# Leçon 4 : Sauvegarde et planification

## 4.1 Introduction

Dans cette leçon, nous allons étudier les moyens de sauvegarder *en-ligne* l'information. Comme les serveurs Linux sont souvent présents sur Internet et connectés en permanence, il est très courant de sauvegarder les éléments essentiels (configuration, répertoire web, bases de données, …) sur un autre serveur.

Il existe également des logiciels de sauvegarde plus classique. Ainsi, comme sur tous les systèmes, il existe des logiciels chargés de faire des copies de sauvegarde sur bande magnétique ou tout autre média utilisé pour le backup ou l'archivage. Cependant, si l'on souhaite réaliser un backup très rapide, la copie vers un serveur est l'un des moyens les plus simples.

## 4.2 Sauvegarde de fichiers

Pour sauvegarder des fichiers, on peut utiliser des *programmes d'archivage et de compression*. A l'instar des programmes permettant de créer des fichiers .ZIP, il existe, sous Linux, quelques formats courant pour la compression et la distribution de fichiers archivés.

### 4.2.1 Le format standard ZIP

Le format ZIP, très ancien, est une véritable référence en moyen de distribution et de compression. Il n'est, par contre, pas très souvent utilisé dans le monde Unix. Cependant, les commandes standard `zip` et `unzip` sont présentes pour créer, manipuler ou extraire de tels fichiers.

#### *Création d'une archive ZIP*

```
$ zip -r /tmp/save-etc.zip /etc/*
```

*L'option `-r` assure que toute l'arborescence sera traitée, le fichier destination est d'abord mentionné et puis les fichiers à inclure dans l'archive. Dans cet exemple, une archive nommée `save-etc.zip` est créée dans le dossier `/tmp` et reprend l'ensemble des répertoires et sous-répertoires du dossier `/etc`. Les informations propres à Unix (propriétaire, permission, …) sont perdues.*

#### *Extraction d'une archive ZIP*

```
$ unzip monfichier.zip
```

*Cette commande extrait le fichier `monfichier.zip` dans le répertoire courant.*

#### *Lister les fichiers d'une archive ZIP*

```
$ unzip -l monfichier.zip
```

*Cette commande permet de connaître les fichiers et dossiers qui se trouvent dans une archive ZIP.*

### 4.2.2 Les formats tar.gz et tar.bz2

Parmi les formats de fichiers courant sous Linux, nous avons les formats `.tar.gz` ou `.tar.bz2`. La double extension mentionnée fait référence à 2 logiciels très particuliers.

Le premier, `tar`, permet de transformer un répertoire et ses fichiers en un seul fichier (une sorte de « regroupement »).

Les commandes `gzip` ou `bzip2` sont des programmes de compression : elles permettent de compresser un fichier (ici, en l'occurrence, le fichier `.tar`).

Sous **Linux**, il est possible de combiner ces commandes en une seule, alors que certains systèmes Unix nécessitent d'exécuter la commande `tar` puis `gzip` (ou `bzip2` au choix de l'utilisateur) successivement.

### *Création d'une archive*

```
$ tar cvzf /tmp/save-etc.tar.gz /etc
$ tar cvjf /tmp/save-etc.tar.bz2 /etc
```

*Dans le premier exemple, nous allons créer une archive `save-etc.tar.gz`. Cette archive contiendra le dossier `/etc` et tous les fichiers et dossiers contenus dans celui-ci. Le paramètre `c` permet de créer l'archive, le paramètre `v` provoque un affichage verbeux, le paramètre `z` indique d'exécuter `gzip` après la création du fichier `.tar` et enfin, le paramètre `f` indique que la destination est un fichier.*

*Le second exemple diffère simplement par le paramètre `j` et le nom de l'archive. Ici, c'est la commande `bzip2` qui est exécutée suite à la création du fichier `.tar`.*

### *Extraction d'une archive*

```
$ tar xvzf /tmp/save-etc.tar.gz
$ tar xvjf /tmp/save-etc.tar.bz2
```

Dans le premier exemple, nous allons extraire l'archive `save-etc.tar.gz` qui se trouve dans le dossier `/tmp`. L'archive sera extraite dans le répertoire courant. Le format de compression est `gzip`. Enfin, l'option `x` permet d'informer `tar` de réaliser l'extraction.

