Livrable 5

Peran AMBERNY, Clement BARNOUIN, Loucas BURELLIER,
Vladimir KRAINIK–SAUL, Samuel SCHICKE

2025-04-03

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 :

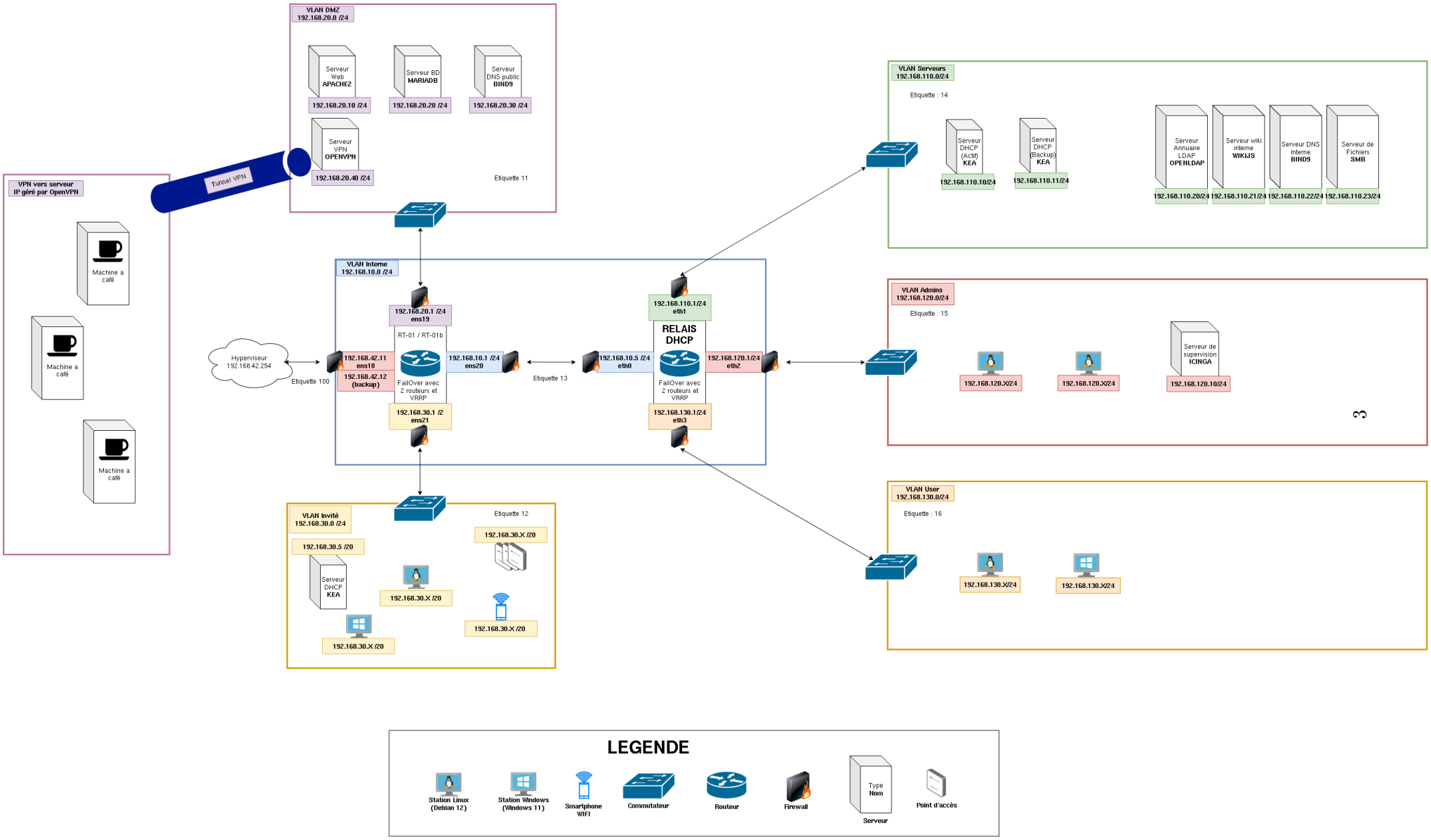
- Divers serveurs d'utilité interne pour les employés et l'équipe informatique;
- Un réseau invité pour permettre à ses fournisseurs d'utiliser du matériel informatique sur place;
- Divers serveurs accessibles en ligne (site Web, serveur DNS public);
- Une meilleure communication entre ses machines à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

Nous allons 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 interfaces :

