# BTS SIO 2026

| | |
|---|---|
| ☒ | **Administration des systèmes et des réseaux (E6 – SISR)** |
| ☐ | **Conception et développement d'applications (E6 – SLAM)** |

## PAGE DE PRÉSENTATION DU DOSSIER

**N° d'inscription[1] :** | 0 | 2 | 5 | 4 | 2 | 6 | 1 | 0 | 6 | 3 | 1 |

**NOM :** Jung

**PRÉNOM :** Jean-Marie

| | |
|---|---|
| **Date de passage[2] : ..... /..... / 2026** | **Heure de passage[2] : .....h.....** |

| **CATÉGORIE CANDIDAT[1] (UNE CASE À COCHER)** | |
|---|---|
| ☒ **Scolaire** | ☐ **Ex-scolaire** |
| ☐ **Apprenti** | ☐ **Ex-apprenti** |
| ☐ **Formation professionnelle continue** | ☐ **Ex-formation professionnelle continue** |
| ☐ **Expérience professionnelle** | |

*[1] Informations communiquées sur votre confirmation d'inscription*
*[2] Informations à ne pas compléter*

**SIEC – maison des examens**
7 rue Ernest Renan
94749 ARCUEIL CEDEX
Tél : 01 49 12 23 00
www.siec.education.fr

**Tampon de L'établissement**

# Rapport de Situation N°1

**Jung**
**Jean-Marie**

**E6**

—

**BTS SIO SISR**

—

**Déploiement d'une Infrastructure Web Haute Disponibilité**

# Table des matières

1. Contexte.....4
2. Client.....4
3. Réseau de l'entreprise.....5
3.1 Avant.....5
3.2 Après.....6
4. Cahier des charges.....7
4.1 Définition des besoins.....7
4.2 Solution proposée.....7
5. Réalisation.....8
5.1 Installation et configuration d'Apache HTTPD.....8
5.2 Configuration HTTPS (SSL/TLS).....12
5.3 Load Balancing avec HAProxy.....13
5.4 Haute Disponibilité avec Keepalived.....17
5.5 Synchronisation des serveurs avec lsyncd.....21
5.6 Installation et configuration de Zabbix.....27
5.7 Configuration des agents Zabbix.....31
6. Conclusion.....33
7. Glossaire.....34

# 1. Contexte

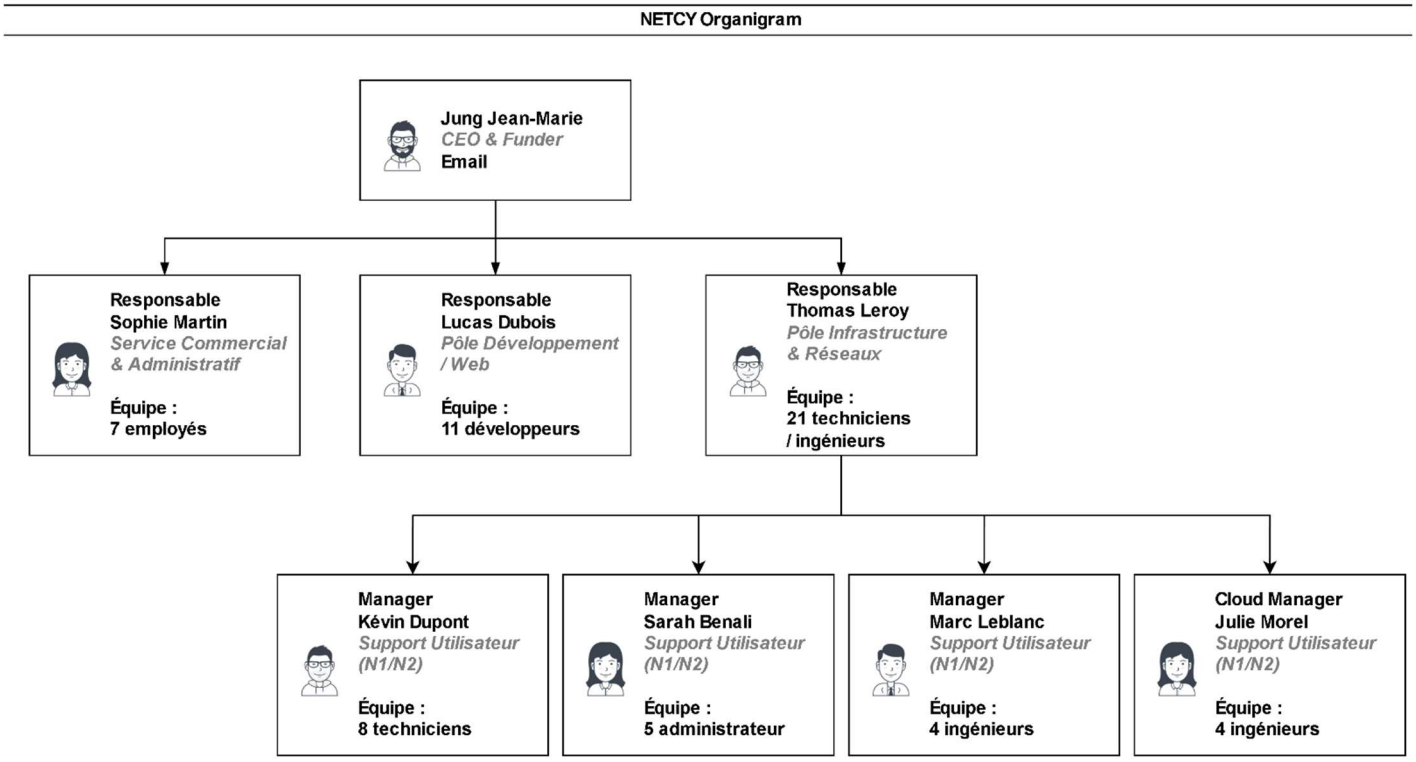
L'entreprise dispose d'un site web interne utilisé par l'ensemble de ses collaborateurs. Ce service hébergé sur un serveur unique présente un point de défaillance critique : toute panne du serveur entraîne une interruption totale du service, impactant directement la productivité des équipes.

Par ailleurs, aucun outil de supervision n'est en place pour détecter les anomalies en temps réel, ce qui contraint les équipes IT à intervenir uniquement en mode réactif, souvent après que les utilisateurs aient signalé une panne.

Dans ce contexte, la direction informatique a décidé de faire évoluer l'infrastructure web vers une architecture haute disponibilité, avec répartition de charge, synchronisation automatique des fichiers entre serveurs et supervision centralisée via Zabbix.

# 2. Client

Entreprise : NETCY
Secteur : Services / Informatique interne
Infrastructure : Serveur web unique, sans redondance
Problématique : Disponibilité insuffisante, absence de supervision
Demande : Infrastructure HA + load balancing + supervision Zabbix



