

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

WWW.HEH.BE

Projet Linux

Rapport Final

Année Académique 2024-2025

IRT2

Hemeryck Quentin
Genart Alex

Table des matières

Table des matières	2
1. Introduction	4
2. Présentation générale du projet.....	4
3. Choix distribution et type d'installation.....	5
4. Plan de partitionnement	5
4.1. choix de RAID :	6
5. Présentation des services configurés.....	7
5.1. SSH	7
5.2. Fail2ban	8
5.3. Firewall	10
5.4. DNS.....	11
Dns Global.....	11
DNS client	13
5.5. Serveur de temps (NTP).....	14
5.6. NFS	15
NFS Global	15
Samba Global.....	16
Samba client (/var/www)	17
5.7. Quota.....	18
5.8. Apache.....	20
Apache domaine principal (linuxserver.lan)	20
Apache client	21
5.9. FTP.....	23
Sécurisation via SSL.....	23
Script d'installation serveur FTP	24
Script d'ajout client FTP.....	24
5.10. Base de données & MariaDB	24
Installation et démarrage de MariaDB.....	24
Sécurisation initiale (script MySqlSecure.sh).....	25
Installation et intégration de phpMyAdmin	25
Création automatisée des bases clients	26
Intégration côté interface web	27
5.11. Antivirus (Clamav)	27
5.12. Monitoring (NetData).....	28
5.13. Suppression d'un client	30

6.	Plan de sauvegarde	33
7.	Sécurisation du serveur.....	34
8.	Problèmes rencontrés et solutions	35
9.	Améliorations.....	36
9.1.	SELinux.....	36
9.2.	Sécurisation supplémentaire via options de montage.....	36
9.3.	Répartition des services sur plusieurs machines	37
10.	Conclusion.....	37
11.	Annexes	38
11.1.	Dépôt GitHub du projet.....	38

1. Introduction

Ce rapport présente le travail réalisé lors de cette seconde semaine de projet dédiée au déploiement et à la configuration de services sur des serveurs virtuels Linux, hébergés sur Amazon Web Services (AWS).



En respectant le cahier des charges, nous avons mis en place une infrastructure complète et automatisée grâce à des scripts Bash, pour installer, configurer et sécuriser divers services (DNS, Apache, FTP, Samba, MariaDB, NFS, NTP, monitoring...).

Le document retrace les choix techniques et méthodologiques, les étapes clés de l'implémentation, ainsi que les problèmes rencontrés et les solutions apportées. Il conclut par des pistes d'amélioration et un retour d'expérience.

Tout commentaire peut être envoyé à quentin.hemeryck@std.heh.be ou alex.genart@std.heh.be

2. Présentation générale du projet

Dans le cadre de notre deuxième année en informatique, spécialisation télécommunications et réseaux, nous avons travaillé par binôme sur la mise en place d'une infrastructure réseau complète et sécurisée sur un serveur virtuel Linux hébergé dans Amazon Web Services (AWS).

L'objectif a été de configurer et sécuriser différents services tout en automatisant leur installation via des scripts Bash, en respectant les bonnes pratiques vues en cours. Une seule instance EC2 sous Amazon Linux 2023 a été utilisée, regroupant à la fois la partie serveur et la partie cliente pour les phases de test.

Les services déployés ont compris :

- SSH sécurisé avec Fail2Ban,
- Partage de dossiers NFS et Samba sans authentification,
- Mise à disposition pour chaque client d'un sous-domaine, d'un serveur web, d'un accès FTP/Samba et d'une base de données dédiée,
- Serveur DNS (cache, maître de zone, zone inverse),
- Monitoring via interface web,
- Serveur NTP pour la synchronisation horaire,
- Plan de sauvegarde,
- Sécurisation avec SELinux, pare-feu et antivirus.

L'accès à la machine, située dans un réseau privé AWS, a été réalisé via un VPN OpenVPN avec un fichier de configuration personnalisé.

3. Choix distribution et type d'installation



La distribution utilisée pour ce projet a été imposée par les enseignants : il s'est agi d'Amazon Linux 2023, fournie par défaut lors de la création des instances EC2 sur AWS. Ce choix s'est révélé pertinent car cette distribution, basée sur RHEL et maintenue directement par AWS, a été spécialement optimisée pour ses services cloud.

Elle a offert plusieurs avantages :

- Optimisation pour AWS et compatibilité maximale avec ses services,
- Légèreté, rapidité de déploiement, stabilité et sécurité,
- Proximité avec RHEL/CentOS, facilitant l'application des notions vues en cours.

Le système a été déployé à partir d'une image AMI officielle, sans installation manuelle. L'essentiel du travail s'est concentré sur la configuration post-déploiement : mises à jour, installation de paquets, configuration des services, utilisateurs et règles de sécurité.

Le type d'installation a été headless (sans interface graphique), réduisant la consommation de ressources et la surface d'attaque potentielle. L'administration s'est faite exclusivement en ligne de commande via SSH.

4. Plan de partitionnement

Pour le choix du partitionnement, nous nous sommes basés sur la simplicité de gestion, la robustesse et la préservation des performances. L'instance EC2 utilisée a été configurée avec le type t3.medium, offrant ainsi davantage de ressources pour supporter l'ensemble des services.

Point de montage	Périphérique	Type de FS	Taille	Option de montage
/	/dev/nvme0n1p1	xfs	8 Go	Defaults, noatime
/boot/efi	/dev/nvme0n1p128	vfat	10 Mo	Defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount
/srv/nfs/share	/dev/vg_raid1/nfs_share	ext4	500 Mo	Defaults,usrquota,grpquota
/var/www	/dev/vg_raid1/web	ext4	500 Mo	Defaults,usrquota,grpquota
/backup	/dev/vg_raid1/backup	ext4	1 Go	defaults

La partition racine (/) située sur /dev/nvme0n1p1 (8 Go) a contenu tous les répertoires standards du système Linux (/home, /var, /etc...). Nous n'avons pas dédié de partitions spécifiques à /home ou /var car :

- /home a été peu sollicité, les utilisateurs se connectant principalement via SSH.
- Les données critiques ont été stockées sur les volumes RAID.
- /var, bien qu'hébergeant les journaux systèmes, est resté peu volumineux.

Une partition /boot/efi a été présente conformément aux exigences du démarrage UEFI.

Pour garantir la redondance, les volumes /dev/nvme1n1 et /dev/nvme2n1 ont été assemblés en RAID 1 logiciel via mdadm. Ce volume RAID a ensuite été divisé en trois volumes logiques LVM :

- /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 a facilité la gestion des droits, des sauvegardes et des quotas, tout en permettant un redimensionnement simple si nécessaire.

Aucune partition swap n'a été mise en place, la configuration t3.medium offrant suffisamment de RAM pour couvrir les besoins des services. Ce choix a permis de préserver les performances des disques NVMe et de réduire leur usure.

4.1. choix de RAID :

Nous avons mis en place un RAID 1 (mirroring) afin d'assurer une redondance complète des données sur les deux volumes de taille similaire disponibles. Ce mode de fonctionnement permet de conserver une copie exacte des données sur chaque disque, offrant ainsi une protection efficace contre la perte en cas de défaillance matérielle.

Le RAID 1 a été choisi pour sa simplicité de mise en œuvre, ses performances de lecture optimisées et sa fiabilité dans le cadre du projet, même si la capacité utile est réduite à 50 % de l'espace total.

5. Présentation des services configurés

5.1. SSH

La configuration du service SSH a été renforcée afin de réduire les risques d'intrusion et de respecter les bonnes pratiques de sécurité. Un script automatisé a été utilisé pour sauvegarder le fichier de configuration existant, appliquer des règles strictes et redémarrer le service pour prendre en compte les changements.

Avant toute modification, le script définit deux variables pour stocker les chemins du fichier de configuration SSH et de sa sauvegarde.

```
CONFIG_FILE="/etc/ssh/sshd_config"
BACKUP_FILE="/etc/ssh/sshd_config.bak"
```

Ces variables permettent d'éviter les répétitions dans le script et de modifier facilement les chemins si nécessaire.

Le script vérifie ensuite si une sauvegarde existe déjà. Si ce n'est pas le cas, il la crée :

```
if [[ ! -f $BACKUP_FILE ]]; then
    cp "$CONFIG_FILE" "$BACKUP_FILE"
    echo "Sauvegarde du fichier de configuration existant effectuée : $BACKUP_FILE"
else
    echo "Une sauvegarde existe déjà : $BACKUP_FILE"
fi
```

Cela garantit qu'en cas d'erreur ou de problème, il sera possible de restaurer la configuration d'origine.

Les paramètres essentiels sont stockés dans un tableau associatif CONFIGS puis parcourus par une boucle pour être ajoutés ou modifiés dans le fichier /etc/ssh/sshd_config.

```
declare -A CONFIGS=(
    ["Protocol"]="2"
    ["PermitRootLogin"]="no"
    ["PasswordAuthentication"]="no"
    ["PubkeyAuthentication"]="yes"
    ["ChallengeResponseAuthentication"]="no"
    ["AllowAgentForwarding"]="no"
    ["PermitTunnel"]="yes"
    ["X11Forwarding"]="no"
    ["MaxAuthTries"]="3"
    ["UsePAM"]="yes"
    ["ClientAliveInterval"]="300"
    ["ClientAliveCountMax"]="2"
    ["LoginGraceTime"]="300"
    ["LogLevel"]="VERBOSE"
)
```

```
for KEY in "${!CONFIGS[@]}"; do
    VALUE="${CONFIGS[$KEY]}"
    if grep -q "^$KEY" "$CONFIG_FILE"; then
        sed -i "s/^$KEY.*$/KEY $VALUE/" "$CONFIG_FILE"
    else
        echo "$KEY $VALUE" >> "$CONFIG_FILE"
    fi
done
```

Ce mécanisme permet d'appliquer automatiquement toutes les options nécessaires sans risquer d'oublier une ligne et en assurant leur cohérence.