/etc/network/interfaces

```
# Interface WAN (connectee a Internet)
auto eth0
iface eth0 inet
mtu 1450  # Taille MTU standard
# Interface LAN 1
(reseau interne 192.168.1.0/24)
auto eth1
iface eth1 inet static
    address 192.168.1.1/24
    gateway 192.168.1.254
    # Facultatif, utilise uniquement si ce reseau doit
    sortir par une autre route
    dns-nameservers 192.168.1.1 8.8.8.8
    dns-domain lan1.local
    mtu 1450  # Optimise pour les reseaux
    locaux rapides

# Interface LAN 2
(reseau interne 10.10.0.0/24)
auto eth2
iface eth2 inet static
    address 10.10.0.1/24
    mtu 1450  # Optimise pour un second
    reseau local

# Activation du routage entre les reseaux
post-up echo 1 > /proc/sys/net/ipv4/ip_forward

# Ajout de routes pour permettre aux deux reseaux de communiquer entre eux
post-up ip route add 192.168.1.0/24 via 192.168.1.1 dev eth1
post-up ip route add 10.10.0.0/24 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
– (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 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 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
```

4.2.3 NAT et règles de pare-feu sur un routeur

Le pare-feu et la traduction d'adresses (NAT) sont essentiels au fonctionnement sécurisé du routeur. Nous avons configuré ces éléments à l'aide de **nftables**, qui remplace iptables dans les systèmes Linux récents.

Structure des fichiers : Le fichier principal `/etc/nftables.conf` agit comme point d'entrée pour charger l'ensemble des règles. Son contenu est minimal et se contente d'inclure des fichiers spécialisés :

/etc/nftables.conf

```
1 #!/usr/sbin/nft -f
2
3 flush ruleset
4
5 include "/etc/nftables/ruleset/sets.nft"
6 include "/etc/nftables/ruleset/filter.nft"
7 include "/etc/nftables/ruleset/nat.nft"
8 include "/etc/nftables/ruleset/logging.nft"
```

Organisation modulaire :

- `sets.nft` contient tous les groupes d'adresses IP (DMZ, LAN, invités, etc.).
- `filter.nft` contient les règles de filtrage (input, output, forward).
- `nat.nft` gère la traduction d'adresse (masquerade vers Internet).
- `logging.nft` contient les chaînes pour journaliser proprement les paquets rejetés.

Mécanisme de NAT : Le routeur 1 agit comme passerelle vers Internet. Pour permettre aux machines internes d'accéder à Internet, il faut faire du NAT (masquage) :

nat.nft

```
table ip nat {
    chain prerouting {
        type nat hook prerouting priority dstnat;
        policy accept;
    }

    chain postrouting {
        type nat hook postrouting priority srcnat;
        policy accept;
        oifname "ens18" masquerade
    }
}
```

Ici, toutes les connexions sortant via l'interface WAN `ens18` seront masquées, permettant aux machines internes d'utiliser l'IP publique du routeur.

Filtrage par zones : Le pare-feu distingue les interfaces selon les zones (WAN, DMZ, LAN, invités) et applique une politique stricte de contrôle :

- **INPUT** : seuls le SSH depuis des IP précises et le protocole VRRP sont autorisés.
- **FORWARD** : contrôle précis des flux inter-réseaux.
- **OUTPUT** : le routeur ne génère que le minimum de trafic (DNS, ICMP, VRRP).
- **IPv6** : toutes les connexions IPv6 sont bloquées (table `ip6 filter_ipv6`).

Fichier avec les règles du routeur 1: Ci-dessous, les règles nftables du routeur 1 :

