

📍 Avenue V. Maistriau 8a
B-7000 Mons
📞 +32 (0)65 33 81 54
✉️ scitech-mons@heh.be

WWW.HEH.BE

Projet Linux

Rapport final

2024-2025

Gillard Robin
Gustin Aimerik



Table des matières

Table des matières	2
1.Introduction.....	3
2.Présentation générale du projet	4
3.Choix de distribution et type d'installation.....	4
4.Plan de partitionnement	5
Justification du choix de RAID.....	6
5.Présentation et description détaillée des services	6
5.1.SSH	6
5.2.Partage NFS/Samba sans authentification	7
5.3.Web.....	7
5.4.DNS	9
5.5.Base de données.....	10
5.6.FTP/Samba dossier web	11
5.7.NTP.....	12
5.8.Monitoring	13
6.Description et explication des scripts.....	13
Script d'Installation : installGlobal.sh	13
Script de Test : Test_Total.sh.....	19
7.Plan de sauvegarde	27
Contexte du projet et objectifs du plan de sauvegarde.....	27
Analyse des contraintes spécifiques et identification des données critiques	27
Justification technique de la sauvegarde complète	28
Méthode de sauvegarde implémentée.....	29
Planification des sauvegardes avec systemd timers.....	30
Processus de sauvegarde détaillé	30
Monitoring et journalisation des sauvegardes	31
Procédure de restauration	31
Comparaison avec les alternatives	32



8.Sécurisation de notre serveur	32
9.Problèmes rencontrés	34
10.Améliorations et conclusions.....	34
11.Bibliographie.....	35
12 Annexes	35
12.1.Lien du GitHub.....	35
12.2.Interface monitoring	35
12.3.Interface web domaine	36
12.4.Interface web client	36
12.5.Database client	37



1. Introduction

Ce document a pour objectif de présenter de manière claire et concise les éléments essentiels du travail réalisé durant la semaine de projet consacrée au développement de différents services sur des serveurs virtuels Linux.

Dans celui-ci, nous passerons en revue nos choix méthodologiques et techniques ainsi que les problèmes rencontrés au cours du projet.

Enfin, nous proposerons une conclusion reprenant une analyse des améliorations possibles pour optimiser les résultats obtenus ainsi qu'un avis personnel.

2. Présentation générale du projet

Ce projet, réalisé par équipe de 2, consiste à mettre en place différents services sur des serveurs virtuels Linux dans une architecture AWS.

Cette semaine enrichissante a pour objectif de mettre en application une méthodologie de travail adéquate combinant cohésion d'équipe et mise en pratique de la matière vue en classe.

Le serveur doit reprendre les points suivants :

- Une connexion SSH sécurisée permettant à l'administrateur de configurer le serveur et d'y exécuter des scripts
- Un partage de dossier sans authentification aussi bien pour l'environnement Linux que Windows à l'aide de NFS et Samba
- Mise à disposition pour un client d'un nom de domaine, un serveur web, un accès FTP et Samba à son dossier web et une base de données
- Un service de monitoring par interface web afin de surveiller nos serveurs ainsi que nos services
- Un serveur NTP permettant de mettre à jour le temps des machines du réseaux
- Faire cache pour les requêtes, être maître de sa zone et posséder une zone inverse
- Plan de sauvegarde
- Une sécurisation du serveur et des services à l'aide de différents outils

3. Choix de distribution et type d'installation

Notre choix de distribution s'est tourné vers Amazon Linux 2023, cette distribution est recommandée et optimisée par AWS pour ses instances EC2.

L'environnement cloud imposant ce choix, fournit une compatibilité maximale avec le reste des services AWS.

Amazon Linux 2023 est basée sur RHEL, cette distribution a pour avantage d'être légère, rapide à déployer et maintenue directement par AWS.

Le type d'installation choisi est headless (sans interface graphique), cela permet une consommation moindre en ressources et une réduction de la surface d'attaque potentielle car moins de services donc moins de failles exploitables.

4. Plan de partitionnement

Pour le choix de notre partitionnement, nous nous sommes basés sur la simplicité de gestion, la robustesse et la préservation des performances.

Point de montage	Périphérique	Type de FS	Taille	Option de montage
/	/dev/nvme0n1p1	xfs	8Go	defaults, noatime
/boot/efi	/dev/nvme0n1p1 28	vfat	10Mo	defaults,noatime, uid=0,gid=0,umas k=0077,shortnam e=winnt,x systemd.automo unt
/mnt/raid1_shar e	/dev/vg_raid1/sh are	ext4	500Mo	defaults
/mnt/raid1_web	/dev/vg_raid1/w eb	ext4	500Mo	defaults
/mnt/raid1_back up	/dev/vg_raid1/ba ckup	ext4	500Mo	defaults

Notre partition principale (/) situé sur le disque /dev/nvme0n1p1 d'une taille de 8 Go contient tous les répertoires standards du système Linux (/home, /var, /etc, ...).

Nous avons choisi de ne pas dédier de partitions aux répertoires /home et /var car le /home est peu sollicité dans ce projet, les utilisateurs se connectent essentiellement via SSH et les données critiques sont déportées sur les volumes RAID et le /var bien qu'hébergeant les journaux systèmes reste peu volumineux dans ce contexte.

L'inclusion de ces répertoires directement dans la partition principale simplifie la maintenance du système global.

Une partition /boot/efi est présente sur le /dev/nvme0n1p128 conformément aux exigences du démarrage UEF.

Afin de garantir la redondance des données critiques, nous avons assemblé les deux volumes /dev/nvme1n1 et /dev/nvme2n en un RAID1 logiciel via mdadm.

Ce volume RAID1 héberge trois volumes logiques VLM séparés selon leur usage :

- /mnt/raid1_share : partage réseau (NFS/Samba)
- /mnt/raid1_web : répertoires web des clients
- /mnt/raid1_backup : zone de sauvegarde automatisée

La séparation logique via LVM facilite la gestion des droits, des sauvegardes et des quotas tout en restant souple à redimensionner si nécessaire.

Nous n'avons pas mis en place de partition dédiée à la swap car nous avons estimé que la machine virtuelle était dotée de suffisamment de RAM pour couvrir les besoins en mémoire des services.

Cela permet de préserver les performances sur disque NVMe et réduire l'usure sur le long terme.

En cas de besoin, nous pouvons tout de même ajouter dynamiquement celle-ci.

4.1. Justification du choix de RAID :

Nous avons opté pour du RAID1 (mirroring) à la place du RAID5 qui aurait nécessité 3 disques afin de mettre en place une redondance basé sur la machine que nous avions (2 disques de tailles similaires).

Ce RAID nous permet donc d'éviter toute perte ainsi qu'une facilité de mise en œuvre utile au projet.

Critère	RAID 1 (Implémenté)	RAID 5 (Alternative)
Redondance	Miroir complet	Parité distribuée
Capacité utile	50%	N-1/N
Performances	Lecture rapide	Écritures plus lentes
Coût	Élevé (100% redondant)	Modéré

5. Présentation et description détaillée des services

5.1. SSH

Ce service est un protocole réseau permettant d'établir une connexion sécurisée à distance directement sur la machine virtuelle.

Son objectif principal étant d'effectuer une administration à distance tout en assurant l'authentification du client.

Afin de garantir l'intégrité des données fournit par le client, l'authentification s'effectue par clé et non par mot de passe.

Les mots de passe étant vulnérable aux attaques par force brute et pouvant facilement être interceptés, l'authentification par clé améliore la sécurité du serveur.

Nous avons ensuite désactivé la connexion directe en root via SSH.