Détails des principaux paramètres appliqués

- PasswordAuthentication no
-> Désactive l'authentification par mot de passe, vulnérable aux attaques par force brute.
- PermitRootLogin no
-> Empêche toute connexion directe au compte root.
- PubkeyAuthentication yes
-> Active l'authentification par clé publique, plus sécurisée.
- MaxAuthTries 3
-> Limite à 3 les tentatives d'authentification avant coupure de la connexion.
- ClientAliveInterval 300 et ClientAliveCountMax 2
-> Déconnecte un utilisateur inactif après environ 10 minutes.
- Protocol 2
-> Force l'utilisation de SSH version 2, plus sûre que la v1.
- X11Forwarding no
-> Bloque le transfert graphique inutile et potentiellement risqué.
- LogLevel VERBOSE
-> Augmente le niveau de journalisation pour faciliter le suivi et le débogage.

Une fois les modifications appliquées, le script redémarre le service SSH pour qu'elles soient immédiatement effectives :

```
sudo systemctl restart sshd
```

Le redémarrage est indispensable pour que les nouvelles règles prennent effet.

5.2. Fail2ban

Fail2Ban a été mis en place pour protéger le serveur contre les tentatives d'accès non autorisées en bannissant automatiquement les adresses IP suspectes après plusieurs échecs de connexion.

Une variable stocke l'adresse IP d'administration afin qu'elle ne soit jamais bannie par erreur :

```
# Variables
IP_SRV=$(hostname -I | awk '{print $1}')
IP_ADMIN="172.20.10.2"
```

Cela évite qu'un administrateur réseau perde l'accès au serveur à cause d'un bannissement.

Le script vérifie ensuite si Fail2Ban est installé et l'installe si nécessaire :

```
# Vérifie si Fail2Ban est déjà installé
if ! command -v fail2ban-server &> /dev/null; then
    sudo dnf install -y fail2ban || { echo "Échec de l'installation de Fail2Ban. Abandon."; exit 1; }
else
    echo "Fail2Ban est déjà installé."
fi
```

Cette vérification empêche une réinstallation inutile et garantit que Fail2Ban est disponible.

Création du dossier de configuration :

```
# Crée le dossier de configuration si nécessaire
sudo mkdir -p /etc/fail2ban/jail.d
```

Les fichiers de configuration spécifiques (jails) seront placés dans ce répertoire.

Fail2Ban a été configuré pour protéger le service SSH via un fichier sshd.local :

Explications des paramètres principaux :

- enabled = true → Active la surveillance pour SSH.
- port = 22 → Port du service SSH.
- maxretry = 3 → Bannissement après 3 tentatives échouées.
- bantime = 600 → Bannissement de 10 minutes.
- findtime = 600 → Les tentatives échouées sont comptées sur 10 minutes.
- ignoreip → Exclusion de l'IP de l'administrateur.

```
# Configure Fail2Ban pour protéger le service SSH
cat <<EOF | sudo tee /etc/fail2ban/jail.d/sshd.local > /dev/null
[sshd]
enabled = true
port = 22
action = %(action_mwl)s
logpath = %(sshd_log)s
maxretry = 3
bantime = 600
findtime = 600
ignoreip = $IP_ADMIN
EOF
```

Un fichier vsftpd.local a été créé sur le même principe pour protéger le service FTP :

```
# Configure Fail2Ban pour protéger le service FTP
cat <<EOF | sudo tee /etc/fail2ban/jail.d/vsftpd.local > /dev/null
[vsftpd]
enabled = true
port = 21
action = %(action_mwl)s
logpath = /var/log/vsftpd.log
maxretry = 3
bantime = 600
findtime = 600
ignoreip = $IP_ADMIN
EOF
```

Seules les valeurs du port et du chemin des logs diffèrent par rapport à la configuration SSH.

Une fois les jails configurées, le service Fail2Ban est activé et démarré automatiquement :

```
# Active et démarre le service Fail2Ban
sudo systemctl enable fail2ban || { echo "Échec de l'activation de Fail2Ban. Abandon."; exit 1; }
sudo systemctl start fail2ban || { echo "Échec du démarrage de Fail2Ban. Abandon."; exit 1; }
```

Cette étape rend la protection active immédiatement et persistante au redémarrage.

5.3. Firewall

Afin de n'exposer que les services nécessaires tout en bloquant le reste, le pare-feu firewalld a été configuré et activé. Les règles ci-dessous reprennent ce qui a été appliqué par notre script.

a) Mise en service de firewalld

```
# Démarrer le service firewalld
sudo systemctl start firewalld
sudo systemctl enable firewalld
```

Lance le démon et l'active au démarrage de la machine.

b) Autorisations par service / port

```
# Script pour configurer le pare-feu avec firewalld
sudo firewall-cmd --permanent --add-service=dns
sudo firewall-cmd --permanent --add-service=ssh
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --permanent --add-service=ftp
sudo firewall-cmd --permanent --add-port=30000-30100/tcp
sudo firewall-cmd --permanent --add-service=samba
sudo firewall-cmd --permanent --add-service=nfs
sudo firewall-cmd --permanent --add-service=mysql
sudo firewall-cmd --permanent --add-service=ntp
sudo firewall-cmd --permanent --add-port=19999/tcp
```

- SSH/DNS : nécessaires à l'admin distante et à la résolution interne.
- HTTP/HTTPS : publication des sites (redirection HTTPS active côté Apache).
- FTP : ouverture de la plage 30000–30100/tcp pour le mode passif (indispensable derrière pare-feu/NAT).
- Samba/NFS : partages de fichiers internes.
- MySQL (MariaDB) : accès BDD lorsque requis par les clients/outils.
- NTP : synchronisation horaire du réseau.
- 19999/tcp : tableau de bord NetData.

c) Application des règles

```
sudo firewall-cmd --reload
```

Recharge la configuration et rend les règles persistantes sans couper les connexions actives

d) Principes suivis

Notre configuration du pare-feu s'appuie sur ces principes clairs :

- Blocage par défaut → tout trafic entrant est refusé, seules des ouvertures explicites sont permises.
- Besoin strict → on n'ouvre que les ports/protocoles indispensables au fonctionnement prévu.
- Superposition défensive → le pare-feu complète Fail2ban, SELinux, TLS et le durcissement des services.
- Segmentation précise → interfaces rangées par zones (public/interne/VPN) et règles ciblées par service, port et source IP en IPv4/IPv6 ; seuls les réseaux/hôtes attendus atteignent le service.

e) Spécificités de la configuration

- FTP passif : ouverture strictement bornée à 30000–30100/TCP pour les transferts.
- Monitoring : exposition du 19999/TCP pour l'interface NetData uniquement.
- Partages : autorisation des services Samba et NFS pour l'échange de fichiers.
- Base de données : MariaDB limité à localhost par défaut ; ouverture distante ponctuelle et ciblée si besoin.

Ces règles complètent la défense en profondeur tout en ne laissant passer que le trafic nécessaire.

5.4. DNS

Dns Global

Le service DNS a été installé et configuré automatiquement grâce à un script Bash. L'objectif était de mettre en place un serveur BIND jouant à la fois le rôle de serveur maître pour notre domaine et de cache DNS pour les requêtes externes, avec une zone directe et une zone inverse.

Le script commence par installer BIND et ses utilitaires, puis définit les variables nécessaires pour le domaine, l'adresse IP du serveur et les chemins des fichiers de zones :

```
DOMAIN="linuxserver.lan"
REVERSE_ZONE="0.42.10.in-addr.arpa"
DNS_IP=$(hostname -I | awk '{print $1}')
LAST_OCTET=$(echo "$DNS_IP" | awk -F. '{print $4}')
ZONE_FILE="/var/named/${DOMAIN}.zone"
REVERSE_ZONE_FILE="/var/named/0.42.10.rev"
NAMED_CONF="/etc/named.conf"
SERIAL=$(date +%Y%m%d)01
```

Ces variables centralisent la configuration et facilitent les modifications ultérieures.

a) Création de la zone directe

Le fichier de zone directe associe les noms de domaine à leurs adresses IP et définit le serveur d'autorité.

```
echo "[*] Création de la zone directe"
sudo tee "$ZONE_FILE" > /dev/null <<EOF
\$TTL 86400
@ IN SOA ns1.${DOMAIN}. admin.${DOMAIN}. (
    $SERIAL ; Serial
    3600    ; Refresh
    1800    ; Retry
    1209600 ; Expire
    86400   ; Minimum TTL
)
@ IN NS ns1.${DOMAIN}.
@ IN A ${DNS_IP}
ns1 IN A ${DNS_IP}
EOF
```

b) Création de la zone inverse

```
echo "[*] Création de la zone reverse"
sudo tee "$REVERSE_ZONE_FILE" > /dev/null <<EOF
\$TTL 86400
@ IN SOA ns1.${DOMAIN}. admin.${DOMAIN}. (
    $SERIAL ; Serial
    3600    ; Refresh
    1800    ; Retry
    1209600 ; Expire
    86400   ; Minimum TTL
)
@ IN NS ns1.${DOMAIN}.
${LAST_OCTET} IN PTR ns1.${DOMAIN}.
EOF
```

La zone inverse permet la résolution IP → nom de domaine grâce à un enregistrement PTR.

c) Configuration du fichier named.conf

```
sudo cat <<EOF > "$NAMED_CONF"
options {
    listen-on port 53 { 127.0.0.1; $DNS_IP; }; ### DNS principal #####
    listen-on-v6 { none; };
    directory      "/var/named";
    dump-file     "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query { localhost; any; }; ### Clients VPN #####
    recursion yes;
    dnssec-validation yes;
    bindkeys-file "/etc/named.iscdlv.key";
    managed-keys-directory "/var/named/dynamic";
    pid-file     "/run/named/named.pid";
    session-keyfile "/run/named/session.key";
    forwarders {
        1.1.1.1;
        8.8.8.8;
    };
};
```

```
logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

zone "${DOMAIN}" IN {
    type master;
    file "$ZONE_FILE";
    allow-update { none; };
};

zone "$REVERSE_ZONE" IN {
    type master;
    file "$REVERSE_ZONE_FILE";
    allow-update { none; };
};

EOF
```

Le serveur écoute sur 127.0.0.1 et 10.42.0.1, accepte les requêtes locales et VPN, et redirige vers Cloudflare et Google si la résolution locale échoue. Les deux zones (directe et inverse) sont déclarées comme zones maîtres.

