Gestion Automatisée des Conteneurs Docker avec Timeout et Recréation / Automated Docker Container Management with Timeout and Auto-Recreation

□□ Objectif :

Créer un script pour :

- **Lister les conteneurs actifs** (sauf **Portainer**)
- **Sauvegarder les informations essentielles** (ID, nom, ports, volumes)
- **Arrêter les conteneurs** pour la maintenance (sans toucher à Portainer)
- **Redémarrer automatiquement après 4 heures** (ou sur commande)
- **Recréer les conteneurs supprimés** avec leurs anciens ports et volumes

Cela vous permet de gérer vos conteneurs sereinement tout en conservant l'accès à votre **interface Portainer** ! □□

□□ Le Script : `manage_containers.sh`

Voici le code complet à copier :

```
#!/bin/bash

# Fichier pour stocker les informations des conteneurs
CONTAINER_FILE="/tmp/containers_info.txt"
TIMEOUT=14400  # 4 heures en secondes
EXCLUDED_CONTAINER="portainer"  # Conteneur à exclure de l'arrêt/redémarrage

# Fonction pour lister et sauvegarder les conteneurs avec des détails
list_and_save_containers() {
    echo "□□ Liste des conteneurs en cours d'exécution (hors $EXCLUDED_CONTAINER)..."
    docker ps --format "{{.ID}} {{.Names}} {{.Ports}} {{.Mounts}}" | grep -v "$EXCLUDED_CONTAINER" >
"$CONTAINER_FILE"

    if [[ ! -s $CONTAINER_FILE ]]; then
        echo "□  Aucun conteneur en cours d'exécution (hors $EXCLUDED_CONTAINER)."
        exit 0
    fi

    echo "□□ Informations des conteneurs enregistrées dans : $CONTAINER_FILE"
    cat "$CONTAINER_FILE"
}

# Fonction pour arrêter les conteneurs (hors Portainer)
stop_containers() {
    echo "□□ Arrêt des conteneurs (hors $EXCLUDED_CONTAINER)..."
    while read -r container_id container_name container_ports container_mounts; do
        if [[ "$container_name" != "$EXCLUDED_CONTAINER" ]]; then
            echo "➡ Arrêt du conteneur : $container_name ($container_id)"
            docker stop "$container_id"
        fi
    done < "$CONTAINER_FILE"
    echo "✔ Tous les conteneurs listés (sauf $EXCLUDED_CONTAINER) sont arrêtés."
}

# Fonction pour redémarrer ou recréer les conteneurs
restart_or_recreate_containers() {
    echo "□□ Redémarrage ou recréation des conteneurs (hors $EXCLUDED_CONTAINER)..."
    while read -r container_id container_name container_ports container_mounts; do
        if [[ "$container_name" != "$EXCLUDED_CONTAINER" ]]; then
            # Vérifier si le conteneur existe encore
```

```
            if docker ps -a --format "{{.ID}}" | grep -q "$container_id"; then
                echo "➡ Redémarrage du conteneur : $container_name ($container_id)"
                docker start "$container_id"
            else
                echo "⚠ Conteneur $container_name introuvable. Recréation..."

                # Recréer le conteneur avec des options basiques
                recreate_command="docker run -d --name $container_name"

                # Ajouter les ports si présents
                if [[ -n $container_ports ]]; then
                    for port_mapping in $(echo $container_ports | tr ',' ' '); do
                        recreate_command+=" -p $port_mapping"
                    done
                fi

                # Ajouter les volumes si présents
                if [[ -n $container_mounts ]]; then
                    for volume_mapping in $(echo $container_mounts | tr ',' ' '); do
                        recreate_command+=" -v $volume_mapping"
                    done
                fi

                # Afficher la commande de recréation (l'image reste à préciser)
                echo "□□ Commande de recréation :"
                echo "$recreate_command <image>"
                echo "□  Remplacez '<image>' par l'image Docker d'origine pour finaliser la recréation."
            fi
        fi
    done < "$CONTAINER_FILE"
    echo "□□ Redémarrage/recréation terminé."
}

# Exécution du script
case "$1" in
    start)
        list_and_save_containers
        stop_containers

        echo "□□ Conteneurs arrêtés (hors $EXCLUDED_CONTAINER). Vous avez 4 heures pour effectuer votre
```

```
action."
        echo "⏳ Après 4 heures, les conteneurs seront redémarrés automatiquement."

        # Délai de 4 heures
        sleep "$TIMEOUT"

        echo "⏰ Timeout atteint : redémarrage automatique des conteneurs..."
        restart_or_recreate_containers
        ;;
    restart)
        if [[ ! -f $CONTAINER_FILE ]]; then
            echo "❌ Fichier des conteneurs non trouvé. Lancez './manage_containers.sh start' d'abord."
            exit 1
        fi
        restart_or_recreate_containers
        ;;
    *)
        echo "Usage : $0 {start|restart}"
        exit 1
        ;;
esac
```