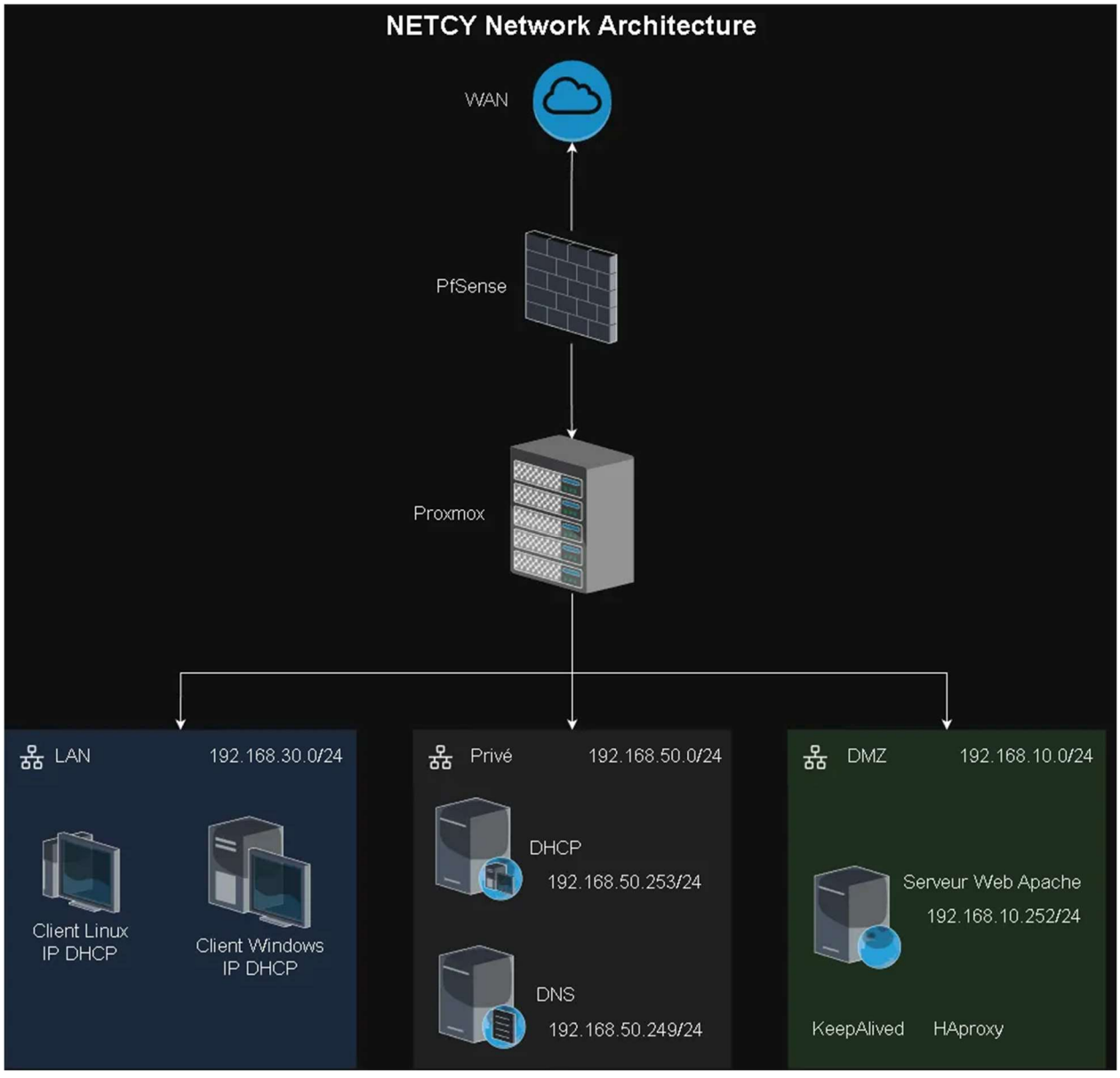
# 3. Réseau de l’entreprise

## 3.1 Avant

L'infrastructure initiale repose sur un seul serveur web Apache accessible directement par les clients du réseau interne. Il n'existe aucun mécanisme de redondance, aucune répartition de charge, et aucun outil de surveillance des équipements.

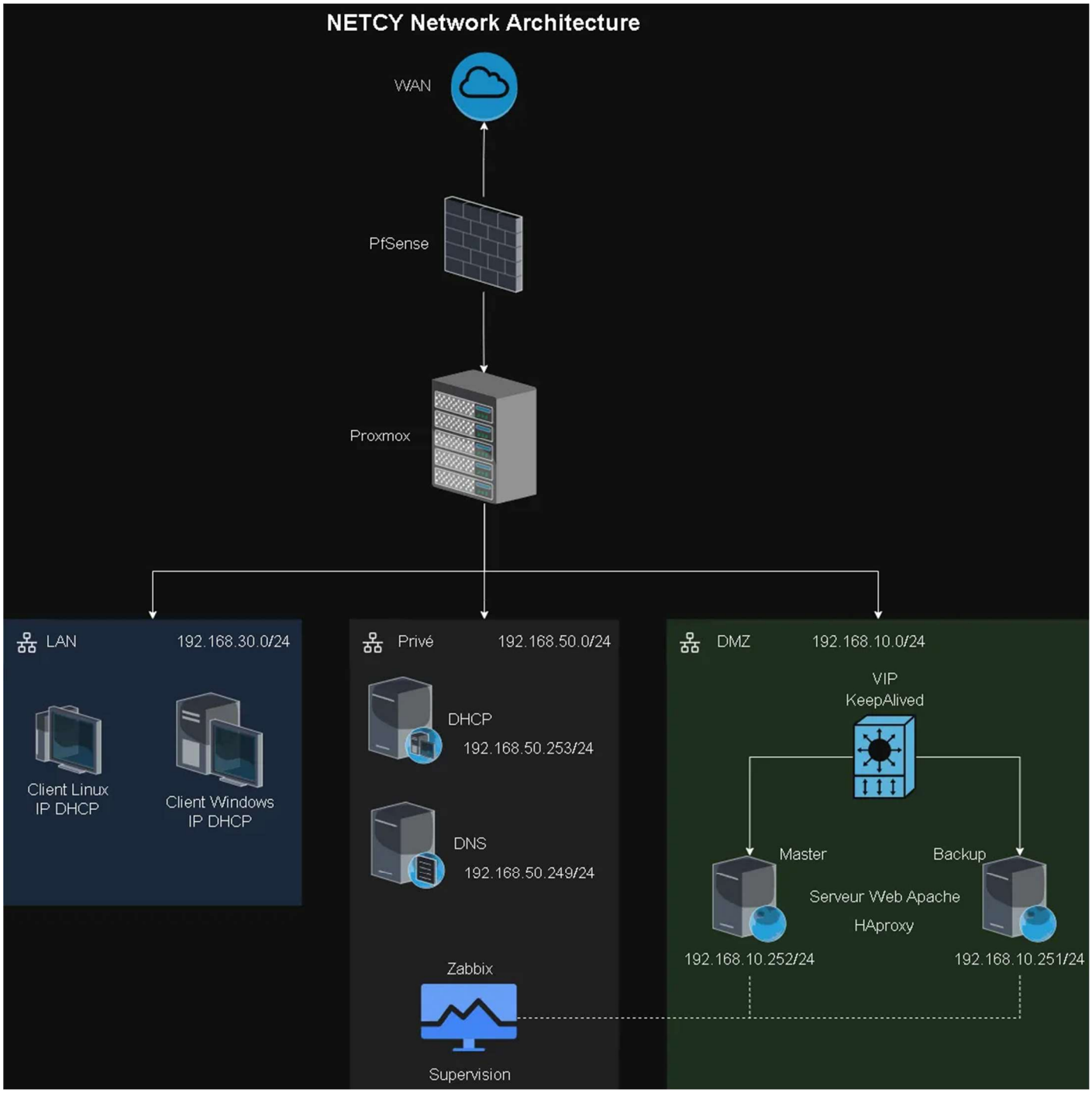
Limites identifiées :

- Point de défaillance unique (SPOF — Single Point of Failure)
- Aucune tolérance aux pannes
- Aucune supervision des services et équipements
- Charge non répartie en cas de montée en charge



## 3.2 Après

L'infrastructure cible est composée de deux serveurs web Apache en haute disponibilité, d'un répartiteur de charge HAProxy, d'une adresse IP virtuelle gérée par Keepalived, d'un mécanisme de synchronisation des fichiers par lsyncd, et d'un serveur de supervision Zabbix.




