BORG: Les bases

Table of Contents

Présentation

Borg est un outil de sauvegarde incrémentielle en ligne de commande écrit en Python. C'est un fork d'Attic mais un peu plus en avance puisqu'il corrige pas mal de bug d'Attic et propose des fonctionnalités supplémentaires (choix de la compression, par exemple).En outre le projet est très actif et en constante évolution.

Une des particularités de Borg est qu'il supporte la déduplication, c'est-à-dire que les fichiers sauvegardés sont découpés en une multitude de tronçons, et Borg ne sauvegarde que les tronçons qui ont été modifiés depuis la dernière sauvegarde, d'où une économie substantielle en termes d'espace disque et un gain lors de transfert des sauvegardes distantes. De plus Borg gère différents

types de compression permettant de diminuer encore la taille des sauvegardes ainsi que le chiffrement en AES 256-bit.

Particularités principales

- **Optimisation de l'espace disque**: la déduplication, basée sur une segmentation du contenu, est utilisée pour réduire le nombre d'octets stockés: chaque fichier est divisé en un certain nombre de morceaux de longueur variable et seuls les morceaux qui n'ont jamais été sauvegardés auparavant sont ajoutés au dépôt. Peu importe qu'ils viennent de machines différentes, de sauvegardes précédentes ou de la même sauvegarde. Il n'y aura aucun fichier en double dans ces sauvegardes, tout reposant sur les empreintes de chaque fichier sauvegardé. Tous les morceaux dans le même dépôt sont référencés, peu importe s'ils proviennent de différentes machines, à partir de sauvegardes précédentes, à partir de la même sauvegarde ou même à partir du même fichier unique.
- **Vitesse**: Le code critique (chunking, compression, chiffrage) est implémenté en C / Cython et Borg gère la mise en cache locale des fichiers / données d'index des morceaux ainsi que la détection rapide des fichiers non modifiés.
- **Chiffrement des données**: Toutes les données peuvent être protégées en utilisant le méthode de chiffrage AES 256-bit, l'intégrité des données et l'authenticité sont vérifiées en utilisant HMAC-SHA256. Les données sont chiffrées côté client.
- **Compression**: Toutes les données peuvent être compressées au choix par LZ4 (super rapide, faible compression), zlib (vitesse moyenne et compression aussi) ou lzma (basse vitesse, compression élevée).
- **Sauvegardes hors site**: Borg peut stocker des données sur un hôte distant accessible via SSH. Si Borg est installé sur l'hôte distant, des gains importants de performance peuvent être atteints par rapport à un système de fichiers réseau (sshfs, nfs, ...).
- **Sauvegardes montables comme un simple système de fichier**: Les archives de sauvegarde peuvent être montées comme des systèmes de fichiers dans l'espace utilisateur pour un examen interactif, rapide et faciles des sauvegardes, la restauration se faisant alors par une simple copie de fichier.

Installation

Le paquet borgbackup, disponible dans les dépôts de la distribution ou via pip, doit être installé sur tous les serveurs.

Installation sur Ubuntu

Le paquet borgbackup, disponible dans les dépôts des distributions récentes.

Pour les ancienns versions ajouter le repository borgbackup backports

```
$ sudo -HE add-apt-repository ppa:costamagnagianfranco/borgbackup
```

puis

```
$ sudo apt install borgbackup

Les NOUVEAUX paquets suivants seront installés :
   borgbackup libb2-1 libzstd1 python3-llfuse python3-msgpack
```

Installation sur Centos

Le paquet borgbackup, est disponible dans les dépôts EPEL.

Activer le dépôt dans /etc/yum.repos.d/epel.repo

```
[epel]
enabled=1
name=Extra Packages for Enterprise Linux 7 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch
mirrorlist=http://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basea
rch
failovermethod=priority
```

Installer le rpm borgbackup

```
$ yum install borgbackup

 Installé :
   borgbackup.x86_64 0:1.1.8-2.el7

 Dépendances installées :
   fuse.x86_64 0:2.9.2-11.el7
   fuse-libs.x86_64 0:2.9.2-11.el7
   libzstd.x86_64 0:1.3.8-1.el7
   lz4.x86_64 0:1.7.5-2.el7
   python34.x86_64 0:3.4.9-1.el7
   python34-libs.x86_64 0:3.4.9-1.el7
   python34-llfuse.x86_64 0:1.0-1.el7
   python34-msgpack.x86_64 0:0.5.6-4.el7
   python34-setuptools.noarch 0:39.2.0-2.el7

 Dépendances mises à jour :
   openssl.x86_64 1:1.0.2k-16.el7       openssl-libs.x86_64 1:1.0.2k-16.el7
```

Interface graphique

Vorta permet d’offrir la possibilité de faire des sauvegardes chiffrées, dédupliquées et compressées facilement.