🛠️ Comment l’utiliser :

1. **Créer le script :**

   ```
   nano manage_containers.sh
   ```

2. **Donner les droits d’exécution :**

   ```
   chmod +x manage_containers.sh
   ```

3. **Lancer le processus d’arrêt et de sauvegarde :**

   ```
   ./manage_containers.sh start
   ```

4. **Après votre intervention (ou après le timeout de 4h) :**
   - Les conteneurs sont **redémarrés automatiquement**.
   - Si un conteneur a été supprimé, la **commande de recréation** s’affiche pour vous guider ! 🛠️

5. **Redémarrer manuellement avant le timeout :**

```
./manage_containers.sh restart
```

□□ Pourquoi stocker plus d’infos sur les conteneurs ?

On sauvegarde les :

- **ID du conteneur**
- **Nom**
- **Ports exposés**
- **Volumes montés**

➡ Ça permet de **recréer** un conteneur avec sa configuration si jamais il a été supprimé par accident.

□□ Timeout de 4 heures :

Avec le `sleep 14400`, vous avez **4 heures** pour faire votre maintenance avant que les conteneurs ne redémarrent tout seuls. Vous pouvez ajuster cette valeur à votre convenance.

□□ Conclusion :

Avec ce script, vous pouvez :
✔ **Sauvegarder l’état de vos conteneurs**
✔ **Arrêter les conteneurs proprement**
✔ **Redémarrer automatiquement après un délai**
✔ **Recréer les conteneurs supprimés** avec leurs ports et volumes
✔ **Gagner du temps** et **réduire les erreurs** lors des maintenances

C’est une base solide pour **sécuriser vos déploiements Docker** sur Debian ! □□

Here’s your article translated into English, ready to save in **Bookstack**! □□ □□

□□ Goal:

Create a script to:

- **List active containers** (except **Portainer**)
- **Save essential information** (ID, name, ports, volumes)
- **Stop containers** for maintenance (without affecting Portainer)
- **Automatically restart after 4 hours** (or on command)
- **Recreate deleted containers** with their previous ports and volumes

This allows you to manage your containers with peace of mind while maintaining access to your **Portainer interface**! □□

