Script : Sauvegarde des bases de données MariaDB.

Les scripts shell sont des fichiers exécutables permettant de lancer successivement plusieurs commandes. Pour créer un script shell, il faut éditer un fichier, y entrer les commandes et le rendre exécutable. Imaginez un mini langage de programmation intégré à Linux. Ce n'est pas un langage aussi complet que peuvent l'être le C, le C++ ou le Java par exemple, mais cela permet d'automatiser la plupart de vos tâches : sauvegarde des données, surveillance de la charge de votre machine, etc.

Voici les noms de quelques-uns des principaux shells qui existent :

- **sh** : *Bourne Shell*. L'ancêtre de tous les shells.
- **bash** : *Bourne Again Shell*. Une amélioration du *Bourne Shell*, disponible par défaut sous Linux et Mac OS X.
- **ksh** : *Korn Shell*. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
- **csh** : *C Shell*. Un shell utilisant une syntaxe proche du langage C.
- **tcsh** : *Tenex C Shell*. Amélioration du *C Shell*.
- **zsh** : *Z Shell*. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

Pour ce tutoriel, nous allons utiliser seulement le shell « bash ».

Notre script consiste à effectuer plusieurs actions permettant la sauvegarde des bases de données sur un serveur distant.

```
############################################################
######## Début de la sauvegarde des bases de données ! #######
########.................................................#######
########.............Compression du fichier..............#######
########.................................................#######
########.......Suppression des fichiers obsolètes......#######
########.................................................#######
########......Envoi du fichier compressé en cours......#######
########.................................................#######
#################### Sauvegarde terminée ! ####################
```

Avant la création du script, commencez par créer les répertoires contenant votre script (et vos futurs scripts) et vos sauvegardes, notre script contiendra des informations sensibles (mot de passe et utilisateur), je vous conseille donc de placer votre script dans un dossier accessible seulement en lecture et écriture en root.

- Sur les 2 serveurs : `root@ServeurWeb:~# mkdir /root/scripts/`

- Sur le serveur web : `root@ServeurWeb:~# mkdir /root/scripts/backups/`

- Sur le serveur distant : `root@ns1:~# mkdir /opt/backupsDB/`

1. <u>Création et automatisation du script sur le Serveur Web</u>

Pour faciliter la création du script plaçons nous dans le répertoire « */root/scripts/* », utilisons la commande : « **cd /root/scripts/** ».

Maintenant nous pouvons enfin créer le script : `root@ServeurWeb:~/scripts# nano backup_bdd.sh`

La page sera vide, enregistrez la pour créer le fichier !

Commencez par écrire sur la première ligne : `#!/bin/bash`

Nous venons de définir le fichier comme un script, occupons-nous des commandes à exécuter.

```
 GNU nano 2.7.4                          Fichier : backup_bdd.sh

#!/bin/bash
#
## Sauvegarde BDD
#
############################

## Variables
DIR=/root/scripts/backups                # Répertoire où l'on placera les sauvegardes avant de les envoyer
DATE=`date +%d-%m-%Y`                    # Correspond à la date de création du fichier de sauvegarde (JJ-MM-AAAA)
############################

## Serveur Distant
IP='192.168.3.11'                        # Adresse IP du serveur distant
USER='thomas'                            # Nom d'utilisateur du serveur distant (ne doit pas etre root)
PASS='2312'                              # Mot de passe de l'utilisateur ci-dessus
DIRD='/opt/backupsDB/'                   # Direction du répertoire où l'on souhaite envoyer les sauvegardes
############################
cd $DIR

## Sauvegarde des BDD en .sql
mysqldump --user=root --password=2312 --all-databases > $DIR/SaveDB_$DATE.sql
          # Utilisateur et mot de passe MySQL

## Compression en tar.bz2
tar jcf SaveDB_$DATE.sql.tar.bz2 SaveDB_$DATE.sql
```

Nous avons créé les variables nécessaires, puis nous avons sauvegardé les bases de données vers le fichier se trouvant dans « */root/scripts/backups/* » que nous avons nommé : « **SaveDB_JJ-MM-AAAA.sql** » (JJ-MM-AAAA correspond à la date, par exemple 13-04-2018).

Nous avons ensuite compressé le fichier pour éviter qu'il prenne trop de place et pour l'envoyer plus rapidement.