Vorta est open source et permet de gérer différents profils de sauvegardes, plusieurs sources, plusieurs destinations, ainsi qu’une planification des sauvegardes.

Installation

Installer les dépendances

```
$ sudo apt install -y python3-pip libqt5x11extras5-dev libxcb-xinerama0
```

Installer les packages python

```
$ pip3 install vorta

Installing collected packages: appdirs, pytz, six, setuptools, tzlocal,
apscheduler, asn1crypto, pycparser,
cffi, cryptography, jeepney, secretstorage, entrypoints, keyring, bcrypt,
pynacl, pyasn1, paramiko, peewee,
psutil, PyQt5-sip, pyqt5, python-dateutil, qdarkstyle, vorta
```

intégration sur le bureau

À moins que Vorta ne soit démarré avec l'option **–foreground**, il sera réduit dans la barre d'état système sans ouvrir de fenêtre de paramètres. Assurez-vous que votre environnement de bureau prend en charge les icônes de la barre des tâches, par exemple. installer l'extension xfce4-indicator-plugin

```
 $ sudo apt-get install -y xfce4-indicator-plugin
```

Utilisation

Variables d'environnement

Borg permet de définir plusieurs variables d'environnement afin de simplifier les commandes exécutées, dont :

- **BORG_REPO**: Chemin d'accès au dépôt de sauvegardes.
- **BORG_PASSPHRASE**: Phrase de passe, si nécessaire, pour débloquer la clé de chiffrement.
- **BORG_PASSCOMMAND**: Commande utilisée pour fournir la phrase de passe.

Lors de l'utilisation de Borg avec sudo, il est nécessaire de passer le paramètre -i à sudo, pour que les données (cache, clé de chiffrement, etc.) soit écrites et lues dans le répertoire /root.

Opérations de base

Initialisation

L'initialisation d'un dépôt de sauvegarde Borg se fait à l'aide de la commande **borg init**. Le paramètre chemin passé en paramètre peut être un emplacement local (absolu ou relatif), ou un emplacement distant en utilisant la syntaxe <user>@<host>:<path>. Dans le cas d'un dépôt distant, l'accès se fait par SSH, et nécessite que Borg soit installé sur la machine distante.

```
$ borg init --encryption {none,keyfile,repokey} /home/babar/borg_backup
```

Le paramètre –encryption (abrégé -e) permet de définir la manière dont la sauvegarde est chiffrée ::

- **none**: La sauvegarde n'est pas chiffrée.
- **repokey (par défaut)**: La clé de chiffrement est écrite dans le fichier de configuration du dépôt de sauvegarde (le fichier /home/babar/borg_backup/config). Si un attaquant récupère le repository, il a la clé mais pas le mot de passe, il ne peut pas déchiffrer les sauvegardes.
- **keyfile**: La clé de chiffrement est écrite dans un fichier séparé dans le home (dans /home/babar/.config/borg/keys). Si un attaquant récupère le repository, il n'a ni la clé ni le mot de passe, il ne peut pas déchiffrer les sauvegardes.

Il est vital de sauvegarder le fichier /home/babar/borg_backup/config dans le mode repokey et le dossier /home/babar/.config/borg/keys dans le mode keyfile car en cas de perte on ne pourra pas déchiffrer les sauvegardes.

Création d'une sauvegarde

borg create est la commande la plus importante car c'est elle qui effectue la sauvegarde. Évidemment la première sauvegarde est la plus longue.

```
 $ borg create -v --stats /home/babar/borg_backup::{hostname}_{now:%d.%m.%Y}
/home/babar/syncthing /etc /var/www
```

- **-v**: verbose
- **–stats**: Montrer les statistiques de la sauvegarde créée
- **/home/babar/borg_backup**: Le dossier contenant les sauvegardes
- **{hostname}_{now:%d.%m.%Y}** : Il s'agit du nom de la sauvegarde que nous allons créer.
- **/home/babar/syncthing /etc /var/www** : Liste des dossiers à sauvegarder

Le premier paramètre est le nom de la sauvegarde, de la forme <RepoPath::SaveName>, suivi des fichiers et répertoires à sauvegarder.

