

# Livrable 5

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

2025-04-03

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Architecture</b>	<b>2</b>
<b>3</b>	<b>Ressources Matérielles utilisées</b>	<b>4</b>
<b>4</b>	<b>Installation et Configuration des éléments de l'infrastructure</b>	<b>4</b>
4.1	Réseaux Virtuel . . . . .	4
4.2	Routeurs . . . . .	4
4.2.1	Configuration des interfaces . . . . .	4
4.2.2	Configuration des routes entre les réseaux . . . . .	6
4.3	Serveur DHCP . . . . .	6
4.3.1	Installation . . . . .	6
4.3.2	Configuration . . . . .	6
4.4	Serveur DNS . . . . .	8
4.4.1	Installation de BIND9 . . . . .	8
4.4.2	Configuration des options DNS . . . . .	8
4.4.3	Configuration de zone . . . . .	9
4.4.4	Configuration de DoT (DNS over TLS . . . . .	10
4.5	Wiki . . . . .	13
4.5.1	Installation/Configuration de WikiJS . . . . .	13
4.5.2	Configuration de Nginx comme Proxy . . . . .	14
4.5.3	Accès interne . . . . .	15

# 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 :

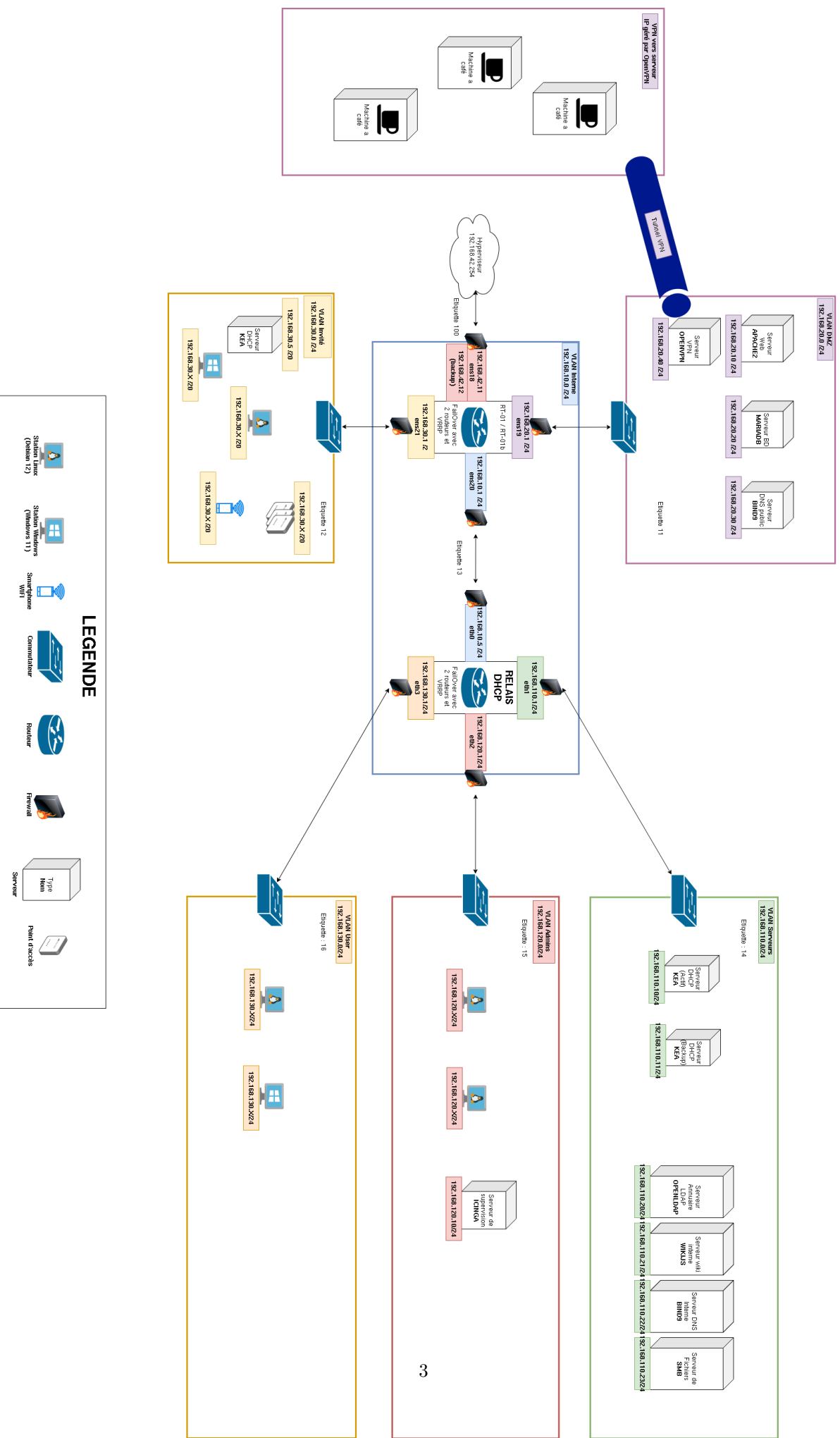
- Divers serveurs d'utilité interne pour les employés et l'équipe informatique;
- Un réseau invité pour permettre à ses fournisseurs d'utiliser du matériel informatique sur place;
- Divers serveurs accessibles en ligne (site Web, serveur DNS public);
- Une meilleure communication entre ses machines à 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.