Plan d'adressage :

| Rôle | Nom | Adresse IP |
|---|---|---|
| IP Virtuelle (VIP) | — | 192.168.10.250 |
| Serveur MASTER (Web1 + HAProxy) | Web1 | 192.168.10.252 |
| Serveur BACKUP (Web2 + HAProxy) | Web2 | 192.168.10.251 |
| Serveur Zabbix (supervision) | Zabbix | À définir |

# 4. Cahier des charges

## 4.1 Définition des besoins

| N° | Besoin | Priorité |
|---|---|---|
| 1 | Mettre en place deux serveurs web Apache redondants | Haute |
| 2 | Configurer HTTPS (SSL/TLS) sur les serveurs web | Haute |
| 3 | Mettre en œuvre la répartition de charge (HAProxy) | Haute |
| 4 | Assurer la haute disponibilité via une IP virtuelle (Keepalived) | Haute |
| 5 | Synchroniser automatiquement les fichiers web (lsyncd) | Moyenne |
| 6 | Déployer un serveur de supervision Zabbix | Haute |
| 7 | Superviser tous les serveurs via des agents Zabbix | Haute |

## 4.2 Solution proposée

| Composant | Technologie retenue | Justification |
|---|---|---|
| Serveur web | Apache HTTPD | Open-source, standard entreprise |
| Chiffrement | OpenSSL | Sécurisation HTTP → HTTPS |
| Répartiteur de charge | HAProxy | Round-robin, health checks |
| Haute disponibilité | Keepalived (VRRP) | Basculement automatique via VIP |
| Synchronisation | lsyncd + rsync | Temps réel via inotify |
| Supervision | Zabbix 6.4 | Solution complète, agents Linux |
| Base de données | MariaDB | Moteur de BDD pour Zabbix |

# 5. Réalisation

## 5.1 Installation et configuration d'Apache HTTPD

Cette étape est à répéter sur les deux serveurs (web1 et web2).

### Installation des packages

Commande :

```
dnf install httpd mod_ssl -y
```

Explication :

- httpd : paquet du serveur web Apache
- mod_ssl : module SSL/TLS nécessaire pour activer HTTPS

Vérification de l'installation :

```
rpm -qa | grep httpd
```

Explication :

- rpm -qa : liste tous les paquets RPM installés (-q = query, -a = all)
- | grep httpd: filtre la sortie pour n'afficher que les paquets "httpd"

**Résultat attendu :**

```
httpd-2.4.57-5.el9.x86_64
httpd-tools-2.4.57-5.el9.x86_64
httpd-filesystem-2.4.57-5.el9.noarch
mod_ssl-2.4.57-5.el9.x86_64
```

### Configuration du serveur Apache

Commande :

```
vi /etc/httpd/conf/httpd.conf
```

Explication :

- vi : éditeur de texte en mode terminal
- /etc/httpd/conf/httpd.conf : fichier de configuration principal d'Apache

Localiser les lignes suivantes et les modifier :

Avant :

```
Listen 80
#ServerName www.example.com:80
```

Après :

```
    Listen *:80
    ServerName web:80
```

Explication :

- Listen *:80 : Apache écoute sur toutes les interfaces réseau sur le port 80.

```
Le * remplace une IP spécifique pour permettre l'écoute multi-interfaces.

- ServerName : définit le nom canonique du serveur.
Doit correspondre au hostname configuré dans /etc/sysconfig/network.
```

## Démarrage et activation du service

Commandes :

```
    systemctl start httpd.service
    systemctl enable httpd.service
```

Explication :

- start : démarre immédiatement le service Apache
- enable : crée un lien symbolique systemd pour que le service démarre automatiquement

## Vérification du service

Commande :

```
    ss -atnp | grep httpd
```

Explication :

- ss : outil de statistiques sur les sockets réseau (remplace netstat)
- -a : affiche tous les sockets (connectés et en écoute)
- -t : filtre sur le protocole TCP uniquement
- -n : affiche les numéros de port au lieu de leur nom (80 au lieu de "http")
- -p : affiche le nom du processus associé à chaque socket

**Résultat attendu :**

```
    LISTEN 0  511  0.0.0.0:80   0.0.0.0:*  users:(("httpd",pid=1234,fd=4))
    LISTEN 0  511  0.0.0.0:443  0.0.0.0:*  users:(("httpd",pid=1234,fd=6))
```

Les ports 80 (HTTP) et 443 (HTTPS) doivent apparaître en état LISTEN.

## Ouverture des ports dans iptables

Commande :

```
vi /etc/sysconfig/iptables
```

Ajouter les règles suivantes :

-I INPUT 2 -p tcp --dport 80 -j ACCEPT

-I INPUT 3 -p tcp --dport 443 -j ACCEPT

**Explication :**

- -I INPUT 2 : insère une règle en position 2 de la chaîne INPUT
- -p tcp : s'applique au protocole TCP
- --dport 80 : cible les connexions à destination du port 80 (HTTP)
- --dport 443 : cible les connexions à destination du port 443 (HTTPS)
- -j ACCEPT : action = accepter le paquet

Appliquer les changements :

```
systemctl reload iptables
```

**Explication :**

- reload : recharge la configuration du pare-feu SANS interrompre les connexions existantes (contrairement à restart)

Vérifier les règles :

```
iptables -nvL
```

**Explication :**

- -n : affiche les adresses IP et ports en format numérique
- -v : mode verbose (compteurs de paquets et octets)
- -L : liste les règles de toutes les chaînes

**Résultat attendu :**

```
pkts bytes target  prot opt in out source      destination
 81  5840 ACCEPT  all  --  *   *  0.0.0.0/0  0.0.0.0/0  state RELATED,ESTABLISHED
 0     0 ACCEPT  tcp  --  *   *  0.0.0.0/0  0.0.0.0/0  tcp dpt:80     ← IMPORTANT
 0     0 ACCEPT  tcp  --  *   *  0.0.0.0/0  0.0.0.0/0  tcp dpt:443    ← IMPORTANT
 0     0 ACCEPT  icmp --  *   *  0.0.0.0/0  0.0.0.0/0
 0     0 ACCEPT  all  --  lo  *  0.0.0.0/0  0.0.0.0/0
 0     0 ACCEPT  tcp  --  *   *  0.0.0.0/0  0.0.0.0/0  state NEW tcp dpt:22
 0     0 REJECT  all  --  *   *  0.0.0.0/0  0.0.0.0/0  reject-with icmp-host-prohibited
```

## Création de la page d'accueil

Commande :

```
vi /var/www/html/index.html
```

**Explication :**

- /var/www/html/ : répertoire racine par défaut d'Apache (DocumentRoot)
- index.html : fichier chargé automatiquement lorsqu'un client accède à la racine du site

Attribuer les droits au répertoire web :

```
chown apache.apache -R /var/www/html
```

**Explication :**

- chown apache.apache : définit l'utilisateur ET le groupe propriétaires comme "apache"

```
 (utilisateur système créé lors de l'installation d'Apache)
-R : récursif, applique le changement à tous les fichiers et sous-dossiers
- /var/www/html : répertoire cible
```

Vérification finale :

```
curl 192.168.10.251
```

**Explication :**

- curl : outil en ligne de commande pour effectuer des requêtes HTTP
- Le contenu de la page index.html doit s'afficher dans le terminal

**Résultat attendu :**

```
cc (ou le contenu de votre page index.html)
```

# 5.2 Configuration HTTPS (SSL/TLS)

HTTPS chiffre les échanges entre le client et le serveur, au niveau de la couche transport (protocole TLS), sur le port 443.

### Génération d'un certificat auto-signé

Commande :

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/pki/tls/private/apache.key \
-out /etc/pki/tls/certs/apache.crt
```

**Explication :**

- openssl req : outil OpenSSL pour gérer les demandes de certificat
- -x509 : génère directement un certificat auto-signé
- -node : la clé privée n'est PAS chiffrée par un mot de passe, ce qui permet à Apache de démarrer sans intervention manuelle
- -days 365 : durée de validité du certificat = 365 jours
- -newkey rsa:2048 : génère une nouvelle paire de clés RSA de 2048 bits
- -keyout ...apache.key : chemin de sauvegarde de la clé privée
- -out ...apache.crt : chemin de sauvegarde du certificat public

⚠ ATTENTION : En environnement de production, un certificat signé par une CA reconnue (ex : Let's Encrypt) est indispensable.

### Configuration Apache pour HTTPS

Commande :

```
vi /etc/httpd/conf.d/ssl.conf
```

Localiser et modifier les lignes :

```
SSLCertificateFile    /etc/pki/tls/certs/apache.crt
SSLCertificateKeyFile /etc/pki/tls/private/apache.key
```

**Explication :**

- SSLCertificateFile : chemin vers le certificat public (.crt) envoyé au navigateur
- SSLCertificateKeyFile : chemin vers la clé privée (.key) utilisée pour le chiffrement.

⚠ ATTENTION : Ce fichier contenant la clé privée ne doit jamais être partagé.

Redémarrer Apache :

