Récapitulatif — saveAll.algo et createAuto.algo

Scripts Bash d'automatisation — Serveur mutualisé TSSR

1. Organisation générale des scripts

```
/root/saveSite.algo
→ sauvegarde un seul espace client

/root/saveAll.algo
→ parcourt tous les clients et lance saveSite.algo pour chacun

/root/createSite.sh
→ crée un seul espace client

/root/createAuto.algo
→ lit une liste de clients et lance createSite.sh pour chacun

/root/siteACreer.txt
→ contient les noms des clients à créer

/root/saves
→ contient les archives de sauvegarde
```

À retenir :

```
saveAll.algo ne réalise pas directement la sauvegarde
→ il appelle saveSite.algo

createAuto.algo ne réalise pas directement la création
→ il appelle createSite.sh
```

2. Script saveAll.algo

Objectif

```
Parcourir tous les dossiers présents dans /home
→ ignorer le premier utilisateur administrateur
→ ignorer les dossiers qui ne correspondent pas à un utilisateur Linux
→ vérifier qu'une base MariaDB existe
→ lancer saveSite.algo pour chaque véritable client
```

Création du script

Passer en root :

```
su -
```

Se placer dans /root :

```
cd /root
```

Vérifier les fichiers présents :

```
ls
```

Créer ou modifier le script :

```
nano /root/saveAll.algo
```

Contenu final de saveAll.algo

```
#!/bin/bash

# PATH complet pour permettre l'exécution du script avec cron.
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
export PATH

# Le script doit être exécuté avec les droits root.
if [ "$(id -u)" -ne 0 ]; then
    echo "Erreur : exécutez ce script en root."
    exit 1
fi

# Dossier dans lequel se trouvent saveAll.algo et saveSite.algo.
SCRIPT_DIR="$(cd "$(dirname "$0")" && pwd)"
SAVE_SCRIPT="$SCRIPT_DIR/saveSite.algo"

# Récupère le premier utilisateur humain créé sur le serveur.
FIRSTUSER=$(getent passwd | awk -F: '$3 >= 1000 && $3 < 65534 && $6 ~ /^\/home\// {print $3 ":"
$1}' | sort -t: -k1,1n | head -n 1 | cut -d: -f2)

if [ -z "$FIRSTUSER" ]; then
    echo "Erreur : impossible de déterminer le premier utilisateur du serveur."
    exit 1
fi

if [ ! -f "$SAVE_SCRIPT" ]; then
    echo "Erreur : le script $SAVE_SCRIPT est introuvable."
    exit 1
fi

echo "Premier utilisateur ignoré : $FIRSTUSER"

# Parcourt tous les sous-dossiers présents dans /home.
for FILE in /home/*; do

    # Ignore les éléments qui ne sont pas des dossiers.
    [ -d "$FILE" ] || continue

    CLIENT=$(basename "$FILE")

    # Ignore les dossiers qui ne correspondent pas à un utilisateur Linux.
    if ! getent passwd "$CLIENT" > /dev/null; then
        echo "Ignoré : $CLIENT ne correspond pas à un utilisateur Linux"
```

```
        continue
    fi

    # Vérifie que la base MariaDB du client existe.
    if ! mariadb -NBe "SHOW DATABASES LIKE '$CLIENT';" | grep -qx "$CLIENT"; then
        echo "Ignoré : aucune base MariaDB trouvée pour $CLIENT"
        continue
    fi

    # Ne sauvegarde pas le dossier personnel de l'administrateur.
    if [ "$CLIENT" = "$FIRSTUSER" ]; then
        echo "Ignoré : $CLIENT"
        continue
    fi

    echo "Sauvegarde de l'espace client : $CLIENT"

    # Appelle saveSite.algo avec le nom du client en paramètre.
    if /bin/bash "$SAVE_SCRIPT" "$CLIENT"; then
        echo "Sauvegarde réussie : $CLIENT"
    else
        echo "Erreur pendant la sauvegarde : $CLIENT" >&2
    fi

done
```

Enregistrer dans Nano

```
Ctrl + O
Entrée
Ctrl + X
```

Rendre le script exécutable

```
chmod +x /root/saveAll.algo
```

chmod +x ajoute le droit d'exécution au fichier.