Premièrement, le compte root étant le super utilisateur connu universellement, est la première cible des attaques SSH automatisées et sa désactivation réduit déjà progressivement les tentatives d'intrusion.

Secondement, cela force l'utilisation d'un compte utilisateur standard et donc limite les risques de dommage accidentels (utilisation de sudo si besoin d'élever ses priviléges).

La mise en place de l'option “MaxAuthTries 3” permet de déconnecter un utilisateur après 3 mots de passe mauvais entrés.

“LoginGraceTime 2min” limite le temps maximum d'authentification avant déconnexion, cela permet de libérer les ressources utilisées du serveur en cas de session fantôme.

“ClientAliveInterval 300” et ClientAliveCountMax2 se combinent ensemble afin d'envoyer une requête icmp au client toutes les 300 secondes vérifiant si l'utilisateur est bien actif.

Après 2 requêtes non répondues, le client est déconnecté.

Afin de durcir la protection autour du SSH, nous avons décidé de mettre en place fail2ban.

Fail2ban est un outil de sécurité qui protège notre service SSH en bannissant automatiquement les adresses IP après plusieurs tentatives de connexion échouées, cela limite d'autant plus les attaques par force brute.



5.2. Partage NFS/Samba sans authentification

NFS est un protocole qui permet de partager des fichiers entre systèmes UNIX/Linux sans authentification nécessaire au préalable tandis que Samba lui, permet un partage entre Linux et Windows demandant une authentification d'accès au serveur avant d'accéder au dossier.

```
[ec2-user@sherpa ~]$ sudo cat /etc/exports
/mnt/raid1_share *(rw,sync,root_squash)
/srv/samba/public *(rw,sync,no_subtree_check,root_squash)
```

Sur nos partages, on autorise lecture/écriture, écritures synchrones et interdit aux clients d'accéder aux fichiers en tant que root afin de garder un certain niveau de protection autour des fichiers sensibles.

Sur le client, créer un point de montage via la commande : mount -t nfs <IP serveur>:/mnt/raid1_share <rédertoire du client> afin d'accéder au contenu.

Tout fichier créé dans le /mnt/raid1_share du serveur ou inversement chez le client, sera également créé chez l'autre.

5.3. Web

Le serveur Web est un composant fondamental de notre infrastructure permettant l'hébergement et la diffusion de contenu sur Internet. Pour ce projet, nous avons implémenté Apache HTTP Server (httpd), solution robuste et reconnue dans la communauté Linux.

Apache fonctionne selon un modèle de traitement des requêtes HTTP où il écoute les connexions entrantes sur les ports 80 (HTTP) et 443 (HTTPS), analyse les demandes et retourne le contenu approprié en fonction de la configuration des hôtes virtuels.

Notre implémentation utilise le concept de VirtualHost d'Apache, permettant d'héberger plusieurs sites web sur un même serveur avec des noms de domaines différents. Chaque client obtient automatiquement :

- Un nom de domaine personnalisé (client.domaine)
- Un espace web isolé dans /var/www/client
- Une configuration HTTP et HTTPS dédiée



- Un certificat SSL autosigné pour les connexions sécurisées

Exemple de configuration d'un VirtualHost pour un client:

```
ServerName client.domaine
DocumentRoot /var/www/client
```

```
Options -Indexes +FollowSymLinks
AllowOverride All
Require all granted
```

```
ServerName client.domaine
DocumentRoot /var/www/client
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/client.domaine.crt
SSLCertificateKeyFile
/etc/pki/tls/private/client.domaine.key
```

```
Options -Indexes +FollowSymLinks
AllowOverride All
Require all granted

ErrorLog logs/client_ssl_error.log
CustomLog logs/client_ssl_access.log combined
```

La sécurisation du service Web a été implémentée à plusieurs niveaux :

- Désactivation du listing des répertoires avec l'option `-Indexes`
- Utilisation systématique de HTTPS avec des certificats SSL/TLS autosignée et crée pour le domaine et chaque sous-domaine
- Journalisation séparée des accès et erreurs pour faciliter l'audit
- Implémentation de l'entête HSTS (Strict-Transport-Security) pour renforcer l'utilisation de HTTPS

Pour faciliter l'administration des bases de données, phpMyAdmin est installé et configuré sur le domaine principal. Une page d'accueil personnalisée permet d'accéder simplement à l'interface via une URL dédiée.



Le service Web est automatiquement démarré au lancement du système et configuré pour s'adapter à la charge serveur avec un nombre approprié de processus enfants.

5.4. DNS

Le service DNS (Domain Name System) est essentiel pour la résolution des noms de domaines en adresses IP, facilitant l'accès aux ressources réseau. Notre Implémentation repose sur BIND9 (Berkeley Internet Name Domain), une référence dans l'écosystème Linux.

BIND9 joue plusieurs rôles dans notre infrastructure :

- Serveur maître pour notre zone de domaine principale
- Gestion de la résolution inverse (reverse DNS)
- Serveur récursif pour les requêtes externes
- Cache DNS pour optimiser les performances

La création d'un nouveau client dans notre système déclenche automatiquement la configuration DNS appropriée, incluant :

- Ajout d'un enregistrement A pour le sous-domaine client.domaine
- Mise à jour du fichier de zone principal
- Création d'un enregistrement PTR dans la zone de résolution inversée
- Incrémentation du numéro de série de la zone

Exemple de fichier de zone directe :

```
$TTL 86400
@ IN SOA ns1.domaine. admin.domaine. (
    20250519001 ; Serial
    3600      ; Refresh
    1800      ; Retry
    604800    ; Expire
    86400 )   ; Minimum TTL

@     IN  NS  ns1.domaine.
@     IN  A   192.168.1.10

ns1   IN  A   192.168.1.10
```



```
client1 IN A 192.168.1.10
client2 IN A 192.168.1.10
* IN A 192.168.1.10
```

Exemple de zone inverse :

```
$TTL 86400
@ IN SOA ns1.domaine. admin.domaine. (
    20250519001 ; Serial
    3600      ; Refresh
    1800      ; Retry
    1209600   ; Expire
    86400 )   ; Minimum TTL

@ IN NS ns1.domaine.
10 IN PTR ns1.domaine.
```

Notre configuration permet à BIND9 de répondre aux requêtes provenant de n'importe quelle source (`allow-query { any; }`), mais limite la récursion aux réseaux internes pour des raisons de sécurité et de performance (`allow-recursion { 127.0.0.1; localhost; 192.168.0.0/16; 10.0.0.0/8; };`).

L'ajout automatique du serveur DNS local (`nameserver 127.0.0.1`) comme premier serveur de résolution dans le fichier `/etc/resolv.conf` garantit la priorité à notre service DNS pour toutes les requêtes.

Le service est configuré pour démarrer automatiquement avec le système et vérifier systématiquement l'intégrité des zones avec `named-checkzone` avant chaque (re)démarrage.



5.5. Base de données

Le service de base de données repose sur MariaDB (version 10.5), un fork communautaire de MySQL offrant d'excellentes performances et une compatibilité optimale avec notre infrastructure web.

L'architecture de notre solution de base de données s'articule autour de plusieurs principes :

- Création automatique d'une base de données par client
- Attribution d'un utilisateur dédié à chaque base
- Isolation complète entre les clients
- Sécurisation robuste de l'installation

Le processus de provisionnement d'un nouveau client inclut :

```
CREATE DATABASE IF NOT EXISTS client_db;
CREATE USER IF NOT EXISTS 'client'@'localhost' IDENTIFIED BY
'password';
GRANT ALL PRIVILEGES ON client_db.* TO 'client'@'localhost';
FLUSH PRIVILEGES;
```

La sécurisation de MariaDB est effectuée via mysql_secure_installation qui applique automatiquement les bonnes pratiques :

- Suppression des utilisateurs anonymes
- Désactivation de la connexion root à distance
- Suppression de la base de test

L'intégration avec le service web est assurée par phpMyAdmin, accessible via une interface web sécurisée qui permet aux clients de gérer leurs bases de données sans accès direct au serveur de base de données.