```
systemctl restart httpd
```

**Explication :**

- restart : stoppe et relance complètement le service pour appliquer les nouvelles directives SSL

Vérification HTTPS :

```
curl -k https://192.168.10.251
```

**Explication :**

- https:// : force l'utilisation du protocole HTTPS
- -k : ignore les erreurs de validation du certificat (nécessaire pour les certificats auto-signés non reconnus par une CA)

**Résultat attendu :**

```
cc (ou le contenu de votre page index.html)
```

# 5.3 Load Balancing avec HAProxy

HAProxy est un répartiteur de charge (reverse proxy) qui distribue les requêtes HTTP entre plusieurs serveurs backends selon un algorithme défini.

Schéma :

**Client] → [HAProxy :80] → round-robin → [Web1 :8080] ou [Web2 :8080]**

### Préparation des serveurs Apache (port 8080)

Pour que HAProxy prenne le contrôle du port 80, Apache doit écouter sur le port 8080. Cette opération est à réaliser sur CHAQUE serveur web.

Commande :

```
vi /etc/httpd/conf/httpd.conf
```

Avant :

```
Listen *:80
ServerName web:80
```

Après :

```
Listen *:8080
ServerName web:8080
```

**Explication :**

- Le port 80 sera désormais utilisé par HAProxy
- Apache écoute sur 8080 pour rester accessible en tant que backend
- Cette modification libère le port 80 pour HAProxy

Vérification syntaxique de la configuration Apache :

```
httpd -S
```

**Explication :**

- httpd -S : vérifie la syntaxe ET affiche la configuration des Virtual Hosts sans démarrer le service. Permet de détecter les erreurs avant un redémarrage.

**Résultat attendu :**

```
VirtualHost configuration:
*:8080   web (/etc/httpd/conf/httpd.conf:42)
Syntax OK
```

Redémarrage en mode gracieux :

```
httpd -k graceful
```

**Explication :**

- graceful : redémarre Apache SANS couper les connexions en cours. Les workers actifs terminent leurs requêtes avant de se réinitialiser.
- Préférable en production pour une interruption de service nulle.

## Ouverture du port 8080

Commande :

```
vi /etc/sysconfig/iptables
```

Ajouter :

-I INPUT 1 -p tcp --dport 8080 -j ACCEPT

**Explication :**

- -I INPUT 1 : insère la règle en première position, avant toute autre règle
- --dport 8080: autorise les connexions TCP entrantes sur le port 8080 (HAProxy → Apache)

Appliquer et vérifier :

```
systemctl reload iptables
iptables -nvL | grep 8080
```

**Résultat attendu :**

```
0  0  ACCEPT  tcp  --  *  *  0.0.0.0/0  0.0.0.0/0  tcp dpt:8080
```

## Installation HAProxy

Commande :

```
dnf install haproxy -y
```

## Configuration HAProxy

Commande :

```
vi /etc/haproxy/haproxy.cfg
```

Contenu de la configuration :

```
global
    log /dev/log local0
    maxconn 2000
    daemon

defaults
    mode http
    timeout connect 5s
    timeout client 50s
    timeout server 50s

frontend http_front
    bind *:80
    default_backend web_back

backend web_back
    balance roundrobin
    option httpchk
    server web1 192.168.10.252:8080 check
    server web2 192.168.10.251:8080 check
```

Explication détaillée :

Section global :

- log /dev/log local0 : envoie les logs HAProxy au système syslog local
- maxconn 2000 : limite le nombre de connexions simultanées à 2000
- daemon : lance HAProxy en arrière-plan (mode démon)

Section defaults :

- mode http : HAProxy opère en mode HTTP (couche 7), ce qui permet d'inspecter et de manipuler les requêtes
- - timeout connect 5s : délai maximum pour établir une connexion avec un backend
- - timeout client 50s : délai d'inactivité maximal côté client
- - timeout server 50s : délai d'inactivité maximal côté serveur backend

Section frontend http_front :

- bind *:80 : HAProxy écoute sur toutes les interfaces, port 80
- default_backend : redirige tout le trafic entrant vers "web_back"

Section backend web_back :

- balance roundrobin : algorithme round-robin — chaque nouvelle requête est envoyée alternativement à web1, web2, web1, web2...
- option httpchk : active les health checks HTTP : HAProxy vérifie périodiquement que chaque serveur répond. Un serveur qui ne répond plus est automatiquement retiré du pool.
- server web1 ...check : déclare le serveur web1 à l'IP 192.168.10.252 sur le port 8080, avec surveillance active
- server web2 ...check : idem pour web2 à l'IP 192.168.10.251

## Démarrage HAProxy

Commandes :

```
systemctl enable haproxy --now
systemctl status haproxy
```

**Explication :**

- --now : combinaison de enable + start en une seule commande
- status : affiche l'état courant du service (actif/inactif, PID, logs récents)

**Résultat attendu :**

```
haproxy.service - HAProxy Load Balancer
Active: active (running)
Status: "Ready."
```

### Test du Load Balancing

Commande :

```
curl 192.168.10.251
```

En répétant plusieurs fois cette commande, les réponses doivent alterner entre les deux serveurs selon l'algorithme round-robin.

## 5.4 Haute Disponibilité avec Keepalived

Keepalived implémente le protocole VRRP (Virtual Router Redundancy Protocol) pour créer une adresse IP virtuelle (VIP) partagée entre plusieurs serveurs.

Principe de fonctionnement :

- Le serveur MASTER (priorité 100) détient la VIP en temps normal
- Le serveur BACKUP (priorité 90) surveille le MASTER
- En cas de panne du MASTER, le BACKUP prend la VIP en moins de 2 secondes
- Lorsque le MASTER redémarre, il peut reprendre la VIP (selon la config nopreempt)

### Installation sur les deux serveurs

Commande :

```
dnf install keepalived -y
```

### Configuration du serveur MASTER — web1 (192.168.10.252)

Vider le fichier de configuration par défaut :

```
echo > /etc/keepalived/keepalived.conf
```

**Explication :**

- echo > fichier : vide le fichier pour repartir d'une base propre

Éditer le fichier :

```
vi /etc/keepalived/keepalived.conf
```

Contenu de la configuration MASTER :

```
global_defs {
   router_id WEB1
}

vrrp_script chk_haproxy {
    script "pidof haproxy"
    interval 2
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass 1234
    }

    virtual_ipaddress {
        192.168.10.250
    }

    track_script {
        chk_haproxy
    }
}
```

Explication détaillée :

global_defs :

- router_id WEB1 : identifiant unique de ce nœud dans l'infrastructure VRRP

vrrp_script chk_haproxy :

- script "pidof haproxy": commande exécutée périodiquement. Retourne le PID de haproxy (code 0) ou rien si haproxy est arrêté (code 1).
- interval 2 : Vérification toutes les 2 secondes

vrrp_instance VI_1 :

- state MASTER : ce serveur est le MASTER initial (porteur de la VIP)
- interface eth0 : interface réseau sur laquelle la VIP sera attribuée
- virtual_router_id 51 : identifiant du groupe VRRP (identique sur MASTER et BACKUP)
- priority 100 : priorité du MASTER (supérieure à celle du BACKUP = 90)
- advert_int 1 : intervalle d'envoi des heartbeats VRRP = 1 seconde
- auth_pass 1234 : authentification entre MASTER et BACKUP
- virtual_ipaddress : adresse IP virtuelle partagée (VIP) — utilisée par les clients
- track_script : si haproxy s'arrête, la priorité baisse → basculement déclenché

Démarrer le service :

```
systemctl enable keepalived --now
systemctl start keepalived --now
```

## Configuration du serveur BACKUP — web2 (192.168.10.251)

Commande :

```
vi /etc/keepalived/keepalived.conf
```

Contenu de la configuration BACKUP :