Vérifier la syntaxe

```
bash -n /root/saveAll.algo
```

> Si rien ne s'affiche : la syntaxe Bash est correcte.
>
> Attention : bash -n vérifie la syntaxe, mais il ne vérifie pas que toutes les opérations fonctionneront réellement.

Exécuter le script

```
/root/saveAll.algo
```

Résultat obtenu :

```
Premier utilisateur ignoré : yunhee

Ignoré : dossier.txt ne correspond pas à un utilisateur Linux

Sauvegarde réussie : john
Sauvegarde réussie : lulu
Sauvegarde réussie : Toto

Ignoré : yunhee
```

Vérifier les sauvegardes

```
ls -lh /root/saves
```

Cela permet d'afficher :

- Le nom des archives
- Leur taille
- Leur propriétaire
- Leur date de création

Les archives ressemblaient à :

```
john_2026-07-24_11-06-42.tar.gz
lulu_2026-07-24_11-06-42.tar.gz
Toto_2026-07-24_11-06-42.tar.gz
```

Explication des commandes principales

```
for FILE in /home/*
```

→ Parcourt chaque élément présent dans /home.

```
[ -d "$FILE" ] || continue
```

→ Vérifie que l'élément est bien un dossier. Sinon, le script passe au suivant.

```
CLIENT=$(basename "$FILE")
```

→ Transforme /home/john en john.

```
getent passwd "$CLIENT"
```

→ Vérifie qu'un utilisateur Linux portant ce nom existe.

```
mariadb -NBe "SHOW DATABASES LIKE '$CLIENT';"
```

→ Vérifie qu'une base MariaDB portant le nom du client existe.

```
/bin/bash "$SAVE_SCRIPT" "$CLIENT"
```

→ Lance saveSite.algo en lui donnant le nom du client en premier paramètre.

Exemple réel :

```
/bin/bash /root/saveSite.algo john
```

3. Script createAuto.algo

Objectif

```
Lire les noms présents dans siteACreer.txt
→ ignorer les lignes vides
→ ne pas créer l'espace de l'administrateur
→ lancer createSite.sh pour chaque client
→ vider siteACreer.txt à la fin
```

Différence entre les deux fichiers

```
/root/createAuto.algo
→ contient les commandes du script

/root/siteACreer.txt
→ contient seulement les noms des clients
```

Création du script

```
nano /root/createAuto.algo
```

Contenu de createAuto.algo

```
#!/bin/bash

# Redéfinition du PATH pour une future utilisation avec cron
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# Premier utilisateur du serveur : administrateur
FIRSTUSER=$(getent passwd 1000 | cut -d: -f1)

# Parcourt la liste des sites à créer
while IFS= read -r DOSSIER; do

    # Ignore les lignes vides
    [ -z "$DOSSIER" ] && continue

    # Ne crée pas d'espace pour le premier utilisateur
    if [ "$DOSSIER" = "$FIRSTUSER" ]; then
        echo "Ignoré : $DOSSIER est l'administrateur"
    else
        echo "Création de l'espace client : $DOSSIER"
        /bin/bash /root/createSite.sh "$DOSSIER"
    fi

done < /root/siteACreer.txt

# Vide la liste après le traitement
echo "" > /root/siteACreer.txt

echo "Création automatique terminée"
```

Enregistrer dans Nano

```
Ctrl + O
Entrée
Ctrl + X
```

Rendre le script exécutable

```
chmod +x /root/createAuto.algo
```

Vérifier la syntaxe

```
bash -n /root/createAuto.algo
```

> Si rien ne s'affiche : la syntaxe est correcte.

4. Préparer la liste des clients

Ouvrir le fichier :

```
nano /root/siteACreer.txt
```

Mettre un nom par ligne :

```
didier
maya
ruru
```

Pour les futurs essais, tu peux mettre par exemple :

```
alice
paul
marine
```

Il est préférable d'utiliser des noms simples en minuscules :

- Lettres minuscules
- Chiffres si nécessaire
- Pas d'espace
- Pas de chemin /home/
- Pas de nom de domaine

Correct :

```
alice
```

Incorrect :

```
/home/alice
alice.mutualise.fr
Alice Dupont
```

Enregistrer

```
Ctrl + O
Entrée
Ctrl + X
```

Vérifier la liste

```
cat -n /root/siteACreer.txt
```

cat -n affiche le contenu avec les numéros de lignes.