Dans le second exemple, nous allons extraire l'archive `save-etc.tar.bz2` qui se trouve dans le dossier `/tmp`. L'archive sera extraite dans le répertoire courant. Le format de compression est `bzip2`.

### *Tester / lister les fichiers d'une archive*

```
$ tar tvzf /tmp/save-etc.tar.gz
$ tar tvjf /tmp/save-etc.tar.bz2
```

Ces deux commandes permettent de lister les fichiers et dossiers de l'archive `save-etc.tar.gz` ou `save-etc.tar.bz2`.

## 4.3 Sauvegarde d'une image disque

Une autre possibilité intéressante est de réaliser une sauvegarde d'un système complet en vue de sa duplication sur un grand nombre de machines. Cette possibilité est très utilisée dans les parcs d'ordinateur homogène (même configuration matérielle).

Parmi les logiciels connus[13], nous avons Symantec Ghost Solution Suite, Acronis True Image, Paragon Hard Disk Manager. A coté de ces solutions commerciales, il y a, également, CloneZilla (`http://www.clonezilla.org`), comprenant une suite d'outils gratuits et open-sources permettant la duplication des machines.

[13] Cette liste ne reprend que les logiciels que j'ai été amené à utiliser. Il en existe certainement bien d'autres.

## 4.4 Sauvegarde incrémentale et différentielle

A coté des outils permettant des sauvegardes complètes du système, il y a aussi bon nombre de logiciels qui proposent des sauvegardes régulières et se basant sur les sauvegardes précédentes, déjà réalisées.

Ainsi, on parle de *sauvegarde complète* quand l'ensemble du système est sauvegardé. Une *sauvegarde incrémentale* est une sauvegarde reprenant les dernières modifications depuis la dernière sauvegarde incrémentale ou complète réalisée. Dans ce scénario, la restauration des données exige de rétablir la dernière sauvegarde complète et tous les incréments effectués.

On parle également de *sauvegarde différentielle* qui est une sauvegarde des différences par rapport à la dernière sauvegarde complète du système. Dans ce scénario, la restauration du système exige de rétablir la dernière sauvegarde complète et la dernière différentielle effectuée.

Dans le monde Unix, il y a également beaucoup de solution de sauvegarde permettant ce type de backup. A l'instar de Symantec BackupExec, qui est un des logiciels de backup les plus connus dans le monde Windows, des solutions analogues sont disponibles sous Linux :

- **MondoRescue** (http://www.mondorescue.org/) permet de faire une sauvegarde du système sur un périphérique externe, disque dur, disque réseau, …
- **Bacula** (http://www.bacula.org/) permet de faire des sauvegardes complètes, incrémentales et différentielles. Les sauvegardes peuvent être centralisées sur un lecteur de bande, … La solution Bacula est l'une des plus complètes dans le monde libre Linux.
- **Amanda** (http://www.amanda.org/) est un autre logiciel de sauvegarde dont les fonctionnalités sont proches également de Bacula. Ce logiciel a été développé par l'université du Maryland.

## 4.5 Sauvegarde en réseau

Autre moyen de sauvegarde très courant sur les serveurs, c'est la copie de données vers un autre serveur pour backup. Cette copie *en live, planifiée* permet une récupération des fichiers quasi-immédiate.

Les outils standard pour réaliser ces opérations sont :

- `scp` qui permet de faire une copie de fichiers vers une machine distante, en utilisant le service ssh installé par défaut sur la plupart des serveurs Linux.
- `rsync` qui permet de synchroniser deux dossiers (pour créer un *miroir* d'un dossier). Son atout étant que seules les modifications sont transférées.
- `lftp` - dont nous parlerons plus tard - permet de sauvegarder des fichiers sur un site FTP quelconque. Cette option est très intéressante lorsqu'on loue des serveurs *cloud* sur internet car il est courant que le fournisseur propose un espace FTP de backup.

