Administration système

Sauvegarde automatique de type Snapshot avec Linux et Rsync

A quoi ça sert ?

Nous allons étudier une méthode qui a pour but de générer des Snapshots automatiques sur un system orienté Unix. Les sauvegardes de type Snapshot sont utilisées surtout dur les systèmes de fichier industriels ; les snapshots créent l'illusion de pouvoir créer plusieurs sauvegardes complètes par jours sans l'inconvénient d'utiliser de la place disque ou des ressources systèmes. L'intérêt de cette méthode est qu'elle propose une sauvegarde qui utilise au minimum deux fois moins de place qu'une sauvegarde standard.
Notre but sera donc de mettre en place simplement cette procédure de sauvegarde automatique en utilisant l'outil maintenant bien intégré dans la plupart des distributions : Rsync.
Nous allons donc essayer de mettre en place une sauvegarde toute les heures ou toutes les demi-heure et une fois par semaine qui se place dans un dossier /snapshot qui est en lecture seule donc, non modifiable, et qui permet de restaurer le système en cas de problème avec le système de fichier ou les fichiers eux-mêmes. Ce système ne nous protège pas d'un crash Disque.
Ustiliser `rsync` pour faire une sauvegarde
L'utilitaire Rsync est disponible sur la plupart des distributions si malgré tout vous voulez le télécharger vous pouvez a partir de ce site : rsync.samba.org.
Faire un backup simple equivalent d'un copie avec Rsync :
rsync -a source/ destination/
Ce qui est equivalent a :
cp -a source/. destination/
Mais grâce a Rsync on peut aussi effectuer des sauvegardes via un shell sécurisé a distance par le réseau :
rsync -a -e ssh source/ username@remotemachine.com:/path/to/destination/
Cette fonction est donc une fonction avancée qui nous l'utilisation de Rsync plutôt que le simple utilitaire cp.

Utiliser l'option --delete

Si un fichier était au départ dans le répertoire `source/` et `destination/` il est possible de synchroniser le contenu des répertoires et de supprimer le fichier et dans le répertoire source et dans le répertoire destination.
Cette commande permet de supprimer tous les fichiers de `destination/` qui n'est pas dans `source/`, nous utiliserons l'option delete :
rsync -a --delete source/ destination/
Utilisation de la Cron table : `cron`
Nous devons mettre en place un backup fréquent pour cela nous utiliserons la cron table qui automatisera pour nous la procédure.
Pour lancer une procedure avec Rsync tous les jours a 4:20 nous allons editer la crontab (en tant que root) :
crontab -e
Et ajouter la ligne suivante :
20 4 * * * rsync -a --delete source/ destination/
Ainsi la sauvegarde s'effectuera à 4:20 tous les matins et le Root recevra le résultat par mail.

Rappel sur les hard links

Nous pensons qu'un fichier est représente par son nom en réalité le nom est un *hard link*. Un fichier donné peut avoir plus d'un hard link vers lui par exemple un répertoire a au moins deux, hard links: le nom du répertoire et . (Quant on est dedans bien sur). On peut savoir combien de liens a un fichier en utilisant l'utilitaire stat :
stat nomdufichier
Nous pouvons aussi créer des Hardlinks sur des fichiers :
ln a b
Ici, `a` et `b` sont deux noms pour le même fichier, Ils résident dans le même Inode:
ls -i a
232177 a
ls -i b
232177 b

Donc `ln a b` est plus ou moins équivalent a `cp a b`, mais il y a de grosses différences :

- Le contenu des fichier est exactement le même et on n'utilise l'espace qu'une fois .
- si vous changez `a`, vous changez b , et vice-versa.
- si vous changez les droits de `a`, vous changez ceux de b, et vice-versa.
- Si vous écrasez a avec un autre fichier `a`, vous écraserez `b`, a moins que vous demandiez a cp de supprimer le lien entre a et b avant. On peut le faire avec `cp --remove-destination` . **Rsync supprime toujours les liens avant d'écraser un fichier**

Si vous faite un rm du fichier alors vous supprimerez seulement le lien vers celui-ci.
Utiltiser `cp -al`
L'utilitaire cp permet de créer des Hardlink en utilisant cp –l. Pour la commande `cp` on peut utiliser –a qui permet de garder les propriétaires, les informations de date, et les permissions d'accès.
La combinaison `cp -al` crée l'illusion d'une copie complète d'une arborescence mais ce ne sont que des hardlinks.