DNS client

Pour automatiser l'ajout de sous-domaines au serveur DNS, un script DNSClient.sh a été développé. Ce script prend en argument le nom du client et effectue les étapes suivantes : vérification d'existence, ajout de l'entrée DNS, mise à jour du numéro de série et redémarrage de BIND.

Vérification si le sous-domaine existe déjà :

```
# Vérifie si le sous-domaine existe déjà
if grep -q "^$CLIENT\s" "$ZONE_FILE"; then
    echo "Le sous-domaine $CLIENT.$DOMAIN existe déjà."
    exit 0
fi
```

Évite d'insérer plusieurs fois la même entrée dans la zone DNS.

Ajout dynamique de l'entrée A dans la zone directe :

```
# Ajouter l'entrée A
echo "$CLIENT      IN      A      $DNS_IP" | sudo tee -a "$ZONE_FILE"
```

Crée une entrée \$CLIENT.linuxserver.lan pointant vers l'adresse IP du serveur DNS.

Incrémation automatique du numéro de série :

```
# Incrémenter automatiquement le Serial
# Génère une nouvelle date + numéro (ex: 2025051302)
NEW_SERIAL=$(date +%Y%m%d)02

# Remplace l'ancien Serial (ligne contenant SOA)
sudo sed -i -E "s/([0-9]{10}) ; Serial/${NEW_SERIAL} ; Serial/" "$ZONE_FILE"
```

Met à jour le Serial pour que BIND prenne en compte la modification de zone.

Redémarrage de BIND

```
# Redémarrer BIND
sudo systemctl restart named && echo "Sous-domaine $CLIENT.$DOMAIN ajouté et BIND rechargé"
```

Recharge les nouvelles entrées DNS afin qu'elles soient actives immédiatement.

5.5. Serveur de temps (NTP)

Pour garantir une heure fiable sur l'ensemble de la plateforme, chrony a été configuré comme client des pools publics européens et comme source de temps locale pour le réseau.

a) Sauvegarde et remise à plat de la configuration

```
# Sauvegarde de la configuration existante
sudo cp /etc/chrony.conf /etc/chrony.conf.bak

# Nettoyage des anciennes lignes 'pool'
sudo sed -i '/^pool /d' /etc/chrony.conf
```

- Crée un backup avant modification.
- Supprime les directives pool existantes afin de repartir d'une base claire.

b) Déclaration des sources publiques + service local

- Trois serveurs pool.ntp.org (Europe) avec iburst pour une synchro initiale rapide.
- allow 0.0.0.0/0 : le serveur accepte les requêtes NTP des clients du réseau.
- local stratum 10 : sert d'horloge de secours si les sources externes sont inaccessibles

```
# Ajout de serveurs publics et autorisation pour le réseau local
cat <<EOF | sudo tee -a /etc/chrony.conf

# Serveurs de temps publics
server 0.europe.pool.ntp.org iburst
server 1.europe.pool.ntp.org iburst
server 2.europe.pool.ntp.org iburst

# Autoriser tous les clients du réseau
allow 0.0.0.0/0
local stratum 10
EOF
```

c) fuseau horaire

```
# Réglage du fuseau horaire
echo "[~] Mise à jour du fuseau horaire vers Europe/Brussels"
sudo timedatectl set-timezone Europe/Brussels
```

- Aligne le fuseau du système sur Europe/Brussels.

d) Redémarrage et mise à l'heure immédiate

```
# Redémarrage du service NTP
sudo systemctl restart chronyd

# Correction immédiate de l'heure
sudo chronyc makestep
```

- Redémarre chronyd puis force l'application immédiate de l'écart horaire.

e) Pare-feu (port UDP 123)

```
# Ouvrir le port NTP si firewalld est actif
if systemctl is-active --quiet firewalld; then
    sudo firewall-cmd --permanent --add-service=ntp
    sudo firewall-cmd --reload
else
    echo "[!] Firewalld non actif. Vérifie l'ouverture du port UDP 123 si nécessaire."
fi
```

- Ouvre le service NTP si firewalld est en place, sinon rappelle de vérifier la règle réseau.

f) Vérification

```
# Affichage de l'état final
echo "Configuration du serveur NTP terminée. Fuseau horaire et synchro appliqués."
chronyc tracking
```

- Affiche l'état de la synchronisation (source, offset, fréquence, etc.)

Bénéfices de la configuration :

- Horodatages cohérents pour tous les services (logs, certificats, sessions).
- Point de référence de temps pour les clients du réseau (fonction “serveur NTP” local).
- Résilience : fonctionnement local même si les serveurs publics sont momentanément injoignables

5.6. NFS

NFS Global

Le service NFS a été configuré pour permettre un accès réseau au répertoire partagé depuis des machines clientes Linux. L'objectif était de mettre en place un partage simple et sécurisé, accessible en lecture seule, pour éviter toute modification involontaire des fichiers partagés.

a) Crédit du répertoire partagé

```
# Variables
NFS_SHARE_DIR="/srv/nfs/share"
EXPORTS_FILE="/etc/exports"
SAMBA_CONF="/etc/samba/smb.conf"

# Crédit du répertoire pour le partage NFS
echo "Crédit du répertoire de partage NFS...""
sudo mkdir -p "$NFS_SHARE_DIR"
sudo chown -R nobody:nobody "$NFS_SHARE_DIR"
sudo chmod -R 777 "$NFS_SHARE_DIR"
```

Ces commandes créent le répertoire à partager, attribuent l'utilisateur et le groupe nobody pour éviter les accès privilégiés, et accordent des droits complets en lecture/écriture/exécution à tous.

b) Activation du service NFS

```
# Activation et démarrage des services NFS
echo "Activation et démarrage des services NFS...""
sudo systemctl enable rpcbind --now
sudo systemctl enable nfs-server --now
```

Ces services doivent être actifs pour que NFS fonctionne correctement et pour que le partage soit accessible immédiatement.

c) Configuration du partage NFS

```
# Configuration de l'export NFS
echo "Configuration de l'export NFS..."
if ! grep -q "$NFS_SHARE_DIR" "$EXPORTS_FILE"; then
    echo "$NFS_SHARE_DIR *(rw,sync,no_subtree_check)" | sudo tee -a "$EXPORTS_FILE" > /dev/null
fi
```

- rw -> lecture et écriture autorisées
- sync -> Écritures synchronisées, garantissant l'intégrité des données.
- no_subtree_check -> Améliore les performances en évitant la vérification des sous-arborescences.

d) Export du partage

```
sudo exportfs -rav
```

Recharge la configuration et applique les règles définies dans /etc(exports

Samba Global

Configuration du partage Samba pour permettre l'accès réseau à un dossier partagé.

a) Définition des variables globales :

```
# Variables
NFS_SHARE_DIR="/srv/nfs/share"
EXPORTS_FILE="/etc/exports"
SAMBA_CONF="/etc/samba/smb.conf"
```

Stocke les chemins nécessaires pour la configuration Samba et NFS.

b) Crédit du répertoire partagé :

```
# Crédit du répertoire pour le partage NFS
echo "Crédit du répertoire de partage NFS..."
sudo mkdir -p "$NFS_SHARE_DIR"
sudo chown -R nobody:nobody "$NFS_SHARE_DIR"
sudo chmod -R 777 "$NFS_SHARE_DIR"
```

Crée le dossier partagé et définit les permissions pour un accès invité complet.

c) Ajout de la configuration Samba :

- [share] : nom visible du partage.
- path : chemin physique du dossier partagé.
- browseable : rend le partage visible sur le réseau.
- writable / read only : contrôle les droits d'écriture.
- guest ok / guest only : accès sans authentification.

```
[share]
path = $NFS_SHARE_DIR
browseable = yes
writable = yes
read only = no
guest ok = yes
guest only = yes
EOF
fi
```

d) Activation et démarrage des services Samba :

```
# Activation et démarrage des services Samba
echo "Activation et démarrage des services Samba..."
sudo systemctl enable smb --now
sudo systemctl enable nmb --now
```

Active et démarre les services nécessaires au fonctionnement du partage Samba.

