Restic Backup

- Restic : Déploiement
- Restic : Ligne de commandes
- Restic : Annexes
- Restic : Scénarios de sauvegarde

Restic : Déploiement

Docker Restic REST with SSL

Pré-requis :

- Posséder un binaire restic : `apt install restic`
- Posséder la suite openssl : `apt install openssl`
- Posséder docker et docker compose : `apt install docker docker-compose`
- Savoir lire :)

1) Préparation du répertoire projet

Variable de la documentation :

```
NAME_PROJECT=projet
PATH_PROJECT=/mnt/"${NAME_PROJECT}"
FOLDER_DATA_NAME=data
PATH_DATA="${PATH_PROJECT}"/"${FOLDER_DATA_NAME}"
PORT_EXPOSED=8009
USER_PROJECT=projet_user
SRV_RESTIC=127.0.0.1 #TEST EN LOCAL
```

Créer votre répertoire de projet :

```
mkdir /mnt/projet && cd $_
```

Créer le répertoire de sauvegarde dans le projet =

```
mkdir /mnt/projet/data
```

2) Générer un utilisateut restic

Créer un utilisateur pouvant intéragir avec le répertoire de sauvegarde :

```
htpasswd -B -c ${PATH_DATA}/.htpasswd projet_user
```

Un prompt apparaîtra afin de valider votre mot de passe.

3) Générer une paire de clés SSL

Créer un bundle de clés SSL ou importer les vôtre sous le format :

- public_key
- private_key

Méthode 1 : Let's Encrypt

Il est préférable d'utiliser une paire de clés valide via let's encrypt par exemple :

```
certbot certonly --standalone -d ${DOMAINE}
```

Il faudra suite intégrer les clés dans votre espace data :

```
cp -fra /etc/letsencrypt/archive/${DOMAINE}/fullchain1.pem ${PATH_DATA}/public_key
cp -fra /etc/letsencrypt/archive/${DOMAINE}/privkey1.pem ${PATH_DATA}/private_key
```

Méthode 2 : Snakeoil

Voici une manière manuelle de les générer :

```
openssl req -newkey rsa:4096  -x509  -sha512  -days 365 -nodes -out ${PATH_DATA}/public_key -
keyout ${PATH_DATA}/private_key
```

Cette méthode vous obligera à indiquer à restic de ne pas prendre en compte la validité du certificat via l'option :

```
restic --insecure-tls
```

4) Préparer votre docker-compose.yml

Créer votre fichier docker-compose.yml en modifiant les valeurs entre "{{ }}" : `vim ${PATH_PROJECT}/docker-compose.yml`

```
  services:
  rest_server_{{ NAME_PROJECT }}:
    command:
      - "/entrypoint.sh"
    container_name: "rest_server_{{ NAME_PROJECT }}"
    environment:
      - "OPTIONS=--tls --prometheus"
      - "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      - "DATA_DIRECTORY=/data"
      - "PASSWORD_FILE=/data/.htpasswd"
    hostname: "rest_server_{{ NAME_PROJECT }}"
    image: "restic/rest-server"
    ipc: "private"
    labels:
      org.opencontainers.image.source: "https://github.com/restic/rest-server"
      org.opencontainers.image.title: "rest-server"
    logging:
      driver: "json-file"
      options: {}
    ports:
      - "{{ PORT_EXPOSED }}:8000/tcp"
    volumes:
      - "{{ PATH_DATA }}:/data"
version: "3.6"
```

Lancer ensuite votre instance restic

```
cd "${PATH_PROJECT}"
docker-compose up -d
```

5) Initialiser le workspace de sauvegarde

Init : initialiser un répertoire de sauvegarde

```
restic -r rest:https://${USER_PROJECT}:{{ MOT DE PASSE ETAPE 3
}}@${SRV_RESTIC}:${PORT_EXPOSED} init
```

Votre dépôt comporte désormais un répertoire de sauvegarde.