Le nom de la sauvegarde peut contenir des chaînes réservées, remplacées par une valeur spécifique,

dont :

- **{hostname}**: Le nom d'hôte court de la machine.
- **{fqdn}**: Le nom d'hôte complet de la machine.
- **{now}**: La date et l'heure locales.
- **{utcnow}**: La date et l'heure UTC.
- **{user}**: Le nom de l'utilisateur réalisant la sauvegarde.

Pour les valeurs **{now}** et **{utcnow}**, il est possible de définir le format, au moyen de cette syntaxe : {now:%Y%m%d-%H%M%S}. Les éléments de formattage utilisés sont ceux disponibles de Python.

Il est possible d'utiliser - comme nom de fichier pour sauvegarder les données envoyées à l'entrée standard de la commande Borg. Ces données sont enregistrées dans un fichier nommé stdin, placé à la racine de la sauvegarde.

Concernant les sauvegardes par script on peut utiliser export BORG_PASSPHRASE='monjolimotdepasse'. On peut également choisir le mode keyfile et renseigner un mot de passe vide.

```
#!/bin/bash

export BORG_PASSPHRASE='monjolimotdepasse'
REPOSITORY='/home/babar/borg_backup'

/usr/local/bin/borg create -v --stats $REPOSITORY::{hostname}_{now:%d.%m.%Y}
/home/babar/syncthing /etc /var/www
/usr/local/bin/borg prune -v --keep-within=10d --keep-weekly=4 --keep-
monthly=-1 $REPOSITORY
```

Parcourir les données sauvegardées

La commande **borg list** permet de lister le contenu d'un repository ou d'une sauvegarde.

Exemples :

```
$ # Lister les sauvegardes disponibles
$ borg list <RepoPath>
$ # Lister le contenu d'une sauvegarde
$ borg list <RepoPath>::<SaveName>
```

La commande **borg info** permet d'afficher quelques informations basiques sur une sauvegarde, comme les dates de début et de fin, la commande utilisée pour générer cette sauvegarde, la taille des données sauvegardées, ou encore l'espace disque total occupé :

```
$ borg info <RepoPath>::<SaveName>
```

Pour parcourir le contenu des sauvegardes de manière plus naturelle, la commande **borg mount**

permet de monter le dépôt de sauvegarde dans un répertoire en utilisant fuse (Filesystem in USErspace). Le répertoire racine du point de montage contiendra alors une liste de répertoires, correspondant aux différentes sauvegardes disponibles, qu'il est possible d'explorer avec vos outils habituels, comme n'importe quel répertoire. Cette méthode est particulièrement pratique pour récupérer un fichier, voire une information précise dans un fichier, sans avoir à restaurer la totalité de la sauvegarde.

```
$ # Monter le dépôt de sauvegarde dans un répertoire
$ borg mount <RepoPath> <MountPointPath>
$ # Pour une sauvegarde réalisée en root, il est intéressant de permettre
aux utilisateurs de la parcourir
$ sudo -i borg mount -o allow_other <RepoPath> <MountPointPath>
```

Restauration d'une sauvegarde

La restauration d'une sauvegarde peut être effectuée au moyen de la commande **borg extract**. Cette commande prend en paramètre le nom de la sauvegarde à restaurer, ainsi qu'une liste optionelle de chemins à restaurer depuis cette sauvegarde. Si aucun chemin n'est donné, la totalité de la sauvegarde est restaurée.

Les fichiers et répertoires restaurés sont écrits dans le répertoire courant. Il est possible de restaurer des données sur la sortie standard, au moyen du paramètre –stdout. Cela est utile pour les données sauvegardées depuis l'entrée standard, mais permet aussi de transférer directement un fichier provenant d'une sauvegarde complète à un programme.