/etc/nftables.config

```
        chain input {
                type filter hook input priority filter; policy drop;
                ip version != 4 drop
                iifname "lo" accept
                ct state established,related accept
                ip protocol vrrp ip daddr 224.0.0.18 accept
                iifname { "ens18", "ens19" } ip saddr @ip_ssh_autorise tcp dport 22
                    ct state new jump log_input_ssh
                jump log_input_unknown
        }

        chain output {
                type filter hook output priority filter; policy drop;
                ct state established,related accept
                ip protocol icmp icmp type echo-request accept
                ip daddr 224.0.0.18 ip protocol vrrp accept
                udp dport 53 ct state new accept
        }

        chain forward {
                type filter hook forward priority filter; policy drop;
                ct state established,related accept
                iifname "ens18" oifname "ens19" ip daddr 192.168.20.10 tcp dport {
                    80, 443 } ct state new accept
                iifname "ens19" oifname "ens20" ip saddr @dns_clients ip daddr
                    192.168.110.22 udp dport 53 ct state new accept
                iifname "ens20" oifname "ens18" ct state new accept
                iifname "ens20" oifname "ens19" ip daddr @dmz_net ct state new accept
                iifname "ens19" oifname "ens18" ip saddr @dmz_net ct state new accept
                iifname "ens19" oifname "ens20" ip saddr @dmz_net ip daddr
                    @internal_net jump log_forward_dmz_to_lan
                iifname "ens19" oifname "ens21" ip saddr @dmz_net ip daddr @guest_net
                     jump log_forward_dmz_to_guest
                iifname "ens21" oifname "ens18" ip saddr @guest_net ct state new
                    accept
                iifname "ens21" ip saddr @guest_net ip daddr 192.168.30.10 udp sport
                    68 udp dport 67 accept
                oifname "ens21" ip saddr 192.168.30.10 ip daddr @guest_net udp sport
                    67 udp dport 68 accept
                iifname "ens21" ip saddr @guest_net ip daddr @guest_net jump
                    log_guest_to_guest_block
                iifname "ens21" ip daddr @all_internal_net jump
                    log_forward_guest_to_lan
                jump log_forward_unknown
        }
}
table ip6 filter_ipv6 {
        chain input {
                type filter hook input priority filter; policy drop;
        }

        chain forward {
                type filter hook forward priority filter; policy drop;
        }

        chain output {
                type filter hook output priority filter; policy drop;
        }
}
```

Log structuré : Toutes les actions de rejet sont journalisées via des chaînes spécifiques (ex. : `log_input_unknown`, `log_forward_dmz_to_lan`) ce qui permet un débogage efficace.

4.2.4 Haute disponibilité avec Keepalived avec le routeur

Afin d'assurer la tolérance aux pannes et la continuité de service réseau, nous avons mis en place un mécanisme de *haute disponibilité* (HA) entre deux routeurs à l'aide de l'outil **Keepalived**. Cette solution

repose sur le protocole **VRRP** (Virtual Router Redundancy Protocol) qui permet de partager une IP virtuelle entre plusieurs machines redondantes.

Principe de fonctionnement : Tout repose sur la configuration d'un fichier : avec Keepalived, on choisit un **routeur actif (MASTER)** et un autre de secours (**BACKUP**). Le routeur actif gère l'IP virtuelle (VIP) utilisée par les clients comme passerelle. En cas de défaillance du MASTER (ex. perte d'interface, arrêt du service), le BACKUP avec la priorité suivante prend automatiquement le relais.

Architecture mise en place : Deux routeurs sont configurés :

- **RT-01-Master** : MASTER avec une priorité élevée, détient la VIP
- **RT-01-Backup** : BACKUP avec une priorité inférieure, prend le relais si le MASTER tombe

Une IP virtuelle est utilisée comme passerelle par interface, par exemple : `192.168.30.1/24` pour le LAN Invité.

Configuration de Keepalived : Sur chaque routeur, Keepalived est installé via :

```
# apt install keepalived
```

Le fichier principal de configuration est `/etc/keepalived/keepalived.conf`. Voici un exemple pour l'interface `ens21` (réseau invité) :

⚙/etc/keepalived/keepalived.conf

```
# Script de verification de la sante de nftables
vrrp_script check_nft {
    script "/etc/keepalived/scripts/check_nft.sh"
    interval 3
    fall 2
    rise 3
}

vrrp_instance VI_INVITE {
    state MASTER
    interface ens21
    virtual_router_id 40
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass "mdp"
    }
    virtual_ipaddress {
        192.168.30.1/24
    }
    track_interface {
        ens21
    }
    track_script {
        check_nft
    }
    notify_master "/etc/keepalived/scripts/notify_vrrp.sh MASTER ens21"
    notify_backup "/etc/keepalived/scripts/notify_vrrp.sh BACKUP ens21"
    notify_fault  "/etc/keepalived/scripts/notify_vrrp.sh FAULT ens21"
}
```

Analyse de la configuration :

- **L2** : Déclare un bloc de supervision `vrrp_script` nommé `check_nft`.
- **L3** : Définit le chemin du script qui vérifie si le pare-feu nftables est correctement chargé.
- **L4** : Le script est exécuté toutes les 3 secondes.