```
global_defs {
   router_id WEB2
}

vrrp_script chk_haproxy {
    script "pidof haproxy"
    interval 2
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 90
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass 1234
    }

    virtual_ipaddress {
        192.168.10.250
    }

    track_script {
        chk_haproxy
    }
}
```

Différences par rapport au MASTER :

- router_id WEB2 : identifiant distinct
- state BACKUP : rôle initial = passif, en attente
- priority 90 : priorité inférieure au MASTER (100 > 90 → le MASTER conserve la VIP)

Démarrer le service :

```
systemctl enable keepalived --now
systemctl start keepalived --now
```

## Vérification

Sur le serveur MASTER, afficher les adresses IP :

```
ip a
```

**Explication :**

- ip a (ou ip address) : affiche toutes les interfaces réseau et leurs adresses IP configurées

**Résultat attendu :**

```
inet 192.168.10.252/24 brd 192.168.10.255 scope global eth0
inet 192.168.10.250/32 scope global eth0 ← VIP présente sur le MASTER
```

Test de basculement :

1. Arrêter Keepalived sur le MASTER : systemctl stop keepalived
2. Sur le BACKUP, vérifier que la VIP apparaît : ip a

**Résultat attendu :**

```
inet 192.168.10.251/24 brd 192.168.10.255 scope global eth0
inet 192.168.10.250/32 scope global eth0 ← VIP basculée sur le BACKUP
```

# 5.5 Synchronisation des serveurs avec lsyncd

lsyncd (Live Syncing Daemon) surveille les modifications de fichiers en temps réel via inotify et les synchronise vers les serveurs cibles via rsync.

### Installation

Commande :

```
dnf install lsyncd -y
```

### Génération et échange des clés SSH

La synchronisation automatique nécessite une authentification sans mot de passe entre serveurs.

Sur le serveur 1 (web1 — 192.168.10.252) :

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.10.251
```

**Explication :**

- ssh-keygen : génère une paire de clés SSH (publique + privée)
- -t rsa : algorithme RSA
- -b 4096 : taille de la clé = 4096 bits (robustesse accrue)
- -f ~/.ssh/id_rsa: emplacement de la clé privée générée
- -N "" : passphrase vide (aucun mot de passe) pour permettre l'automatisation
- ssh-copy-id : copie la clé publique vers le serveur distant dans authorized_keys

Sur le serveur 2 (web2 — 192.168.10.251) :

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.10.252
```

### Configuration lsyncd

Créer le répertoire de configuration :

```
mkdir -p /etc/lsyncd
```

**Explication :**
-p : crée les répertoires parents si nécessaire

Créer le fichier de configuration (sur le MASTER) :

```
vi /etc/lsyncd/lsyncd.conf.lua
```

Contenu de la configuration (synchronisation du MASTER vers le BACKUP) :

```
settings {
   logfile    = "/var/log/lsyncd.log",
   statusFile = "/var/log/lsyncd.status",
   nodaemon   = false,
}
```

```
sync {
   default.rsync,
   source = "/var/www/html/",
   target = "root@192.168.10.251:/var/www/html/",
```

```
   rsync = {
      archive = true,
      compress = true,
      _extra = {"--delete"}
   }
}
```

**Explication :**

- logfile : fichier de log des opérations de synchronisation
- statusFile : fichier d'état de lsyncd (utilisé par les health checks Keepalived)
- nodaemon : lsyncd tourne en arrière-plan (mode démon)
- default.rsync : utilise rsync comme backend de transfert
- source : répertoire source à surveiller (racine web du MASTER)
- target : destination via SSH (racine web du BACKUP)
- archive : préserve permissions, dates, liens symboliques (rsync -a)
- compress : compresse les données lors du transfert (rsync -z)
- --delete : supprime sur la cible les fichiers supprimés sur la source

## Adaptation du service systemd

Commande :

```
vi /usr/lib/systemd/system/lsyncd.service
```

Configuration finale :

### Unit

```
Description=Live Syncing (Mirror) Daemon
After=network.target
```

### Service

```
User=root
Type=simple
ExecStart=/usr/bin/lsyncd -nodaemon /etc/lsyncd/lsyncd.conf.lua
Restart=on-failure
```

### Install

```
WantedBy=multi-user.target
```

**Explication :**

- User=root : lsyncd doit s'exécuter en tant que root pour accéder aux clés SSH de root et aux fichiers système
- ExecStart=... : commande de démarrage pointant vers le fichier de configuration personnalisé
- Restart=on-failure : redémarre automatiquement lsyncd en cas de crash

Recharger systemd et activer le service :

```
systemctl daemon-reload
systemctl enable lsyncd
systemctl start lsyncd   ← Uniquement sur le MASTER
```

**Explication :**

- daemon-reload : recharge la configuration systemd après modification d'un fichier .service
- lsyncd ne doit démarrer que sur le MASTER. Le BACKUP l'activera lors d'un basculement.

### Scripts de notification Keepalived

Ces scripts permettent à Keepalived de démarrer/arrêter lsyncd lors des basculements.

Créer le script MASTER :

```
vi /etc/keepalived/master.sh
```

```
#!/bin/bash
systemctl start lsyncd
```

Créer le script BACKUP :

```
vi /etc/keepalived/backup.sh
```

```
#!/bin/bash
systemctl stop lsyncd
```

Rendre les scripts exécutables :

```
chmod +x /etc/keepalived/master.sh
chmod +x /etc/keepalived/backup.sh
```

**Explication :**

- chmod +x : rend le script exécutable (nécessaire pour que le système le lance)
- Ces scripts sont appelés automatiquement par Keepalived lors d'un changement d'état

Créer le script de Vérification de lsyncd :

```
vi /usr/local/bin/check_lsyncd_sync.sh
```

#!/bin/bash

```
STATUS_FILE="/var/log/lsyncd.status"
```

```
systemctl is-active --quiet lsyncd
if [ $? -ne 0 ]; then
    exit 1
fi
```

```
if grep -q "Syncing" "$STATUS_FILE"; then
    exit 1
fi
```

```
exit 0
```

**Explication :**

- is-active --quiet lsyncd : vérifie silencieusement si lsyncd est actif
- $? : code de retour de la dernière commande (0 = succès)
- grep -q "Syncing" : vérifie si une synchronisation est en cours
- exit 1 : retourne une erreur → Keepalived peut baisser la priorité

Rendre le script exécutable :

```
chmod +x /usr/local/bin/check_lsyncd_sync.sh
```

## Reconfiguration Keepalived avec intégration lsyncd

Mettre à jour /etc/keepalived/keepalived.conf :

```
vi /etc/keepalived/keepalived.conf
```

Nouvelle configuration (MASTER) :

```
global_defs {
   router_id WEB1
}
```

```
    vrrp_script check_sync {
        script "/usr/local/bin/check_lsyncd_sync.sh"
        interval 2
        weight 2
    }
```

```
    vrrp_script chk_haproxy {
        script "pidof haproxy"
        interval 2
    }
```

```
    vrrp_instance VI_1 {
        state MASTER
        interface eth0
        virtual_router_id 51
        priority 100
        advert_int 1
        nopreempt
```

```
        authentication {
            auth_type PASS
            auth_pass 1234
        }
```

```
        virtual_ipaddress {
            192.168.10.250
        }
```

```
        track_script {
            chk_haproxy
            check_sync
        }
```

```
        notify_master "/etc/keepalived/master.sh"
        notify_backup "/etc/keepalived/backup.sh"
        notify_fault  "/etc/keepalived/backup.sh"
    }
```

**Explication :**

- nopreempt : lorsque le MASTER redémarre, il ne reprend PAS automatiquement la VIP
- (évite les basculements inutiles)
- notify_master : script exécuté quand ce nœud devient MASTER → démarre lsyncd
- notify_backup : script exécuté quand ce nœud devient BACKUP → arrête lsyncd
- notify_fault : script exécuté en cas de défaut → arrête lsyncd

Redémarrer Keepalived :

```
systemctl restart keepalived
```

## Vérification de la synchronisation

Sur le MASTER, vérifier que lsyncd est actif :

```
systemctl status lsyncd
```

**Résultat attendu :**

```
Active: active (running)
```

Sur le BACKUP, vérifier que lsyncd est arrêté :

```
systemctl status lsyncd
```

**Résultat attendu :**

```
Active: failed
```

Test en conditions réelles :

1. Modifier la page HTML sur le MASTER :