6) Bashrc && alias

Sur votre utilisateur courant (différent de root), éditer votre fichier bashrc `vim /home/user/.bashrc`

```
export RESTIC_PASSWORD="${USER_MDP}"
export
RESTIC_REPOSITORY="rest:https://${USER_PROJECT}:${USER_MDP}@${SRV_RESTIC}:${PORT_EXPOSED}/"
```

Se reconnecter sur sa session ou charger le fichier bashrc :

```
source /home/user/.bashrc
```

Cela permet ensuite d'utiliser le binaire vers votre dépôt de sauvegarde sans à divulguer toutes les informations :

```
zayad@vmi2007645:~$ restic stats
repository 6d0ed1e0 opened (repository version 2) successfully, password is correct
scanning...
Stats in restore-size mode:
Snapshots processed:   2
   Total File Count:   281
         Total Size:   24.191 KiB
```

Restic : Ligne de commandes

Bashrc & Alias

Sur votre utilisateur courant (différent de root), éditer votre fichier bashrc `vim /home/user/.bashrc`

```
alias restic='restic --insecure-tls'
export RESTIC_PASSWORD="${USER_MDP}"
export
RESTIC_REPOSITORY="rest:https://${USER_PROJECT}:${USER_MDP}@${SRV_RESTIC}:${PORT_EXPOSED}/"
```

Cela permet ensuite d'utiliser le binaire vers votre dépôt de sauvegarde sans à divulguer toutes les informations :

```
zayad@vmi2007645:~$ restic stats
repository 6d0ed1e0 opened (repository version 2) successfully, password is correct
scanning...
Stats in restore-size mode:
Snapshots processed:   2
   Total File Count:   281
         Total Size:   24.191 KiB
```

Backing Up

```
restic backup ~/directory-to-backup --pack-size=60
```

Cleanup

With every backup, Restic is creating a new snapshot with contents of a directory at the moment. To remove old and unused snapshots we need to execute the forget command:

```
restic forget --keep-last 2 --prune
```

Check

If you want to verify the consistency of your backup, run the check command:

```
restic check
```

Restore

To restore the latest snapshot of your backup:

```
restic restore latest --target ~/restore
```

Restic : Annexes

Logiciels clients

Testé et approuvé :

- Restic-browser https://github.com/emuell/restic-browser

Restic

- Restatic : https://github.com/Mebus/restatic
- autorestic : https://github.com/cupcakearmy/autorestic
- crestic : https://github.com/nils-werner/crestic
- Runrestic : https://github.com/sinnwerkstatt/runrestic
- restic-windows-backup : https://github.com/kmwoley/restic-windows-backup

Restic : Scénarios de sauvegarde

Sauvegarde MariaDB

```
#!/bin/bash
DATE=$(date +%Y%m%d-%Hh%M)
DUMP="/mnt/data/dump"
## Récupérer les bdd présent dans le SGBDR
listBdd=$(echo "SHOW DATABASES" |mysql |grep -v "Database" |grep -v "schema")

##Backup bases de données
for bdd in ${listBdd};
do
    echo "Dump des bases de données avant sauvegarde"
    echo "Création du répertoire temporaire pour les dump"
    mkdir -p ${DUMP}
    echo "dump de la base $bdd "
    echo ""
    mysqldump $bdd | gzip -9 > ${DUMP}/${bdd}.sql.tar.gz && bddName=${bdd}.sql.tar.gz
    echo "Lancement de la sauvegarde pour la BDD ${bdd}"
    restic backup --tag="bdd-$bdd-${DATE}" ${DUMP}/${bddName} --quiet
    if [ $? == 0 ]
    then
        echo "La sauvegarde de la bdd ${bdd} est effective"
    else
        echo "Echec de la sauvegarde"
        let "ERROR+=1"
    fi
done
```

Sauvegarde Conf & Logs

```
#!/bin/bash
DATE=$(date +%Y%m%d-%Hh%M)
CONF="/etc/apache2 /etc/mysql /etc/ssh /root /etc/letsencrypt /etc/fail2ban /etc/php
/etc/redis /etc/memcached.conf"
SITE="/var/www/html/"
LOG="/var/log"

## Récupérer les bdd présent dans le SGBDR
listBdd=$(echo "SHOW DATABASES" |mysql |grep -v "Database" |grep -v "schema")

##Backup conf & logs
for var in ${CONF} ${SITE} ${LOG}
do
    echo "Lancement de la sauvegarde pour les données du répertoire ${var}"
    DATE=$(date +%Y%m%d-%Hh%M)
    restic backup --tag="$var-${DATE}" $var --quiet
    if [ $? == 0 ]
    then
        echo "La sauvegarde du répertoire ${var} est effective"
    else
        echo "Echec de la sauvegarde"
        let "ERROR+=1"
    fi
done
```