Samba client (/var/www)

Configuration d'un partage Samba dédié à un client, situé dans /var/www/\$CLIENT. Chaque client dispose d'un dossier personnel, accessible uniquement par lui via Samba.

a) Variables :

- CLIENT → Nom du client passé en argument au script.
- SAMBA_PWD → Mot de passe Samba attribué au client.
- DOCUMENT_ROOT → Chemin du dossier web du client.
- SAMBA_CONF → Fichier de configuration Samba.

```
# Variables
CLIENT=$1
SAMBA_PWD=$2
DOCUMENT_ROOT="/var/www/$CLIENT"
SAMBA_CONF="/etc/samba/smb.conf"
```

b) Création de l'utilisateur et configuration Samba

```
# Créer l'utilisateur Samba s'il n'existe pas
if ! id "$CLIENT" &>/dev/null; then
    sudo useradd -M -s /sbin/nologin "$CLIENT"
    echo "Utilisateur UNIX $CLIENT créé."
fi

# Définir un mot de passe Samba (invite l'admin)
echo "Définir un mot de passe Samba pour l'utilisateur $CLIENT :"
echo -e "$SAMBA_PWD\n$SAMBA_PWD" | sudo smbpasswd -s -a "$CLIENT"
```

- Vérifie si l'utilisateur existe déjà, sinon le crée.
- Définit le mot de passe Samba du client.

d) Droits sur le dossier client :

```
# Donner les droits sur le dossier
sudo mkdir -p "$DOCUMENT_ROOT"
sudo chown -R "$CLIENT:$CLIENT" "$DOCUMENT_ROOT"
sudo chmod -R 770 "$DOCUMENT_ROOT"
```

- Crée le dossier du client s'il n'existe pas.
- Associe le dossier au client et à son groupe.
- Donne les droits lecture/écriture/exécution au propriétaire et groupe

e) Redémarrage du service Samba

- Vérifie si le partage existe déjà, sinon l'ajoute.
- browseable = yes → visible sur le réseau.
- writable = yes → autorise l'écriture.
- valid users → seul le client a accès.
- create mask et directory mask → définissent les droits sur les fichiers et dossiers créés.

```
# Ajouter le partage Samba si non existant
if ! grep -q "^\[$CLIENT\]$" "$SAMBA_CONF"; then
    sudo bash -c "cat >> $SAMBA_CONF <<EOF

[$CLIENT]
path = $DOCUMENT_ROOT
browseable = yes
writable = yes
valid users = $CLIENT
create mask = 0770
directory mask = 0770
EOF"
    echo "Partage ajouté pour $CLIENT"
else
    echo "Le partage $CLIENT existe déjà dans smb.conf"
fi
```

f) Redémarrage du service Samba

- Recharge la configuration Samba pour appliquer les changements

```
# Redémarrer Samba
sudo systemctl restart smb
```

5.7. Quota

Les quotas ont été mis en place pour limiter l'espace disque alloué à chaque client sur les volumes critiques. Leur configuration s'est déroulée en deux phases : activation technique lors de la mise en place des volumes (LVM.sh) et attribution lors de la création d'un environnement client (Environnement.sh).

a) Activation et initialisation des quotas (LVM.sh)

Lors de la configuration LVM et RAID, les volumes /var/www (sites web) et /srv/nfs/share (partages NFS) sont montés avec les options usrquota,grpquota :

```
sudo mkdir -p /srv/nfs/share
UUID_NFS=$(blkid -s UUID -o value /dev/vg_raid1/nfs_share)
grep -q "$UUID_NFS" /etc/fstab || echo "UUID=$UUID_NFS /srv/nfs/share ext4 defaults,usrquota,grpquota 0 0" >> /etc/fstab
sudo mount /srv/nfs/share 2>/dev/null || sudo mount -o remount,usrquota,grpquota /srv/nfs/share
```

```
sudo mkdir -p /var/www
UUID_WEB=$(blkid -s UUID -o value /dev/vg_raid1/web)
grep -q "$UUID_WEB" /etc/fstab || echo "UUID=$UUID_WEB /var/www ext4 defaults,usrquota,grpquota 0 0" >> /etc/fstab
sudo mount /var/www 2>/dev/null || sudo mount -o remount,usrquota,grpquota /var/www
```

Ensuite, avec le script :

```
# Remontage avec quotas avant initialisation
echo "[INFO] Remontage avec quotas..."
sudo mount -o remount,usrquota,grpquota /var/www
sudo mount -o remount,usrquota,grpquota /srv/nfs/share

# Initialiser les quotas
echo "[INFO] Initialisation des quotas..."
# Désactiver les quotas temporairement s'ils sont actifs
sudo quotaoff /var/www 2>/dev/null || true
sudo quotaoff /srv/nfs/share 2>/dev/null || true

# Forcer la vérification des quotas (création des fichiers quota)
sudo quotacheck -cufm /var/www
sudo quotacheck -cufm /srv/nfs/share

# Créer les fichiers de quotas de groupe s'ils n'existent pas
sudo quotacheck -cugm /var/www 2>/dev/null || true
sudo quotacheck -cugm /srv/nfs/share 2>/dev/null || true

# Réactiver les quotas
sudo quotaon /var/www 2>/dev/null || echo "[WARNING] Quota utilisateur activé pour /var/www, groupe ignoré si inexistant"
sudo quotaon /srv/nfs/share 2>/dev/null || echo "[WARNING] Quota utilisateur activé pour /srv/nfs/share, groupe ignoré si inexistant"
```

- Remontage des volumes avec quotas activés.
- Vérification et création des fichiers de suivi (aquota.user, aquota.group).
- Activation des quotas utilisateurs et groupes.

b) Attribution des limites à un client (Environnement.sh)

```
SOFT_LIMIT=${SOFT_LIMIT:-102400} # 100 Mo
HARD_LIMIT=${HARD_LIMIT:-153600} # 150 Mo

# Quotas
sudo setquota -u "$CLIENT" $SOFT_LIMIT $HARD_LIMIT 0 0 /var/www
sudo setquota -u "$CLIENT" $SOFT_LIMIT $HARD_LIMIT 0 0 /srv/nfs/share
```

- SOFT_LIMIT : seuil d'avertissement (100 Mo) avant blocage.
- HARD_LIMIT : limite maximale (150 Mo) au-delà de laquelle aucune donnée supplémentaire ne peut être stockée.
- Application simultanée sur /var/www et /srv/nfs/share.

Avantages :

- Les quotas sont actifs dès la configuration du serveur grâce à LVM.sh.
- Chaque client dispose d'un espace contrôlé et personnalisé via Environnement.sh.
- Évite la saturation des volumes partagés et garantit une répartition équitable des ressources.

5.8. Apache

Apache domaine principal (`linuxserver.lan`)

Création et configuration du domaine principal sur Apache, avec certificat SSL auto-signé et redirection HTTP → HTTPS.

c) Variables

```
# Variables
DOMAIN="linuxserver.lan"
DOC_ROOT="/var/www/$DOMAIN"
SSL_DIR="/etc/ssl/$DOMAIN"
CONF_FILE="/etc/httpd/conf.d/$DOMAIN.conf"
```

- DOMAIN → Nom du domaine principal.
- DOC_ROOT → Chemin du dossier racine du site web.
- SSL_DIR → Répertoire de stockage des fichiers SSL.
- CONF_FILE → Chemin vers le fichier de configuration Apache du domaine

b) Création du certificat SSL auto-signé

```
# Préparation
echo "[INFO] Création du dossier de certificat : $SSL_DIR"
mkdir -p "$SSL_DIR"

echo "[INFO] Génération du certificat auto-signé pour $DOMAIN"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout "$SSL_DIR/$DOMAIN.key" \
    -out "$SSL_DIR/$DOMAIN.crt" \
    -subj "/C=BE/ST=Hainaut/L=Mons/O=LinuxServerCorp/OU=Web/CN=$DOMAIN"
```

- mkdir -p "\$SSL_DIR" → Crée le dossier pour stocker le certificat.
- openssl req -x509 ... → Génère un certificat auto-signé valable 1 an.
- -keyout → Fichier contenant la clé privée.
- -out → Fichier contenant le certificat public.
- -subj → Informations associées au certificat.

c) Création de la page d'accueil

```
# Création de la page d'accueil
mkdir -p "$DOC_ROOT"
cat <<'EOF' > "$DOC_ROOT/index.html"
<!DOCTYPE html>
<html lang="fr">
<head>...
</head>
<body>...
</body>
</html>
EOF
```

- Crée le dossier racine du site.
- Ajoute une page HTML servant de portail (vers NetData ou phpMyAdmin).

d) Fichier de configuration Apache

- *VirtualHost :80 -> Écoute sur le port 80 et redirige vers HTTPS.
- *VirtualHost :443 -> Écoute sur le port 443 en HTTPS.
- DocumentRoot -> Définit le répertoire racine du site.
- SSLCertificateFile -> Chemin vers le certificat.
- SSLCertificateKeyFile -> Chemin vers la clé privée.
- <Directory> -> Autorise tous les visiteurs et active .htaccess

```
# Création du fichier de configuration Apache
echo "[INFO] création du fichier de configuration Apache"
cat <<EOF > "$CONF_FILE"
<VirtualHost *:80>
    ServerName $DOMAIN
    Redirect permanent / https://$DOMAIN/
</VirtualHost>

<VirtualHost *:443>
    ServerName $DOMAIN
    DocumentRoot $DOC_ROOT

    SSLEngine on
    SSLCertificateFile $SSL_DIR/$DOMAIN.crt
    SSLCertificateKeyFile $SSL_DIR/$DOMAIN.key

    <Directory $DOC_ROOT>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
EOF
```

e) Redémarrage d'Apache

```
# Redémarrage du service Apache
echo "[INFO] Redémarrage d'Apache"
systemctl restart httpd
```

Recharge la configuration pour appliquer les changements.

Apache client

a) Variables

```
# Variables
CLIENT=$1
DOMAIN="$CLIENT.linuxserver.lan"
DOCUMENT_ROOT="/var/www/$CLIENT"
VHOST_CONF="/etc/httpd/conf.d/$CLIENT.conf"
CERT_FILE="/etc/pki/tls/certs/$DOMAIN.crt"
KEY_FILE="/etc/pki/tls/private/$DOMAIN.key"
```

- CLIENT -> Nom du client passé en paramètre, utilisé pour générer le domaine et les chemins.
- DOMAIN -> Nom de domaine du client (ex. client1.linuxserver.lan).
- DOCUMENT_ROOT -> Chemin du dossier web du client.