### 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'infrastructure

Nous allons détailler dans cette partie comment nous avons configuré les éléments de notre infrastructure réseau.

#### 4.1 Réseaux 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 exemple de chaque paramètre que l'on peut mettre :

Voici un fichier de configuration type pour un routeur avec plusieurs 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

```

### Détail de la configuration

- 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 l'on souhaite plusieurs adresses IP sur la même interface
- (L4) **mtu 1450** : MTU maximum
- (L3) **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 configuré, il redémarrer le service avec

```
# systemctl restart networking.service
```

## 4.2.2 Configuration des routes entre les réseaux

Pas sur de l'explication les sangs ? Par défaut, une machine Linux ne fait pas passer n'importe quel paquet c'est le but d'avoir des routeurs ? 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
```

<La partie qui suit devrait être dans les routeurs non ? Mettre en place le NAT (fichier nft a exécuter)

```
</> 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

Afin de pouvoir attribuer des @IP de façon automatique, nous allons installer un serveur DHCP, le logiciel que nous allons utiliser pour le DHCP s'appelle **Kea**. Kea est un serveur DHCP développé par l'ISC, conçu pour être plus flexible et performant que ISC DHCP. Kea supporte IPv4 et IPv6 et est particulièrement adapté aux environnements à grande échelle nécessitant une gestion avancée des adresses IP

### 4.3.1 Installation

- Nous créons une VM avec une @IP statique car c'est cette dernière qui attribuera les @IP.
- Installation de Kea :

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

### 4.3.2 Configuration

- Nous sauvegardons la configuration pour la restaurer en cas de problème

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

- Nous créons par la suite le fichier `/etc/kea/kea-dhcp4.conf` qui doit contenir la configuration suivante :