Exemples :

```
$ # Restaurer une sauvegarde complète
$ borg extract <RepoPath>::<SaveName>
$ # Restaurer une partie de la sauvegarde
$ borg extract <RepoPath>::<SaveName> <Filename> <Path>/<Directory>
$ # Restaurer une base de données PostgreSQL
$ borg extract --stdout <RepoPath>::<SaveName> stdin | pg_restore -d
<DbName>
$ # Lister le contenu d'une archive tar présente dans la sauvegarde
$ borg extract --stdout <RepoPath>::<SaveName> <FileName>.tgz | tar zt
```

Contrôler l'intégrité

La commande **borg check** permet de vérifier l'intégrité des données sauvegardées. Elle peut travailler sur le dépôt complet, ou sur une ou plusieurs sauvegardes.

Le paramètre **–prefix (abrégé -P)** détermine le préfixe du nom des sauvegardes à vérifier. Par défaut, toutes sont vérifiées.

Exemples :

```
$ # Vérifier le dépôt complet
$ borg check <RepoPath>
```

```
$ # Vérifier une unique sauvegarde
$ borg check <RepoPath>::<SaveName>
$ # Vérifier toutes les branches de la machine desktop
$ borg check --prefix=desktop <RepoPath>
```

Il existe aussi un paramètre **–repair**, permettant de tenter la réparation des données corrompues détectées, mais son utilisation est actuellement déconseillée, puisque cette fonctionnalité est encore expérimentale.

Libérer de l'espace disque

Lorsque le dépôt de sauvegarde devient trop gros, il peut être nécessaire de réduire l'espace disque utilisé.

Borg propose deux commandes pour cela :

- **borg delete**: Supprime le dépôt complet, ou une sauvegarde précise.
- **borg prune**: Supprime des sauvegardes selon des critères.

borg delete

borg delete permet de supprimer une sauvegarde ou un repository entier.

```
$ borg delete /home/babar/borg_backup # Va supprimer le repository et donc
toutes les sauvegardes contenues dedans
$ borg delete /home/babar/borg_backup::host_19.08.2016 # Va supprimer la
sauvegarde du 19/08
```

borg prune

borg prune s’occupe de la rotation c’est-à-dire de respecter les règles de conservation/suppression des sauvegardes.

Les paramètres possibles sont :

- **–prefix (abrégé -P)**: Définit le préfixe des sauvegardes à prendre en compte (toutes par défaut).
- **–keep-within**: Définit une période dans laquelle toutes les sauvegardes doivent être conservées.
- **–keep-hourly (abrégé -H)**: Conserver la dernière sauvegarde des X dernières heures.
- **–keep-daily (abrégé -d)**: Conserver la dernière sauvegarde des X derniers jours.
- **–keep-weekly (abrégé -w)**: Conserver la dernière sauvegarde des X dernières semaines.
- **–keep-monthly (abrégé -m)**: Conserver la dernière sauvegarde des X derniers mois.
- **–keep-yearly (abrégé -y)**: Conserver la dernière sauvegarde des X dernières années.
- **–list**: Associé à –verbose, permet d'afficher la liste des sauvegardes conservées et supprimées.

- **–dry-run (abrégé -n)**: Permet de tester ce qui serait fait, sans supprimer réellement les données, avec –list.

Exemple de commande :

```
$ borg prune -v --keep-within=10d --keep-weekly=4 --keep-monthly=-1
/home/babar/borg_backup
```

- **–keep-within=10d**: Conserver toutes les sauvegardes effectuées durant le temps indiqué (ici 10 jours)
- **–keep-weekly=4**: Nombre de sauvegardes hebdomadaires à conserver
- **–keep-monthly=-1**: Nombre de sauvegardes mensuelles à conserver. On note le **-**, qui spécifie un nombre négatif de sauvegardes signifiant qu'on conserve une sauvegarde indéfiniment ici la mensuelle
- **/home/babar/borg_backup** : Le dossier contenant les sauvegardes

Débloquer un dépôt verrouillé

Lorsqu'il travaille sur le dépôt, Borg le verrouille pour empêcher qu'une autre opération ne se lance en parallèle dessus. Tenter de lancer la création d'une sauvegarde, la restauration, ou simplement de lister les sauvegardes disponible renverra alors une erreur, indiquant que le dépôt de sauvegarde est verouillé.

Cependant, il peut arriver que le dépôt reste bloqué, suite à une coupure de la connexion réseau, ou un crash de Borg ou du système, par exemple. Dans ce cas, il est nécessaire de forcer le déverrouillage, au moyen de la commande **borg break-lock**.