- VHOST_CONF -> Fichier de configuration Apache dédié au client.
- CERT_FILE / KEY_FILE -> Emplacement du certificat SSL et de sa clé privée.

b) Génération du certificat auto-signé

- Génère un certificat SSL valide un an si inexistant.
- Utilise RSA 2048 bits et désactive la protection par mot de passe (-nodes).

```
# Générer certificat auto-signé s'il n'existe pas déjà
if [[ ! -f "$CERT_FILE" || ! -f "$KEY_FILE" ]]; then
    echo "[*] Générer un certificat auto-signé pour $DOMAIN"
    sudo openssl req -x509 -nodes -days 365 \
        -newkey rsa:2048 \
        -keyout "$KEY_FILE" \
        -out "$CERT_FILE" \
        -subj "/C=BE/ST=Hainaut/L=Mons/O=LinuxServerCorp/OU=Web/CN=$DOMAIN"
else
    echo "[*] Certificat déjà existant pour $DOMAIN, pas de régénération"
fi
```

c) Configuration Apache (HTTP → HTTPS)

```
# Crée un fichier de configuration Apache pour le client avec HTTP + HTTPS
sudo bash -c "cat > $VHOST_CONF <<EOF
# Redirection HTTP -> HTTPS
<VirtualHost *:80>
    ServerName $DOMAIN
    Redirect permanent / https://$DOMAIN/
</VirtualHost>

# VirtualHost HTTPS
<VirtualHost *:443>
    ServerAdmin quentin.hemeryck@std.heh.be
    DocumentRoot $DOCUMENT_ROOT
    ServerName $DOMAIN
    ErrorLog /var/log/httpd/${CLIENT}_error.log
    CustomLog /var/log/httpd/${CLIENT}_access.log combined

    SSLEngine on
    SSLCertificateFile $CERT_FILE
    SSLCertificateKeyFile $KEY_FILE

    <Directory $DOCUMENT_ROOT>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
EOF"
```

- Port 80 -> Redirige toutes les requêtes HTTP vers HTTPS.
- Port 443 -> Active SSL et pointe vers le répertoire du client.
- AllowOverride All -> Autorise l'utilisation de fichiers .htaccess.
- Require all granted -> Accès libre à tout le répertoire.

d) Page d'accueil par défaut

```
# Crée un fichier index.html par défaut
if [ ! -f "$DOCUMENT_ROOT/index.html" ]; then
    echo "<h1>Bienvenue sur votre domaine $CLIENT: $DOMAIN </h1>" | sudo tee "$DOCUMENT_ROOT/index.html" > /dev/null
fi
```

- Crée un fichier index.html personnalisé si absent

f) Permissions Apache

```
sudo usermod -aG $CLIENT apache
sudo chmod 750 /var/www/$CLIENT
```

- Ajoute Apache au groupe du client.
- Autorise uniquement le propriétaire et le groupe à accéder aux fichiers.

g) Redémarrage du service

```
# Redémarrer Apache
sudo systemctl restart httpd
echo "Le site $DOMAIN est configuré avec HTTP → HTTPS"
echo "DocumentRoot : $DOCUMENT_ROOT"
echo "Fichier de configuration : $VHOST_CONF"
```

- Recharge Apache pour appliquer la configuration.
- Affiche un résumé des paramètres appliqués.

5.9. FTP

Le service FTP a été mis en place avec vsftpd afin de permettre aux utilisateurs d'accéder à leurs fichiers tout en garantissant la sécurité et l'isolation des comptes.

Configuration du service FTP dans le fichier /etc/vsftpd/vsftpd.conf :

- anonymous_enable=NO → Interdit les connexions anonymes pour éviter les accès non autorisés.
- local_enable=YES → Autorise les utilisateurs locaux à se connecter.
- write_enable=YES → Permet l'écriture sur le serveur FTP.
- chroot_local_user=YES → Confine l'utilisateur dans son répertoire personnel.
- allow_writeable_chroot=YES → Évite les blocages lorsque le dossier personnel est accessible en écriture.
- userlist_enable=YES → Active une liste blanche des utilisateurs autorisés.
- local_root=/var/www/\$USER → Définit le répertoire racine de l'utilisateur sur son dossier web.
- pasv_min_port=30000 & pasv_max_port=30100 → Définit la plage de ports passifs pour faciliter le passage du pare-feu.

```
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=002
user_sub_token=\$USER
local_root=/var/www/\$USER
secure_chroot_dir=/var/run/vsftpd/empty

pasv_min_port=30000
pasv_max_port=30100

setproctitle_enable=YES

listen_port=21
listen=YES
listen_ipv6=NO

pam_service_name=vsftpd
```

Sécurisation via SSL

```
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
EOF
```

- Active et impose l'utilisation de SSL/TLS pour les connexions FTP.
- Utilise un certificat auto-signé pour chiffrer les communications.

Génération du certificat :

```
echo "[*] Génération du certificat SSL auto-signé"
sudo mkdir -p /etc/pki/tls/private /etc/pki/tls/certs
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/pki/tls/private/vsftpd.key \
-out /etc/pki/tls/certs/vsftpd.pem \
-subj "/C=BE/ST=Hainaut/L=Mons/O=LinuxServerCorp/CN=ftp.linuxserver.lan"
```

Script d'installation serveur FTP

```
# Installer vsftpd si pas encore installé
echo "[*] Installation de vsftpd"
sudo dnf install -y vsftpd

echo "[*] Sauvegarde de l'ancienne configuration"
sudo cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.bak || true

echo "[*] Création du répertoire de sécurité"
sudo mkdir -p /var/run/vsftpd/empty
```

- Installe vsftpd.
- Sauvegarde l'ancienne configuration.
- Crée le répertoire chroot requis.
- Redémarre le service.

```
echo "Redémarrage du service FTP"
sudo systemctl restart vsftpd
```

Script d'ajout client FTP

```
# Variables
CLIENT=$1
DOCUMENT_ROOT="/var/www/$CLIENT"

# Installer vsftpd si nécessaire
sudo dnf install -y vsftpd

# Ajouter un utilisateur FTP
echo "$CLIENT" | sudo tee -a /etc/vsftpd/user_list
sudo usermod -d $DOCUMENT_ROOT $CLIENT

# Configurer les permissions d'accès
sudo chmod 755 $DOCUMENT_ROOT
sudo chown $CLIENT:$CLIENT $DOCUMENT_ROOT

# Redémarrer vsftpd
sudo systemctl restart vsftpd
echo "FTP configuré pour $CLIENT avec accès au dossier $DOCUMENT_ROOT."
```

- Ajoute le client dans la liste blanche FTP.
- Définit son dossier personnel dans /var/www/<client>.
- Configure les permissions.
- Redémarre le service pour appliquer les changements.

5.10. Base de données & MariaDB

La base de données, MariaDB 10.6 a été installée depuis le dépôt officiel, puis sécurisée via un script automatisé. phpMyAdmin a été déployé sous le vHost principal pour une administration web en HTTPS.

Installation et démarrage de MariaDB

```
# Installer mariadb
sudo tee /etc/yum.repos.d/MariaDB.repo > /dev/null <<EOF
[mariadb]
name = MariaDB
baseurl = https://yum.mariadb.org/10.6/rhel9-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF

sudo dnf install -y MariaDB-server MariaDB-client

sudo systemctl start mariadb
```

- Ajoute le repo officiel MariaDB 10.6 puis installe serveur + client.
- Démarrer le service et l'active au boot.

Sécurisation initiale (script MySqlSecure.sh)

```
# Choix mdp root MariaDB
read -s -p "Entrez le mot de passe root que vous désirez : " MARIADB_ROOT_PASSWORD
echo
read -s -p "Confirmez le mot de passe root : " MARIADB_ROOT_PASSWORD_CONFIRM
echo

if [ "$MARIADB_ROOT_PASSWORD" != "$MARIADB_ROOT_PASSWORD_CONFIRM" ]; then
    echo "Les mots de passe ne correspondent pas. Abandon."
    exit 1
fi

# Sauvegarde du mot de passe root pour les scripts
echo "$MARIADB_ROOT_PASSWORD" | sudo tee /root/.mariadb_root_pass > /dev/null
sudo chmod 600 /root/.mariadb_root_pass

echo "[*] Configuration sécurisée de MariaDB..."

# Forcer l'utilisation de mysql_native_password pour root
sudo mysql <<EOF
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '$MARIADB_ROOT_PASSWORD';
FLUSH PRIVILEGES;
EOF
```

- Demande un mot de passe root, le valide et le stock de façon restreinte pour les scripts.

- Uniformise l'authentification de root (compat outils)

```
# Sécurisation classique
if command -v mysql_secure_installation &> /dev/null; then
    sudo mysql_secure_installation <<EOF
$MARIADB_ROOT_PASSWORD
n
n
Y
Y
Y
Y
EOF
```

- Automatisation de la sécurisation: supprimer les utilisateurs anonymes, interdire root distant, drop la base test, flush des privilèges.

