Sauvegardes

La gestion de la sauvegarde revêt bien des aspects sur les systèmes Linux. Des outils historiques qui géraient dans le meilleur des cas un lecteur de bande local jusqu'aux outils modernes sophistiqués et aux logiciels de sauvegarde commerciaux, l'éventail est large. L'essentiel est de connaître les moyens disponibles, et d'adapter sa stratégie de sauvegarde à ses besoins en fonction du temps et de l'argent qu'on est prêt à investir dans la sauvegarde.

1. Les utilitaires d'archivage

Les utilitaires d'archivage permettent de réaliser les sauvegardes les plus simples, et grâce à cette simplicité, sans doute les plus fiables. Leur principe est simple : elles envoient un ensemble de fichiers (en général une arborescence de répertoires) vers un fichier, qu'il s'agisse d'un fichier ordinaire ou d'un fichier spécial qui désigne un périphérique de stockage.

a. La commande tar

La commande **tar**, d'usage universel dans les environnements Linux, est à connaître absolument. Sa richesse fonctionnelle peut impressionner, mais si la commande **tar** présente de très nombreuses options, moins d'une dizaine sont utilisées dans la plupart des situations.

Syntaxe de la commande tar pour créer une archive

```
tar action compression verbosité -f fichier_archive répertoire
```

Syntaxe de la commande tar pour lister ou extraire une archive

```
tar action compression verbosité -f fichier_archive
```

Commande tar : options et paramètres		
action	-c	Crée une archive. Il faut alors indiquer en dernier paramètre le répertoire à partir duquel l'archive est créée.
	-t	Liste le contenu d'une archive existante.
	-x	Extrait le contenu d'une archive existante dans le répertoire courant.
compression		Pas de compression sur l'archive manipulée.
	-z	Compression au format gzip de l'archive manipulée.
	-j	Compression au format bz2 de l'archive manipulée.
verbosité		Pas de verbosité, affichage minimum.
	-v	Verbosité, affichage détaillé.
fichier_archive		Le fichier qui reçoit ou héberge l'archive. Ce fichier peut être un fichier spécial en mode bloc ou en mode caractères. Aujourd'hui presque toujours un fichier ordinaire.
répertoire		Dans le cadre d'une création d'archive, désigne le répertoire à partir duquel l'archive est créée.

Même si ça n'est pas une obligation, il est d'usage d'affecter une extension « .tar » aux fichiers contenant une archive tar, suivie d'une extension liée au mode de compression « .gz » ou « .bzip2 ».

Dans un usage de la commande **tar** à des fins de sauvegarde, on dirigera l'archive de sauvegarde vers un périphérique amovible ou vers un espace de stockage distant.

Exemple d'utilisation de la commande tar

Dans cet exemple, on crée une archive tar compressée à partir d'un répertoire, on efface le répertoire, puis on le restaure à partir de l'archive.

```
A:~# ls
trucs
A:~# # remarque : création de l'archive
A:~# tar czf sauvegarde.tar.gz trucs
A:~# ls
sauvegarde.tar.gz  trucs
A:~# # remarque : destruction du répertoire trucs
A:~# rm -r trucs
A:~# ls
sauvegarde.tar.gz
A:~# # remarque : restauration de l'archive
A:~# tar xzf sauvegarde.tar.gz
A:~# ls
sauvegarde.tar.gz  trucs
A:~#
```

Si la commande **tar** est employée pour créer une archive sur bande magnétique et non sur disque, il est recommandé de ne pas utiliser d'option de compression. Le format compressé empêcherait une récupération partielle des données en cas de détérioration de la bande.

b. La commande cpio

La commande **cpio** dont l'usage tend à disparaître en environnement Linux permet de réaliser des archives non compressées d'un ensemble de fichiers et répertoires.

