Mission 4 : Configuration de Rsync

0. Pré-requis

Afin de pouvoir réaliser cette mission il faut :

- Un serveur
- Un routeur
- un conteneur template et ns configuré
- Deux humains

1. Introduction

On souhaite mettre en place une sauvegarde automatique des fichiers de configuration des différents services mis en place :

Le stockage des sauvegardes des configurations sera effectué sur une nouvelle machine **(conteneur)** nommé **backup.**

La sauvegarde doit être réalisée :

- toutes les 5 minutes (pour les tests) et toutes les nuits lorsque la solution sera fonctionnelle,

- sans intervention humaine.

Les outils utilisés seront **ssh, rsync et cron.** La sauvegarde sera réalisée grace à un script BASH qui sera exécuté automatiquement et qui devra générer des fichiers de logs pour garder une trace des sauvegardes effectuées et des éventuels problèmes.

rsync est généralement utilisé pour réaliser des sauvegardes locales ou distantes (mirroring).

Il est possible d'utiliser Rsync de deux manières différentes :

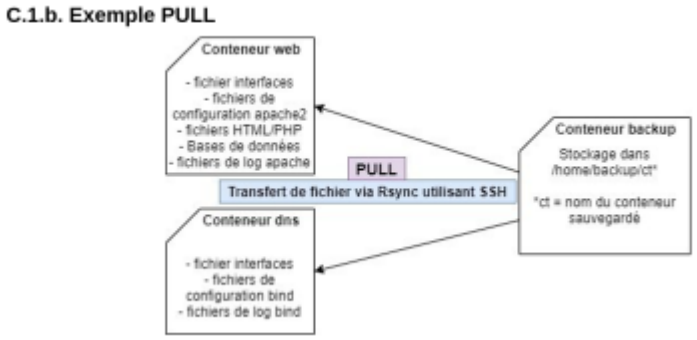
1. Mode démon (client/serveur) : non sécurisé par défaut

2. Via SSH : sécurisé (les données ne sont pas transférées en clair) Dans cet atelier, seul le transfert de données via SSH sera traité.

2. Installation et configuration de l'outil rsync

2.0. Mémo Rsync

Il existe deux méthode pour effectuer une sauvegarde de fichier en **ssh** avec l'outil **rsync**, la méthode **PUSH** et la méthode **PULL**. Ici, on s'intéressera à la méthode **PULL.**



Dans le cas de la méthode PULL, c'est la machine de sauvegarde qui va se connecter aux machines à sauvegarder et qui va initier le transfert.

La commande rsync est utilisée depuis la machine de sauvegarde (la machine backup) et la source est un répertoire distant et la destination est un répertoire local sur la machine de sauvegarde. Donc la machine de sauvegarde PULL les répertoires à sauvegarder des différentes machines.

2.1. Creation du compte backup1

Pour autoriser le conteneur backup à se connecter a nos machines à sauvegarder, on crée d'abord un utilisateur **backup1** sur chacun des 3 conteneurs (ns1, web et backup) et on lui donne le droit d'effectuer la commande **sudo rsync**.

```
root@web:~# adduser backup1 #crée l'utilisateur backup1
```

Une fois l'utilisateur **backup1** crée, on se rend dans le fichier **sudoers** pour lui autoriser à effectuer la commande **sudo rsync** :

```
root@backup1:~# nano /etc/sudoers       #ouvre le fichier sudoers
```

On y ajoute une ligne :

```
/etc/sudoers
```

```
root   ALL=(ALL:ALL) ALL  #privilèges du compte root
backup1 ALL=NOPASSWD: /usr/bin/rsync       #privilèges du compte backup1
```

Une fois configuré, on se connecte dessus

```
root@backup1:~# su - backup1        #se connecte au compte backup1
```

2.2. Installation et mis en place de rsync

On installe l'outil **rsync** sur tous les conteneurs :

```
root@:~# apt install rsync   #installe l'outil rsync
```

Désormais, on doit s'assurer que notre conteneur **backup** puisse se connecter à l'utilisateur backup1 de nos deux autres conteneurs **web** et **ns1** en **ssh**.