```

1 {
2   "Dhcp4": {
3     "interfaces-config": {
4       "interfaces": [
5         "ens18"
6       ]
7     },
8     "valid-lifetime": 691200,
9     "renew-timer": 345600,
10    "rebind-timer": 604800,
11    "authoritative": true,
12    "lease-database": {
13      "type": "memfile",
14      "persist": true,
15      "name": "/var/lib/kea/kea-leases4.csv",
16      "lfc-interval": 3600
17    },
18    "subnet4": [
19      {
20        "subnet": "192.168.120.0/24",
21        "pools": [
22          {
23            "pool": "192.168.120.10 - 192.168.120.200"
24          }
25        ],
26        "option-data": [
27          {
28            "name": "domain-name-servers",
29            "data": "192.168.110.22"
30          },
31          {
32            "name": "domain-search",
33            "data": "bav4.local"
34          },
35          {
36            "name": "routers",
37            "data": "192.168.120.1"
38          }
39        ]
40      },
41      [... Ajouter autant de subnet que de sous reseaux sont concernes]
42    ]
43  }
44 }

```

### Détail de la configuration

- (L2) **"Dhcp4"** : Indique que l'attribution est faite avec des @IPv4
- (L3-5) Indique l'interface qui émettra les DHCP response
- (L8-11) Configuration des différents 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 différents intervalles IP attribués
- (L27-L39) Configure les options DHCP (@IP du serveur DNS, adresse du serveur DNS, @IP du routeur)

## 4.4 Serveur DNS

Afin de pouvoir lier un nom à une machine, nous allons installer des serveurs DNS (Domain Name Server), ces derniers feront le lien entre les @IP des différentes machines et le nom que nous leur avons attribué. Le logiciel en charge du DNS s'appelle **BIND 9**. BIND 9 est un serveur DNS open source développé par l'ISC. Il est reconnu pour sa stabilité, sa sécurité et sa compatibilité avec les standards du DNS. Doté de nombreuses fonctionnalités avancées, il prend en charge DoT (DNS over TLS), le contrôle d'accès, la journalisation fine, ainsi que la gestion en mode maître/esclave. BIND 9 est configurable via des fichiers texte et s'adapte aussi bien aux petits réseaux qu'aux infrastructures de grande taille

### Marche à suivre :

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

### 4.4.1 Installation de BIND9

Nous installons Bind9 via apt avec la commande suivante :

```
# apt install bind9 dnsutils
```

### 4.4.2 Configuration des options DNS

Nous allons d'abord copier la configuration 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** qui contient les options du serveur DNS.

```
🔧 /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      forwarders{
10         152.77.1.22
11     }
12     allow-query { lan; };
13 };
```

### Détail de la configuration

- (L1) **acl "lan"** : On déclare une ACL appelée "lan", qui permet que seule les @Ip spécifiées auront accès au serveur DNS
- (L2-7) Déclarations des sous-réseaux concernés par l'ACL



- (L9-11) On change le forwarder (le serveur DNS qui résoudra les noms si le nôtre ne les contient pas) par le DNS de l'UGA
- (L12) **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 désormais créer la zone `bav4.local`, c'est cette dernière qui contiendra les enregistrements dns (par exemple `wiki.bav4.local`). Dans **/etc/bind/named.conf.local**, nous ajoutons donc la zone suivante:

```

1 zone "bav4.local" {
2     type master;
3     file "/etc/bind/db.bav4.local";
4     allow-update { none; };
5 };

```

#### Détail de la configuration

- (L2) Indique que ce serveur DNS est l'autorité principale (ou maître) pour la zone `bav4.local`.
- (L3) Indique l'emplacement du fichier qui contient les enregistrements (voir suite)
- (L4) Interdit toute mise à jour dynamique des enregistrements DNS pour la zone concernée.

Nous allons ensuite dupliquer le fichier **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 :

```

1 $TTL      604800
2 @         IN      SOA      srv-dns.bav4.local.  root.bav4.local. (
3             1          ; Serial
4             604800     ; Refresh
5             86400      ; Retry
6             2419200    ; Expire
7             604800 )    ; Negative Cache TTL
8 ;
9 @         IN      NS       srv-dns.bav4.local.
10 srv-dns   IN      A        192.168.110.22
11 ldap     IN      A        192.168.110.30
12
13 [... ajouter autant d'enregistrement que necessaire]

```

#### Détail de la configuration

- (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'hôte> IN A <@IP>
2. **Lien alias-nom** : <nom-de-l'alias> IN CNAME <nom-de-l'enregistrement-de-référence>

Une fois cela fait, relançons **bind9** et notre serveur DNS est maintenant opérationnel :

```
# systemctl restart bind9
```

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

Pour le vérifier, utilisons la commande suivante :

```
$ nslookup google.com
```

#### 4.4.4 Configuration de DoT (DNS over TLS)

Afin de rajouter une couche de sécurité dans les requêtes DNS, nous allons utiliser DoT (DNS over TLS) qui permet de chiffrer nos requêtes DNS. Nous utiliserons par la suite **systemd-resolved** du côté client.

##### 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. Nous créons et se place dans un nouveau dossier **ssl**

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

2. Nous générons une clé et un certificat (auto-signé), nécessaires pour TLS :

