Scripte

Mathieu Carteron

Table des matières

Sauvegarde

Sauvegarde complète

```
#!/bin/sh

        # Fichier            : save_all.sh
        # Fonction           : Permet de sauvegarder l'ensemble de la configuration du serveur
        HTTP.
        # Planification      : Oui, tout les mois.



        # Déclaration des constantes :
        ERR_FILE="/var/log/apache2/http_generate_stats.log"      readonly ERR_FILE
        SAVE_DIR="/var/snap/apache2"                                    readonly SAVE_DIR
        SAVE_SRC="/etc/apache2/"                                               readonly
        SAVE_SRC


        # On se place dans le dossier des "snapshots" :
        cd $SAVE_DIR


        # On supprime la dernière sauvegarde si existante :
        LAST_SAVE=$(ls -l | grep -oE "config_.*")
        if [ $? -eq 0 ]
        then
               rm "$LAST_SAVE"
        fi


        # On réalise la sauvegarde de la configuration :
        tar -cvzf "$SAVE_DIR/config_$(date)" $SAVE_SRC 2> ERR_FILE
```

Sauvegarde du site web

```
#!/bin/sh

# Fichier               : save_website.sh
# Fonction              : Permet de sauvegarder le site web en conservant les 7 dernières
sauvegardes.
# Planification : Oui, toutes les semaines à 3h.



# Déclaration des constantes :
ERR_FILE="/var/log/apache2/http_generate_stats.log"     readonly ERR_FILE
SAVE_DIR="/var/snap/apache2"                                    readonly SAVE_DIR
SAVE_SRC="/etc/apache2/sites-enabled/"                          readonly SAVE_SRC


# On se place dans le dossier des "snapshots" :
cd $SAVE_DIR


# On compte le nombre de sauvegardes déjà présentes (on trie les fichiers par date) :
ls -lt | while read -r line
do
        I=$(($I + 1))
        echo $line > lst # On récupère le plus ancien fichier
        echo $I > num    # On récupères la dernière valeur de I
done
NUM=$(cat num)
LAST=$(cat lst | grep -oE 'carnofluxe_sv_.*')


rm num lst                      # On supprime les fichiers temporaires


# Si elles sont au nombre de 7, on supprime la plus ancienne :
if [ $NUM -gt 7 ]
then
        rm "$LAST"
fi


# Enfin, on réalise la sauvegarde du site :
tar -cvzf "$SAVE_DIR/carnofluxe_sv_$(date)" $SAVE_SRC 2> ERR_FILE
```

Scripte de supervision

Statistique de connexion au DNS

```
#!/bin/sh

# Fichier               : save_website.sh
# Fonction              : Permet de sauvegarder le site web en conservant les 7
dernières sauvegardes.
# Planification : Oui, toutes les semaines à 3h.


# Déclaration des constantes :
ERR_FILE="/var/log/apache2/http_generate_stats.log"     readonly ERR_FILE
SAVE_DIR="/var/snap/apache2"                                    readonly
SAVE_DIR
SAVE_SRC="/etc/apache2/sites-enabled/"                          readonly SAVE_SRC


# On se place dans le dossier des "snapshots" :
cd $SAVE_DIR


# On compte le nombre de sauvegardes déjà présentes (on trie les fichiers par date) :
ls -lt | while read -r line
do
        I=$(($I + 1))
        echo $line > lst # On récupère le plus ancien fichier
        echo $I > num    # On récupères la dernière valeur de I
done
NUM=$(cat num)
LAST=$(cat lst | grep -oE 'carnofluxe_sv_.*')


rm num lst                      # On supprime les fichiers temporaires


# Si elles sont au nombre de 7, on supprime la plus ancienne :
if [ $NUM -gt 7 ]
then
        rm "$LAST"
fi


# Enfin, on réalise la sauvegarde du site :
tar -cvzf "$SAVE_DIR/carnofluxe_sv_$(date)" $SAVE_SRC 2> ERR_FILE
```

Génération de la page web

```
#!/bin/sh

        # Fichier            : http_generate_stats.sh
        # Fonction           : Permet de générer une page HTML a partir des fichiers CSV
        pécédemment créés.
        # Planification      : Oui, toutes les 5 minutes.



        # Déclaration des constantes :
        ERR_FILE="/var/log/apache2/http_generate_stats.log"      readonly ERR_FILE
        HTTP_CSV="/var/log/apache2/access.csv"                          readonly HTTP_CSV
        DNS_CSV="/var/log/apache2/state.csv"                            readonly DNS_CSV
        TEMP_CSV="/var/log/apache2/input.csv"                           readonly TEMP_CSV
        WEBSITE="/etc/apache2/sites-enabled/index.html"          readonly WEBSITE


        # On concatène les deux fichiers :
        cat $HTTP_CSV $DNS_CSV > $TEMP_CSV


        # On génère le site Web (nécessite le script python "csv2http") :
        csv2html -o $WEBSITE $TEMP_CSV 2> ERR_FILE
```

Récupération des informations des connexion entrante

```
#!/bin/sh

        # Fichier     : http_log_access.sh
        # Fonction    : Permet de générer un fichier CSV à partir du journal de
connexion du serveur
        #                     Apache, renseignant l'IP du visiteur et sa
géolocalisation.
        # Planification      : Oui, toutes les heures.



        # Déclaration des constantes :
        ERR_FILE="/var/log/apache2/http_log_access.log";
                readonly ERR_FILE
        LOG_FILE="/var/log/apache2/access.log";
                                readonly LOG_FILE
        CSV_FILE="/var/log/apache2/access.csv";
                                readonly CSV_FILE
        HOUR=$(date | grep -oE '[0-9]{2}:[0-9]{2}:[0-9]{2}' | cut -d ':' -f1);
                readonly HOUR


        # On insère les catégories dans le CSV local :
        echo "IP,Country,Region,Ville" > $CSV_FILE


        # Récupération de l'IP :
        grep -E '' $LOG_FILE | while read -r line
        do
                # On vérifie si la ligne est vide :
                if [ "$line" = "" ]
                then
                        continue
                fi


                # On vérifie si la connexion date de la dernière heure :
```

```
	HOURS=$(echo $line | grep -oE ':[0-9]{2}:[0-9]{2}:[0-9]{2}' | cut -d ':' -f2)
	if [ "$HOURS" -eq "$(($HOUR - 1))" ]
	then
		# Récupération de l'adresse IP :
		IP=$(echo $line | grep -oE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}') 2> ERR_FILE

		# Récupération des informations liées à cet IP (curl doit être installé) :
		IP_INFO=$(curl -sg http://ip-api.com/csv/$IP)  2> ERR_FILE

		# On contrôle les échecs de la requête :
		if [ $(echo $IP_INFO | cut -d ',' -f1) != "success" ]
		then
			echo "Echec, impossible de collecter les informations !"

			IP_COUNTRY="Indisponible"
			IP_REGION="Indisponible"
			IP_CITY="Indisponible"
		else
			# Récupération des informations de localisation :
			IP_COUNTRY=$(echo $IP_INFO | cut -d ',' -f2)      # Le pays
			IP_REGION=$(echo $IP_INFO | cut -d ',' -f5)       # La région
			IP_CITY=$(echo $IP_INFO | cut -d ',' -f6)         # La ville
		fi

		# Ecriture du CSV local :
		echo $IP,$IP_COUNTRY,$IP_REGION,$IP_CITY >> $CSV_FILE
	fi
done
```