Continuons le script :

```
## Suppression des fichiers non compressés
rm SaveDB_$DATE.sql

## Suppression automatique tout les 3 jours
find $DIR/ -type f -mtime +3 |xargs rm -f

## Envoi du fichier
sshpass -p "$PASS" scp $DIR/SaveDB_$DATE.sql.tar.bz2 $USER@$IP:$DIRD


^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^J Justifier  ^C Pos. cur.  ^Y Page préc.
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^T Analyse sta^_ Aller lig. ^V Page suiv.
```

Ici, nous supprimons les fichiers non compressés ainsi que les fichiers ayant été créé 3 jours avant. Pour finir nous envoyons les fichiers sur le serveur distant grâce à la commande « **sshpass** » qui permet de ne pas avoir à rentrer le mot de passe de l'utilisateur distant manuellement, pour cela il vous faut l'installer sur les 2 serveurs (« **apt-get install sshpass** ») ainsi que de *vérifier que l'on puisse bien se connecter en ssh sur les 2 machines*.

Le script est terminé, exécutons le pour vérifier s'il est bien fonctionnel :

```
root@ServeurWeb:~/scripts# bash backup_bdd.sh

root@ServeurWeb:~/scripts#
```

Aucun message d'erreur, le script a donc bien été exécuter mais nous n'avons aucun retour, vérifions *sur le serveur distant* en affichant le contenu du répertoire de sauvegarde :

```
root@ns1:~# ls /opt/backupsDB/
SaveDB_15-04-2018.sql.tar.bz2
root@ns1:~#
```

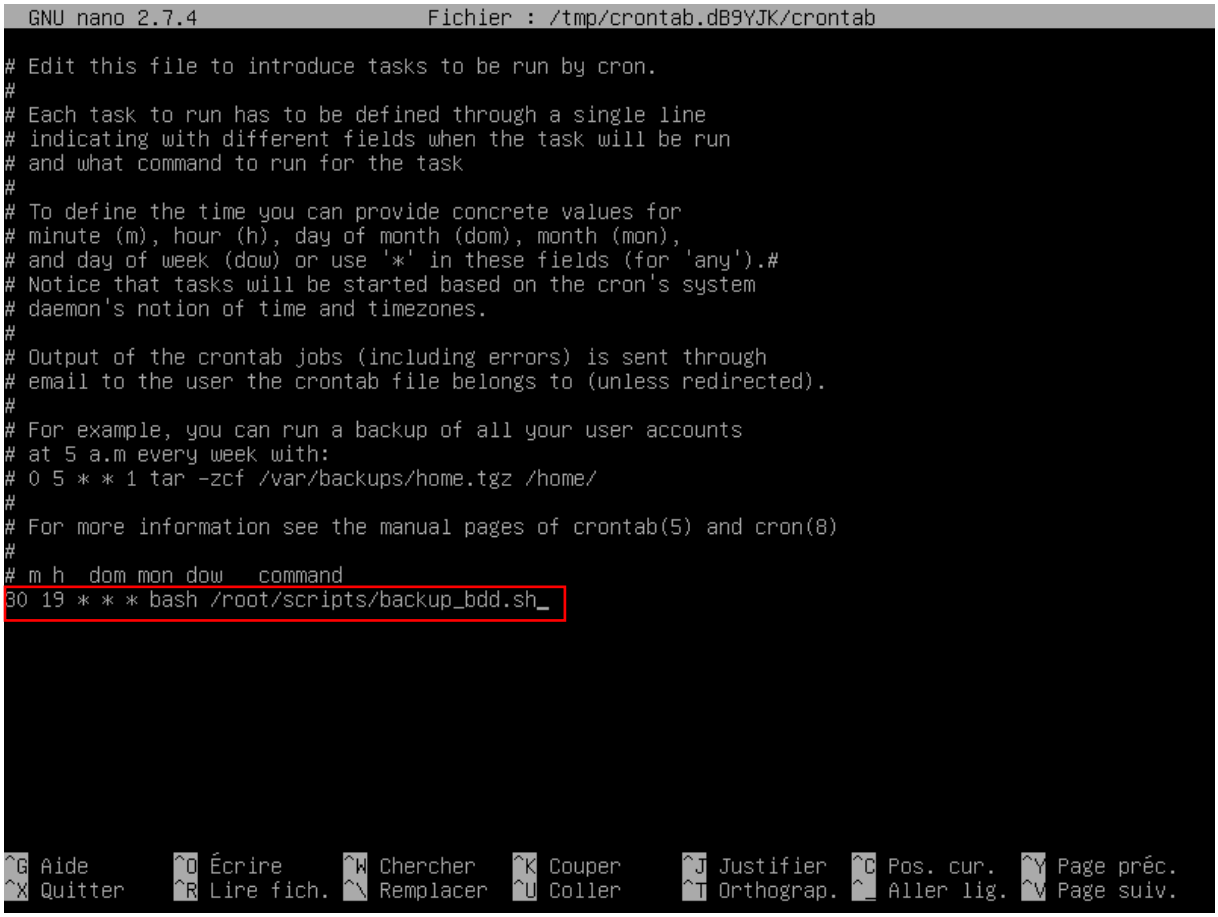
La sauvegarde à bien eu lieu et le fichier est bien compressé !