```
# 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. Nous donnons ensuite l'ownership de la clé à **bind**

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

4. Finalement, nous modifions le fichier de configuration **/etc/bind/named.conf.options**, vu précédemment :

```

/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 }
```

##### Détail de la configuration

- (L1) Déclaration d'un bloc tls "**servertls**"
- (L2-3) Indique le chemin vers le certificat et la clé
- (L10) Indique que le serveur écoute les requêtes TLS liées au blocs **servertls** sur toutes les adresses (restreintes en réalité à l'ACL "lan" vu précédemment)

## Niveau Client

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

1. Nous installons systemd-resolved :

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

2. Nous allons 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  [...]
```

### Détail de la configuration

- (L4) Déclaration du 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) Activation le mode DNSOverTLS

3. Nous copions ensuite 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, nous rechargeons les certificats avec :

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

5. Finalement, dans **/etc/nsswitch.conf**, nous changeons la ligne **hosts** :

```
⚙️/etc/nsswitch.conf

1  [...]
2  hosts:  files  resolve dns
3  [...]
```

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, nous testons si resolvectl fonctionne avec :

```
# resolvectl status
```

Et la résolution de nom (chiffrée) 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 déjà existantes.

## Script à exécuter sur les machines client

```
</> scriptDoTClient.sh

1  #!/bin/bash
2
3  # Définir l'adresse IP du serveur DNS et son nom de domaine
4  DNS_IP="192.168.110.24"
5  DNS_HOSTNAME="monserveur.local"
6
7  echo "Installation de systemd-resolved..."
8
9  apt update && apt install -y systemd-resolved
10
11 # Activer et démarrer systemd-resolved
12 systemctl enable --now systemd-resolved
13
14 echo "systemd-resolved est installé et actif."
15
16 # Sauvegarde des fichiers avant modification
17 echo "Sauvegarde des fichiers de configuration..."
18 cp /etc/systemd/resolved.conf /etc/systemd/resolved.conf.bak
19 cp /etc/nsswitch.conf /etc/nsswitch.conf.bak
20 echo "Changement de la config"
21 rm /etc/nsswitch.conf
22 rm /etc/systemd/resolved.conf
23 cp /tmp/nsswitch.conf /etc/nsswitch.conf
24 cp /tmp/resolved.conf /etc/systemd/resolved.conf
25 cp /tmp/certDNS.crt /usr/local/share/ca-certificates/certDNS.crt
26
27 echo "Update du certificat"
28 update-ca-certificates
29
30 # Redémarrer systemd-resolved pour appliquer les changements
31 echo "Redémarrage de systemd-resolved..."
32 systemctl restart systemd-resolved
33
34 # Tester la résolution DNS
35 echo "Test de résolution DNS avec systemd-resolved..."
36 if resolvectl query google.com | grep -q "google.com"; then
37     echo "Test réussi : la résolution DNS fonctionne"
38     echo "Configuration terminée !"
39 else
40     echo "Echec du test DNS. Vérifie la configuration."
41 fi
```

## Détail du script

- (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) Redémarrage de **systemd-resolved**
- (L36-40) Test de connectivité final

### Script à exécuter sur Bastion

Ce script nécessite que la machine Bastion ait les fichiers `nsswitch.conf`, `resolve.conf`, `scriptDoTClient.sh` dans le répertoire dans lequel se trouve le script.

```
</> scriptDoTServeur.sh

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

A chaque connection, nous effectuerons en plus la commande

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

### Détail du script

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

## 4.5 Wiki

Pour le wiki nous avons comparé plusieurs choix, on a choisi Wiki.js pour plusieurs raisons essentielles. Tout d'abord, Wiki.js est simple à installer et à administrer, tout en étant très puissant. Il fonctionne avec une base de données (PostgreSQL, MySQL, etc.), ce qui permet une gestion efficace du contenu. Son interface utilisateur moderne et intuitive facilite la navigation et la rédaction des pages. Ensuite, Wiki.js offre une gestion avancée des permissions, ce qui permet de contrôler précisément qui peut voir ou modifier les pages. Il prend aussi en charge plusieurs formats d'édition, dont Markdown, facilitant ainsi la contribution des utilisateurs. Enfin, il est open-source, léger et extensible grâce à ses nombreux modules. Il permet aussi l'intégration avec Git, ce qui est un gros avantage pour la collaboration et la gestion des versions du wiki.