### 4.5.1 Copie de fichiers à distance

```
$ scp monfichier.tar.gz p010544@dartagnan.cg.helmo.be:/tmp
```

*Copie le fichier `monfichier.tar.gz` sur la machine `dartagnan`, dans le répertoire `/tmp`, en utilisant le compte `p010544`.*

Dans ces commandes, on doit toujours mentionner *la source* puis *la destination*. La *source* ou la *destination* peuvent être locale ou distante. Ainsi, si l'on souhaite copier le fichier depuis `dartagnan`, nous pourrions écrire :

```
$ scp p010544@dartagnan.cg.helmo.be:/tmp/monfichier.tar.gz .
```

*Dans cet exemple, nous copions le fichier `monfichier.tar.gz` depuis le serveur `dartagnan` vers le répertoire courant (notez le « . » comme destination, désignant le dossier courant).*

Pour que cette copie fonctionne, il faut que le système distant exécute le service SSH. Nous étudierons celui-ci plus loin dans le cadre de ce cours.

### 4.5.2 Création d'un miroir

La notion de dossier miroir est assez simple à comprendre : il s'agit de *synchroniser* deux dossiers de sorte que le dossier destination (appelé miroir du premier) contiennent au moins les fichiers du dossier source. L'intérêt étant que la synchronisation ne transfère que les fichiers modifié ou ajouté et n'existant pas sur le dossier destination.

Cette possibilité est donc intéressante si le répertoire source est important et que les mises à jour ne concerne que certains fichiers.

```
$ rsync -Cavz /home p010544@dartagnan.cg.helmo.be:~/backup -e ssh
```

*Cette commande crée un dossier miroir du dossier `/home` du serveur. Ainsi, l'objectif est que le dossier miroir distant contiennent tous les fichiers contenu dans `/home` pour des raisons de backup. La commande `rsync` permet de réaliser cette synchronisation. Comme pour `scp`, il faut mentionner le dossier source (qui sera copié) et le dossier destination (qui sera le miroir). Le dossier source ou destination peut être local ou distant.* ***Dans cet exemple****, le dossier local est `/home` et le dossier distant est `~/backup` (pour rappel « ~ » fait référence au dossier personnel de l'utilisateur, ici `p010544`, ce qui revient à dire qu'il s'agit du dossier distant `/home/p010544/backup`).*



## 4.6 Planification des tâches

Nous avons vu comment il était possible de créer des sauvegardes. Cependant, une sauvegarde ponctuelle ou réalisée par l'utilisateur, n'est pas très intéressante. Seule les sauvegardes planifiées, automatiques et programmées sont réellement intéressante car l'utilisateur n'intervenant pas dans le processus, il n'y a pas de danger que celle-ci soit oubliée.

Il y a deux types de planification possibles :

1. Les tâches *ponctuelles* qui sont lancées une seule fois à un moment déterminé. Ces tâches sont exécutées puis leur planification est supprimée.
2. Les tâches *répétitives* qui sont lancées régulièrement sur le système. La planification de celles-ci est mémorisée et le système maintient celles-ci.

Les commandes `at` et `crontab` sont utilisées pour ces deux types. La commande `at` permet de planifier l'exécution d'une tâche ponctuelle alors que `crontab` permet de planifier l'exécution de tâches répétitives.

Pour des raisons de sécurité, il est possible de limiter les utilisateurs autorisés à exécuter ces commandes. Par défaut, tous les utilisateurs peuvent programmer des tâches ponctuelles ou répétitives sur les systèmes CentOS 7. Il est possible de limiter les utilisateurs en ajoutant des éléments dans `/etc/cron.allow`, `/etc/cron.deny`, `/etc/at.allow` ou `/etc/at.deny`. Pour plus d'information, référez-vous aux pages de manuel de ces commandes.

**Attention !** Comme il s'agit de tâches planifiées, lancées automatiquement, ***il ne peut pas y avoir d'interaction avec l'utilisateur***. Ces tâches doivent s'exécuter complètement automatiquement. L'environnement utilisé est réduit : il n'y a pas de *stdin* (ou entrée standard), ni de *stdout* (ou sortie standard). Ainsi, si l'application utilisée nécessite une entrée au clavier, il peut est intéressant de faire une redirection. Par défaut, toutes les sorties produites sont envoyées par mail à l'utilisateur ayant programmé la tâche.