cpio est d'un usage particulièrement non intuitif, et n'est en général utilisé que dans des cas spécifiques. Le problème de **cpio** vient de ce que cette commande n'accepte pas qu'on lui désigne les éléments à sauvegarder en tant que paramètre comme le fait la commande **tar**. Il faut lui indiquer ces éléments sous forme de liste de fichiers sur son entrée standard. De même, toutes les manipulations en sortie se font par redirection de la sortie standard. Si la commande **cpio** a survécu malgré ces handicaps d'un autre temps, c'est justement grâce à ces limitations syntaxiques : la liste de fichiers à sauvegarder est presque toujours fournie par redirection du résultat d'une commande **find**. Or, la commande find est capable de faire des recherches extrêmement précises sur de très nombreux critères. C'est donc dans les cas où l'on veut faire des sauvegardes très sélectives que l'on utilisera **cpio**.

Syntaxe de la commande cpio pour créer une archive

```
find répertoire critère -print | cpio options > fichier_archive
```

Syntaxe de la commande cpio pour lister ou extraire une archive

```
cpio options < fichier_archive
```

Commande cpio : options et paramètres		
répertoire		Le répertoire de base à partir duquel se fait la recherche.
critère		Critères de recherche selon la syntaxe de la commande find.
options	-o	Mode copy-out. Indique qu'on est en mode de création d'archive. Exclusif des options i et t.
	-t	Associée à l'option i, liste le contenu d'une archive existante. Exclusif de l'option o.
	-i	Mode copy-in. Indique qu'on est en mode d'extraction ou de consultation d'archive. Exclusif de l'option o.
	-v	Facultatif : rend la commande bavarde.

fichier_archive	Le fichier (spécial ou ordinaire) qui recevra l'archive.

Exemple d'utilisation de la commande cpio

S'il est indispensable de savoir utiliser la commande tar naturellement, on peut raisonnablement ne pas se souvenir de la syntaxe cpio.

```
A:~# ls
trucs
A:~# # remarque : création de l'archive
A:~# find trucs -print | cpio -o > archive.cpio
1 block
A:~# ls
archive.cpio  trucs
A:~# # remarque : destruction du répertoire trucs
A:~# rm -rf trucs
A:~# # remarque : restauration de l'archive
A:~# cpio -i < archive.cpio
1 block
A:~# ls
archive.cpio  trucs
A:~#
```

2. Les logiciels de sauvegarde

a. AMANDA

AMANDA : *Advanced Maryland Automatic Network Disk Archiver* est une solution de sauvegarde crée initialement par l'université du Maryland sous licence BSD. Disponible sous licence communautaire (gratuite) ou commerciale, AMANDA permet de sauvegarder localement ou en réseau, sur disques ou sur bandes, les données des systèmes Linux/Unix ou Windows.

b. Bacula

Bacula est une solution de sauvegarde sous licence GPL qui permet de sauvegarder localement ou en réseau, sur disques ou sur bandes, les données des systèmes Linux/Unix ou Windows.

c. BackupPC

BackupPC est une solution de sauvegarde sous licence GPL qui permet de sauvegarder localement ou en réseau, sur disques ou sur bandes, les données des systèmes Linux/Unix ou Windows.

d. Les logiciels commerciaux

La plupart des grands éditeurs de logiciels de sauvegarde supportent, souvent en option, la sauvegarde des systèmes Linux. Il faudra alors installer sur chaque système un agent de sauvegarde qui permettra de renvoyer les données vers le serveur de sauvegarde.

3. Duplication et synchronisation de données

a. Copie binaire avec dd

La commande de copie bloc à bloc dd permet de réaliser des copies de bas niveau d'un périphérique. Elle est utilisée notamment pour la duplication de disques durs, mais aussi pour la création d'images binaires de périphériques de stockage.

Syntaxe générique de la commande dd

```
dd if=entrée of=sortie bs=taille_blocs count=nombre_blocs
```

Commande dd : options et paramètres	
entrée	Le fichier à copier. Généralement un fichier spécial en mode bloc.
sortie	Le fichier vers lequel copier. Fichier spécial en mode bloc ou fichier ordinaire.
taille_blocs	Facultatif. Désigne la taille des blocs à copier.
nombre_blocs	Facultatif. Le nombre de blocs à copier. Si le paramètre est omis, la copie s'arrête dès qu'elle n'est plus possible.

