ClamAV : automatisation avancée

Voici une proposition en deux temps pour gérer **freshclam** et **clamAV** sur un serveur (ou un PC personnel).

Cet how to suppose que **fresclam** et **clamav** soient installés sur le PC (et éventuellement **clamav-daemon**).

//1- Gestion automatiser de la mise à jour de la base de virus via **fresclam. //**

//2- Utiliser **clamAV de façon intelligent afin de réduire au mieux la charge serveur (si scan de gros disque)//**

PS, - j'ai mis sudo (mais on peut aussi utiliser un compte root (#) - idem, j'ai mis nano car installé de base, mais on peut utiliser micro qui est plus induitif

PS2, Les scripts sont stockés ici, dans un répertoire : **/root/scripts/clamAV/** Je trouve cela plus clair de les regrouper au même endroit, ensuite libre à vous. Si vous voulez les stocker ailleurs, c'est possible, à vous d'adapter les configurations.

Script de vérification si mise à jour de la base est nécessaire

Tout d'abord, créons un script dont le but sera de ne lancer la mise à jour de la base de données ClamAV que si celle -ci **date de plus de X heures (par exemple 24 h)**. Cela évite de gaspiller de la bande passante ou des ressources si le système redémarre souvent. Un ensemble de log seront également générés.

| download

```
sudo nano /root/scripts/clamAV/freshclam-auto.sh
```

Avec le contenu

| download

```
#!/bin/bash
# Script : freshclam-auto.sh
# Objectif : mettre à jour la base ClamAV uniquement si elle a plus de
24h

LOGFILE="/var/log/freshclam-update.log"
DBDIR="/var/lib/clamav"
MAX_AGE_HOURS=24

# Trouver le fichier principal de la base (daily.cvd ou daily.cld)
```

```
DBFILE=$(ls -t "$DBDIR"/daily.* 2>/dev/null | head -n 1)

# Si la base existe, calculer son âge
if [ -f "$DBFILE" ]; then
    LAST_UPDATE=$(stat -c %Y "$DBFILE")
    NOW=$(date +%s)
    AGE_HOURS=$(( (NOW - LAST_UPDATE) / 3600 ))

    if [ "$AGE_HOURS" -lt "$MAX_AGE_HOURS" ]; then
        echo "$(date '+%F %T') - Base récente ($AGE_HOURS h), mise à
jour ignorée." >> "$LOGFILE"
        exit 0
    fi
fi

# Si la base n'existe pas ou trop vieille, on met à jour
echo "$(date '+%F %T') - Lancement de freshclam..." >> "$LOGFILE"
/usr/bin/freshclam --quiet >> "$LOGFILE" 2>&1
RET=$?

if [ "$RET" -eq 0 ]; then
    echo "$(date '+%F %T') - Mise à jour réussie." >> "$LOGFILE"
else
    echo "$(date '+%F %T') - Erreur freshclam (code $RET)." >>
"$LOGFILE"
fi
```

On le sauvegarde et on le rend exécutable

| download

```
sudo chmod +x /root/scripts/clamAV/freshclam-auto.sh
```

Maintenant, on crée une unité **systemd** (se lancera 10 minutes après le démarrage) et lancera le script précédent.

Crée le fichier** /etc/systemd/system/freshclam.service** :

| download

```
[Unit]
Description=Update ClamAV database 10 minutes after boot (only if
outdated)
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
ExecStartPre=/bin/sleep 600
```

```
ExecStart=/root/scripts/clamAV/freshclam-auto.sh
Nice=19
IOSchedulingClass=idle

[Install]
WantedBy=multi-user.target
```

On recharge systemd, on active ce script et le lance :

| download

```
sudo systemctl daemon-reload
sudo systemctl enable freshclam.service
sudo systemctl start freshclam.service
```

On peut également vérifier les logs :

download

```
cat /var/log/freshclam-update.log
```

(Optionnel, mais bien utile ;)), configurer la rotation du log

Créer le fichier **/etc/logrotate.d/freshclam-update** avec le contenu

| download

```
/var/log/freshclam-update.log {
    weekly
    rotate 4
    compress
    missingok
    notifempty
    create 640 root adm
}
```

Afin de vérifier sa bonne prise en charge et tester la configuration sans attendre la rotation réelle :

| download

```
sudo logrotate --debug /etc/logrotate.conf
```

Ce qui donnera finalement dans l'arborescence des logs

| download

```
/var/log/freshclam-update.log       # log actuel
/var/log/freshclam-update.log.1.gz  # 1 semaine avant
/var/log/freshclam-update.log.2.gz  # 2 semaines avant
/var/log/freshclam-update.log.3.gz  # 3 semaines avant
/var/log/freshclam-update.log.4.gz  # 4 semaines avant
```

Et voici un exemple de leur contenu

| download

```
/var/log/freshclam-update.log
2025-11-13 10:12:00 - Lancement de freshclam...
2025-11-13 10:12:05 - Mise à jour réussie.

/var/log/freshclam-update.log.1.gz
2025-11-06 10:12:00 - Lancement de freshclam...
2025-11-06 10:12:05 - Mise à jour réussie.
2025-11-06 11:12:00 - Base récente (1 h), mise à jour ignorée.
2025-11-06 12:12:00 - Base récente (1 h), mise à jour ignorée.
```

Bonus, voici un petit script Bash pratique qui te permet de voir rapidement les dernières mises à jour de ClamAV, en parcourant automatiquement le fichier de log actuel et les anciens logs compressés.