```
vi /var/www/html/index.html
```

2. Vérifier la synchronisation sur le BACKUP :

```
cat /var/www/html/index.html
```

→ Le fichier doit être identique sur les deux serveurs quasi instantanément.

# 5.6 Installation et configuration de Zabbix

Zabbix est une solution de supervision open-source permettant de monitorer en temps Réel l'état des serveurs, services et équipements réseau.

## Installation des packages

Importer le dépôt officiel Zabbix 6.4 :

```
    rpm -Uvh https://repo.zabbix.com/zabbix/6.4/rhel/9/x86_64/zabbix-release-6.4-1.el9.noarch.rpm
```

**Explication :**

- rpm -Uvh : installe ou met à jour (-U) un paquet RPM avec affichage verbeux (-v) et barre de progression (-h)
- L'URL pointe vers le dépôt officiel Zabbix pour RHEL 9.
- Cette commande ajoute le fichier .repo dans /etc/yum.repos.d/ permettant à dnf de trouver les paquets Zabbix.

Installer les composants Zabbix :

```
    dnf -y install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-sql-scripts zabbix-agent mariadb-server
```

**Explication :**

- zabbix-server-mysql : serveur Zabbix avec support MySQL/MariaDB
- zabbix-web-mysql : interface web PHP de Zabbix configurée pour MySQL
- zabbix-apache-conf : fichier de configuration Apache pour l'interface web Zabbix
- zabbix-sql-scripts : scripts SQL d'initialisation du schéma de base de données
- zabbix-agent : agent Zabbix local pour superviser le serveur Zabbix lui-même
- mariadb-server : serveur de base de données MariaDB (fork de MySQL, 100% compatible)

## Configuration de la base de données MariaDB

Activer et sécuriser MariaDB :

```
    systemctl enable --now mariadb
    mysql_secure_installation
```

**Explication :**

- enable --now mariadb : démarre MariaDB ET l'active au démarrage
- mysql_secure_installation : script interactif qui :
- Définit le mot de passe root MariaDB
- Supprime les utilisateurs anonymes
- Désactive la connexion root à distance
- Supprime la base de test
- Recharge les privilèges

Connexion à MariaDB :

```
mysql -u root -p
```

**Explication :**

- mysql : client en ligne de commande MariaDB/MySQL
- -u root : connexion avec l'utilisateur root
- -p : demande le mot de passe (saisi de façon masquée)

Création de la base de données et de l'utilisateur Zabbix :

```
CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

**Explication :**

```
CREATE DATABASE ... utf8mb4 COLLATE utf8mb4_bin
```

- Crée la base avec l'encodage UTF-8 4 octets (nécessaire pour les caractères spéciaux) et un collation binaire (requis par Zabbix pour la sensibilité à la casse).

```
CREATE USER 'zabbix'@'localhost'
```

- Crée un utilisateur zabbix qui ne peut se connecter QUE depuis localhost (sécurité).

```
IDENTIFIED BY 'password'
```

- Définit le mot de passe. À remplacer par un mot de passe fort en production.

```
GRANT ALL PRIVILEGES ON zabbix.*
```

- Accorde tous les droits sur toutes les tables de la base zabbix uniquement.

```
FLUSH PRIVILEGE
```

- Recharge immédiatement les tables de droits.

Initialisation du schéma Zabbix :

```
zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql -u
zabbix -p zabbix
```

**Explication :**

- zcat : décompresse un fichier .gz et l'envoie vers la sortie standard
- server.sql : script SQL compressé contenant toutes les instructions de création des tables
- | : redirige la sortie de zcat vers le client MySQL
- mysql -u zabbix -p zabbix : se connecte à la base "zabbix" avec l'utilisateur "zabbix"

⚠ **ATTENTION : Cette commande peut prendre plusieurs minutes. Elle crée des centaines de tables et insère les données de configuration initiales de Zabbix**

## Configuration du serveur Zabbix

Commande :

```
vi /etc/zabbix/zabbix_server.conf
```

Modifier la ligne :

```
DBPassword=MotDePasseFort
```

**Explication :**

- Ce paramètre doit correspondre au mot de passe défini lors de la création de l'utilisateur MariaDB "zabbix". Le fichier zabbix_server.conf contient toutes les options du serveur Zabbix (connexion BDD, timeouts, threads, etc.).

## Configuration du fuseau horaire PHP

Commande :

```
vi /etc/php-fpm.d/zabbix.conf
```

Modifier ou Ajouter :

```
php_value[date.timezone] = Europe/Paris
```

**Explication :**

- Zabbix utilise PHP-FPM (FastCGI Process Manager) pour exécuter son interface web.
- date.timezone : paramètre PHP définissant le fuseau horaire pour l'affichage des dates et heures dans l'interface Zabbix. Sans ce paramètre, les horodatages s'afficheraient en UTC.

## Démarrage des services

Commandes :

```
systemctl enable --now zabbix-server zabbix-agent httpd php-fpm
systemctl start zabbix-server zabbix-agent httpd php-fpm
```

**Explication :**

- zabbix-server : service principal qui collecte les données et exécute les actions
- zabbix-agent : agent local pour superviser le serveur Zabbix lui-même
- httpd : Apache qui sert l'interface web PHP de Zabbix
- php-fpm : gestionnaire de processus PHP requis par l'interface web

## Ouverture des ports Zabbix

Commande :

```
vi /etc/sysconfig/iptables
```

Ajouter :

-A INPUT -p tcp -m state --state NEW -m tcp --dport 10051 -j ACCEPT

-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT

-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT

**Explication :**

- -m state --state NEW : n'autorise que les nouvelles connexions (protection contre attaques)
- --dport 10051 : port d'écoute du serveur Zabbix (communication avec agents actifs)
- --dport 80 / 443 : accès à l'interface web Zabbix via HTTP/HTTPS

```
systemctl reload iptables
```

## Accès à l'interface web

Depuis un navigateur, saisir l'adresse IP du serveur Zabbix.

Identifiants par défaut :

Utilisateur : Admin

Mot de passe : zabbix

**⚠ IMPORTANT : Modifier le mot de passe par défaut immédiatement après la première connexion pour des raisons de sécurité.**

# 5.7 Configuration des agents Zabbix

L'agent Zabbix est installé sur chaque machine à superviser. Il collecte les métriques locales et les envoie au serveur Zabbix.

### Installation de l'agent (sur chaque machine supervisée)

Commandes :

```
    rpm -Uvh https://repo.zabbix.com/zabbix/6.4/rhel/9/x86_64/zabbix-release-6.4-1.el9.noarch.rpm
    dnf install zabbix-agent -y
```

### Configuration de l'agent

Commande :

```
    vi /etc/zabbix/zabbix_agentd.conf
```

Modifier les trois paramètres suivants :

```
    Server=IP_DU_SERVEUR_ZABBIX
    ServerActive=IP_DU_SERVEUR_ZABBIX
    Hostname=Nom_Exact_DuServeur_A_Superviser
```

**Explication :**

- Server : adresse IP du serveur Zabbix. L'agent accepte les connexions (checks passifs) UNIQUEMENT depuis cette IP.
- ServerActive : adresse IP du serveur Zabbix pour les checks actifs (l'agent envoie proactivement ses données au serveur).
- Hostname : nom de la machine tel qu'il apparaîtra dans l'interface Zabbix.Doit correspondre EXACTEMENT au champ "Host name" saisi dans l'interface web.

Démarrer l'agent :

```
    systemctl enable --now zabbix-agent
    systemctl start zabbix-agent
```

## Ouverture du port de l'agent

Commande :

```
vi /etc/sysconfig/iptables
```

Ajouter :

-A INPUT -p tcp -s IP_Serveur_Zabbix --dport 10050 -j ACCEPT

**Explication :**

- --dport 10050 : port d'écoute de l'agent Zabbix (checks passifs depuis le serveur)
- -s IP_Serveur_Zabbix : restreint l'accès à ce port UNIQUEMENT depuis l'IP du serveur Zabbix (sécurité renforcée — le port 10050 n'est pas exposé à tous)

```
systemctl reload iptables
```

## Ajout de l'hôte dans l'interface Zabbix

Depuis l'interface web, naviguer vers :

```
Data collection → Hosts → Create host
```

Remplir les champs suivants :

| Champ | Valeur |
|---|---|
| Host name | Nom exact du paramètre Hostname dans zabbix_agentd.conf |
| Templates | Linux by Zabbix agent (ou le template approprié) |
| Host groups | Créer un groupe ou en sélectionner un existant |
| Interfaces | Sélectionner "Agent" et renseigner l'adresse IP de la machine |

Cliquer sur "Add" pour valider.