Pour garantir la disponibilité et l'intégrité des données, nous avons mis en place :

- Démarrage automatique du service au boot du système
- Configuration des journaux d'erreurs et de requêtes



- Optimisation des paramètres de performance en fonction des ressources serveur

Les bases de données sont stockées dans le système de fichiers RAID pour assurer une redondance des données et limiter les risques de corruption en cas de défaillance matérielle.

Cette architecture permet de gérer efficacement les bases de données de multiples clients sur un unique serveur tout en maintenant une isolation stricte et une sécurité optimale

5.6. *FTP/Samba dossier web*

Le FTP (File Transfer Protocol) est un protocole de communication standard utilisé pour le transfert de fichiers par le biais d'un client (FileZilla dans notre cas).

Il fonctionne par architecture client-serveur en passant par les ports 20 et 21.

Dans notre cas afin d'ajouter une sécurité supplémentaire, nous avons mis en place une connexion chiffrée TLS entre client et serveur.

Le serveur passe donc par l'utilisation d'un certificat SSL garantissant confidentialité et intégrité.

En complément, Fail2ban surveille les tentatives de connexion FTP grâce à un système de bannissement temporaire après plusieurs échecs d'authentification.

Samba permet un accès à ce fichier depuis Windows par le biais de l'explorateur de fichiers.

Un dossier web se crée automatiquement dans le /var/www lors de la création d'un utilisateur et c'est ce même dossier qui sera partagé.



```
[ec2-user@sherpa ~]$ cat /etc/vsftpd/vsftpd.conf
cat: /etc/vsftpd/vsftpd.conf: Permission denied
[ec2-user@sherpa ~]$ sudo cat /etc/vsftpd/vsftpd.conf
vsftpd_banner=Bienvenue sur le serveur FTP sécurisé.
xferlog_enable=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
chroot_local_user=YES
allow_writeable_chroot=YES
userlist_enable=YES
userlist_deny=NO
local_umask=022
user_sub_token=$USER
local_root=/var/www/$USER
secure_chroot_dir=/var/run/vsftpd/empty
pasv_min_port=30000
pasv_max_port=30100
listen_port=21
listen=YES
listen_ipv6=NO
pam_service_name=vsftpd
ssl_enable=YES
rsa_cert_file=/etc/pki/tls/certs/vsftpd.pem
rsa_private_key_file=/etc/pki/tls/private/vsftpd.key
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
```

[malaise]

```
path = /var/www/malaise
valid users = malaise
writable = yes
create mask = 0770
directory mask = 0770
force user = malaise
force group = malaise
```

[public]

```
path = /srv/samba/public
guest ok = yes
guest only = yes
writable = yes
force user = nobody
force group = nobody
create mask = 0666
directory mask = 0777
browseable = yes
```



5.7. NTP

Le protocole NTP est utilisé pour synchroniser l'heure de notre serveur avec un serveur de temps public fiable.

Nous avons mis en place un système flexible offrant 3 possibilités :

- Définir par défaut le fuseau horaire sur Europe/Bruxelles
- Sélectionner une timezone parmi 598 choix
- Afficher en temps réel le statut NTP

L'importance principale d'une horloge fiable étant la cohérence des logs, les authentifications et les services réseau.

Le service utilisé est Chrony à la place du service NTPd suite à une comparaison réalisée, recommandé pour sa précision et sa rapidité de synchronisation.

Métrique	Chrony	NTPd
Convergence	1-2 secondes	10-15 minutes
Précision réseau instable	±1 ms	±10 ms
Consommation RAM	5 MB	15 MB
Gestion des veilles	Optimisé	Limité

5.8. Monitoring

Afin de visualiser en continu les services actifs et l'état du système, nous avons choisi de passer par le monitoring web nommé "Netdata".

Netdata est un outil léger en ressources et possédant un large choix de services à surveiller.

Nous passons par le port 19999 depuis l'url d'un navigateur web et pouvons ainsi surveiller :

- L'utilisation du CPU, mémoire, disques
- Les services DNS, Samba, SSH, NTP, FTP, HTTP, NFS, ...



- Les fichiers logs
- Les connexions actives et le trafic réseau

Et surtout, netdata propose un système d'alerte en cas d'anomalie ce qui nous permet de réagir rapidement et efficacement.

Pour la mise en place de netdata, nous utilisons docker afin de minimiser l'utilisation de ressources.

6. Description et explication des scripts

Ce point présente une analyse détaillée des deux scripts principaux utilisés dans notre projet : le script d'installation globale (`installGlobal.sh`) et le script de test (`Test_Total.sh`).

Ces scripts constituent le cœur de notre infrastructure, permettant respectivement de déployer l'ensemble des services et de vérifier leur bon fonctionnement.

6.1. *Script d'Installation : installGlobal.sh*

Le script `installGlobal.sh` est l'outil central de notre projet, permettant l'installation automatique de tous les services nécessaires à notre infrastructure.

Ce script suit une architecture modulaire basée sur un système de menus interactifs, permettant une installation flexible et adaptée aux besoins spécifiques.

Architecture générale du script

Le script est structuré autour d'un menu principal offrant un accès à différentes fonctionnalités :

```
display_menu() {  
    echo ""  
    echo " | -----"  
    echo "----- | "  
    echo -e " | ${BLUE}Welcome to the server assistant ${NC}  
| "  
    echo " | Please select the tool you want to use | "
```



```
echo "|-----"
-----| "
echo "| 0. Set server hostname |"
echo "| 1. RAID Configuration |"
echo "| 2. NFS |"
echo "| 3. Web services management |"
echo "| 4. NTP Time Server |"
echo "| 5. Install security services |"
echo "| 6. Backup |"
echo "| 7. Consult Logs Dashboard |"
echo "| 8. Installer Netdata (monitoring) |"
echo "|-----"
-----| "
echo "| q. Quit |"
echo "|-----"
-----| "
}
```

Cette organisation permet une gestion modulaire de l'installation, où l'administrateur peut choisir de configurer uniquement certains aspects du serveur si nécessaire.

Composants clés

Installation initiale

Le script commence par l'installation des packages essentiels pour notre infrastructure :

```
dnf -y install nfs-utils samba mlocate bind chrony fail2ban
vsftpd rsync bind-utils httpd php php-mysqlnd mariadb-server
phpmyadmin
```

Cette première étape garantit que toutes les dépendances nécessaires sont présentes sur le système avant de procéder à la configuration spécifique.