Le dépôt est verrouillé lors de toutes les opérations. Par exemple, monter le dépôt dans un répertoire avec la commande borg mount verrouille aussi le dépôt. Vérifiez donc bien qu'aucune opération ne soit en cours avant de forcer le déverrouillage du dépôt, pour éviter toute corruption de données.

Sauvegarder et restaurer la clé de chiffrement

Sauvegarder la clé de chiffrement est fortement conseillé, surtout lorsqu'elle est conservée dans un fichier séparé du dépôt de sauvegarde. En effet, sans cette clé, les sauvegardes sont inutilisables.

Lors de l'utilisation du mode repokey, la clé étant sauvegardée dans le fichier de configuration du dépôt, il n'est pas vraiment nécessaire de la sauvegarder. Cependant, dans le cas où ce fichier serait endommagé, il peut être utile de le faire.

Borg propose plusieurs formats pour sauvegarder la clé de chiffrement, avec la commande **borg key export** :

- **Par défaut**: Enregistrement de la clé directement dans un fichier.
- **–paper**: Génération d'une série de valeurs, destinées à être imprimées, ou écrites à la main sur

un papier.
- **–qr-html**: Génération d'un QR Code contenant la clé de chiffrement, destiné à être imprimé.

Les deux formats destinés à être imprimés permettent de sauvegarder la clé de chiffrement hors de tout système informatique. Dans le cas du QR Code, le scanner avec un lecteur compatible permet de retrouver le contenu original de la clé. Dans le cas de la série de valeurs générée, elle est faite de manière à pouvoir vérifier, ligne par ligne, s'il y a eu des fautes de frappe, et ainsi simplifier la recherche d'une erreur dans ce cas.

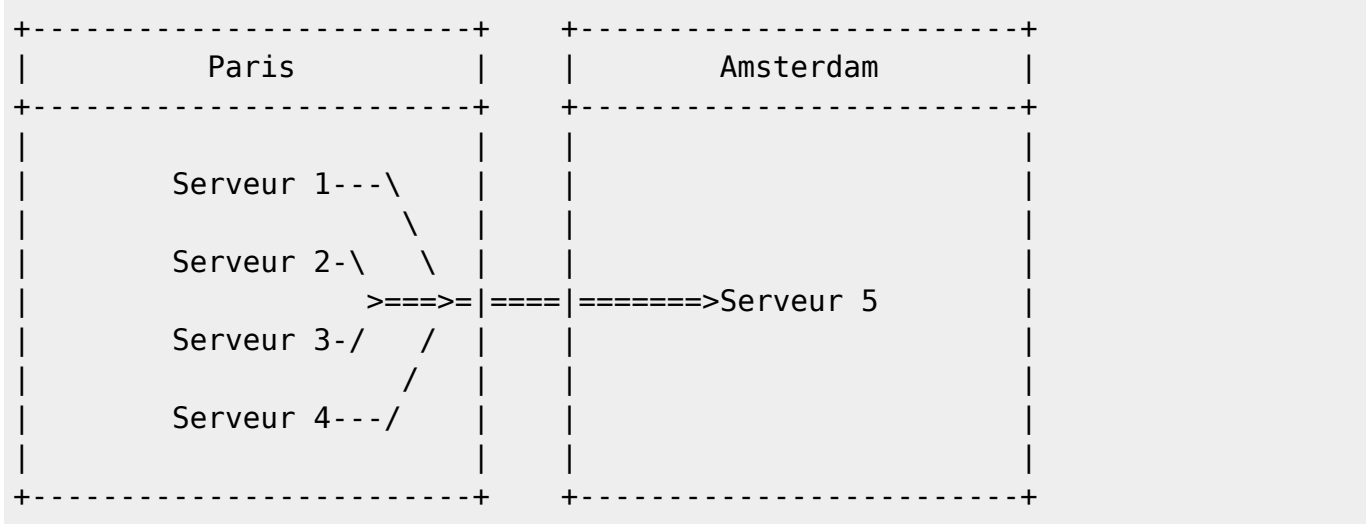
L'import d'une clé est réalisé grâce à la commande **borg key import**. De la même manière que pour l'export, le paramètre **–paper** active le mode papier, permettant de saisir les séries de valeurs, ligne par ligne, en vérifiant la somme de contrôle de chaque ligne. Le mode par défaut demande, en plus du chemin du dépôt de sauvegarde, un paramètre désignant le fichier contenant la clé de chiffrement à importer.