□□ The Script: `manage_containers.sh`

Here’s the full code to copy:

```
#!/bin/bash

# File to store container information
CONTAINER_FILE="/tmp/containers_info.txt"
TIMEOUT=14400  # 4 hours in seconds
EXCLUDED_CONTAINER="portainer"  # Container to exclude from stop/restart

# Function to list and save container details
list_and_save_containers() {
    echo "□□ Listing running containers with details..."
    docker ps --format "{{.ID}} {{.Names}} {{.Ports}} {{.Mounts}}" | grep -v "$EXCLUDED_CONTAINER" > "$CONTAINER_FILE"

    if [[ ! -s $CONTAINER_FILE ]]; then
        echo "□  No running containers (excluding $EXCLUDED_CONTAINER)."
        exit 0
    fi
```

```
    echo "□□ Container information saved to: $CONTAINER_FILE"
    cat "$CONTAINER_FILE"
}

# Function to stop containers (excluding Portainer)
stop_containers() {
    echo "□□ Stopping containers (excluding $EXCLUDED_CONTAINER)..."
    while read -r container_id container_name container_ports container_mounts; do
        if [[ "$container_name" != "$EXCLUDED_CONTAINER" ]]; then
            echo "➡ Stopping container: $container_name ($container_id)"
            docker stop "$container_id"
        fi
    done < "$CONTAINER_FILE"
    echo "✔ All listed containers (except $EXCLUDED_CONTAINER) stopped."
}

# Function to restart or recreate containers
restart_or_recreate_containers() {
    echo "□□ Restarting or recreating containers (excluding $EXCLUDED_CONTAINER)..."
    while read -r container_id container_name container_ports container_mounts; do
        if [[ "$container_name" != "$EXCLUDED_CONTAINER" ]]; then
            # Check if the container still exists
            if docker ps -a --format "{{.ID}}" | grep -q "$container_id"; then
                echo "➡ Restarting container: $container_name ($container_id)"
                docker start "$container_id"
            else
                echo "⚠ Container $container_name not found. Recreating..."

                # Recreate the container with basic options
                recreate_command="docker run -d --name $container_name"

                # Add ports if present
                if [[ -n $container_ports ]]; then
                    for port_mapping in $(echo $container_ports | tr ',' ' '); do
                        recreate_command+=" -p $port_mapping"
                    done
                fi

                # Add volumes if present
                if [[ -n $container_mounts ]]; then
```

```
                for volume_mapping in $(echo $container_mounts | tr ',' ' '); do
                    recreate_command+=" -v $volume_mapping"
                done
            fi

            # Show the recreation command (image needs to be specified)
            echo "□□ Recreation command:"
            echo "$recreate_command <image>"
            echo "□ Replace '<image>' with the original Docker image to complete the recreation."
        fi
    fi
done < "$CONTAINER_FILE"
echo "□□ Restart/recreation complete."
}

# Script execution
case "$1" in
    start)
        list_and_save_containers
        stop_containers

        echo "□□ Containers stopped (excluding $EXCLUDED_CONTAINER). You have 4 hours to complete your action."
        echo "□ After 4 hours, containers will restart automatically."

        # 4-hour timeout
        sleep "$TIMEOUT"

        echo "□ Timeout reached: Automatically restarting containers..."
        restart_or_recreate_containers
        ;;
    restart)
        if [[ ! -f $CONTAINER_FILE ]]; then
            echo "□ Container file not found. Run './manage_containers.sh start' first."
            exit 1
        fi
        restart_or_recreate_containers
        ;;
    *)
        echo "Usage: $0 {start|restart}"
```

```
        exit 1
        ;;
esac
```

□□ How to Use It:

1. **Create the script:**

   ```
   nano manage_containers.sh
   ```

2. **Make it executable:**

   ```
   chmod +x manage_containers.sh
   ```

3. **Start the process to stop and save containers:**

   ```
   ./manage_containers.sh start
   ```

4. **After your maintenance (or after the 4-hour timeout):**
   - Containers **restart automatically**.
   - If a container was deleted, a **recreation command** appears to help you restore it! □□
5. **Restart manually before the timeout:**

   ```
   ./manage_containers.sh restart
   ```

□□ Why Save More Container Info?

We save:

- **Container ID**
- **Name**
- **Exposed ports**
- **Mounted volumes**

➡ This lets you **recreate a container** with its configuration if it gets deleted.

□□ 4-Hour Timeout:

With `sleep 14400`, you get **4 hours** to perform maintenance before containers automatically restart. You can change this value to fit your needs.

□□ Conclusion:

With this script, you can:
✔ **Save container state**
✔ **Stop containers safely**
✔ **Automatically restart after a delay**
✔ **Recreate deleted containers** with their ports and volumes
✔ **Save time and avoid errors** during maintenance

It’s a solid base to **secure your Docker deployments** on Debian! □□

Revision #3
Created 11 March 2025 13:35:00 by Admin
Updated 11 March 2025 13:56:26 by Admin