Utilisation de la commande dd pour une copie de disque dur

Copie du disque sdb vers le disque sdc.

```
root@serveur# dd if=/dev/sdb of=/dev/sdc
root@serveur#
```

Utilisation de la commande dd pour réaliser l'image iso d'un cdrom

Le fichier iso généré est gravable par n'importe quel logiciel ou utilisable dans une machine virtuelle.

```
root@serveur# dd if=/dev/cdrom of=/home/toto/image.iso
root@serveur#
```

Utilisation de la commande dd pour effacer physiquement une clé usb

Effacement physique de tous les blocs d'une clé usb vue comme le périphérique sdd. Attention, les données ne sont récupérables par aucun moyen simple. Ne vous trompez pas de disque !

```
root@serveur# dd if=/dev/zero of=/mnt/sdd
root@serveur#
```

Utilisation de la commande dd pour créer un fichier vide de 100 Mo

Commande dd utilisée pour recevoir un espace de swap, ou générer de gros fichiers pour des tests de copie.

```
root@serveur# dd if=/dev/zero of=/home/toto/fichiervide bs=1024 count=100000
root@serveur#
```

b. Génération de fichier iso avec mkisofs

Les fichiers iso sont des images binaires de cdrom ou dvdrom. Les images iso sont montables par la commande **mount**, gravables par n'importe quel logiciel de gravure, et exploitables depuis les machines virtuelles où elles sont vues comme un cdrom. Il peut être utile de générer des images iso à partir d'une arborescence de fichiers et répertoires ; la commande **mkisofs** est là pour ça.

Syntaxe de la commande mkisofs

```
mkisofs -J -o image répertoire
```

Commande mkisofs : options et paramètres	
-J	Facultatif : génère des enregistrements Joliet en plus de la structure de noms iso9960. Améliore la compatibilité avec les systèmes Windows.

-o *image*	Le fichier iso qui sera généré. Généralement avec l'extension « .iso ».
répertoire	Le répertoire à partir duquel l'image iso sera générée.

Exemple d'utilisation de la commande mkisofs

Le fichier iso généré peut être gravé directement par n'importe quel logiciel de gravure.

```
bob@cuicui:~/Temp$ ls
data
bob@cuicui:~/Temp$ mkisofs -o imgcd.iso data
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 8192
Path table size(bytes): 50
Max brk space used 23000
178 extents written (0 MB)
bob@cuicui:~/Temp$ ls
imgcd.iso  data
bob@cuicui:~/Temp$ file imgcd.iso
imgcd.iso: ISO 9660 CD-ROM filesystem data 'CDROM                         '
bob@cuicui:~/Temp$
```

mkisofs est le nom historique de la commande permettant de créer des fichiers iso. Toutefois, cette commande a été renommée récemment en genisoimage, et mkisofs est présent sur les distributions récentes sous forme de lien symbolique vers genisoimage.

L'image iso ainsi générée est un fichier unique a priori insondable hors de son exploitation par un logiciel adapté. En fait, il est possible de monter le fichier image comme s'il s'agissait d'un périphérique ordinaire.

Montage local d'une image iso

```
mount -o loop fichier_image point_montage
```

Où *fichier_image* représente l'image iso à monter, et *point_montage* le répertoire qui recevra ce montage. L'option loop est indispensable pour le montage d'un fichier image.

c. Synchronisation de données avec rsync

Dans le cadre des stratégies de préservation des données, il peut être utile de répliquer des données d'un serveur sur un autre, soit afin de garantir une disponibilité géographique de données identiques, soit pour se préserver d'une défaillance d'un disque dur ou d'un serveur. La commande **rsync** remplit cet office à merveille.

rsync propose plusieurs modes de fonctionnement, mais le plus courant dans le cadre de synchronisation de données est de disposer d'un service **rsync** sur un serveur, et de planifier des synchronisations régulières depuis les machines contenant les données à répliquer.