On génère d'abord une clé publique dans le conteneur **backup** avec la commande :

```
backup1@backup1:~$ ssh-keygen -t rsa -b 4096     #génere des clés ssh
```

On accède donc à notre fichier qui contient notre clé publique :

```
backup1@backup1:~$ cat /home/backup2/.ssh/id_rsa.pub
```

Ensuite on se rend dans nos deux conteneurs **web** et **ns1** et on crée un dossier **.ssh** qui va contenir un fichier **authorized_keys** dans lequel on mettra notre clé :

```
backup1@web:~$ mkdir .ssh                    #crée un dossier .ssh
backup1@web:~$ cd .ssh                        #se déplace jusqu'au dossier
.ssh
backup1@web:~$ nano authorized_keys           #crée le fichier
authorized_keys
```

On vérifie si ça fonctionne en se connectant en ssh au conteneur **web** :

```
backup1@backup1:~# ssh backup1@10.31.64.80 #connexion ssh au conteneur web
```

Et au conteneur **ns1** :

```
backup1@backup1:~# ssh backup1@10.31.64.53 #connexion ssh au conteneur ns1
```

Ensuite, on exécute la commande suivante depuis le conteneur backup1 :

```
backup1@backup1:~# rsync -azv -e ssh backup2@10.31.65.80:/etc/apache2
/home/backup2/web2
```

Que fait la commande complète ?

Exécutée sur la machine de sauvegarde, elle va **synchroniser** (en archivant, compressant, chiffrant

les données) via **SSH** le répertoire distant **/etc/apache2** de la machine **10.31.64.80** vers le répertoire local **/home/backup1/web** de la machine backup en utilisant **le compte utilisateur backup de la machine distante.**

On vérifie si la commande a bien crée un répertoire **web** dans lequel se trouve les fichiers de configuration d'apache2 du conteneur **web** :

```
backup1@backup1:~# ls /home/backup1/web  #verifie le contenue du dossier web
```

3. Script de sauvegarde et Cron

3.0. Script de lancement de synchronisation

Désormais, nous allons créer un script **BASH** qui va nous permettre de synchroniser tous les répertoires à sauvegarder sur tous nos serveurs avec un répertoire dans **/home/backup/** de la machine de sauvegarde (la machine **backup** puisque l'on va choisir la méthode **PULL**).

PULL : le script est éxecuté sur la machine de sauvegarde

Le script devra :