Créer le script** /root/scripts/clamAV/freshclam-log.sh** avec le contenu suivant :

| download

```
#!/bin/bash
# Script pour afficher les dernières mises à jour ClamAV
# Prend en compte le log actuel et les logs compressés

LOGDIR="/var/log"
LOGFILE="freshclam-update.log"
NUM_LINES=20  # nombre de lignes à afficher

echo "===== Dernières mises à jour ClamAV ====="

# Affiche le log actuel
if [ -f "$LOGDIR/$LOGFILE" ]; then
    echo "--- Log actuel ($LOGFILE) ---"
    tail -n $NUM_LINES "$LOGDIR/$LOGFILE"
fi

# Affiche les logs compressés
for f in $(ls -1t $LOGDIR/$LOGFILE.*.gz 2>/dev/null); do
    echo "--- Log compressé ($(basename $f)) ---"
    zcat "$f" | tail -n $NUM_LINES
```

```
done
```

Comme d'habitude, on le rend exécutable

| download

```
sudo chmod +x /usr/local/bin/freshclam-log.sh
```

Et on peut le tester

| download

```
sudo /usr/local/bin/freshclam-log.sh
```

Ce qui donnera une sortie de ce style

| download

```
===== Dernières mises à jour ClamAV =====
--- Log actuel (freshclam-update.log) ---
2025-11-13 10:12:00 - Lancement de freshclam...
2025-11-13 10:12:05 - Mise à jour réussie.

--- Log compressé (freshclam-update.log.1.gz) ---
2025-11-06 10:12:00 - Lancement de freshclam...
2025-11-06 10:12:05 - Mise à jour réussie.
2025-11-06 11:12:00 - Base récente (1 h), mise à jour ignorée.
2025-11-06 12:12:00 - Base récente (1 h), mise à jour ignorée.
```

Scanner son système

On va écrire un script intelligent pour ne scanner que les fichiers modifiés depuis la dernière exécution, ce qui est idéal pour des disques volumineux (comme des disques contenant plein de MP3 par exemple). Cela réduira drastiquement le temps et la charge.

Principe

- Pour chaque disque, on garde une date de dernière exécution dans un fichier de suivi (/var/log/clamscan-last-<disk>.timestamp).
- On utilise find pour ne sélectionner que les fichiers modifiés après cette date (-newermt).
- On scanne uniquement ces fichiers avec clamscan -i.
- On met à jour la date de dernière exécution après le scan.

Les avantages

- Ultra rapide sur gros disques : scan uniquement les fichiers nouveaux ou modifiés.
- Scan incrémental → gain énorme sur les gros disques (MP3, vidéos…)
- Alertes email immédiates en cas de virus.
- Lock par disque → pas de scans simultanés sur le même disque
- Logs propres : historique conservé dans /var/log/clamscan-auto.log.
- Priorité CPU et I/O très faible → serveur reste réactif
- Compatible avec logrotate.
- Prise en charge des noms de fichiers avec des espaces et des caractères spéciaux lors des scans

Créer le script **/root/scripts/clamAV/clamscan-timer-inc-lock.sh** avec le contenu suivant