Installation et intégration de phpMyAdmin

```
# Chemin d'installation correct
INSTALL_DIR="/var/www/linuxserver.lan/phpMyAdmin"
TARBALL="phpMyAdmin-latest-all-languages.tar.gz"
DOWNLOAD_URL="https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz"

echo "[*] Installation de wget et tar"
sudo dnf install -y wget tar > /dev/null 2>&1

# Créer le dossier racine s'il n'existe pas
sudo mkdir -p /var/www/linuxserver.lan

cd /var/www/linuxserver.lan

echo "[*] Téléchargement de phpMyAdmin"
sudo wget "$DOWNLOAD_URL" -O "$TARBALL" > /dev/null 2>&1

echo "[*] création du dossier $INSTALL_DIR"
sudo mkdir -p "$INSTALL_DIR"

echo "[*] Extraction de l'archive"
sudo tar -xvzf "$TARBALL" -C "$INSTALL_DIR" --strip-components=1 > /dev/null 2>&1

echo "[*] Suppression de l'archive $TARBALL"
sudo rm -f "$TARBALL"
```

- Déploie phpMyAdmin dans le site du domaine principal /var/www/<domaine>/phpMyAdmin.
- L'accès se fait en HTTPS via le vHost 443 déjà configuré (certificat auto-signé). (*Conforme au déploiement web effectué plus haut.*)

Création automatisée des bases clients

Pour éviter les manipulations répétitives, la génération d'un couple base/utilisateur par client a été industrialisée via un script. Celui-ci demande un mot de passe, le valide, puis crée la base, le compte SQL et assigne les droits.

a) saisie et contrôle du mot de passe :

- Entrée masquée + confirmation pour limiter les erreurs.
- En cas d'incohérence, arrêt propre du script.

```
# Demande un mot de passe à l'utilisateur pour la base de données
read -s -p "Mot de passe DB pour $CLIENT : " DB_PASSWORD
echo
read -s -p "Confirmez le mot de passe : " DB_PASSWORD_CONFIRM
echo

if [ "$DB_PASSWORD" != "$DB_PASSWORD_CONFIRM" ]; then
    echo "Les mots de passe ne correspondent pas. Abandon."
    exit 1
fi
```

b) Provisionnement SQL (base + utilisateur + priviléges) :

```
# Variables
CLIENT=$1
DOMAIN=$2
FTP_PASSWORD=$3
DB_PASS=$4

DOCUMENT_ROOT="/var/www/$CLIENT"
FTP_USER=$CLIENT
DB_NAME="$CLIENT"
DB_USER="$CLIENT"

# Base MariaDB + utilisateur
sudo mysql -u root -p"$MYSQL_ROOT_PWD" <<EOF
CREATE DATABASE IF NOT EXISTS \`$DB_NAME\`;
CREATE USER IF NOT EXISTS '$DB_USER'@'localhost' IDENTIFIED BY '$DB_PASS';
CREATE USER IF NOT EXISTS '$DB_USER'@'%' IDENTIFIED BY '$DB_PASS';
GRANT ALL PRIVILEGES ON \`$DB_NAME\`.* TO '$DB_USER'@'localhost';
GRANT ALL PRIVILEGES ON \`$DB_NAME\`.* TO '$DB_USER'@'%';
FLUSH PRIVILEGES;
EOF
```

- Crée une base dédiée nommée comme le client.
- Ajoute l'utilisateur SQL pour localhost et (optionnel) pour toute IP (%).
- Accorde les droits uniquement sur la base du client (moindre privilège).
- Le mot de passe root MariaDB est lu depuis /root/.mariadb_root_pass (permissions 600) pour l'automatisation.

Atouts :

- Isolation forte -> 1 base + 1 compte par client ; aucun droit hors de sa base.
- Moindre privilège -> priviléges limités à db_client.* ; aucun GRANT global.
- Traçabilité claire -> même identifiant pour la base et l'utilisateur => logs/audits simples.
- Gestion facilitée -> création/suppression/rotation des mots de passe par client sans impacter les autres.
- Compatibilité outils -> accès via phpMyAdmin et CLI ; support localhost et (optionnel) accès distant contrôlé (%).
- Réversibilité -> supprimer un client = supprimer son user et sa base, sans effets de bord.
- Secrets mieux protégés -> mots de passe distincts ; secret root lu depuis /root/.mariadb_root_pass (600).
- Scalable & reproductible -> même script réutilisable pour un grand nombre de clients.

Intégration côté interface web

Pour l'administration graphique, phpMyAdmin a été raccordé au vHost HTTPS du domaine principal.

- Ajout d'un lien d'accès vers phpMyAdmin depuis la page d'accueil du site principal.
- Servi sous `https://linuxserver.lan/phpMyAdmin` via le VirtualHost :443 déjà configuré.
- Transport chiffré (TLS) en s'appuyant sur le certificat du site.
- Chaque client s'authentifie avec son compte SQL et ne voit que sa base (droits limités côté SGBD).

Résultat : une administration web simple et sécurisée, tout en conservant l'isolation stricte entre clients.

5.11. Antivirus (Clamav)

Pour renforcer la sécurité du serveur, ClamAV a été installé et configuré afin d'analyser régulièrement les répertoires critiques (`/var/www`, `/etc`). Les signatures virales ont été mises à jour automatiquement et un scan quotidien a été planifié, avec journalisation centralisée des résultats.

- a) Installation de ClamAv et de son démon clamd

```
# Installer ClamAV + clamd (le démon)
sudo dnf install -y clamav clamav-update clamav-server
```

Installe le moteur ClamAV, l'outil de mise à jour des signatures et le service de scan (clamd).

- b) Mise à jour de la base virale (freshclam)

```
# Mettre à jour la base de signatures virales
sudo sed -i 's/^Example/#Example/' /etc/freshclam.conf
sudo freshclam
echo " Base virale mise à jour (freshclam)"
```

Active la configuration de freshclam (en commentant Example) puis télécharge les dernières signatures.

- c) Configuration du démon clamd

```
# Configurer clamd pour qu'il fonctionne avec le socket
sudo sed -i 's/^Example/#Example/' /etc/clamd.d/scan.conf
sudo sed -i 's/^#LocalSocket /LocalSocket /' /etc/clamd.d/scan.conf
sudo sed -i 's|^LocalSocket .*|LocalSocket /run/clamd.scan/clamd.sock|' /etc/clamd.d/scan.conf
echo "Configuration du démon clamd prête"
```

Configure l'usage du socket local `/run/clamd.scan/clamd.sock` requis par le service.

d) Activation du service clamd

```
# Activer et démarrer le démon
sudo systemctl enable --now clamd@scan
echo "Service clamd lancé"
```

Démarre clamd et l'active au démarrage.

e) Vérification du socket

```
# Vérifier que le socket est bien lancé
if [ ! -S /run/clamd.scan/clamd.sock ]; then
    echo "[...] Attente du socket clamd"
    sleep 5
fi
```

Attend que le socket soit disponible avant de poursuivre.

f) Premier scan initial

```
# Créer le dossier des logs si besoin
sudo mkdir -p /var/log/clamav
sudo touch /var/log/clamav/scan.log
echo "Log prêt : /var/log/clamav/scan.log"
```

Analyse les répertoires sensibles (/var/www, /etc).

g) Planification automatique (scan quotidien à 3 h)

```
# Configurer un cron pour scanner automatiquement chaque nuit
CRON_FILE="/etc/cron.d/clamav-scan"

sudo mkdir -p /etc/cron.d
sudo tee "$CRON_FILE" > /dev/null <<EOF
0 3 * * * root clamscan --log=/var/log/clamav/scan.log /var/www /etc
EOF
```

Lance un scan chaque nuit à 03:00 et consigne le résultat dans /var/log/clamav/scan.log

5.12. Monitoring (NetData)

Pour superviser en temps réel l'état du serveur (CPU, RAM, disque, réseau, services), **NetData** a été déployé dans un conteneur **Docker**. Le choix du conteneur simplifie l'installation, l'isolation et les mises à jour.

a) installation de Docker

```
echo "Installation de Docker"
sudo dnf install -y docker
```

Installe le moteur de conteneurs requis pour exécuter NetData.

b) Activation du service Docker

```
echo "Démarrage et activation de Docker"
sudo systemctl enable --now docker
```

Active Docker au démarrage et lance le service immédiatement.

c) Ajout de l'utilisateur au groupe docker

```
echo "Ajout de ec2-user au groupe docker"
sudo usermod -aG docker ec2-user
```

Autorise l'utilisateur par défaut (ec2-user) à lancer des conteneurs sans sudo (reconnexion requise pour prise en compte du groupe).

d) Déploiement de NetData (conteneur Docker)

```
# [AJOUT] Nettoyage : supprimer un conteneur Netdata existant pour éviter le conflit
sudo docker rm -f netdata >/dev/null 2>&1 || true

echo "Déploiement du conteneur Netdata"
sudo docker run -d --name=netdata \
-p 19999:19999 \
-v netdataconfig:/etc/netdata \
-v netdatalib:/var/lib/netdata \
-v netdatacache:/var/cache/netdata \
-v /etc/passwd:/host/etc/passwd:ro \
-v /etc/group:/host/etc/group:ro \
-v /proc:/host/proc:ro \
-v /sys:/host/sys:ro \
-v /etc/os-release:/host/etc/os-release:ro \
--cap-add SYS_PTRACE \
--security-opt apparmor=unconfined \
netdata/netdata
```

- -p 19999:19999 → expose l'interface Web de NetData.
- Volumes netdataconfig/lib/cache → persistent la config et les données NetData.
- Montages passwd, group, proc, sys, os-release (en **lecture seule**) → permettent la collecte de métriques hôte.

- SYS_PTRACE + apparmor=unconfined → autorisent certains collecteurs/sondes de NetData.
- Le conteneur tourne en arrière-plan (-d).

e) Ouverture du port 19999 (pare-feu)

```
echo "Ouverture du port 19999 dans Firewalld si actif"
if systemctl is-active --quiet firewalld; then
    # règle existante (zone par défaut)
    sudo firewall-cmd --permanent --add-port=19999/tcp
    # [AJOUT] aussi dans la zone docker si elle est active
    sudo firewall-cmd --zone=docker --permanent --add-port=19999/tcp >/dev/null 2>&1 || true
    sudo firewall-cmd --reload
else
    echo "Firewalld non actif, tu dois ouvrir le port 19999 dans les règles de sécurité AWS si nécessaire."
fi
```

- Ouvre l'accès au tableau de bord si firewalld est utilisé.
- Si firewalld n'est pas actif, ouvrir le Security Group AWS sur le port 19999/TCP.

f) Accès au tableau de bord

```
echo "Netdata installé. Accès : http://linuxserver.lan:19999"
```

- URL : <http://linuxserver.lan:19999>
- Vue temps réel : charge CPU/RAM, I/O disque, réseau, services, alertes.

