

# Livrable 5

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

2025-04-02

## 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 . . . . .	8
4.4.4	Configuration de DoT (DNS over TLS . . . . .	9

# 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 :

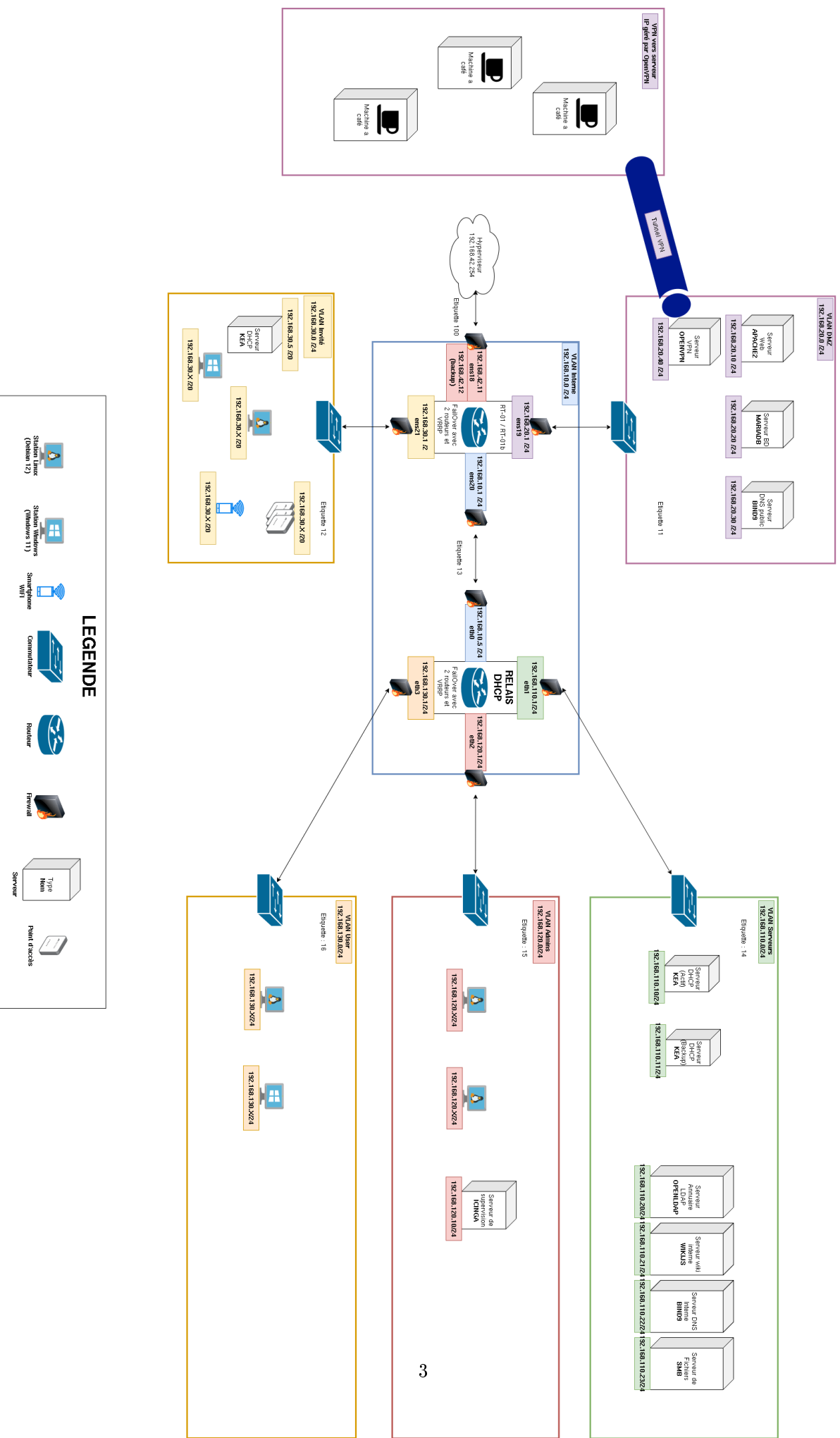
- Divers serveurs d'utilité interne pour les employés et l'équipe informatique;
- Un réseau invité pour permettre à ses fournisseurs d'utiliser du matériel informatique sur place;
- Divers serveurs accessibles en ligne (site Web, serveur DNS public);
- Une meilleure communication entre ses machines à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

On va 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 interface :

```

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
- (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

```
$ 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 à 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

#### 4.3.1 Installation

- Créer 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 restaurer 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`

```

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
24                             - 192.168.120.200"
25                         }
26                     ],
27                     "option-data": [
28                         {
29                             "name": "domain-name-servers",
30                             "data": "192.168.110.22"
31                         },
32                         {
33                             "name": "domain-search",
34                             "data": "bav4.local"
35                         },
36                         {
37                             "name": "routers",
38                             "data": "192.168.120.1"
39                         }
40                     ]
41                 },
42                 [... Ajouter autant de subnet que de sous reseaux sont
43                     concernees]
44             ]
45         }

```

- (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'extérieur, et un serveur DNS Interne, qui va être accessible uniquement depuis l'intérieur, 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 crée 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 appelée "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 désormais créer la zone bav4. Dans **/etc/bind/named.conf.local**, on ajoute donc la zone suivante:



```
🔧 /etc/bind/named.conf.local
```

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

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
```

```
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
```

- (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

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

- (L1) 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
- (L2) 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
1 [...]
2 hosts:  resolve files  dns
3 [...]

```

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

- (a) On va tenter d'utiliser **systemd-resolved**
- (b) Si cela échoue, le système vérifie le fichier local **/etc/hosts**
- (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 queries 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 déjà existantes.

```

</> 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
28 # Redémarrer systemd-resolved pour appliquer les changements
29 echo "Redémarrage de systemd-resolved..."
30 systemctl restart systemd-resolved
31
32 # Tester la résolution DNS
33 echo "Test de résolution DNS avec systemd-resolved..."
34 if resolvectl query google.com | grep -q "google.com"; then
35     echo "Test réussi : la résolution DNS fonctionne"
36 else
37     echo "Echec du test DNS. Vérifie la configuration."
38 fi
39
40 echo "Configuration terminée !"

```

- dziedbzdzn
- AAA

Le script suivant est a esecuter sur la machine bastion, en ayant les fichiers **nsswitch.conf**, **resolve.conf**, **scriptDoTClient.sh**, a chaque connection, on effectuera en plus la commande

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

</> 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
```