Combinaison

Nous pouvons combiner `rsync` et `cp -al` pour créer une sauvegarde multiple d'un système de fichier sans besoin d'espace disque :
rm -rf backup.3
// Suppression récursive du contenu des répertoires sans confirmation
mv backup.2 backup.3
// transforme backup.2 en Backup.3
mv backup.1 backup.2
// transforme backup.1 en Backup.2
cp -al backup.0 backup.1
// copie en utilisant les hardlinks et en gardant les informations relatives aux fichier backup.0 vers backup.1
rsync -a --delete source_directory/ backup.0/
// synchronise le contenu du répertoire source dans le fichier Backup.0

Si les commandes ci-dessus s'exécutent tous les jours alors les sauvegardes `backup.0`, `backup.1`, `backup.2`, et `backup.3` vont être des sauvegardes complètes du `source_directory/` d'aujourd'hui, d'hier, d'il y a deux jours, et d'il y a trois jours, a part que les droits et l'appartenance des fichiers dans les anciens snapshots aurons les valeurs les plus récentes c'est-à-dire celles de la sauvegarde backup.0.En fait la place occupée sera égale a la taille du `source_directory/` plus la taille de tous les changement faits les trois derniers jours .

Isoler la sauvegarde du reste du système

Si vous faites vos sauvegardes dans un répertoire de votre système de fichiers alors il est possible que lors d'un problème vos sauvegardes soient endommagées. Nous allons donc séparer nos sauvegardes de nos données.
La meilleure solution consiste a faire nos sauvegardes sur une partition séparée ainsi si le système de fichiers est abîmé nous pouvons tout de même récupérer nos sauvegardes.
Si la partition se trouve aussi sur un disque dur séparé alors cela protégé aussi la sauvegarde d'un crash disque.
Il est possible aussi de créer un serveur de sauvegarde avec une machine physiquement éloignée de la machine à sauvegarder.
Sur la machine source, on exporte les répertoires à sauvegarder via NFS en read-only vers la machine de sauvegarde. Le serveur de sauvegarde lance la routine de snapshot comme si elles étaient en local. Si nous optons pour cette approche nous serons vulnérable si :

- Un trou de sécurité sur le compte root à distance est découvert sur NFS en read-only
- La machine source a été attaquée et corrompue

La meilleure solution dans ce cas est de créer deux serveurs de sauvegarde pour pouvoir en garder un toujours éteint.
Si vous n'avez pas de serveur avec une ligne dédiée à la sauvegarde, il vaut mieux laisser tomber l'idée de NFS et utiliser `rsync -e ssh`.
Voici encore un exemple on peut aussi sauvegarder des postes sous Windows en partageant sous SAMBA. La sauvegarde s'effectuera sur le système SAMBA comme sur un répertoire normal.

Faire en sorte que la sauvegarde soit en Read-only

Nous voulons éviter de laisser la sauvegarde SnapShot monté en read-write dans un endroit publique. Malheureusement nous ne pouvons pas le laisser monté read-only tout le temps car le processus de sauvegarde lui-même a besoin de l'accès en écriture. Le mieux serait que la sauvegarde soit montée en read-only dans un endroit publique, et en même temps, en read-write dans un répertoire privé accessible seulement par le root, comme par exemple `/root/snapshot`.

La solution : utiliser NFS sur le localhost

Pour créer un disque en lecture seule mais aussi en écriture, nous utiliserons NFS. Pour cela on monte la partition ou les sauvegardes sont stockées et accessible seulement par le root, par exemple `/root/snapshot`. Ensuite on l'exporte en read-only, via NFS, mais seulement sur la même machine. Il suffit d'ajouter à `/etc/exports`:
/root/snapshot 127.0.0.1(secure,ro,no_root_squash)
Ensuite lancer `nfs` et `portmap` a partir de `/etc/rc.d/init.d/`. Enfin monter le répertoire exporté en , read-only, sur `/snapshot`:
mount -o ro 127.0.0.1:/root/snapshot /snapshot
Et vérifier que ça a fonctionné: mount ...
/dev/hdb1 on /root/snapshot type ext3 (rw)
127.0.0.1:/root/snapshot on /snapshot type nfs (ro,addr=127.0.0.1)
Nous avons donc réalisé ce que nous avions besoin : Seulement le root auras la permission d'écrire dans la sauvegarde (en utilisant `/root/snapshot`). Les autres utilisateurs verrons seulement le répertoire `/snapshot` en read-only . Pour plus de sécurité nous pouvons garder `/root/snapshot` monté en read-only et le passer en écriture seulement pour les sauvegardes.