**⚠ ATTENTION : Le champ Host name doit correspondre EXACTEMENT au paramètre Hostname du fichier zabbix_agentd.conf. Toute différence (casse, espace) empêchera la communication.**

Après quelques instants, l'indicateur "Availability" de l'hôte passe au VERT, confirmant que l'agent communique correctement avec le serveur.

Répéter cette procédure pour chaque machine à superviser.

# 6. Conclusion

Cette situation professionnelle a permis de mettre en œuvre une infrastructure web complète répondant aux exigences de haute disponibilité, de performances et de supervision.

Récapitulatif des réalisations :

| Composant | Statut | Fonction |
|---|---|---|
| Apache HTTPD (HTTPS) | Opérationnel | Serveur web sécurisé avec certificat SSL |
| HAProxy | Opérationnel | Répartition de charge round-robin |
| Keepalived (VRRP) | Opérationnel | Basculement automatique sur panne |
| lsyncd | Opérationnel | Synchronisation temps réel des fichiers web |
| Zabbix Server | Opérationnel | Supervision centralisée |
| Agents Zabbix | Déployés | Collecte de métriques sur chaque serveur |

Ce projet m'a permis de mobiliser un ensemble de compétences techniques variées, allant de l'installation et la configuration de services Linux sous RHEL 9 à la mise en œuvre d'une architecture haute disponibilité complète. J'ai pu approfondir ma maîtrise de la configuration du pare-feu iptables, de la gestion des services systemd, ainsi que de la sécurisation des communications web via SSL/TLS. Le déploiement de Zabbix comme solution de supervision professionnelle a été particulièrement formateur, car il m'a confronté à des problématiques concrètes de monitoring en environnement de production.

Sur le plan personnel, ce projet m'a fait prendre conscience de l'importance de la haute disponibilité dans un contexte professionnel. Travailler sur le basculement automatique entre serveurs m'a permis de comprendre que la continuité de service n'est pas un simple concept théorique, mais un enjeu réel auquel les administrateurs systèmes et réseaux font face quotidiennement.

Cependant, plusieurs axes d'amélioration pourraient être envisagés pour renforcer cette infrastructure :

- Mise en place d'un certificat SSL signé par une CA reconnue (Let's Encrypt)
- Configuration de triggers et alertes e-mail dans Zabbix
- Ajout de dashboards de supervision personnalisés
- Mise en place d'une sauvegarde de la base de données Zabbix
- Documentation des procédures de reprise sur incident (PRA)

# 7. Glossaire

**Apache HTTPD**

Serveur web open-source de la fondation Apache. Il traite les requêtes HTTP/HTTPS des clients et sert les fichiers web. Dans ce projet, il s'exécute sur le port 8080 en tant que backend derrière HAProxy.

**CA (Certificate Authority — Autorité de certification)**

Organisme de confiance qui émet et signe des certificats SSL/TLS. Un certificat signé par une CA reconnue (ex : Let's Encrypt) est accepté par les navigateurs sans avertissement, contrairement à un certificat auto-signé.

**Certificat auto-signé**

Certificat SSL/TLS généré et signé par le serveur lui-même, sans validation par une CA tierce. Utilisé en environnement de test ou interne. Généré avec la commande openssl req -x509.

**DNF (Dandified YUM)**

Gestionnaire de paquets par défaut sous RHEL 9 / Rocky Linux / AlmaLinux. Remplace YUM et permet d'installer, mettre à jour ou supprimer des logiciels depuis des dépôts RPM.

**DocumentRoot**

Répertoire racine où Apache cherche les fichiers à servir aux clients. Par défaut sous RHEL : /var/www/html/.

**HAProxy (High Availability Proxy)**

Répartiteur de charge (load balancer) et reverse proxy open-source. Il distribue les requêtes HTTP entrantes entre plusieurs serveurs backends selon un algorithme (ex : round-robin). Il effectue également des health checks pour retirer automatiquement un serveur défaillant du pool.

**Haute disponibilité (HA — High Availability)**

Architecture visant à garantir la continuité de service en cas de panne d'un composant. Elle repose sur la redondance des équipements et le basculement automatique (failover). L'objectif est de minimiser le temps d'interruption (downtime).

**Health Check (Vérification de santé)**

Mécanisme par lequel HAProxy vérifie périodiquement si chaque serveur backend répond correctement. Un serveur qui ne passe plus le health check est exclu temporairement du pool de répartition.

**HTTP / HTTPS**

HTTP (HyperText Transfer Protocol) est le protocole de communication web (port 80). HTTPS est sa version sécurisée (port 443) : les échanges sont chiffrés via TLS, garantissant la confidentialité et l'intégrité des données.

**iptables**

Pare-feu intégré au noyau Linux, basé sur Netfilter. Il filtre le trafic réseau entrant (INPUT), sortant (OUTPUT) et transité (FORWARD) selon des règles configurées dans /etc/sysconfig/iptables.

**Keepalived**

Démon Linux implémentant le protocole VRRP pour gérer une adresse IP virtuelle (VIP) partagée entre plusieurs serveurs. En cas de panne du serveur MASTER, le serveur BACKUP prend automatiquement la VIP en moins de 2 secondes.

**lsyncd (Live Syncing Daemon)**

Démon de synchronisation en temps réel qui surveille les modifications de fichiers via inotify et les réplique vers des serveurs distants via rsync/SSH. Utilisé pour maintenir la cohérence du contenu web entre le MASTER et le BACKUP.

**MariaDB**

Système de gestion de bases de données relationnelles (SGBDR) open-source, fork communautaire de MySQL. 100 % compatible MySQL. Utilisé comme moteur de base de données pour stocker les configurations et données de supervision de Zabbix.

**mod_ssl**

Module Apache qui active le support SSL/TLS. Il permet à Apache d'écouter sur le port 443 et de chiffrer les communications HTTP via le protocole TLS.

**OpenSSL**

Bibliothèque et boite à outils open-source implémentant les protocoles SSL et TLS ainsi que les algorithmes de chiffrement associés. Utilisé pour générer les clés RSA et les certificats SSL auto-signés.

**PHP-FPM (FastCGI Process Manager)**

Gestionnaire de processus PHP fonctionnant via le protocole FastCGI. Il exécute les scripts PHP de l'interface web Zabbix indépendamment d'Apache, permettant une meilleure gestion des ressources et des performances.

**PRA (Plan de Reprise d'Activité)**

Document décrivant les procédures et ressources nécessaires pour restaurer les systèmes informatiques après un sinistre ou une panne majeure. Il définit les objectifs de temps de reprise (RTO) et de point de reprise (RPO).

**Round-robin**

Algorithme de répartition de charge utilisé par HAProxy : chaque nouvelle requête est envoyée au serveur suivant dans la liste de manière cyclique (web1, web2, web1, web2…), assurant une distribution équitable de la charge.

**rsync**

Utilitaire de synchronisation de fichiers en local ou via réseau (SSH). Il ne transfère que les différences entre la source et la destination, ce qui le rend très efficace. Utilisé par lsyncd comme backend de transfert.

**SPOF (Single Point of Failure — Point de défaillance unique)**

Composant dont la défaillance provoque l'arrêt total du service. Dans l'architecture initiale de NETCY, le serveur web unique constituait un SPOF : toute panne entraînait une interruption complète du service.

**SSH (Secure Shell)**

Protocole réseau chiffré permettant l'administration à distance et le transfert sécurisé de fichiers. Dans ce projet, l'authentification SSH par clés RSA (sans mot de passe) est utilisée pour automatiser la synchronisation lsyncd entre les serveurs.