5. Lancer createAuto.algo

```
/root/createAuto.algo
```

Le script a lu successivement :

```
didier
maya
ruru
```

Puis il a lancé :

```
/bin/bash /root/createSite.sh didier
/bin/bash /root/createSite.sh maya
/bin/bash /root/createSite.sh ruru
```

Résultat obtenu :

```
Création de l'espace client : didier
Création de l'espace client : maya
Création de l'espace client : ruru
Création automatique terminée
```

6. Ce que réalise createSite.sh

Pour chaque client, createSite.sh réalise réellement la création :

- Création de l’utilisateur Linux
- Création de son dossier dans /home
- Création automatique d’une page d’accueil index.html
- Création de la base MariaDB

- Création du VirtualHost Apache
- Activation du VirtualHost
- Rechargement d'Apache

Exemple pour Ruru :

```
Utilisateur : ruru
Dossier : /home/ruru
Base MariaDB : ruru
VirtualHost : ruru.mutualise.fr
Page de test : /home/ruru/index.html
```

createAuto.algo sert uniquement à automatiser l'appel de createSite.sh plusieurs fois.

Création automatique d'une page d'accueil de test

Pour éviter une réponse « 403 Forbidden » lorsqu'un nouvel espace client est vide, une page index.html de validation est maintenant créée automatiquement par createSite.sh.

Cette page ne constitue pas un site complet : elle sert uniquement à confirmer que le dossier du client, ses permissions et son VirtualHost Apache fonctionnent correctement.

Emplacement du bloc dans createSite.sh : après les permissions du dossier /home/${USER}/ et avant la partie MariaDB.

```
#3 on définit les droits sur son dossier home pour donner l'accès à Apache + SGID
chgrp www-data /home/${USER}/
chmod g+w,o-rwx /home/${USER}/
chmod g+s /home/${USER}/

# Création automatique d'une page d'accueil de test
cat > /home/${USER}/index.html <<EOF
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Espace de ${USER}</title>
</head>
<body>
    <h1>Site de ${USER}</h1>
    <p>L'espace client de ${USER} fonctionne.</p>
</body>
</html>
EOF

# Propriétaire et permissions de la page
chown ${USER}:www-data /home/${USER}/index.html
chmod 0640 /home/${USER}/index.html

## mariaDB
```

Explication du bloc ajouté

→ cat > /home/${USER}/index.html <<EOF crée le fichier index.html et y écrit tout le contenu compris jusqu'à la ligne EOF.

→ ${USER} est remplacé automatiquement par le nom transmis à createSite.sh, par exemple polux ou hamtaro.

→ chown ${USER}:www-data attribue le fichier au client et au groupe utilisé par Apache.

→ chmod 0640 autorise la lecture et l'écriture au propriétaire, la lecture au groupe et aucun droit aux autres.

Vérification après modification

```
bash -n /root/createSite.sh
```

Si aucune erreur ne s'affiche, ajoute de nouveaux noms dans siteACreer.txt, puis relance createAuto.algo.

```
nano /root/siteACreer.txt
/root/createAuto.algo
```

Vérifie ensuite le fichier avec le nom réellement créé. Par exemple, si la liste contenait polux et hamtaro :

```
ls -l /home/polux/index.html
ls -l /home/hamtaro/index.html
```

7. Vérifications effectuées

Vérifier l'utilisateur Linux

```
getent passwd ruru
```

Résultat obtenu :

```
ruru:x:1014:1015::/home/ruru:/bin/false
```

Cela indique notamment :

```
Nom de l'utilisateur : ruru
Dossier personnel : /home/ruru
Shell : /bin/false
```

/bin/false empêche généralement une connexion directe à un shell interactif.

Vérifier le dossier

```
ls -ld /home/ruru
```

Cela affiche :

- Les permissions
- Le propriétaire
- Le groupe
- Le chemin du dossier

Vérifier la base MariaDB

```
mariadb -e "SHOW DATABASES LIKE 'ruru';"
```

Résultat attendu :

```
ruru
```

Vérifier le VirtualHost Apache

```
apache2ctl -S | grep ruru
```

Résultat obtenu :

```
port 80 namevhost ruru.mutualise.fr
(/etc/apache2/sites-enabled/ruru.conf:1)
```

Cela confirme que le site est bien activé dans Apache.