### 4.6.1 Les tâches ponctuelles

Une tâche ponctuelle se planifie grâce à la commande `at`. L'expression du moment de démarrage est assez large :

```
$ at 14:10
at> /usr/sbin/reboot
at> ^D
job 1 at Sun Sep 20 14:10:00 2015
```

Dans cette exemple, nous planifions l'exécution de la commande `reboot` aujourd'hui à 14h10 (soit le dimanche 20 septembre à 14h10). Par précaution, il vaut toujours mieux préciser le chemin complet de la commande ou du script qui doit être démarré. Une fois la tâche planifiée, on quitte et sauvegarde la planification en appuyant sur CTRL+D.

La commande `atq` permet de lister les tâches planifiées pour l'utilisateur courant et `atrm` permet de supprimer une tâche planifiée avec `at`.

Il est possible de préciser bien d'autres planifications.

| Exemples | Explication |
|---|---|
| `at 14:3009202015` | Planification le 20 septembre 2015 à 14h30 |
| `at 14:30 sep 20 2015` | Planification le 20 septembre 2015 à 14h30 |
| `at now + 5 hours` | Planification dans 5 heures à partir de maintenant |
| `at now + 1 day` | Planification dans 1 jour à partir de maintenant |
| `at 11:00 next month` | Planification le même jour dans un mois, à 11h |
| `at 22:00 tomorrow` | Planification pour demain à 22h |

### 4.6.2 Les tâches répétitives

La planification de tâches répétitives se fait grâce à la commande `crontab`. Le format, un peu étrange, à utiliser permet d'être précis dans la planification.

Pour planifier une tâche répétitive, il faut utiliser :

```
$ crontab -e
```[14]

[14] Cette commande lance l'éditeur par défaut. Il est possible de changer d'éditeur en entrant:
`$ EDITOR=gedit crontab -e`

La planification se fait en colonne, dans l'ordre suivant :

```
M      H      j      m      J      commande
```

Avec :

- `M` représentant la minute
- `H` représentant l'heure
- `j` représentant le jour
- `m` représentant le mois
- `J` représentant le jour de la semaine.
- `commande` représentant la commande à exécuter

Dans chaque colonne, on mentionne la donnée souhaitée ou « * » pour dire « à chaque ».

| Planification | | | | | | Explication |
|---|---|---|---|---|---|---|
| **0** | **0** | **20** | **09** | **1** | **cal** | Exécute la commande `cal` tous les 20 septembre à minuit lorsque c'est un lundi (lundi = jour 1 ; dimanche = jour 7) |
| **0** | **0** | **20** | ***** | ***** | **cal** | Exécute la commande `cal` tous les 20 de chaque mois à minuit, peu importe le jour de la semaine |
| **0** | **0** | ***** | ***** | ***** | **cal** | Exécute la commande `cal` tous les jours à minuit quelque soit le jour, le mois ou le jour de la semaine. |

On remarque que cette planification est finalement assez précise et simple à comprendre. Il est très courant de lancer des scripts par `crontab` afin de réaliser des tâches précises comme les backup automatiques.

## 4.7 Exercices

1. Réaliser une sauvegarde du répertoire `/var` dans un fichier `/tmp/var-back.bz2`. Ce fichier sera compressé par bzip2.

2. Réalisez le même exercice que le précédent, en utilisant la commande zip : créer l'archive `/tmp/var-back.zip` contenant l'ensemble du dossier `/var`.

3. Testez et décompressez l'une des archives précédentes dans votre dossier personnel

4. Programmez une tâche ponctuelle, pour le cours prochain, au milieu de celui-ci, et lancer la commande `poweroff`.

5. Programmez une tâche répétitive s'exécutant tous les début de cours et exécutant la commande `ntpdate` (synchronisation d'horloge avec un serveur distant) vers le serveur `time.belnet.be`

6. Programmez pour la fin du cours la synchronisation de vos répertoires personnels (`/home`) vers le serveur `dartagnan`. Créer dans votre dossier personnel sur `dartagnan` un dossier `back` qui deviendra le miroir du votre dossier `/home`. Est-ce que cela fonctionne ? Pourquoi ?