Configuration d'un serveur rsync

La configuration se fait par le biais de deux fichiers : le fichier **/etc/default/rsync** qu'il faudra modifier, et le fichier **/etc/rsyncd.conf** qu'il faudra créer.

Modification du fichier /etc/default/rsync

```
RSYNC_ENABLE=true
```

Ce paramètre permet le démarrage automatique ou manuel de **rsync** en tant que service.

Création du fichier /etc/rsyncd.conf

```
uid = utilisateur
```

```
read only = false
[instance]
  path = répertoire
```

Fichier /etc/rsyncd.conf : directives et paramètres	
utilisateur	Le compte au nom duquel les opérations d'écritures seront réalisées sur le serveur.
read only = false	Indispensable pour que le service puisse écrire sur le disque.
instance	Nom au choix, il y aura autant d'instances que de clients à répliquer. Ce nom sera repris sur le client lors de la demande de synchronisation.
répertoire	Le répertoire dans lequel les données synchronisées seront écrites. Le compte utilisateur employé doit avoir des droits d'écriture sur ce répertoire.

Il faudra après configuration relancer le service **rsync** par les moyens habituels.

```
/etc/init.d/rsync restart
```

Synchronisation des données depuis un client

La synchronisation se fera à la demande ou depuis une tâche planifiée avec la commande **rsync**.

Syntaxe de la commande rsync pour une synchronisation ponctuelle

```
rsync -av --delete /répertoire/ ip_serveur::instance
```

Commande rsync : options et paramètres	
-a	Mode archive : réplique les données à l'identique, en préservant notamment les permissions et les propriétaires.
-v	Facultatif : affiche le détail de chaque opération. Permet de visualiser la progression de la synchronisation.
--delete	Copie miroir : les données effacées sur le client le sont aussi sur le serveur.
répertoire	Le répertoire des données locales à dupliquer.
instance	Le nom de l'instance paramétrée dans /etc/rsyncd.conf sur le serveur.

Synchronisation sécurisée de données avec rsync

Si la synchronisation de données doit se faire en environnement hostile, il est possible de s'en remettre à SSH pour le transport des données. Dans ce mode de fonctionnement, le démon rsync ne s'exécute pas sur le serveur, et l'exécutable est lancé à la volée par SSH pour toute connexion entrante.

Synchronisation sécurisée avec rsync

```
rsync -av --delete -e ssh répertoire
utilisateur@adresse_serveur:/chemin_cible
```

rsync avec ssh : options et paramètres	
-a	Mode archive : réplique les données à l'identique, en préservant notamment les permissions et les propriétaires.
-v	Facultatif : affiche le détail de chaque opération. Permet de visualiser la progression de la synchronisation.

--delete	Copie miroir : les données effacées sur le client le sont aussi sur le serveur.
répertoire	Le répertoire des données locales à dupliquer.
utilisateur	Le compte utilisateur existant sur la machine cible qui sera utilisé pour la session ssh.
adresse_serveur	Adresse IP du serveur cible.
chemin_cible	Répertoire cible pour la synchronisation de données sur la machine cible.

Exemple de synchronisation sécurisée

*La commande **rsync** permet de créer un miroir entre disques sur systèmes différents à peu de frais.*

```
[root@beta data]# rsync -av --delete -e ssh /root/data
root@192.168.200.106:/root/svg
root@192.168.200.106's password:
building file list ... done
created directory /root/svg
data/
data/deux/
data/deux/fichier2
data/trois/
data/un/
data/un/fichier1

sent 50047 bytes  received 88 bytes  14324.29 bytes/sec
total size is 49785  speedup is 0.99
[root@beta data]#
[root@beta data]# rm -rf un
[root@beta data]# rsync -av --delete -e ssh /root/data
root@192.168.200.106:/root/svg
root@192.168.200.106's password:
building file list ... done
deleting data/un/fichier1
deleting data/un/
data/

sent 109 bytes  received 26 bytes  38.57 bytes/sec
total size is 35964  speedup is 266.40
[root@beta data]#
```