Configuration du Hostname

La fonction `set_hostname()` permet de définir et gérer le nom d'hôte du serveur, facilitant ainsi son identification sur le réseau :

```
set_hostname() {  
    while true; do  
        clear  
        display_hostname_menu  
        read -p "Enter your choice: " hostname_choice  
        case $hostname_choice in  
            1) read -p "Enter the new hostname: "  
new_hostname  
                hostnamectl set-hostname $new_hostname  
                # [...]
```

Configuration RAID

La fonction `raid()` implémente la création et la configuration d'un système RAID1 (miroir) pour garantir la redondance des données :

```
raid() {  
    # [...]  
    sudo mdadm --create --verbose $RAID_DEVICE --level=1 --  
    raid-devices=2 $RAID_DISKS  
    sudo pvcreate $RAID_DEVICE  
    sudo vgcreate vg_raid1 $RAID_DEVICE  
    # [...]
```

Cette configuration inclut la création de volumes logiques (LVM) dédiés pour différents usages :

- Un volume pour le partage réseau
- Un volume pour l'hébergement web
- Un volume dédié aux sauvegardes

Services de Partage Réseau

Les fonctions nfs() mets en place l'un des services de partage de fichiers sans authentification :

```
nfs () {  
    echo "Installing NFS share"  
    sudo mkdir -p /mnt/raid1_share  
    # [...]  
    echo "/mnt/raid1_share *(rw,sync,root_squash)" >  
/etc/exports  
    exportfs -a  
    # [...]
```

Services Web et DNS

La fonction webservices() est particulièrement complexe, gérant :

- La configuration du serveur DNS (BIND)
- L'installation et la configuration d'Apache
- La mise en place de MariaDB
- La génération de certificats SSL pour HTTPS
- La création automatisée de sites pour les clients
- La création d'un dossier samba pour le partage de fichier



```
configure_dns() {  
    DOMAIN=$1  
    IP=$(hostname -I | awk '{print $1}')  
    # [...]  
    cat > "$ZONE_FILE" << EOF  
$TTL 86400  
@ IN SOA ns1.$DOMAIN. admin.$DOMAIN. (  
    $(date +%Y%m%d%H) ; Serial  
    3600 ; Refresh  
    1800 ; Retry  
    604800 ; Expire  
    86400 ) ; Minimum TTF  
# [...]
```

Serveur NTP

La fonction `setup_ntp()` configure le service de synchronisation temporelle :

```
setup_ntp() {  
    # [...]  
    sudo systemctl enable chronyd  
    sudo systemctl start chronyd  
    # [...]
```

Services de Sécurité

Le script implémente plusieurs couches de sécurité :

1. Fail2Ban: Protection contre les attaques par force brute

```
configure_fail2ban() {  
    # [...]  
    cat << EOF | sudo tee /etc/fail2ban/jail.d/sshd.conf >
```

```
/dev/null
[sshd]
enabled = true
backend = systemd
port = 22
action = %(action_mwl)s
maxretry = 3
# [...]
```

2. SELinux: Contrôle d'accès obligatoire

```
configure_SELinux() {
    # [...]
    echo "Activation de SELinux en mode enforcing
(redémarrage requis)"
    sudo sed -i 's/^SELINUX=disabled/SELINUX=enforcing/'
"$SELINUX_CONF"
    # [...]
```

3. inotify: Surveillance des modifications de fichiers

```
configure_inotify() {
    # [...]
    cat << 'EOF' | sudo tee
/usr/local/bin/inotify_monitor.sh > /dev/null
#!/bin/bash
LOGFILE="/var/log/inotify_monitor.log"
WATCHED_DIRS="/etc /var/www"
inotifywait -m -r -e modify,create,delete,move
${WATCHED_DIRS} --format '%T %w %f %e' --timefmt '%F %T' |
    # [...]
```



Système de Backup

La fonction `backup()` met en place un système de sauvegarde automatisé utilisant `systemd timers` :

```
backup() {
    # [...]
    sudo tee /usr/local/bin/auto_backup.sh > /dev/null << EOF
    #!/bin/bash
    TIMESTAMP=\$(date +"%Y-%m-%d_%H-%M-%S")
    BACKUP_DIR="\$MOUNT_POINT/\$TIMESTAMP"
    LOG_FILE="\$MOUNT_POINT/backup.log"
    # [...]
    sudo systemctl enable --now backup.timer
    # [...]
```

Monitoring avec Netdata

La fonction `install_netdata()` déploie un conteneur Docker pour la surveillance du système :

```
install_netdata() {
    # [...]
    sudo docker run -d --name=netdata \
    -p 19999:19999 \
    -v netdataconfig:/etc/netdata \
    -v netdatalib:/var/lib/netdata \
    # [...]
```

Intégration et exécution

La fonction principale `main()` orchestre l'exécution du script en affichant le menu et en traitant les choix de l'utilisateur :



```
main() {  
    while true; do  
        clear  
        display_menu  
        read -p "Enter your choice: " choice  
        case $choice in  
            0) set_hostname ;;  
            1) raid ;;  
            # [...]
```

6.2. *Script de Test : Test_Total.sh*

Le script Test_Total.sh est un outil de diagnostic complet permettant de vérifier l'état de tous les services installés sur le serveur. Il produit un rapport détaillé sur la configuration et le fonctionnement de chaque composant, facilitant ainsi le dépannage et la validation de l'installation.

Architecture du script

Le script est structuré autour de plusieurs fonctions de test spécifiques à chaque service, suivies d'un résumé global. Il utilise un système de code couleur pour améliorer la lisibilité :

```
RED='\[0;31m'  
GREEN='\[0;32m'  
YELLOW='\[0;33m'  
BLUE='\[0;34m'  
NC='\[0m' # No Color
```

Fonctions principales

Affichage d'en-têtes et vérifications de base



```

print_header() {
    echo -e "\n${BLUE}===== $1 =====${NC}"
}

check_service() {
    local service=$1
    local display_name=$2
    if systemctl is-active --quiet "$service"; then
        echo -e " ${GREEN} ✅ $display_name est actif${NC}"
        return 0
    else
        echo -e " ${RED} ❌ $display_name est inactif${NC}"
        return 1
    fi
}

check_port() {
    local port=$1
    local service_name=$2
    if nc -z localhost "$port" 2>/dev/null || ss -tuln | grep -q ":$port "; then
        echo -e " ${GREEN} ✅ Port $port ($service_name) est ouvert${NC}"
        return 0
    else
        echo -e " ${RED} ❌ Port $port ($service_name) n'est pas ouvert${NC}"
        return 1
    fi
}

```



Ces fonctions utilitaires permettent d'afficher des sections distinctes dans le rapport et de vérifier l'état des services et des ports.

Composants testés

Configuration du Hostname

```
print_header "CONFIGURATION DU HOSTNAME"
echo -e " Hostname actuel : $(hostnamectl --static)"
echo -e " FQDN : $(hostname -f)"
```

Vérification du Firewall

```
print_header "FIREWALL"
check_service "firewalld" "Firewall"
if systemctl is-active --quiet firewalld; then
    echo -e "\n Services autorisés par le firewall:"
    firewall-cmd --list-services | sed 's/ /\n - /g' | sed
's/^/ - /' | grep -v "^\s* - \$"
    echo -e "\n Ports autorisés par le firewall:"
    firewall-cmd --list-ports | sed 's/ /\n - /g' | sed 's/^/
- /' | grep -v "^\s* - \$"
fi
```

Test RAID

```
print_header "SYSTEME RAID"
if cat /proc/mdstat 2>/dev/null | grep -q 'md'; then
    echo -e " ${GREEN}☒ RAID configuré${NC}"
    echo -e "\n Détails des dispositifs RAID:"
    cat /proc/mdstat | grep -E 'md|blocks'
```



```
# Vérifier l'état de chaque dispositif RAID
for md in $(cat /proc/mdstat | grep ^md | cut -d : -f 1);
do
    echo -e "\n Détails pour $md:"
    mdadm --detail /dev/$md | grep -E "State :|Active
Devices :|Working Devices :|Failed Devices :|Spare Devices :"
    done
else
    echo -e " ${YELLOW}⚠ Aucun RAID détecté${NC}"
fi
```