**SSL/TLS (Secure Sockets Layer / Transport Layer Security)**

Protocoles cryptographiques assurant le chiffrement des communications sur réseau. SSL est l'ancêtre de TLS (aujourd'hui obsolète). TLS est la version actuelle utilisée pour sécuriser HTTPS, en chiffrant les échanges entre le client et le serveur.

**systemd**

Système d'initialisation et gestionnaire de services sous Linux (remplace SysVinit). Il gère le démarrage, l'arrêt et la supervision des services (unités .service). Commandes principales : systemctl start/stop/enable/disable/status.

**VIP (Virtual IP Address — Adresse IP Virtuelle)**

Adresse IP partagée entre plusieurs serveurs, gérée par Keepalived via VRRP. Dans ce projet : 192.168.10.250. Les clients se connectent toujours à cette adresse unique, quel que soit le serveur physique qui la détient.

**VRRP (Virtual Router Redundancy Protocol)**

Protocole réseau standardisé (RFC 5798) permettant à plusieurs routeurs ou serveurs de partager une adresse IP virtuelle. L'un des noeuds est MASTER (priorité 100), les autres sont BACKUP (priorité 90). En cas de panne du MASTER, le BACKUP de plus haute priorité prend la VIP.

**Zabbix**

Solution de supervision réseau et système open-source. Il collecte des métriques (CPU, mémoire, disponibilité des services) via des agents installés sur les machines supervisées, et les affiche dans une interface web centralisée avec gestion des alertes.

**Agent Zabbix**

Démon installé sur chaque machine à superviser. Il écoute sur le port 10050 (checks passifs depuis le serveur) et peut envoyer proactivement des données au serveur Zabbix (checks actifs, port 10051). Il collecte les métriques locales du système.

**BTS SERVICES INFORMATIQUES AUX ORGANISATIONS** **SESSION 2026**

**ANNEXE VII-1-A : Fiche descriptive de réalisation professionnelle (recto)**

**Épreuve E6 - Administration des systèmes et des réseaux (option SISR)**

| **DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE** | **N° réalisation : 1** |
|---|---|
| **Nom, prénom : Jung Jean-Marie** | **N° candidat : 02542610631** |
| **Épreuve ponctuelle ☒** **Contrôle en cours de formation ☐** | **Date :** ..... / ..... /..... |

***Organisation support de la réalisation professionnelle***

**NETCY est une entreprise de services informatiques dont les collaborateurs utilisent quotidiennement un site web interne pour leurs activités métier.**
**Elle peut assurer l'hébergement et la mise à disposition de services web internes, mais également intervenir sur des problématiques d'infrastructure plus avancées, comme la haute disponibilité, la supervision des équipements et la sécurisation des communications.**
**La direction informatique de NETCY fait face à une fragilité critique de son infrastructure web : hébergé sur un serveur unique sans redondance ni supervision, le site interne est exposé à un risque d'interruption totale en cas de panne. Elle souhaite migrer vers une architecture haute disponibilité avec répartition de charge, synchronisation automatique des fichiers entre serveurs, et supervision centralisée via Zabbix.**

***Intitulé de la réalisation professionnelle***

Déploiement d'une Infrastructure Web Haute Disponibilité

***Période de réalisation :*** ..... ***Lieu :*** .....

***Modalité :*** ☐ ***Seul(e)*** ☒ ***En équipe***

***Compétences travaillées***

☒ Concevoir une solution d'infrastructure réseau
☒ Installer, tester et déployer une solution d'infrastructure réseau
☒ Exploiter, dépanner et superviser une solution d'infrastructure réseau

**Conditions de réalisation[1] (ressources fournies, résultats attendus)**

**Ressources fournies :**

- Hyperviseur Proxmox avec 3 machines virtuelles préconfigurées (Web1, Web2, Zabbix)
- Accès root SSH à chaque machine virtuelle
- Réseau virtualisé segmenté en trois zones (LAN, DMZ, zone privée)
- Documentation technique officielle des outils (Apache, HAProxy, Keepalived, lsyncd, Zabbix)
- Pare-feu pfSense en place sur le périmètre réseau

**Résultats attendus :**

- Site web interne accessible en permanence via l'IP virtuelle (VIP 192.168.10.250)
- Basculement automatique vers le serveur secondaire en cas de panne du serveur principal (< 2 secondes)
- Répartition de charge round-robin entre les deux serveurs Apache via HAProxy
- Synchronisation des fichiers web en temps réel entre Web1 et Web2 via lsyncd
- Communications web chiffrées en HTTPS via certificat SSL auto-signé
- Supervision centralisée de l'ensemble des serveurs via Zabbix Server 6.4

[1] En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Administration des systèmes et des réseaux » prévues dans le référentiel de certification du BTS SIO.

**Description des ressources documentaires, matérielles et logicielles utilisées**[2]

- Schéma réseau NETCY (avant/après)
- Documentation d'installation et de configuration d'Apache HTTPD
- Documentation d'installation et de configuration de HAProxy
- Documentation d'installation et de configuration de Keepalived
- Documentation d'installation et de configuration de lsyncd
- Documentation d'installation et de configuration de Zabbix 6.4

**Modalités d'accès aux productions**[3] **et à leur documentation**[4]

**Lien de production:** https://netcy.fr/portfolio
**Lien de documentation:**
**Wiki :** https://wiki.epsi.tala-informatique.fr/index.php

[2] Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.
[3] Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments nécessaires peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.
[4] Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation, par exemples schéma complet de réseau mis en place et configurations des services.

**BTS SERVICES INFORMATIQUES AUX ORGANISATIONS** **SESSION 2026**

**ANNEXE VII-1-A : Fiche descriptive de réalisation professionnelle (verso, éventuellement pages suivantes)**

**Épreuve E6 - Administration des systèmes et des réseaux (option SISR)**

**Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs**

**NETCY est une entreprise de services informatiques dont les collaborateurs utilisent quotidiennement un site web interne pour leurs activités métier.**

**Elle peut assurer l'hébergement et la mise à disposition de services web internes, mais également intervenir sur des problématiques d'infrastructure plus avancées, comme la haute disponibilité, la supervision des équipements et la sécurisation des communications.**

**La direction informatique de NETCY fait face à une fragilité critique de son infrastructure web : hébergé sur un serveur unique sans redondance ni supervision, le site interne est exposé à un risque d'interruption totale en cas de panne.**

**Elle souhaite migrer vers une architecture haute disponibilité avec répartition de charge, synchronisation automatique des fichiers entre serveurs, et supervision centralisée via Zabbix.**

**Solution et procédure d'intégration :**

- **Création et configuration des machines virtuelles RHEL 9 pour les serveurs web (Web1 et Web2)**
- **Installation et configuration d'Apache HTTPD en backend sur chaque serveur web**
- **Configuration du chiffrement HTTPS via un certificat auto-signé généré avec OpenSSL**
- **Installation et configuration de HAProxy comme répartiteur de charge en mode round-robin**
- **Configuration de Keepalived (protocole VRRP) pour la gestion de l'IP virtuelle partagée avec Web1 en MASTER et Web2 en BACKUP**
- **Configuration de lsyncd pour la synchronisation en temps réel du répertoire web entre le MASTER et le BACKUP via rsync/SSH**
- **Déploiement du serveur Zabbix avec MariaDB comme base de données et installation des agents Zabbix sur chaque serveur supervisé**
-

**Test du bon fonctionnement :**

- **Tester la répartition de charge en répétant des requêtes HTTP vers la VIP et vérifier l'alternance entre Web1 et Web2 selon l'algorithme round-robin**
- **Tester le basculement automatique en arrêtant Keepalived sur le MASTER et vérifier que la VIP migre vers le BACKUP en moins de 2 secondes**
- **Tester la synchronisation lsyncd en modifiant un fichier HTML sur le MASTER et vérifier sa réplication immédiate sur le BACKUP**
- **Vérifier la supervision Zabbix en contrôlant que tous les agents remontent leurs métriques au vert dans l'interface web**

**Schéma réseau :**