- **L5** : Si deux échecs consécutifs ont lieu, on considère que l'état est défaillant.
- **L6** : Il faut trois vérifications réussies pour revenir à un état sain.
- **L10** : Ce routeur démarre comme MASTER.
- **L11** : Interface réseau concernée (`ens21`).
- **L12** : Identifiant unique de l'instance pour éviter les collisions.
- **L13** : Priorité élevée (100), donc ce routeur est favorisé.
- **L15–L18** : Bloc d'authentification VRRP avec mot de passe partagé.
- **L19–L21** : Attribution de l'adresse virtuelle `192.168.30.1` à cette interface.
- **L22–L24** : Vérifie que l'interface réseau est active.
- **L25–L27** : Supervision du bon fonctionnement de nftables grâce au script `check_nft`.
- **L28–L30** : Déclenche des scripts selon le changement d'état VRRP (MASTER, BACKUP, FAULT).

Sur le routeur BACKUP, seul le champ `state MASTER` devient `state BACKUP`, et la `priority` est diminuée (par exemple `priority 90`). Une instance VRRP est configurée par interface réseau.

Fichiers de scripts et de journalisation : Les changements d'état (MASTER, BACKUP, FAULT) déclenchent des scripts dans `/etc/keepalived/scripts/`. Ces scripts journalisent les transitions dans `journalctl` au format JSON lisible par un SIEM :

- `notify_vrrp.sh` : reçoit en argument l'état et l'interface, et logue un message structuré.

</> /etc/keepalived/scripts/notify_vrrp.sh

```
#!/bin/bash

STATE="$1"
INTERFACE="$2"
HOST=$(hostname)
TIMESTAMP=$(date -Is)

# Verifie que les deux arguments sont bien fournis
if [ -z "$STATE" ] || [ -z "$INTERFACE" ]; then
  echo "[ERROR] Usage: $0 <STATE> <INTERFACE>" >&2
  exit 1
fi

# Recupere la VIP secondaire associee a l'interface passee
VIP=$(ip -o addr show dev "$INTERFACE" | awk '/secondary/ {print $4}' | head -n1
    )

# Log structure en JSON dans journalctl
logger -t KEEPALIVED \
  "{\"event\":\"VRRP\",\"router\":\"$HOST\",\"state\":\"$STATE\",\"interface\":\
      "$INTERFACE\",\"vip\":\"$VIP\",\"timestamp\":\"$TIMESTAMP\"}"
```

- `check_nft.sh` : vérifie que `nft list ruleset` renvoie un jeu de règles non vide.

Avantages obtenus :

- Continuité d'accès au réseau même si le routeur principal tombe
- Pas de changement d'adresse IP pour les clients (VIP fixe)
- Récupération automatique sans intervention manuelle
- Logs compatibles avec Wazuh

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 :

⚙/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]
        ]
}
}
```

Détail de la configuration

- (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)

4.3.3 Ajout d'un relais DHCP

Dans notre réseau, il nécessaire de permettre à des clients situés dans des sous-réseaux différents d'obtenir une configuration IP automatique à partir d'un serveur DHCP centralisé. Pour cela, nous avons mis en place un relais DHCP (*DHCP relay*) sur le routeur (routeur 2 dans notre cas) assurant l'interconnexion des réseaux.

Configuration utilisée : Le paquet utilisé est `isc-dhcp-relay`, installé sur le routeur. La commande d'installation est la suivante :

```
# apt install isc-dhcp-relay
```

Il faut ensuite adapté le fichier de configuration **`/etc/default/isc-dhcp-relay`** avec les paramètres suivants :

- **SERVERS** : adresse IP du serveur DHCP, par exemple `192.168.110.11`
- **INTERFACES** : interfaces du routeur à écouter (ex. `ens19, ens20, ens21`)

⚙/etc/default/isc-dhcp-relay

```
1 # What servers should the DHCP relay forward requests to?
2 SERVERS="192.168.110.11"
3
4 # On what interfaces should the DHCP relay (dhrelay) serve DHCP requests?
5 INTERFACES="ens19 ens20 ens21"
```

- (L2) Adresse IP du serveur DHCP Kea
- (L5) Ici les interfaces sur lequels le serveur DHCP va pour être relayer.

Résultat attendu : Grâce à cette configuration, les clients présents dans les sous-réseaux reliés à `ens19`, `ens20` et `ens21` peuvent désormais obtenir dynamiquement leur adresse IP à partir du serveur DHCP, même s'ils ne sont pas directement connectés à lui.

4.4 Serveur DNS

Afin de pouvoir lier un nom a 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 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. Nous allons commencer par créer le DNS privé puis nous le cloneront 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 appellée "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 desormais créer la zone bav4.local, c'est cette derniere qui contiendra les enregistrements dns (par exemple wiki.bav4.local). Dans **/etc/bind/named.conf.local**, nous ajoutons 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  };
```

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 :

/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
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'hote> 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 requetes DNS, nous allons utiliser DoT (DNS over TLS) qui permet de chiffrer nos requetes DNS. Nous utiliseront par la suite **systemd-resolved** du coté 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 a **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 réalités a 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 déja existantes.