Partages Réseau

Le script vérifie la configuration et le fonctionnement des services NFS et Samba :

```
print_header "PARTAGES RÉSEAU"
# 4.1 Test NFS
echo -e "${YELLOW}NFS:${NC}"
check_service "nfs-server" "Service NFS"
if systemctl is-active --quiet nfs-server; then
    echo -e "\n Partages NFS exportés:"
    if exportfs -v | grep -v "^\$"; then
        exportfs -v
    else
        echo -e " ${YELLOW}⚠ Aucun partage NFS
configuré${NC}"
    fi
    # [...]
}
```

Services Web et Base de Données

```
print_header "SERVICES WEB"

# 5.1 Test Apache
echo -e "${YELLOW}Apache:${NC}"
check_service "httpd" "Serveur Apache"
check_port "80" "HTTP"
check_port "443" "HTTPS"
if systemctl is-active --quiet httpd; then
    echo -e "\n Sites web configurés:"
    ls /etc/httpd/conf.d/*.conf 2>/dev/null | while read
conf; do
    servername=$(grep -i "ServerName" $conf 2>/dev/null |
head -1 | awk '{print $2}')
    if [ -n "$servername" ]; then
        echo -e " - $servername ($basename $conf)"
    fi
done
# [...]
}
```

Serveur DNS

```
print_header "SERVEUR DNS"

check_service "named" "Serveur DNS (Named/Bind)"
check_port "53" "DNS"
if systemctl is-active --quiet named; then
    echo -e "\n Zones DNS configurées:"
    if [ -f /etc/named.conf ]; then
        grep -E "zone\s+\".*\"" /etc/named.conf | grep -v
"localhost" | grep -v "in-addr.arpa"
    fi
fi
```



Synchronisation NTP

```
print_header "SYNCHRONISATION HORAIRE (NTP)"  
check_service "chrony" "Service Chrony"  
if systemctl is-active --quiet chrony; then  
    echo -e "\n Statut de synchronisation:"  
    chronyc tracking | head -2  
    echo -e "\n Sources NTP:"  
    chronyc sources | head -4  
fi
```

Sécurité

Le script vérifie plusieurs aspects de sécurité :

1. Fail2Ban:

```
echo -e "${YELLOW}Fail2Ban:${NC}"  
check_service "fail2ban" "Service Fail2Ban"  
if systemctl is-active --quiet fail2ban; then  
    echo -e "\n Jails Fail2Ban configurés:"  
    fail2ban-client status 2>/dev/null | grep "Jail list"  
| sed -E 's/^[:]+:[ \t]+//'  
# [...]  
}
```

2. SELinux:

```
echo -e "\n${YELLOW}SELinux:${NC}"  
if command -v getenforce >/dev/null 2>&1; then  
    selinux_status=$(getenforce)
```



```

        if [ "$selinux_status" == "Enforcing" ]; then
            echo -e " ${GREEN}☒ SELinux est actif en mode
Enforcing${NC}"
            # [...]
    }

```

3. ClamAV:

```

echo -e "\n${YELLOW}ClamAV:${NC}"
if command -v clamscan >/dev/null 2>&1; then
    echo -e " ${GREEN}☒ ClamAV est installé:${NC}
$(clamscan --version)"
    # [...]
}

```

Système de Backup

```

print_header "SAUVEGARDE"
if systemctl list-unit-files | grep -q backup.timer; then
    if systemctl is-active --quiet backup.timer; then
        echo -e " ${GREEN}☒ Timer de backup actif${NC}"
        echo -e " Prochain déclenchement: $(systemctl show
backup.timer | grep NextElapseUsecRealtime | cut -d= -f2)"
    else
        echo -e " ${RED}☒ Timer de backup inactif${NC}"
    fi
    # [...]
}

```



Monitoring Netdata

```

print_header "MONITORING NETDATA"

if command -v docker >/dev/null 2>&1; then
    if docker ps 2>/dev/null | grep -q netdata; then
        echo -e " ${GREEN} ✅ Container Netdata en cours
d'exécution${NC}"
        container_id=$(docker ps | grep netdata | awk '{print
$1}')
        port=$(docker port $container_id 2>/dev/null | grep -
i tcp | head -1 | cut -d ":" -f2)
        if [ -n "$port" ]; then
            ip_addr=$(hostname -I | awk '{print $1}')
            echo -e " Interface web disponible sur:
http://\$ip\_addr:\$port"
        fi
    else
        echo -e " ${RED} ❌ Container Netdata non trouvé${NC}"
    fi
else
    echo -e " ${RED} ❌ Docker n'est pas installé${NC}"
fi

```

Résumé global

Le script se termine par un résumé global de tous les services testés :

```

print_header "RÉSUMÉ GLOBAL DES SERVICES"
services=(
    "firewalld:Firewall"
    "httpd:Serveur Web (Apache)"
    "mariadb:Base de données (MariaDB)"
    "named:Serveur DNS (Bind)"

```

```
"chronyd:Synchronisation NTP"
"nfs-server:Partage NFS"
"smb:Partage Samba"
"vsftpd:Serveur FTP"
"fail2ban:Protection Fail2Ban"
)

for svc in "${services[@]}"; do
    name="${svc#*:}"
    service="${svc%:*}"
    if systemctl is-active --quiet "$service"; then
        echo -e " ${GREEN}✓ ${name}${NC}"
    else
        echo -e " ${RED}✗ ${name}${NC}"
    fi
done
```

Conclusion sur les Scripts

Ces deux scripts constituent l'épine dorsale de notre projet, permettant respectivement :

1. **installGlobal.sh**: Une installation modulaire, flexible et complète de tous les services requis, avec des options de configuration avancées.
2. **Test_Total.sh**: Une vérification complète et détaillée de la configuration et du fonctionnement correct de chaque service installé.

Ensemble, ils forment un système robuste pour le déploiement, la configuration et la vérification d'une infrastructure serveur Linux complète répondant aux exigences du cahier des charges.

L'entièreté de ces 2 scripts peuvent être retrouvés dans notre GitHub en annexe



7. Plan de sauvegarde

Contexte du Projet et Objectifs du Plan de Sauvegarde

Le cahier des charges demande la mise en place d'un plan de sauvegarde automatisé pour un serveur Linux hébergeant des services (partage de fichiers NFS/Samba, bases de données utilisateur, serveur web, monitoring). L'objectif est de garantir la disponibilité et la restauration rapide en cas d'incident, tout en respectant les bonnes pratiques de sécurisation.

Notre plan de sauvegarde vise spécifiquement à :

- Assurer la disponibilité des données critiques en cas d'incident
- Garantir une sauvegarde régulière et automatisée des ressources importantes
- Faciliter la restauration en cas de besoin
- Offrir une traçabilité complète des opérations de sauvegarde
- Optimiser les ressources de stockage et de performance système

Notre solution est entièrement automatisée et intégrée au système d'exploitation via systemd, offrant ainsi une fiabilité et une traçabilité supérieures aux solutions classiques basées sur cron.

Analyse des Contraintes Spécifiques et Identification des Données Critiques

Structure des Données et Ressources Critiques

Le système comprend trois catégories principales de données critiques nécessitant une sauvegarde régulière :

1. **Partages réseau** : Deux volumes RAID (/mnt/raid1_share) contenant des données utilisateurs partagées via NFS et Samba, contenant des documents potentiellement irremplaçables.
2. **Contenu web des clients** : Les répertoires web situés dans /mnt/raid1_web hébergent les sites et applications des clients. Ces données représentent l'activité principale de l'infrastructure.
3. **Bases de données utilisateurs** : Chaque utilisateur dispose d'une base de données MySQL individuelle qui doit être sauvegardée séparément.