Exemple de serveur de sauvegardes

Configuration du serveur de stockage des sauvegardes

Contexte

On a quatre serveurs persos hébergés dans un datacenter de Paris et pour la sécurité des données, on a un cinquième serveur, situé dans le datacenter d'Amsterdam (au cas où Paris prend feu) pour servir de serveur de sauvegarde.



Création des dépôts

Sur Serveur 5 créer un utilisateur dédié à recevoir les sauvegardes, avec les dossiers qui vont être nécessaires pour la suite :

```
useradd -d /home/backup -m -r -U backup
```

```
su - backup
mkdir /home/backup/.ssh
mkdir /home/backup/repos/srv{1,2,3,4}
```

Configuration de clients

Configuration des accés ssh

Chaque client (les serveurs à sauvegarder) dispose de son dépôt dédié dans le dossier /home/backup/repos, par exemple /home/backup/repos/srv1 pour Serveur 1 et ainsi de suite.

Sur le client Serveur 1, en root, on va créer une clé SSH qui servira à l'authentification sur serveur Serveur 5 :

```
ssh-keygen -t ed25519
```

Valider en appuyant sur la touche « Entrée » à toutes les questions, à moins de savoir exactement ce q'on fait.

On doit maintenant avoir une clé privée contenue dans le fichier /root/.ssh/id*ed25519, et une clé publique contenue dans le fichier /root/.ssh/id*ed25519.pub. La clé privée est à garder en sécurité et à ne jamais partager.

Sur le serveur Serveur 5, il va falloir ajouter dans le fichier /home/backup/.ssh/authorized_keys la clé publique de chacun des clients avec l'option command, sur le modèle suivant :

```
command="cd /home/backup/repos/<SRV#>; borg serve --restrict-to-path
/home/backup/repos/<SRV#>",no-port-forwarding,no-x11-forwarding,no-agent-
forwarding,no-pty,no-user-rc <CLÉ_PUBLIQUE_DE_SRV#>
```

Exemple pour Serveur 1 :

```
command="cd /home/backup/repos/srv1; borg serve --restrict-to-path
/home/backup/repos/srv1",no-port-forwarding,no-x11-forwarding,no-agent-
forwarding,no-pty,no-user-rc ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIG+n17i8h2PF1/OTv62Ax/mOYhpWpDNDHXukGaK3Noaf
root@srv1
```

L'option **command** permet de restreindre l'utilisation de la connexion SSH à une commande donnée, pour le client qui utilisera la clé publique associée.

Configuration des tâches de backup

Ceci étant fait pour tous les clients, il reste à configurer et planifier la sauvegarde sur ceux-ci.

Sur chaque client, faudra commencer par initialiser son dépôt avec la commande suivante :

```
borg init backup@<ADRESSE_DE_SRV5>:repo
```

La phrase de passe est à conserver soigneusement, c'est elle qui chiffre et déchiffre les données à sauvegarder. Si vous la perdez vos sauvegardes seront inexploitables.

Pour faciliter l'exécution de la sauvegarde on va faire un script, à placer par exemple dans /usr/local/bin/borg-backup :

```
#!/bin/bash

REPO='backup@<ADRESSE_DE_SRV5>:repo'

LIST=(
  /etc
  /root
  /usr/local
  /var/backups/mariadb
  /var/www
)

export BORG_PASSPHRASE='<VOTRE_PHRASE_DE_PASSE>'

function main() {
  # backup
  borg create --compression lz4 \
    ${REPO}::'{now:%Y-%m-%d}' \
    "${LIST[@]}"

  # rotation
  borg prune ${REPO} \
    --keep-daily=7 --keep-weekly=4 --keep-monthly=6
 }

if [[ "${BASH_SOURCE}" == "${0}" ]]; then
  main "${@}"
fi
```

Avec ce script, chacun des fichiers et dossiers présents dans la variable LIST seront sauvegardés.

On oublie pas de rendre le script exécutable :

```
chmod +x /usr/local/bin/borg-backup
```

Il ne reste plus qu'à planifier l'exécution du script toutes les nuits via une tâche cron. Pour ceci on ajoute la ligne suivante dans le fichier /etc/crontab :

```
30 3 * * *    root    /usr/local/bin/borg-backup
```

Et voilà, les serveurs sont sauvegardés ! Du moins à la prochaine exécution de la crontab…