Nous allons maintenant faire en sorte que le script s'exécute automatiquement tous les jours à une certaine heure. Linux vous propose toute une série d'outils qui vous permettent de programmer à l'avance l'exécution d'une tâche, comme par exemple la commande **crontab** que nous allons utiliser.

Il y a trois paramètres différents à connaitre pour l'utilisation de cette commande :

- crontab -e (permet de modifier la crontab)
- crontab -l (permet d'afficher la crontab actuelle)
- crontab -r (supprimer toute la crontab (sans confirmation))

Par défaut, nous n'avons rien dans notre crontab, nous allons ajouter l'exécution de notre script :



- *30 19 * * **: correspond au temps, dans notre cas, tous les jours à 19h30. (pour plus d'informations voir ce tableau)
- *bash /root/scripts/backup_bdd.sh* : permet l'exécution du script.

2. Création et automatisation d'un script sur le serveur distant

Nous allons maintenant créer un script nous permettant de supprimer automatiquement les bases de données de plus de 3 jours pour éviter de surcharger le disque dur.

Créons notre script :

```
  GNU nano 2.7.4                  Fichier : /root/scripts/clear_bdd.sh
#!/bin/bash
#
## Suppression Auto des Sauvegardes BDD Anciennes
#
################################################

## Variables
DIR=/opt/backupsDB/
################################################

cd $DIR

## Supression Automatique tout les 3 jours
find $DIR -type f -mtime +3 |xargs rm -f

#################### END ######################



^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^J Justifier  ^C Pos. cur.  ^Y Page préc.
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^T Analyse sta^_ Aller lig. ^V Page suiv.
```

Voilà maintenant essayer de vous débrouiller pour l'exécuter automatiquement via la **crontab**, exécutez-le automatiquement un peu après qu'il est reçu les fichiers du serveur web.

Pour aller plus loin :

Les scripts s'exécutent bien mais nous n'avons aucun message nous permettant de confirmer l'envoi ou aucun log, nous devons vérifier manuellement sur le serveur distant. Pour afficher un message sur linux, nous avons la commande « **echo "message"** ».

Par exemple :

```
root@ServeurWeb:~# echo "Hello World !"
Hello World !
root@ServeurWeb:~#
```

Nous pouvons donc implémenter des « **echo** » pour avoir un affichage pendant que le script s'exécute comme par exemple :

```
################################################################
######## Début de la sauvegarde des bases de données ! ########
########.....................................................########
########.....Compression du fichier.....########
########.....................................................########
########.....Suppression des fichiers obsolètes.....########
########.....................................................########
########.....Envoi du fichier compressé en cours.....########
########.....................................................########
#################### Sauvegarde terminée ! #####################
```

Vous pouvez bien évidemment le modifier et l'améliorer en utilisant des structures conditionnelles, un système de log, etc… Il suffit de laisser place à son imagination.

Dans mon cas, l'affichage ne change pas même si la sauvegarde échoue, le texte est seulement « décoratif » !

Voici pour le tableau :

```
 GNU nano 2.7.4                    Fichier : /root/scripts/backup_bdd.sh

echo ""
echo "############################################################"
echo "######## Début de la sauvegarde des bases de données ! ########"
echo "########.....................................................########"

cd $DIR

## Sauvegarde des BDD en .sql
mysqldump --user=root --password=2312 --all-databases > $DIR/SaveDB_$DATE.sql

echo "########.............Compression du fichier.............########"
echo "########.....................................................########"

## Compression en tar.bz2
tar jcf SaveDB_$DATE.sql.tar.bz2 SaveDB_$DATE.sql

echo "########.......Suppression des fichiers obsolètes......########"

## Suppression des fichiers non compressés
rm SaveDB_$DATE.sql

echo "########.....................................................########"

## Suppression automatique tout les 3 jours
find $DIR/ -type f -mtime +3 |xargs rm -f

echo "########......Envoi du fichier compressé en cours......########"
echo "########.....................................................########"

## Envoi du fichier
sshpass -p "$PASS" scp $DIR/SaveDB_$DATE.sql.tar.bz2 $USER@$IP:$DIRD

echo "#################### Sauvegarde terminée ! ####################"
echo ""
```