| download

```
#!/bin/bash
# Scan ClamAV incrémental avec priorité CPU/I/O faible et lock
# Supporte noms de fichiers avec espaces/accents
# Usage : cron toutes les heures

LOGFILE="/var/log/clamscan-auto.log"
MAILTO="admin@example.com"  # <-- Y mettre votre email
CURRENT_HOUR=$(date '+%H')

# Disques et horaires (heures pleines)
declare -A MOUNTS_SCHEDULE=(
    ["/diskTOTO"]="02"
    ["/diskTITI"]="03"
    ["/diskTATA"]="04"
    ["/"]="05"
)

for MOUNT in "${!MOUNTS_SCHEDULE[@]}"; do
    if [ "${MOUNTS_SCHEDULE[$MOUNT]}" == "$CURRENT_HOUR" ]; then
        DATE=$(date '+%F %T')
        echo "===== Scan ClamAV incrémental sur $MOUNT démarré à $DATE
=====" >> "$LOGFILE"

        if mountpoint -q "$MOUNT"; then

            # Normalise le nom du lock (évite le cas de "/")
            NAME=$(basename "$MOUNT")
            [ -z "$NAME" ] && NAME="root"
            mkdir -p /var/run/clamscan-locks
            LOCKFILE="/var/run/clamscan-locks/${NAME}.lock"

            # Lock par disque pour éviter scans simultanés
            exec 200>"$LOCKFILE"
            if ! flock -n 200; then
                echo "Scan sur $MOUNT déjà en cours. Ignoré." >>
"$LOGFILE"
                continue
```

```
            fi

            # Fichier de suivi de la dernière exécution
            TIMESTAMP_FILE="/var/log/clamscan-last-$(basename
"$MOUNT").timestamp"
            if [ -f "$TIMESTAMP_FILE" ]; then
                LAST_RUN=$(cat "$TIMESTAMP_FILE")
            else
                LAST_RUN="1970-01-01 00:00:00"
            fi

            # Liste temporaire de fichiers modifiés depuis le dernier
scan (séparateur \0 pour xargs)
            FILE_LIST=$(mktemp)
            find "$MOUNT" -type f -newermt "$LAST_RUN" -print0 >
"$FILE_LIST"

            if [ -s "$FILE_LIST" ]; then
                # Limiter l'usage CPU et I/O avec nice et ionice
                SCAN_OUTPUT=$(ionice -c3 nice -n19 xargs -0 -a
"$FILE_LIST" /usr/bin/clamscan -i 2>&1)
                echo "$SCAN_OUTPUT" >> "$LOGFILE"

                # Alerte mail si virus trouvé
                if echo "$SCAN_OUTPUT" | grep -q "FOUND"; then
                    echo -e "Sujet: [ALERTE] Virus détecté sur
$MOUNT\n\n$SCAN_OUTPUT" | /usr/sbin/sendmail "$MAILTO"
                fi
            else
                echo "Aucun fichier modifié depuis le dernier scan." >>
"$LOGFILE"
            fi

            # Nettoyage
            rm -f "$FILE_LIST"
            date '+%F %T' > "$TIMESTAMP_FILE"

        else
            echo "Ignoré : $MOUNT n'est pas monté." >> "$LOGFILE"
        fi

        echo "===== Scan terminé à $(date '+%F %T') =====" >>
"$LOGFILE"
        echo "" >> "$LOGFILE"
    fi
done
```

Il va falloir que vous adaptiez cette fonction bash à votre environnement

| download

```
declare -A MOUNTS_SCHEDULE=(
    ["/diskTOTO"]="02"
    ["/diskTITI"]="03"
    ["/diskTATA"]="04"
    ["/"]="05"
)
```

Y mettre le point de montage de votre disque et l'heure souhaitée de lancement de l'analyse à coté. Seules des heures pleines sont possibles afin de limiter la charge du serveur.

Pensez également à adapter cette ligne

| download

```
MAILTO="admin@example.com"  # <-- Y mettre votre email
```

On va positionner ce script sur un **cron** qui se lancera toutes les heures

| download

```
0 * * * * /root/scripts/clamAV/clamscan-timer-inc-lock.sh
```

Reste ensuite à gérer les logs générés de façon propre

Créer le fichier **/etc/logrotate.d/clamscan-auto** avec le contenu suivant :

| download

```
/var/log/clamscan-auto.log {
    weekly
    rotate 4
    compress
    missingok
    notifempty
    create 640 root adm
}
```

Les logs seront compressés après rotation et seules les 4 dernières semaines seront conservées.



Bonus

BONUS ! Si sur votre serveur vous utilisez un “**dashboard**” à la connexion ou en appel, voici une fonction qui permettra d'insérer la date des derniers scans (avec couleur qui évolue en fonction de

celle ci) ou affichera “Jamais” si jamais scanné.

| download

```
print_clamav_last_run() {
    local mount=$1
    local base=$(basename "$mount")
    [ -z "$base" ] && base="root"
    local ts_file="/var/log/clamscan-last-${base}.timestamp"
    local color=$GREEN
    local last_run="never"

    if [ -f "$ts_file" ]; then
        last_run=$(cat "$ts_file")
        local last_epoch=$(date -d "$last_run" +%s)
        local now_epoch=$(date +%s)
        local age_hours=$(( (now_epoch - last_epoch) / 3600 ))
        if [ "$age_hours" -le 24 ]; then
            color=$GREEN
        elif [ "$age_hours" -le 72 ]; then
            color=$ORANGE
        else
            color=$RED
        fi
    else
        color=$RED
    fi

    # Alignement à 19 caractères pour la date
    printf "   ClamAV last scan on %-12s : ${color}%-19s${RESET}\n"
"$mount" "$last_run"
}
```

Et pour insérer les entrées, ajouter les lignes suivantes à votre dashboard

| download

```
echo "🖴 Disques :"
MOUNTS=("/diskTOTO" "/diskTITI" "/diskTATA" "/")
for MOUNT in "${MOUNTS[@]}"; do
    print_clamav_last_run "$MOUNT"
```

Bah, voilà, je pense avoir fait le tour ;)

From:
https://cbiot.fr/dokuwiki/ - **Cyrille BIOT**

Permanent link:
https://cbiot.fr/dokuwiki/clamav?rev=1762795897

Last update: **2025/11/10 17:31**