Synchroniser les répertoires,

Créer un fichier de log permettant de garder une trace des sauvegardes,

Un fichier de log différent devra être créé chaque jour.

```
#!/bin/bash

# Log file pour la sauvegarde
LOGFILE="/home/backup1/backuplog_$(date +%d-%m-%Y).log"

# Sauvegarde des fichiers de configuration Apache
rsync -azv --rsync-path="sudo rsync" -e ssh backup1@10.31.64.80:/etc/apache2
/home/backup1/web/apache2 >> $LOGFILE

# Sauvegarde des logs d'Apache
rsync -azv --rsync-path="sudo rsync" -e ssh
backup1@10.31.64.80:/var/log/apache2 /home/backup1/web/apache_logs >>
$LOGFILE

# Sauvegarde des sites web
rsync -azv --rsync-path="sudo rsync" -e ssh backup1@10.31.64.80:/var/www
/home/backup1/web/sites >> $LOGFILE

# Sauvegarde des fichiers de configuration du serveur DNS 1 (ns1)
rsync -azv --rsync-path="sudo rsync" -e ssh backup1@10.31.64.53:/etc/bind
/home/backup1/ns1 >> $LOGFILE
```

```
# Sauvegarde des fichiers de logs de bind (serveur DNS 1)
rsync -azv --rsync-path="sudo rsync" -e ssh
backup1@10.31.64.53:/var/log/bind /home/backup1/ns1_logs >> $LOGFILE

# Sauvegarde des fichiers de configuration du serveur DNS 2 (ns2)
rsync -azv --rsync-path="sudo rsync" -e ssh backup1@10.31.64.99:/etc/bind
/home/backup1/ns2 >> $LOGFILE

# Sauvegarde des fichiers de logs de bind (serveur DNS 2)
rsync -azv --rsync-path="sudo rsync" -e ssh
backup1@10.31.64.99:/var/log/bind /home/backup1/ns2_logs >> $LOGFILE

# Sauvegarde des bases de données MySQL
ssh backup1@10.31.64.80 "mysqldump -u root --all-databases >
/tmp/mysql_backup.sql"
rsync -azv --rsync-path="sudo rsync" -e ssh
backup1@10.31.64.80:/tmp/mysql_backup.sql /home/backup1/mysql_backup.sql >>
$LOGFILE
ssh backup1@10.31.64.80 "rm /tmp/mysql_backup.sql"  # Nettoyage du fichier
temporaire

echo "Sauvegarde terminée pour le $(date)" >> $LOGFILE
```

3.1 Mémo Cron

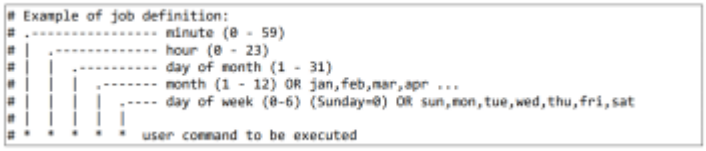
cron est un service serveur bien utile et assez simple à mettre en œuvre. Il permet de programmer des actions à intervalles réguliers.

Un démon nommé **cron** lit les fichiers qui sont dans une crontab. Elle se trouve dans **/etc/crontab**

Chaque utilisateur possède une crontab personnelle et pour l'éditer, il faut taper :

crontab -e

Voici la structure d'une crontab :



La syntaxe des entrées est la suivante : <minute> <heure> <jour du mois> <mois> <jour de la semaine> <commande> (avec un espace entre chacun)

minute : de 0 à 59

heure : de 0 à 23

jour du mois de : 1 à 31

mois de : 1 à 12

jour de la semaine : de 0 à 6, 0 étant le dimanche et ainsi de suite.

commande : peut comporter plusieurs commandes.

Les mois et les jours peuvent aussi être donnés avec les abréviations anglaises : **jan, feb…** et les jours avec **mon, tue…**

On sépare les jours, les mois par des virgules, par exemple pour lancer une action tous les 15 et 30 du mois tapez **15,30.**

Le '-' signifie jusqu'à, ainsi 15-30 signifie du 15 au 30. Le '/' permet de spécifier une répétition, */3 indique toutes les 3 minutes. '*' peut être utilisé et indique tous les jours de semaine, tous les mois, toutes les heures

3.2 Mise en place de la sauvegarde régulière avec cron

On souhaite que la sauvegarde des fichiers se fasse **tous les jours, à 1 heure du matin.**

On va donc faire en sorte que le **script** que l'on a créer **s'éxecute** tous les jours à 1 heure du matin grâce à l'outil **cron** :

D'abord, on ouvre le **crontab** pour l'utilisateur **backup1** du conteneur **backup** :

```
backup1@backup1:~# crontab -e
```

On ajoute cette ligne à la fin du fichier crontab pour programmer l'éxecution à 1heure du matin tous les jours :

```
0 1 * * * /bin/bash /home/backup1/backup.sh
```

Une fois la ligne ajoutée et sauvegardée, on vérifie si le cron est bien configuré avec la commande suivante :

```
backup1@backup1:~# crontab -l #liste des tâches cron programmées pour l'utilisateur.
```