Script a éxécuter sur les machines client

</> scriptDoTClient.sh

```
#!/bin/

# 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
```

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) 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, 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

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

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

- (L2) specifie le type de base de données
- (L3) specifie quel serveur heberge 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. Nous allons d'abord générer le certificats est la clé pour pouvoir signer le wiki

```
# sudo mkdir -p /etc/ssl/private /etc/ssl/certs sudo openssl req -x509 -nodes
-days 365 -newkey rsa:4096  -keyout /etc/ssl/private/wiki-js-key-no-pass.pem
-out /etc/ssl/certs/wiki-js-cert.pem
```

2. Créons un fichier de configuration :

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

3. Dans ce dernier, ajoutons le contenu suivant :

/etc/nginx/sites-available/wiki

```
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 heberge 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).

4. 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 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

4.6 SIEM

4.6.1 Installation de Wazuh

1. Ajout du dépôt Wazuh

```
# curl -sO <https://packages.wazuh.com/key/GPG-KEY-WAZUH>
```

Cette commande utilise curl pour télécharger la clé GPG publique de Wazuh, ce qui permet au système de vérifier l'authenticité des paquets de Wazuh lors de leur installation.

```
# apt-key add GPG-KEY-WAZUH
```

Ajoute la clé GPG téléchargée au gestionnaire de paquets APT pour valider les paquets provenant du dépôt Wazuh.

```
# echo 'deb <https://packages.wazuh.com/4.x/apt/> stable main' | tee
/etc/apt/sources.list.d/wazuh.list
```

Ajoute le dépôt Wazuh à la liste des sources d'APT, ce qui permet au système d'installer des paquets depuis ce dépôt.

```
# apt update
```

2. Installation du serveur Wazuh

```
# apt install wazuh-manager
```

```
# systemctl enable -now wazuh-manager
```

Active et démarre immédiatement le service wazuh-manager pour qu'il fonctionne au démarrage du système.

4.6.2 Installation de l'Elastic Stack

1. Installation d'Elasticsearch

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
```

Télécharge et ajoute la clé GPG de l'Elastic Stack afin de valider les paquets d'Elasticsearch.

```
# echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" | tee -a
/etc/apt/sources.list.d/elastic-8.x.list
```

Ajoute le dépôt officiel d'Elasticsearch à la liste des sources APT pour que le système puisse télécharger et installer Elasticsearch

```
# apt update && apt install elasticsearch
```

```
# systemctl enable -now elasticsearch
```

2. Installation de Kibana

```
# apt install kibana
```

```
# systemctl enable -now kibana
```

4.6.3 Configuration de Filebeat sur le routeur

On configure Filebeat sur une machine pour pouvoir envoyer des informations(logs, traffic réseaux...) au SIEM.

1. Installation de Filebeat

```
# apt install filebeat
```

2. Configuration de Filebeat pour Elasticsearch et Kibana

⚙/etc/filebeat/filebeat.yml

```
1  filebeat.inputs:
2  - type: log
3    enabled: true
4    paths:
5      - /var/log/*.log
6
7  output.elasticsearch:
8    hosts: ["<http://192.168.110.23:9200>"]
9    username: "elastic"
10   password: "enamont"
11 setup.kibana:
12   host: "<http://192.168.110.23:5601>"
```

   - (L1) Specification des fichiers a envoyer au SIEM
   - (L2) Type des fichiers envoyées
   - (L4) Chemin vers les fichiers à envoyées
   - (L8) URL vers l'interface elasticsearch
   - (L9) username utilisé pour se connecter à elasticsearch
   - (L10) Mot de passe utilisé pour se connecter à elasticsearch
   - (L12) URL pour acceder à kibana

3. Activation du module Netflow

```
# filebeat modules enable netflow
```

   Module qui permet d'enregistrer le traffic réseaux

```
# systemctl restart filebeat
```