4. **Configuration de services complexes** : Les configurations des services NTP, DNS et monitoring nécessitent également une sauvegarde.

Exigences de Récupération

- **RPO** (Recovery Point Objective) : Implicitement fixé à 2 heures, ce qui correspond à l'intervalle de sauvegarde de 1h58.
- **RTO** (Recovery Time Objective) : Nécessite une restauration immédiate et complète pour éviter les interruptions durant les démonstrations.

Contraintes Pédagogiques

- Nécessité de simplicité opérationnelle pour des étudiants en apprentissage.
- Validation rapide des sauvegardes durant l'évaluation.
- Redémarrage automatique du serveur toutes les 4 heures.

Justification Technique de la Sauvegarde Complète

Cohérence des Données et Intégrité des Services

La sauvegarde complète garantit une image atomique de tous les services à un instant T :

- Synchronisation complète via rsync -avz des volumes RAID.
- Dump intégral des bases de données avec mysqldump, évitant les états partiels.
- Horodatage précis (`TIMESTAMP=$(date +"%Y-%m-%d_%H-%M-%S")`) pour un versionnage clair.

Contrairement à une sauvegarde incrémentielle, cette approche élimine les risques :

- Dépendances à une chaîne de sauvegardes antérieures.
- Incohérences entre fichiers systèmes et bases de données.

Alignement avec les Bonnes Pratiques AWS

Bien que AWS recommande généralement des sauvegardes incrémentielles pour l'optimisation des coûts, le contexte pédagogique justifie une exception :

- Contrôle total sur le processus de restauration durant les démonstrations.
- Compatibilité avec le stockage sur volumes RAID locaux plutôt que sur S3.

Sécurité et Auditabilité

Chaque sauvegarde complète constitue une preuve indépendante du système :

- Isolation des versions dans des répertoires dédiés (\$BACKUP_DIR/\$TIMESTAMP).
- Journalisation centralisée (\$MOUNT_POINT/backup.log) pour le traçage.
- Protection contre les corruptions en cascade (risque des sauvegardes incrémentielles).

Optimisation pour l'Environnement Académique

- Déploiement prévisible : Les ressources AWS allouées (t2.micro) limitent l'impact des transferts complets.
- Périmètre maîtrisé : La taille modeste des données pédagogiques rend les sauvegardes complètes rapides.

Gestion des Redémarrages Systématiques et Synchronisation des Sauvegardes

Le service AWS impose un redémarrage automatique du serveur toutes les 4 heures. Cette contrainte opérationnelle unique nécessite une adaptation spécifique :

- Intervalle de Sécurité : Le choix d'une sauvegarde toutes les 1h58 (118 minutes) garantit qu'au moins une sauvegarde complète est effectuée entre chaque cycle de redémarrage.
- Protection Contre les Échecs de Redémarrage : Les redémarrages fréquents augmentent statistiquement les risques de corruption de données. Chaque sauvegarde complète constitue un point de restauration indépendant.
- Alignement Temporel : L'intervalle de 1h58 évite toute interférence avec le scheduler systemd tout en maintenant un RPO effectif de 2 heures.

Méthode de Sauvegarde Implémentée

Technologie de Sauvegarde

Notre solution s'appuie sur une combinaison de technologies éprouvées :

- **rsync** pour les fichiers : Outil de synchronisation efficace qui ne copie que les changements, économisant ainsi bande passante et espace disque.
- **mysqldump** pour les bases de données : Export complet de chaque base de données utilisateur dans des fichiers SQL distincts.

- **systemd timers** pour la planification : Alternative moderne à cron offrant une meilleure intégration système, un meilleur contrôle et une journalisation plus complète.

Stockage des Sauvegardes

Les sauvegardes sont stockées sur un volume dédié, configuré en RAID pour garantir la redondance des données. L'administrateur peut choisir le point de montage optimal lors de la configuration du système de sauvegarde, en tenant compte des volumes RAID disponibles pour maximiser la sécurité des données.

Chaque sauvegarde est conservée dans un répertoire dont le nom inclut un horodatage précis au format YYYY-MM-DD_HH-MM-SS, permettant une identification immédiate et facilitant la récupération ciblée.

Planification des Sauvegardes avec Systemd Timers

Fréquence et Déclenchement

Le système de sauvegarde suit une planification précise :

- **Première sauvegarde** : 5 minutes après le démarrage du système (paramètre OnBootSec=5min)
- **Sauvegardes suivantes** : Toutes les 1h58 (paramètre OnUnitActiveSec=118m)
- **Décalage aléatoire** : Un délai aléatoire de 30 secondes est appliqué pour éviter les pics de charge (paramètre RandomizedDelaySec=30s)
- **Précision d'exécution** : Garantie à 1 seconde près (paramètre AccuracySec=1s)

Cette fréquence élevée permet de minimiser la perte potentielle de données en cas d'incident, tout en maintenant une empreinte système raisonnable grâce à l'efficacité de rsync.

Avantages de Systemd Timers par Rapport à Cron

L'utilisation de systemd timers présente plusieurs avantages significatifs :

- Journalisation intégrée et détaillée via journalctl
- Gestion des dépendances et suivi d'état
- Redémarrage automatique en cas d'échec
- Meilleure intégration au système et à ses services



- Précision accrue dans la planification temporelle

Processus de Sauvegarde Détailé

Le processus de sauvegarde s'exécute selon la séquence suivante :

1. Création d'un nouveau répertoire de sauvegarde horodaté
2. Journalisation du début de la sauvegarde
3. Sauvegarde des données partagées via rsync avec options de conservation des attributs
4. Sauvegarde des contenus web via rsync avec options de conservation des attributs
5. Création d'un sous-répertoire pour les bases de données
6. Export de chaque base de données utilisateur via mysqldump
7. Journalisation de la fin du processus de sauvegarde

Les options -avz de rsync garantissent que tous les attributs des fichiers sont préservés, que la sauvegarde est effectuée en mode archive, et que les données sont compressées pendant le transfert pour optimiser les performances.

Monitoring et Journalisation des Sauvegardes

Fichier de Log Dédié

Toutes les opérations de sauvegarde sont enregistrées dans un fichier de log dédié (backup.log) situé à la racine du répertoire de sauvegarde. Ce fichier contient :

- La date et l'heure de début de chaque sauvegarde
- Les détails des opérations rsync (fichiers copiés, taille, etc.)
- Les éventuelles erreurs rencontrées lors des exports de bases de données
- La date et l'heure de fin de chaque sauvegarde

Surveillance du Service

L'état du service de sauvegarde peut être vérifié à tout moment via la commande :

```
systemctl status backup.timer
```

Cette commande fournit des informations en temps réel sur :

- L'état actuel du service (actif ou inactif)

- La dernière exécution du service
- La prochaine exécution planifiée
- Les éventuelles erreurs rencontrées

Procédure de Restauration

En cas de besoin de restauration, l'administrateur peut suivre cette procédure :

1. Identifier la sauvegarde la plus pertinente à restaurer en fonction de l'horodatage
2. Pour les fichiers : utiliser rsync pour copier les données depuis le répertoire de sauvegarde vers l'emplacement d'origine
3. Pour les bases de données : importer les fichiers SQL via la commande mysql
4. Vérifier l'intégrité des données restaurées
5. Redémarrer les services concernés si nécessaire

Comparaison avec les Alternatives

Critère	Sauvegarde Complète (Implémentée)	Sauvegarde Incrémentielle AWS	Sauvegarde Différentielle
Complexité	Faible (script bash simple)	Élevée (intégration AWS Backup/Snapshots)	Moyenne (gestion des deltas)
RTO	Minutes (copie directe)	Variable (dépend de la chaîne)	Heures (reconstruction)
RPO	2 heures	Quelques minutes	2 heures
Risque d'Erreur	Faible (auto-contenu)	Élevé (dépendances multiples)	Moyen (calculs différentiels)
Coût Stockage	Élevé (duplication)	Faible (blocs modifiés)	Moyen (croissance progressive)

Le choix d'une sauvegarde complète s'avère donc optimal dans ce contexte spécifique :

-
- Adéquation parfaite aux contraintes pédagogiques et techniques du cahier des charges et des contraintes AWS
 - Fiabilité et traçabilité supérieures grâce à l'utilisation de systemd timers

Cette stratégie garantit une protection optimale des données critiques tout en minimisant l'impact sur les performances du système. La combinaison du stockage sur RAID et de sauvegardes fréquentes offre une excellente résilience face aux incidents potentiels.