Lancer avec `cron`

Pour lancer le snapshot automatiquement, ajouter a la crontab root le fichier :
0 */4 * * * /usr/local/bin/make_snapshot.sh
0 13 * * * /usr/local/bin/daily_snapshot_rotate.sh
Le script `make_snapshot.sh` se lance toutes les 4 heures
Et le script `daily_snapshot_rotate.sh` tous les jours a 13:00
Pour supprimer l'envoi de mail de résultat par la crontab toutes les 4 heures on envoie le résultat vers `/dev/null` :
0 */4 * * * /usr/local/bin/make_snapshot.sh >/dev/null 2>&1
Mais cette méthode feras que nous ne verrons pas si il y a des erreurs du script `make_snapshot.sh` ou qu'il ne peut pas s'exécuter. Dans ce cas il suffit de créer un Log qui sauvegarde n'importe quel comportement inhabituel qui envoie stdout vers /dev/null mais qui garde stderr :
0 */4 * * * /usr/local/bin/make_snapshot.sh >/dev/null

Site Web reférence :

http://www.mikerubel.org/computers/rsync_snapshots/

Script : **`make_snapshot.sh`**

```
#!/bin/bash
unset PATH        # evite l'utlisation de $PATH
# ------------ commandes systeme utlisees par le script ------------------
ID=/usr/bin/id;
ECHO=/bin/echo;
MOUNT=/bin/mount;
RM=/bin/rm;
MV=/bin/mv;
CP=/bin/cp;
TOUCH=/bin/touch;
RSYNC=/usr/bin/rsync;
# ------------ Placement des fichiers -------------------------------------
MOUNT_DEVICE=/dev/hdb1;
SNAPSHOT_RW=/root/snapshot;
EXCLUDES=/usr/local/etc/backup_exclude;
# ------------ Le script lui-même -----------------------------------------
# Nous devons etre logge en tant que root
if (( `$ID -u` != 0 )); then { $ECHO "Vous devez etre root.Sortie..."; exit; } fi
# Monte le point RW en read-write; sinon annule
$MOUNT -o remount,rw $MOUNT_DEVICE $SNAPSHOT_RW ;
if (( $? )); then
{
        $ECHO "snapshot: ne peut pas remonter $SNAPSHOT_RW en readwrite";
        exit;
}
fi;
# Snapshots tournants de /home
# etape 1: supprime le snapshot le plus ancient si il existe :
if [ -d $SNAPSHOT_RW/home/hourly.3 ] ; then
$RM -rf $SNAPSHOT_RW/home/hourly.3 ;
fi ;
# etape 2: fait tourner le snapshot du milieu par le dernier :
if [ -d $SNAPSHOT_RW/home/hourly.2 ] ; then
$MV $SNAPSHOT_RW/home/hourly.2 $SNAPSHOT_RW/home/hourly.3 ;
fi;
if [ -d $SNAPSHOT_RW/home/hourly.1 ] ; then
$MV $SNAPSHOT_RW/home/hourly.1 $SNAPSHOT_RW/home/hourly.2 ;
fi;
# etape 3: faire un hard-link et copie le dernier snapshot, si il existe
if [ -d $SNAPSHOT_RW/home/hourly.0 ] ; then
$CP -al $SNAPSHOT_RW/home/hourly.0 $SNAPSHOT_RW/home/hourly.1 ;
fi;
# etape 4: rsync du systeme vers le dernier snapshot
$RSYNC
        -va --delete --delete-excluded
        --exclude-from="$EXCLUDES"
        /home/ $SNAPSHOT_RW/home/hourly.0 ;
# etape 5: adapter le mtime a hourly.0 pour donner l'heure du snapshot
$TOUCH $SNAPSHOT_RW/home/hourly.0 ;
# maintenant remonter le point snapshot en readonly
$MOUNT -o remount,ro $MOUNT_DEVICE $SNAPSHOT_RW ;
if (( $? )); then
{
        $ECHO "snapshot: could not remount $SNAPSHOT_RW readonly";
        exit;
} fi;
```