Vérifier la page d'accueil automatique

```
ls -l /home/ruru/index.html
cat /home/ruru/index.html
```

Ces commandes vérifient que le fichier existe, affichent ses permissions et montrent son contenu HTML.

Tester le VirtualHost avec curl

```
curl -H "Host: ruru.mutualise.fr" http://127.0.0.1
curl -H "Host: ruru.mutualise.fr" http://192.168.190.130
```

→ 127.0.0.1 est l'adresse de boucle locale : elle signifie « cette machine elle-même ». Ce n'est pas l'adresse d'Apache.

→ 192.168.190.130 est l'adresse réseau réelle de la VM Debian.

→ L'option -H "Host: ruru.mutualise.fr" indique à Apache quel VirtualHost doit répondre.

→ Les deux tests doivent afficher la même page HTML. Un résultat 403 Forbidden signifie généralement que le VirtualHost répond, mais qu'aucune page d'accueil accessible n'est présente.

```
<h1>Site de ruru</h1>
<p>L'espace client de ruru fonctionne.</p>
```

Vérifier que la liste a été vidée

```
cat -n /root/siteACreer.txt
```

Après l'exécution, le fichier ne contient plus les noms.

Cette ligne réalise le vidage :

```
echo "" > /root/siteACreer.txt
```

Le symbole > remplace le contenu du fichier.

8. Rechargement d'Apache dans createSite.sh

Dans createSite.sh, tu avais déjà :

```
# On active le virtualhost
a2ensite ${USER}.conf
apache2ctl configtest && systemctl reload apache2
```

a2ensite affiche automatiquement :

```
To activate the new configuration, you need to run:
systemctl reload apache2
```

Mais ce message est uniquement informatif.

Ton script exécute juste après :

```
apache2ctl configtest && systemctl reload apache2
```

Donc Apache est bien rechargé.

```
a2ensite
→ active le VirtualHost

apache2ctl configtest
→ vérifie la syntaxe de la configuration Apache

systemctl reload apache2
→ demande à Apache de relire sa configuration
```

9. Configuration globale ServerName

Apache affichait l'avertissement :

```
AH00558: Could not reliably determine the server's fully qualified domain name
```

Ce n'était pas une panne. Apache fonctionnait, mais aucun nom global n'était défini.

Nous avons créé :

```
nano /etc/apache2/conf-available/servername.conf
```

Contenu :

```
ServerName 192.168.190.130
```

Puis, directement dans le terminal :

```
a2enconf servername
apache2ctl configtest
systemctl reload apache2
apache2ctl -S
```

Explications :

```
a2enconf servername
```

→ Active le fichier de configuration générale.

```
apache2ctl configtest
```

→ Vérifie la syntaxe de toute la configuration Apache.

```
systemctl reload apache2
```

→ Recharge Apache sans l'arrêter complètement.

```
apache2ctl -S
```

→ Affiche les VirtualHosts connus par Apache.

Résultat attendu :

```
Syntax OK
```

Cette configuration est faite une seule fois pour le serveur.

Les clients conservent chacun leur propre nom :

```
didier.mutualise.fr
maya.mutualise.fr
ruru.mutualise.fr
```

10. Utiliser à nouveau createAuto.algo

Puisque tu as ajouté de nouveaux noms dans :

```
/root/siteACreer.txt
```

tu peux vérifier :

```
cat -n /root/siteACreer.txt
```

Puis relancer :

```
/root/createAuto.algo
```

Après le traitement :

```
cat -n /root/siteACreer.txt
```

La liste doit de nouveau être vide.

À retenir

SAVEALL	CREATEAUTO
/home → contient les espaces des utilisateurs saveAll.algo → parcourt les espaces clients saveSite.algo → sauvegarde réellement chaque client /root/saves → contient les archives	siteACreer.txt → contient les noms à créer createAuto.algo → lit la liste createSite.sh → crée réellement chaque espace client À la fin → siteACreer.txt est vidé

Fonctionnement global :

```
Plusieurs clients à sauvegarder
→ saveAll.algo
→ saveSite.algo
→ archives dans /root/saves

Plusieurs clients à créer
→ noms dans siteACreer.txt
→ createAuto.algo
→ createSite.sh
→ nouveaux espaces dans /home
```