8. Sécurisation de notre serveur

La sécurisation du serveur a été renforcée par l'implémentation de plusieurs mécanismes de protection mettant en avant la prévention et la détection aux menaces.

1. Firewall (firewalld) : Pare-feu dynamique permettant de contrôler le trafic réseau sur le serveur en autorisant différents ports selon les zones de confiance.

Par défaut, tous les ports sont bloqués.

Exemple de configuration :

```
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
    services: dhcpcv6-client ftp http https mdns mountd nfs rpc-bind samba ssh
    ports: 53/tcp 53/udp 30000-31000/tcp
    protocols:
    forward: yes
    masquerade: no
    forward-ports:
    source-ports:
    icmp-blocks:
    rich rules:
```

La zone public est la zone par défaut dans firewalld, nous acceptons dans celle-ci tous les services nécessaires à la réalisation de notre projet.

2. Fail2ban : Protection contre les attaques par force brute, fail2ban surveille constamment les fichiers de logs à la recherche de comportements suspects.

S'il détecte une tentative d'intrusion échouée, il bannit automatiquement cette IP en la bloquant pour une durée configuration.

Ce système protège les services sensibles (SSH, FTP) et réduit donc les risques d'accès non autorisé.

```
[ec2-user@sherpa ~]$ sudo fail2ban-client status
Status
`- Number of jail:      2
`- Jail list: sshd, vsftpd
```

3. SELinux : Contrôle d'accès par politique basée sur des contextes de sécurité (chaîne de 4 champs s'appliquant à un fichier et permettant de contrôler son accès).

SELinux permet de limiter ce qu'un service peut faire, même s'il est compromis.

Cette limitation permet de contenir les effets d'une attaque.

```
[ec2-user@sherpa ~]$ sestatus
SELinux status:          enabled
SELinuxfs mount:         /sys/fs/selinux
SELinux root directory:  /etc/selinux
Loaded policy name:     targeted
Current mode:           enforcing
Mode from config file:  enforcing
Policy MLS status:      enabled
Policy deny_unknown status: allowed
Memory protection checking: actual (secure)
Max kernel policy version: 33
```

4. Inotify : Service permettant de surveiller en temps réel des fichiers ou dossiers du système et de déclencher des actions en cas d'évènements détectés.

Exemple de commande utilisée :

```
inotifywait -m /mnt/raid1_web >> /var/log/web_changes.log
```

Cette commande permet de passer en mode surveillance continue sur le dossier web partagé et enregistre ensuite tous les événements détectés dans un fichier log.

Cela nous permet de retracer par la suite, toutes les activités sur les fichiers web des clients.

5. ClamAV : Antivirus open-source permettant de scanner des fichiers à la recherche de virus, chevaux de Troie ou tout autre type de logiciels malveillants.

Des analyses manuelles ou automatisées sont possibles.

Dans notre cas, souhaitant respecter la demande du client et voulant fournir un service utile sur tout type de machine, nous avons choisi de configurer ClamAV sans activer le service clamd.

Le service clamd impactant lourdement les ressources du système est donc déconseillé pour une machine de 2Go de RAM.

À la place de cela, nous avons opté pour un scan quotidien ciblé via clamscan planifié par cron ce qui nous permet de conserver une protection antivirus tout en maintenant la stabilité du serveur.

```
[ec2-user@sherpa ~]$ crontab -l
0 2 * * * clamscan -r /var/www >> /var/log/clamav/clamav_www.log 2>&1
```

Cette commande permet d'effectuer un scan sur le dossier web des clients chaque jour à 2 heure du matin et renvoie ensuite le résultat de cela dans un fichier clamav_www_log tout en capturant aussi les messages d'erreur.

9. Problèmes rencontrés

Le principal problème que nous avons rencontré concernait le manque de stabilité des instances EC2 vis-à-vis du réseau.

Étant donné que nous devions passer par un VPN et utiliser SSH pour travailler sur des machines hébergées dans le cloud, toute instabilité du réseau entraînait une perte temporaire de connexion, ce qui perturbait notre travail.

Nous avons également eu pas mal de soucis au niveau de clamd et de la RAM.

Clamd étant un démon qui consomme énormément de ressources, lors de sa mise en place, il est arrivé que la machine ne réponde plus car trop surchargée ce qui nous a valu de résilier cette machine et d'en recréer une.

La mise en place du FTP et Samba nous a posé quelques problèmes lors de l'ajout de SELinux, celui-ci bloquait partiellement ces services.

Après plusieurs ajustements, ces services sont redevenus opérationnels, mais cela nous a néanmoins occasionné une perte de temps.

10. Améliorations et conclusions

Premièrement, une meilleure gestion du temps et une anticipation des blocages auraient dû être mises en place.

Malgré une gestion globale satisfaisante, certaines difficultés techniques nous ont fait perdre un temps évitable, temps que nous aurions pu consacrer à peaufiner les autres services qui n'étaient pas encore parfaits.

Il aurait été utile de planifier davantage de tests pour détecter rapidement les problèmes potentiels.

Deuxièmement, approfondir nos compétences techniques pour la suite.

Dans l'ensemble, tout s'est bien passé, mais la mise en place de certains scripts a néanmoins posé des difficultés techniques.

Troisièmement, la mise en place de ClamAV aurait pu être améliorée.

Lors de la présentation, nous avons constaté que l'antivirus ne détectait pas un shell PHP introduit malgré son installation.

L'utilisation exclusive de clamscan en mode manuel, sans recourir au démon clamd n'empêche donc pas la détection de ce type de menaces.

Bien que clamd consomme plus de ressources, nous aurions pu trouver un compromis afin d'améliorer la détection tout en maîtrisant la consommation.

En conclusion, ce projet nous a permis de mettre à profit nos connaissances tout en nourrissant notre envie d'en apprendre davantage.

Malgré quelques obstacles techniques et imprévus, nous sommes fiers d'avoir livré une infrastructure fonctionnelle, cohérente et relativement robuste.

Ce travail nous a permis non seulement d'approfondir nos compétences en gestion d'infrastructure Linux, mais aussi de prendre pleinement conscience de l'importance de la collaboration, de la répartition des tâches et de la communication dans un projet technique.