5.13. Suppression d'un client

Pour retirer complètement un client de l'infrastructure, le script qui a été mis en place supprime toutes ses ressources (Web/SSL, DNS, Base de données, comptes Linux/FTP/Samba, quotas) et met en place un blocage de domaine empêchant toute réutilisation accidentelle.

a) Saisie et contrôle

```
# Demande du nom du client
read -p "Entrez le nom du client à supprimer : " CLIENT
if [[ -z "$CLIENT" ]]; then
    echo "[ERREUR] Le nom du client est obligatoire."
    exit 1
fi

# Vérification anti-suppression root
if [[ "$CLIENT" == "root" ]]; then
    echo "[ERREUR] Impossible de supprimer l'utilisateur root."
    exit 1
fi
```

- Demande le nom du client à supprimer.
- Refuse si aucun nom ou si l'utilisateur est root.

b) Protection et vérification

- Interdit la suppression du domaine principal linuxserver.lan.
- Vérifie que l'utilisateur existe dans le système.

```
# Protection du domaine principal
if [[ "$CLIENT" == "linuxserver" ]]; then
    echo "[ERREUR] Impossible de supprimer le domaine principal linuxserver.lan."
    exit 1
fi

# Vérification existence utilisateur
if ! id "$CLIENT" &>/dev/null; then
    echo "[ERREUR] L'utilisateur '$CLIENT' n'existe pas."
    exit 1
fi
```

c) Variables de travail

```
DOMAIN="$CLIENT.linuxserver.lan"

DOCUMENT_ROOT="/var/www/$CLIENT"
VHOST_CONF="/etc/httpd/conf.d/$CLIENT.conf"
CERT_FILE="/etc/pki/tls/certs/$DOMAIN.crt"
KEY_FILE="/etc/pki/tls/private/$DOMAIN.key"
ZONE_FILE="/var/named/linuxserver.lan.zone"

DB_NAME="$CLIENT"
DB_USER="$CLIENT"
```

- Définit toutes les variables pour chemins, noms de domaine, certificats et base de données.

d) Web et certificats

```
# 1. Suppression Apache
if [ -f "$VHOST_CONF" ]; then
    sudo rm -f "$VHOST_CONF"
    echo "[✓] Fichier de configuration Apache supprimé"
fi
if [ -d "$DOCUMENT_ROOT" ]; then
    sudo rm -rf "$DOCUMENT_ROOT"
    echo "[✓] Dossier web supprimé"
fi
sudo rm -f "$CERT_FILE" "$KEY_FILE" && echo "[✓] Certificats SSL supprimés"
```

- Supprime le vhost Apache, le document root et les certificats dédiés

e) VirtualHost de blocage (HTTP/HTTPS)

- Interdit toute réutilisation du domaine ; pour HTTPS, utilise le certificat central.

```
# 1bis. Création d'un VirtualHost de blocage
BLOCK_VHOST="/etc/httpd/conf.d/${CLIENT}_blocked.conf"
sudo bash -c "cat > $BLOCK_VHOST <<EOF
<VirtualHost *:80>
    ServerName $DOMAIN
    Redirect 403 /
</VirtualHost>

<VirtualHost *:443>
    ServerName $DOMAIN
    SSLEngine on
    SSLCertificateFile /etc/ssl/linuxserver.lan/linuxserver.lan.crt
    SSLCertificateKeyFile /etc/ssl/linuxserver.lan/linuxserver.lan.key
    Redirect 403 /
</VirtualHost>
EOF"
```

f) DNS (zone locale)

- ```
2. Suppression DNS
if [-f "$ZONE_FILE"] && grep -q "^\$CLIENT\s" "$ZONE_FILE"; then
 sudo sed -i "/^\$CLIENT\s/d" "$ZONE_FILE"
 echo "[✓] Entrée DNS supprimée"
 NEW_SERIAL=$(date +%Y%m%d)$(printf "%02d" 01)
 sudo sed -i -E "s/([0-9]{10}) ; Serial/${NEW_SERIAL} ; Serial/" "$ZONE_FILE"
 sudo systemctl restart named
fi
```
- Retire l'entrée DNS du client dans le fichier de zone.
  - Met à jour le numéro de série (serial) et redémarre named pour appliquer.

#### g) Suppression Base MariaDB

```
3. Suppression Base de données + utilisateur MariaDB
if [-f /root/.mariadb_root_pass]; then
 MYSQL_ROOT_PWD=$(cat /root/.mariadb_root_pass)
 sudo mysql -u root -p"$MYSQL_ROOT_PWD" <<EOF
DROP DATABASE IF EXISTS `\$DB_NAME`;
DROP USER IF EXISTS '\$DB_USER'@'localhost';
DROP USER IF EXISTS '\$DB_USER'@'%';
FLUSH PRIVILEGES;
EOF
```

- Supprime la base de données et les utilisateurs SQL liés.
- Rafraîchit les priviléges pour appliquer les changements.

#### h) Suppression comptes Linux/FTP/Samba

- Supprime le compte Samba et l'utilisateur Linux.
- Efface le dossier personnel /home/\$CLIENT.

```
4. Suppression utilisateur FTP / Samba / Linux + /home
sudo smbpasswd -x "$CLIENT" 2>/dev/null
sudo pkill -u "$CLIENT" 2>/dev/null
sudo userdel -r "$CLIENT" 2>/dev/null
sudo rm -rf "/home/$CLIENT" 2>/dev/null
echo "[√] utilisateur et /home supprimés"
```

#### i) Réinitialisation des quotas

```
5. Suppression quota
sudo setquota -u "$CLIENT" 0 0 0 0 /var/www 2>/dev/null
sudo setquota -u "$CLIENT" 0 0 0 0 /srv/nfs/share 2>/dev/null
echo "[√] Quotas réinitialisés"
```

- Remet les quotas à zéro sur les volumes web et NFS.

#### j) Nettoyage Samba

```
6. Nettoyage du partage Samba
SAMBA_CONF="/etc/samba/smb.conf"
if grep -q "^\[$CLIENT\]$" "$SAMBA_CONF"; then
 sudo sed -i "/^\[$CLIENT\]/,/^\$/d" "$SAMBA_CONF"
 echo "[√] Partage Samba supprimé"
 sudo systemctl restart smb
fi
```

- Supprime le partage Samba dédié au client.
- Redémarre Samba pour appliquer la modification.

#### k) Redémarrage des services

```
7. Redémarrage des services principaux
sudo systemctl restart httpd
sudo systemctl restart named
sudo systemctl restart vsftpd
```

- Redémarre Apache, DNS et FTP pour appliquer toutes les suppressions.

#### Avantages :

- Automatisation complète : tout est retiré en une seule exécution.
- Sécurité : garde-fous pour root et domaine principal, confirmation manuelle, blocage du domaine supprimé.
- Cohérence : nettoyage Web, DNS, SQL, comptes Linux/FTP/Samba, quotas.
- Fiabilité : exécution répétable sans erreurs grâce aux suppressions conditionnelles.

## 6. Plan de sauvegarde

Le plan de sauvegarde définit précisément quelles données doivent être protégées, où elles seront stockées et à quelle fréquence les opérations seront effectuées. L'objectif est de limiter au maximum la perte de données en cas d'incident (panne matérielle, erreur humaine, attaque) et de garantir une restauration rapide.

Afin d'élaborer une stratégie efficace, plusieurs points ont été définis :

Données sauvegardées :

- Fichiers de configuration système : /etc
- Sites web : /var/www
- Répertoires utilisateurs : /home
- Bases de données MySQL/MariaDB : export complet via mysqldump
- Archives de sauvegardes précédentes : présentes dans /backup

Les capacités de stockage étant limitées, seules les données essentielles sont incluses dans la sauvegarde.

Stratégie de sauvegarde :

- Fréquence : Sauvegarde complète quotidienne à 03h00 via une tâche cron planifiée.
- Emplacement : Les archives sont stockées dans /backup avec un dossier horodaté (exemple : 2025-08-14\_03-00-00).
- Format :
  - Compression en .tar.gz pour les répertoires /etc, /var/www et /home
  - Fichier .sql pour les bases de données
- Automatisation : Script backup.sh qui :
  - Crée automatiquement le dossier de destination avec date et heure
  - Lance l'export des données
  - Stocke la sauvegarde au bon emplacement
  - Configure la tâche cron si nécessaire

Le temps nécessaire à l'exécution n'est pas un problème car la sauvegarde est effectuée à un moment où le serveur n'est pas sollicité.

Politique de conservation :

Une rotation des sauvegardes aurait pu être mise en place afin de maîtriser l'espace disque. Par exemple :

- Conservation des sauvegardes quotidiennes pendant 7 jours
- Conservation des sauvegardes hebdomadaires pendant 1 mois
- Conservation des sauvegardes mensuelles pendant 1 an
- Suppression automatique des sauvegardes plus anciennes

Cette stratégie n'a toutefois pas été implémentée dans le script actuel.

#### Procédure de restauration :

Une procédure de restauration aurait pu être également définie afin de garantir l'efficacité des sauvegardes. Par exemple :

- Un fichier ou répertoire : extraction via tar -xvf archive.tar.gz
- Une base de données : restauration avec mysql -u root -p < all\_databases.sql
- Un serveur complet : réimportation successive des archives /etc, /var/www, /home et des bases SQL
- Tests réguliers : une restauration de test au moins une fois par mois pour valider l'intégrité des sauvegardes

Cependant, cette procédure n'a pas été mise en place ni testée dans le cadre de ce projet

#### Avantages :

- Sauvegardes entièrement automatisées, sans intervention manuelle
- Organisation claire grâce à l'horodatage des dossiers
- Restauration simplifiée grâce à la séparation par catégories (configurations, données web, utilisateurs, bases de données)

#### Limites actuelles :

- Pas de rotation automatique : les anciennes sauvegardes ne sont pas supprimées, ce qui peut saturer l'espace disque
- Sauvegardes uniquement locales : pas encore de synchronisation vers un stockage externe (AWS S3, NAS, etc.)
- Pas de vérification automatique d'intégrité : une restauration test manuelle est nécessaire pour valider la sauvegarde

## 7. Sécurisation du serveur

La sécurisation du serveur a été réalisée en intégrant plusieurs couches de protection afin de limiter les risques d'intrusion, de perte de données et de défaillance des services. Ces mesures ont couvert les aspects réseau, accès utilisateur, services exposés, intégrité des données et surveillance système.

