Tutoriel : Mettre en place un serveur de backup avec `rsync` sur Debian 12

Objectif

Mettre en place un serveur de backup sous **Debian 12** permettant de sauvegarder les **fichiers de configuration et bases de données** des services **Zabbix, Webzine, Loki et Gitea** via `rsync`.

---

1 Configuration du Serveur de Backup (Debian 12)

1.1 Installation de `rsync` et SSH

```
sudo apt update && sudo apt install rsync openssh-server -y
```

1.2 Création de l'utilisateur `backup` avec un mot de passe

```
sudo useradd -m -s /bin/bash backup
sudo passwd backup  # Saisir un mot de passe sécurisé
```

1.3 Autoriser l'utilisateur `backup` à se connecter en SSH

Modifie le fichier **`/etc/ssh/sshd_config`** :

```
sudo nano /etc/ssh/sshd_config
```

Ajoute à la fin :

```
AllowUsers backup
PasswordAuthentication yes
```

Recharge le service SSH :

```
sudo systemctl restart ssh
```

1.4 Création des répertoires de sauvegarde

```
sudo mkdir -p /backup/{zabbix,terraform,ansible,wazuh}/{config,db}
sudo chown -R backup:backup /backup
```

---

2 Configuration des Serveurs Applicatifs

Sur chaque serveur applicatif (**Zabbix, Webzine, Loki, Gitea**) :

2.1 Installation de `rsync`

```
sudo apt install rsync -y
```

2.2 Configurer l’accès SSH

Sur chaque serveur applicatif, exécute :

```
ssh-keygen -t rsa -b 4096
```

Copie la clé publique sur le serveur de backup :

```
ssh-copy-id backup@Srv-Backup
```

Test SSH :

```
ssh backup@Srv-Backup
```

Si la connexion fonctionne sans mot de passe, c'est bon !

---

3 Sauvegarde des Fichiers de Configuration

Sur chaque serveur applicatif, crée un script `/usr/local/bin/backup_config.sh` :

```
#!/bin/bash

BACKUP_SERVER="backup@Srv-Backup"
BACKUP_PATH="/backup"

rsync -avz --delete /etc/zabbix/ $BACKUP_SERVER:$BACKUP_PATH/zabbix/config/
rsync -avz --delete /etc/terraform/ $BACKUP_SERVER:$BACKUP_PATH/terraform/config/
rsync -avz --delete /etc/ansible/ $BACKUP_SERVER:$BACKUP_PATH/ansible/config/
rsync -avz --delete /etc/wazuh/ $BACKUP_SERVER:$BACKUP_PATH/wazuh/config/
```

Ajoute un **cron job** pour l'exécuter tous les jours à 2h du matin :

```
crontab -e
```

Ajoute la ligne suivante :

```
0 2 * * * /usr/local/bin/backup_config.sh
```

---

4 Sauvegarde des Bases de Données

PostgreSQL (Zabbix)

```
#!/bin/bash

BACKUP_SERVER="backup@Srv-Backup"
BACKUP_PATH="/backup"
DATE=$(date +'%Y-%m-%d')

pg_dump -U postgres -h localhost -Fc zabbix > /tmp/zabbix_$DATE.dump
rsync -avz /tmp/zabbix_$DATE.dump $BACKUP_SERVER:$BACKUP_PATH/zabbix/db/
rm -f /tmp/zabbix_$DATE.dump
```

Ajoute un **cron job** pour exécuter la sauvegarde à 3h du matin :

```
crontab -e
```

Ajoute cette ligne :

```
0 3 * * * /usr/local/bin/backup_db.sh
```

---

5 Rotation et Nettoyage des Sauvegardes

Sur le **serveur de backup**, crée un script `/usr/local/bin/cleanup_backups.sh` :

```
#!/bin/bash

find /backup/zabbix/db/ -type f -mtime +7 -delete
find /backup/webzine/db/ -type f -mtime +7 -delete
find /backup/gitea/db/ -type f -mtime +7 -delete
```

Ajoute une tâche **cron** pour exécuter le nettoyage à 4h du matin :

```
crontab -e
```

Ajoute cette ligne :

```
0 4 * * * /usr/local/bin/cleanup_backups.sh
```

---

6 Vérification et Monitoring

Tester les scripts de backup

```
/usr/local/bin/backup_config.sh
/usr/local/bin/backup_db.sh
```

Vérifie sur le **serveur de backup** :

```
ls -lah /backup/zabbix/config/
ls -lah /backup/zabbix/db/
```

Ajouter une alerte en cas d’échec

Ajoute ces lignes en fin de chaque script pour recevoir une alerte par email :

```
if [ $? -ne 0 ]; then
    echo "Backup FAILED on $(hostname)" | mail -s "Backup Failure" admin@example.com
fi
```