11. Bibliographie

a) Sources électroniques

JohanCwiklinski (FEDORA), 2012, « SELinux », [en ligne] ;
<https://doc.fedoraproject.org/wiki/SELinux>

Ubuntu, « inotify - Surveiller les événements des systèmes de fichiers », [en ligne] ;
<https://manpages.ubuntu.com/manpages/bionic/fr/man7/inotify.7.html>

ClamAV, « ClamAV Documentation », [en ligne] ;
<https://docs.clamav.net/>

Linuxtricks.fr, « firewalld : Le pare-feu facile sous Linux », [en ligne] ;
<https://www.linuxtricks.fr/wiki/firewalld-le-pare-feu-facile-sous-linux>

Ubuntu, « Bannir des IP avec fail2ban », [en ligne] ;
<https://doc.ubuntu-fr.org/fail2ban>

b) Syllabus

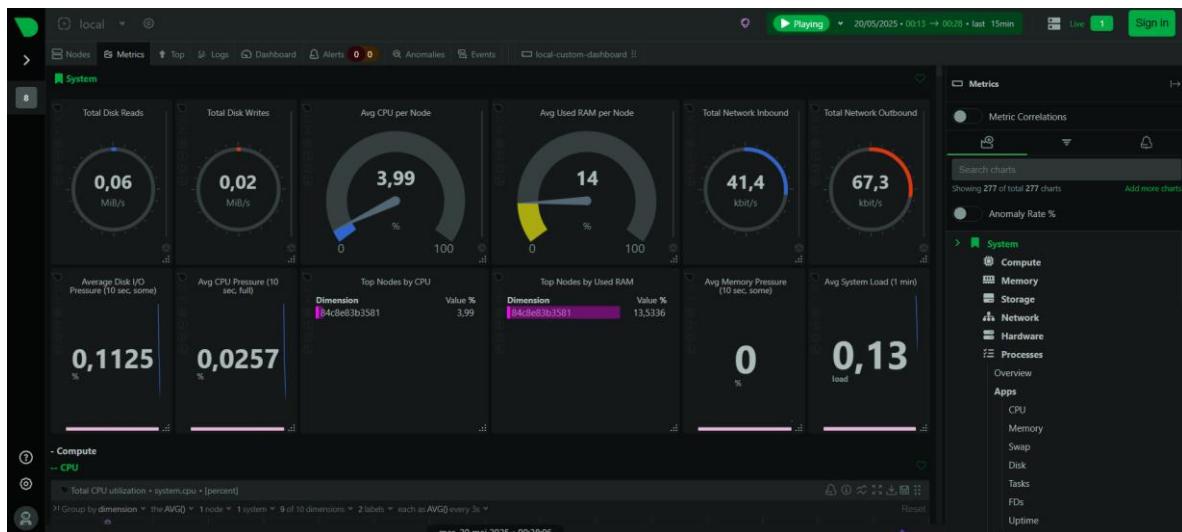
Malaise A., Administration Linux - théorie, Haute École en Hainaut à Mons,
Année académique 2024-2025

12. Annexes

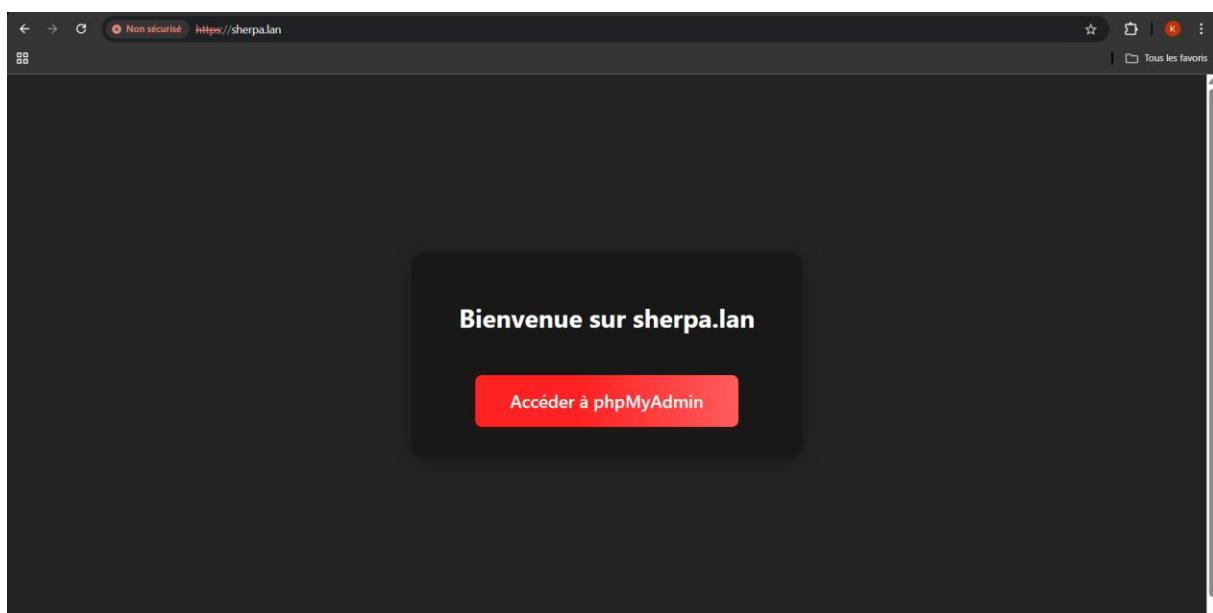
12.1. Lien du GitHub

https://github.com/EI-pabs/Linux_Server_Project

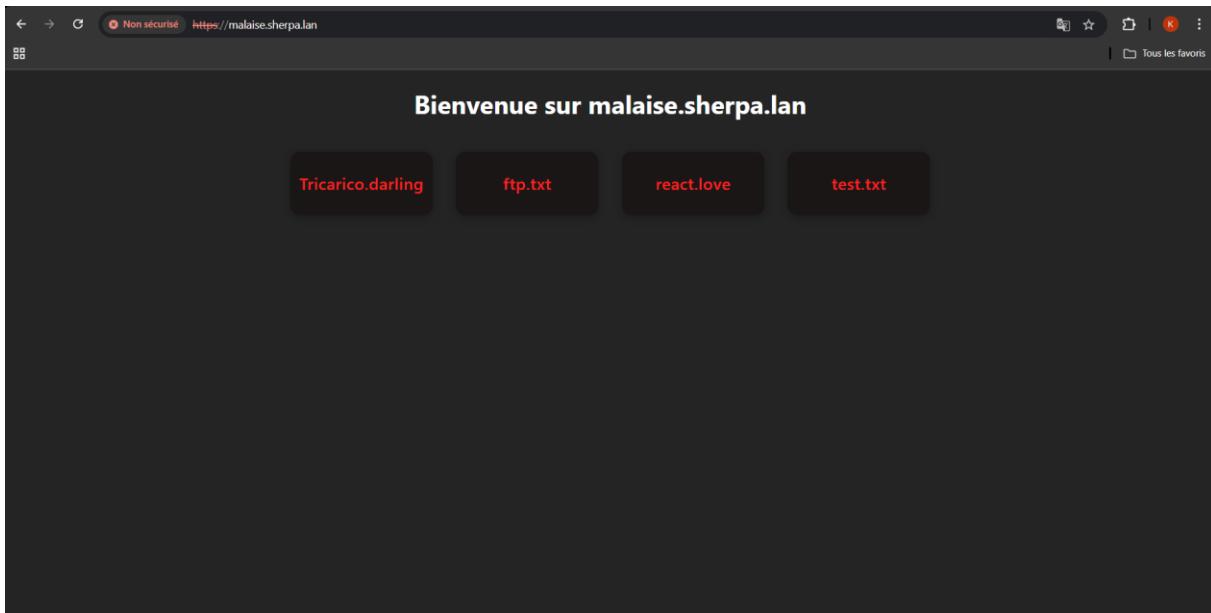
12.2. Interface monitoring



12.3. Interface web domaine



12.4. Interface web client



12.5. Database client