- a) Sécurisation des accès SSH : SSH.sh
  - Interdiction de la connexion directe root.
  - Désactivation de l'authentification par mot de passe au profit des clés SSH.
  - Redémarrage du service pour appliquer les changements.
- b) Pare-feu : Firewall.sh
  - Filtrage strict des ports ouverts.
  - Autorisation uniquement des services nécessaires (SSH, HTTP/HTTPS, DNS, FTP, Samba, etc.).

- 
- c) Protection contre les attaques par force brute : Fail2Ban.sh
    - Surveillance des logs SSH et blocage temporaire des adresses IP suspectes.
    - Fichier de configuration adapté pour réduire les faux positifs
  - d) Sécurité web et chiffrement SSL : SSLDomain.sh
    - Génération de certificats SSL auto-signés pour HTTPS.
    - Redirection automatique HTTP → HTTPS.
  - e) Sécurité des services Samba et FTP : Samba.sh, FTP.sh
    - Samba : restrictions d'accès par IP, authentification obligatoire, partage limité aux répertoires autorisés.
    - FTP : comptes isolés dans leur répertoire (chroot\_local\_user=YES), désactivation des connexions anonymes.
  - f) Sécurité du système et intégrité (SELinux.sh)
    - SELinux configuré en mode permissive afin de journaliser les violations sans bloquer les actions. *Ps : raisons de ce choix dans la section améliorations*
    - Permet de conserver un suivi des tentatives d'accès non autorisé tout en évitant les interruptions de service.
  - g) Protection antivirus (AV.sh)
    - Installation et mise à jour automatiques de ClamAV.
    - Analyse planifiée des répertoires critiques.
  - h) Sauvegardes et restauration (backup.sh)
    - Sauvegardes compressées avec date.
    - Stockage dans un répertoire dédié /backup pour archivage et restauration manuelle.

## 8. Problèmes rencontrés et solutions

Plusieurs problèmes techniques ont été rencontrés durant la réalisation de cette seconde semaine de projet. Tout d'abord, le service Apache échouait au redémarrage (Job for httpd.service failed), ce qui bloquait le script. Le passage à une instance EC2 t3.medium a résolu ce manque de ressources.

Lors de la suppression d'un client, son répertoire /home et certaines ressources restaient présents. Le script DeleteClient.sh a été modifié pour fermer les sessions actives, supprimer complètement les données, réinitialiser les quotas et retirer les partages Samba.

L'accès web à de nouveaux domaines échouait sur le réseau local (port 443 inaccessible), mais fonctionnait en 4G. Le problème venait de la résolution DNS interne et a été corrigé par une configuration adaptée.

---

Avec MariaDB, le root n'affichait pas tous les comptes dans phpMyAdmin, et les utilisateurs créés par SetupClient.sh ne pouvaient pas se connecter. De plus, leurs comptes restaient actifs après suppression. Les scripts ont été corrigés : MySqlSecure.sh force désormais l'authentification mysql\_native\_password pour root, Environnement.sh crée les utilisateurs pour localhost et %, et DeleteClient.sh supprime bases, comptes et accès restants.

Enfin, Netdata a posé plusieurs problèmes liés à son déploiement en conteneur Docker : absence de service netdata.service sous systemd, conflit de nom avec un ancien conteneur déjà présent, et ouverture du port 19999 dans la mauvaise zone firewall. Les correctifs apportés au script NetData.sh automatisent désormais la suppression de l'ancien conteneur, ouvrent le port dans la zone docker et permettent un accès fonctionnel via <http://linuxserver.lan:19999>.

Après ces ajustements, la création et la suppression de clients fonctionnent entièrement, et Netdata est accessible et opérationnel après chaque déploiement.

## 9. Améliorations

### 9.1. SELinux

Une des améliorations notable concerne néanmoins SELinux.

Nous avions prévu de l'activer en mode enforcing afin de renforcer la sécurité, et un script de configuration a été intégré au projet.

Cependant, plusieurs contraintes nous ont empêchés de finaliser pleinement son déploiement :

- Incompatibilités avec certains services déjà configurés
- Difficulté à définir correctement les contextes de sécurité adaptés
- Problèmes empêchant le passage effectif en mode *enforcing* (reste en *permissive*)

En conséquence, SELinux est bien installé mais reste partiellement configuré. Pour une version future, nous prévoyons de consacrer davantage de temps à son intégration complète, afin de bénéficier pleinement des protections supplémentaires qu'il offre.

### 9.2. Sécurisation supplémentaire via options de montage

Lors de la configuration des quotas, certaines options de montage auraient pu être ajoutées afin de renforcer la sécurité des partitions utilisateurs :

- noexec : empêche l'exécution de fichiers binaires depuis la partition (utile pour /home afin qu'un utilisateur ne puisse pas exécuter directement un script ou un binaire depuis son répertoire).

- 
- nosuid : interdit l'utilisation du bit *setuid* ou *setgid* sur les fichiers, évitant ainsi qu'un utilisateur élève ses priviléges via un programme malveillant.
  - nodev : empêche la création et l'utilisation de périphériques spéciaux dans la partition, réduisant les risques d'abus.

Exemple d'entrée /etc/fstab pour /home avec quotas et sécurisation :

```
/dev/sdb1 /home ext4 defaults,usrquota,grpquota,noexec,nosuid,nodev 0 2
```

Ainsi, en plus de la gestion des quotas, le système aurait bénéficié de protections supplémentaires contre l'exécution de fichiers malveillants et l'élévation de priviléges.

### 9.3. Répartition des services sur plusieurs machines

Une autre amélioration envisageable aurait été de répartir les différents services sur plusieurs serveurs distincts, plutôt que de tout centraliser sur une seule machine.

Cela aurait permis :

- Une meilleure sécurité : en isolant par exemple le serveur web du serveur de base de données, une compromission d'un service n'aurait pas automatiquement mis en danger l'ensemble du système.
- Une meilleure disponibilité : chaque machine étant dédiée à une tâche, les pannes ou surcharges seraient limitées à un service précis.
- Une maintenance facilitée : possibilité de mettre à jour ou redémarrer un service sans impacter les autres.

Exemples de répartition possible :

- Machine 1 : serveur DNS et web
- Machine 2 : serveur de bases de données (MariaDB/MySQL)
- Machine 3 : serveur de fichiers (FTP/Samba/NFS)
- Machine 4 : serveur de sécurité et supervision (Fail2Ban, Netdata, etc.)

Cette séparation n'a pas été mise en place dans le cadre du projet par manque de ressources matérielles, mais elle représente une piste d'amélioration importante pour une infrastructure plus robuste et professionnelle.

## 10. Conclusion

Lors de notre première tentative de ce projet Linux AWS, nous n'avions malheureusement pas réussi à obtenir un résultat fonctionnel. Pour cette seconde version, nous avons donc décidé de repartir complètement de zéro, avec une nouvelle organisation et une meilleure méthode de travail.

---

Cette fois, chaque étape a été préparée et testée avec plus de rigueur, ce qui nous a permis d'installer et de configurer tous les services nécessaires sans bloquer l'avancement. Les scripts ont été revus pour être plus fiables et pour corriger les problèmes rencontrés auparavant.

Cette nouvelle méthode de travail a donné des résultats concrets : nous disposons maintenant d'une infrastructure complète, stable et sécurisée, capable de gérer la création et la suppression de clients de manière autonome. Celle-ci intègre un serveur SSH sécurisé avec Fail2ban, un serveur DNS maître avec gestion des sous-domaines, un serveur web Apache avec HTTPS, des partages Samba et NFS, un accès FTP sécurisé, une base de données MariaDB isolée pour chaque client, un monitoring NetData, un serveur NTP, un pare-feu strictement configuré et un SELinux activé en mode permissif pour un contrôle plus souple tout en conservant un suivi de sécurité.

Bien que le résultat final soit pleinement fonctionnel, certaines parties auraient encore pu être améliorées ou optimisées, notamment sur le plan de la sécurité et de l'automatisation avancée. Au-delà du résultat technique, ce projet nous a permis de renforcer nos connaissances en administration serveur, d'améliorer notre organisation et de comprendre l'importance d'anticiper les problèmes pour éviter de répéter les erreurs passées.

## 11. Annexes

### 11.1. Dépôt GitHub du projet

L'ensemble des scripts utilisés pour l'installation, la configuration et l'automatisation des services est disponible dans le dépôt GitHub suivant :

- [https://github.com/Quentin-Hemeryck/projet\\_linuxAWS.git](https://github.com/Quentin-Hemeryck/projet_linuxAWS.git)

Contenu actuel du dépôt :

- Scripts/ – Contient les scripts Bash permettant l'installation, la configuration et l'automatisation des différents services.
- Client/ – Inclut les scripts destinés à la configuration et à la gestion des postes clients.
- Fichiers de configuration/\*\* - Contient un export des fichiers de configuration essentiels du projet :
  - etc/ : SSH, Apache, Samba, FTP, DNS, MariaDB, SELinux, Firewall, Fail2ban, NTP, etc.
  - var/named/ : fichiers de zones DNS (linuxserver.lan.zone, 0.42.10.rev).
- installAll.sh – Script principal permettant d'installer et configurer tous les services via un menu interactif

Remarque : Les fichiers sensibles tels que les clés SSH (.pem) et les profils VPN (.ovpn) ont été retirés du dépôt public afin de garantir la sécurité.