conf**, on change la ligne **hosts** :

/etc/nsswitch.conf

```
[...]
hosts:  files   resolve dns
[...]
```

la ligne **hosts** a donc 3 options, ainsi:

(a) Le système vérifie le fichier local **/etc/hosts**, cette methode est très rapide

(b) Si cela échoue, le système va tenter d'utiliser **systemd-resolved**

(c) Si cela échoue, le système interroge directement un serveur DNS configuré via **/etc/resolv.conf**

Finalement, on teste si resolvectl marche avec :

```
# resolvectl status
```

Et si les querys fonctionnent avec

```
# resolvectl query google.com
```

Pour automatiser cette configuration, nous allons modifier les machines templates pour que les prochaines VMs utilisent DoT et créer un script d'automatisation d'installation pour les machines déja existantes.

Script a éxécuter sur les machines client

</> scriptDoTClient.sh

```
#!/bin/bash

# Definir l'adresse IP du serveur DNS et son nom de domaine
DNS_IP="192.168.110.24"
DNS_HOSTNAME="monserveur.local"

echo "Installation de systemd-resolved..."

apt update && apt install -y systemd-resolved

# Activer et demarrer systemd-resolved
systemctl enable --now systemd-resolved

echo "systemd-resolved est installe et actif."

# Sauvegarde des fichiers avant modification
echo "Sauvegarde des fichiers de configuration..."
cp /etc/systemd/resolved.conf /etc/systemd/resolved.conf.bak
cp /etc/nsswitch.conf /etc/nsswitch.conf.bak
echo "Changement de la config"
rm /etc/nsswitch.conf
rm /etc/systemd/resolved.conf
cp /tmp/nsswitch.conf /etc/nsswitch.conf
cp /tmp/resolved.conf /etc/systemd/resolved.conf
cp /tmp/certDNS.crt /usr/local/share/ca-certificates/certDNS.crt

echo "Update du certificat"
update-ca-certificates

# Redemarrer systemd-resolved pour appliquer les changements
echo "Redemarrage de systemd-resolved..."
systemctl restart systemd-resolved

# Tester la resolution DNS
echo "Test de resolution DNS avec systemd-resolved..."
if resolvectl query google.com | grep -q "google.com"; then
    echo "Test reussi : la resolution DNS fonctionne"
    echo "Configuration terminee !"
else
    echo "Echec du test DNS. Verifie la configuration."
fi
```

- (L4-5) Déclaration de l'@IP du serveur DNS et le nom du certificat
- (L9-12) Installation et activation de **systemd-resolved**
- (L18-19) Sauvegarde des anciens fichiers de configuration
- (L21-23) Remplacement de la configuration
- (L24) Ajout du certificat
- (L28) Update des certificats
- (L32) Redemarrage de **systemd-resolved**
- (L36-40) Test de connectivité final

Script a éxécuter sur Bastion

Ce script nécéssite que la machine Bastion ait les fichiers **nsswitch.conf**, **resolve.conf**, **scriptDoTClient.sh** dans le répertoire dans lequel ce trouve le script.

</> scriptDoTServeur.sh

```
#!/bin/bash
for host in m1 m2 m3 m4; do
    scp scriptDoTClient.sh nsswitch.conf resolved.conf certDNS.crt $host:/tmp/
    ssh $host
done
```

A chaque connection, on effectuera en plus la commande

```
$ su -c /tmp/scriptDoTClient.sh
```

- (L1) Itération dans différentes machines connues par **Bastion**
- (L2) Copie des fichiers nécessaires
- (L3) Ouverture d'une liaison ssh

4.5 Wiki

SAMUEL EXPLIQUE UN PEU LE WIKI ET SON CHOIX

4.5.1 Installation/Configuration de WikiJS

Prérequis

1. Nous allons commencer par installer les packages **curl**, **software-properties-common**, **postgreSQL**, **node.js** :

   ```
   # apt install -y curl software-properties-common
   ```

   ```
   # curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
   ```

   Samuel c'est quoi ca ?

   ```
   # apt install -y nodejs postgresql
   ```

2. On se connecte ensuite la base postgres :

   ```
   $ su -i -u postgres psql
   ```

3. On crée la base wikijs et l'utilisateur postgres wikiuser :

   </> shell postgres

   ```
   CREATE DATABASE wikijs;
   CREATE USER wikiuser WITH ENCRYPTED PASSWORD 'motdepasse';
   GRANT ALL PRIVILEGES ON DATABASE wikijs TO wikiuser;
   \q
   ```

Installation/configuration

eventuellement une intro a l'instal

1. Nous installons wikijs et le serveur node.js :

```
$ mkdir /var/www/wikiJS && cd /var/www/wikiJS
```

```
$ curl -fsSL https://get.requarks.io/wiki/latest.tar.gz | tar xz -C .
```

SAM : Ca me semble treeees limite pour bonnaud, pas d'update possible ? si oui explique pk

```
$ npm install
```

2. Nous allons ensuite modifier la configuration met le chemin ici et au fichier de config

⚙ ton foutu chemin

```
db:
  type: postgres
  host: localhost
  port: 5432
  user: wikiuser
  pass: motdepasse
  db: wikijs
```

- A quoi sert tout ce truc

3. Finalement, démarrons Wiki.js :

```
$ node server
```

4.5.2 Configuration de Nginx comme Proxy

Intro : pk on fait ca, c'est quoi et a quoi ca sert

Installation

Nous installons nginx via apt :

```
# apt install nginx
```

Configuration du Virtual Host

1. Créons un fichier de configuration pourquoi faire ? :

```
# nano /etc/nginx/sites-available/wiki
```

2. Dans ce dernier, ajoutons le contenu suivant :

⚙ /etc/nginx/sites-available/wiki

```
db:
  type: postgres
  host: localhost
  port: 5432
  user: wikiuser
  pass: motdepasse
  db: wikijs
```

- que fait chaque ligne t'as capté

3. On active ensuite la configuration a quoi sert ce lien (le ln -s dans ta commande) :

```
# ln -s /etc/nginx/sites-available/wiki /etc/nginx/sites-enabled/
```

```
# systemctl restart nginx
```

4.5.3 Accès interne

Pour acceder au wiki depuis n'importe quel station interne, il va falloir au préalable l'inscrire dans les enregistrement de notre serveur DNS : 4.4.3

4.5.4 Accès externe

a quoi sert - il ?

Génération de la clé

Si la machine ne possède pas de clé ssh, générez la :

```
$ ssh-keygen -t rsa -b 4096
```

Puis copiez la sur le serveur intermediaire et la machine cible pas compris de quoi tu parle, si t'eclaircis un peu ca sera

Tunnel SSH

Pour lancer le tunnel SSH :

```
# ssh -J krainikv@129.88.210.230,bav-rt@192.168.42.11 bav-server@192.168.110.47 -L
9090:192.168.110.47:443 -fN
```

- 'J' : Chaîne de saut SSH (jump hosts)
- 'L 9090:192.168.110.47:443' : Redirige le port 9090 local vers le 443 distant
- 'fN' : Exécute en arrière-plan sans ouvrir de shell

Pour fermer le tunnel :

```
# pkill -f "ssh -J krainikv@129.88.210.230"
```