### 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 -
```

ajoute le dépôt de Node.js 18 sur une distribution Debian ou Ubuntu et préparer l'installation de Node.js à partir des packages officiels de NodeSource.

```
# 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
```

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

## Installation/configuration

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 .
```

télécharge et extrait la dernière version de Wiki.js dans le répertoire courant.

```
$ npm install
```

2. Nous allons ensuite modifier la configuration `/var/www/wikiJS/config.yml`

```
⚙️ /var/www/wikiJS/config.yml
```

```
1 db:  
2   type: postgres  
3   host: localhost  
4   port: 5432  
5   user: wikiuser  
6   pass: motdepasse  
7   db: wikijs
```

- (L2) spécifie le type de base de données
- (L3) spécifie quel serveur héberge la base de données
- (L4) Port utilisé pour la base de données
- (L5) Nom d'utilisateur de la base de données
- (L6) Mot de passe de l'utilisateur de la base de données
- (L7) Nom de la base de données

3. Finalement, démarrons Wiki.js :

```
$ node server
```

### 4.5.2 Configuration de Nginx comme Proxy

Nous allons maintenant configurer un Proxy avec Nginx pour signé le wiki et avoir un accès HTTPS.

#### Installation

Nous installons nginx via apt :

```
# apt install nginx
```

#### Configuration du Virtual Host

1. Créons un fichier de configuration :

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

2. Dans ce dernier, ajoutons le contenu suivant :

```

1      server {
2          listen 80;
3          server_name 192.168.110.47;
4          return 301 https://$host$request_uri;
5      }
6
7      server {
8          listen 443 ssl;
9          server_name 192.168.110.47;
10
11         ssl_certificate "/etc/ssl/certs/wiki-js-cert.pem";
12         ssl_certificate_key "/etc/ssl/private/wiki-js-key-no-pass.pem";
13
14         ssl_protocols TLSv1.2 TLSv1.3;
15         ssl_ciphers 'HIGH:!aNULL:!MD5';
16
17         location / {
18             proxy_pass http://192.168.110.21:3000;
19             proxy_http_version 1.1;
20             proxy_set_header Upgrade $http_upgrade;
21             proxy_set_header Connection 'upgrade';
22             proxy_set_header Host $host;
23             proxy_cache_bypass $http_upgrade;
24         }
25     }

```

- (L2) port sur lequel tourne le proxy
- (L3) adresse ip de la machine qui héberge le proxy
- (L4) Redirige toutes les requêtes HTTP vers HTTPS en utilisant une redirection permanente (301).
- (L8) Écoute sur le port 443 (HTTPS) et active SSL.
- (L11) Chemin vers le certificat SSL utilisé pour sécuriser la connexion.
- (L12) Clé privée associée au certificat SSL.
- (L14) Autorise uniquement TLS 1.2 et 1.3 (plus sécurisés).
- (L15) Autorise uniquement TLS 1.2 et 1.3 (plus sécurisés).
- (L18) Redirige les requêtes vers Wiki.js qui tourne sur 192.168.110.21:3000.
- (L19) Utilise HTTP/1.1, nécessaire pour les connexions WebSocket.
- (L20) Gère les WebSockets (indispensable pour certaines fonctionnalités).
- (L21) Indique que la connexion doit être mise à niveau (pour WebSockets).
- (L22) Transmet l'hôte original demandé par le client.
- (L23) Désactive la mise en cache si un Upgrade est demandé (important pour les WebSockets).

3. On active ensuite la configuration :

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

```
# systemctl restart nginx
```

### 4.5.3 Accès interne

Pour accéder 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](#)