

# Systeme

- Ajouter une clé GPG de dépôt Debian sans apt-key
- Borg
- Borgmatic
- Crypsetup
- Curl
- Débloquer un RAID coincé en resync=PENDING
- Empêcher l'activation d'un service à son installation
- Exécuter une action à la mise en veille / au réveil
- Firewalld : un firewall simple à utiliser
- Lancer des commandes sudo avec authentification par agent SSH
- LVM
- Monter une ou des partitions contenues dans un fichier qcow2
- Réinstaller les modules Perl installés avec la version précédente de Perl
- Salt
- Sed
- Tmux
- Trucs et astuces
- Supprimer une clé SSH stockée par gpg-agent
- Swap
- Systemd

# Ajouter une clé GPG de dépôt Debian sans apt-key

L'ajout de clé GPG de dépôt Debian avec `apt-key` est une méthode dépréciée. On ajoute maintenant la clé (au bon format) dans le dossier `"/etc/apt/trusted.gpg.d"` :

```
curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key |  
gpg --dearmor |  
sudo tee /etc/apt/trusted.gpg.d/deb.nodesource.com.gpg >/dev/null
```

# Borg

<https://borgbackup.readthedocs.io/>

## Quelques mots sur Borg

- il est **très** simple d'usage ;
- les données sont déduplicuées ;
- les sauvegardes peuvent être compressées ;
- les sauvegardes peuvent être effectuées en local ou à distance ;
- les sauvegardes peuvent être montées (et donc utilisées) comme un système de fichiers classiques.

Pour se simplifier la vie, on peut utiliser Borgmatic qui est un *wrapper* de Borg.

## Installation

```
apt-get install borgbackup
```

On peut aussi utiliser `pip`, c'est vous qui voyez.

## Initialisation

On doit d'abord initialiser le répertoire qui accueillera les sauvegardes :

```
mkdir /opt/backup  
borg init /opt/backup/
```

Au cas où l'on souhaiterait utiliser un serveur distant pour accueillir les sauvegardes, il faut que celui-ci soit joignable par SSH et dispose lui-aussi de Borg.

```
borg init <username>@<server>:/remotepath
```

Dans le cas où il n'est pas possible d'y installer Borg (un espace de stockage fourni par un hébergeur par exemple), il est possible de le monter en *sshfs* :

```
apt-get install sshfs  
sshfs <username>@<server>:/remotepath /opt/backup/
```

Sshfs possède de multiples options améliorant les performances et la stabilité du point de montage, je vous laisse chercher car je ne les ai pas sous la main.

Par défaut, l'initialisation vous demandera un mot de passe qui servira à chiffrer les sauvegardes. Il est possible de désactiver le chiffrement via les options de `borg init`.

```
borg help init
```

Le chiffrement nécessite deux choses pour fonctionner : le mot de passe et une clé de chiffrement. Dans le cas d'une initialisation par défaut (sans options), la clé de chiffrement sera dans le fichier `/opt/backup/config`.

**Il est impératif de sauvegarder cette clé sans laquelle vos sauvegardes seraient inutiles** ! On peut l'exporter via `borg key export /opt/backup` et l'importer via `borg key import /opt/backup keyfile`.

Le mot de passe vous sera demandé pour toute opération sur les sauvegardes (création, restauration, etc). Pour éviter de le taper toutes les cinq minutes, vous pouvez exporter la variable d'environnement `BORG_PASSPHRASE` :

```
export BORG_PASSPHRASE="mot_de_passe"
```

N'oubliez pas que cet `export` se retrouvera dans votre historique bash (ou zsh, ou ce que vous utilisez) ! Je vous conseille de le supprimer de votre historique après usage.

# Création d'une sauvegarde

Rien de plus simple :

```
borg create /opt/backup::nom_de_la_sauvegarde /chemin/a/sauvegarder
```

L'option `--stats` est très appréciable car elle fournit des informations sur la sauvegarde créée (comme sa taille par exemple).

Le nom de la sauvegarde doit être unique et ne doit pas se terminer par `.checkpoint`. On peut utiliser des *placeholders* comme `{hostname}` dans le nom de la sauvegarde. Voir `borg help placeholders` pour plus de détails.

# Manipulation des sauvegardes

## Lister les sauvegardes

```
borg list /opt/backup
```

## Supprimer une sauvegarde

```
borg delete /opt/backup::
```

## Renommer une sauvegarde

```
borg rename /opt/backup::
```

## Extraire le contenu d'une sauvegarde

Attention ! Le contenu sera extrait dans le répertoire où vous vous trouvez !

```
borg extract /opt/backup::
```

Il est possible de n'extraire que certains fichiers :

```
borg extract /opt/backup::borg extract /opt/backup::
```

## Monter une sauvegarde

```
borg mount /opt/backup::
```

Vous pourrez alors parcourir et utiliser (`cp`, `cat` ...) la sauvegarde en parcourant le dossier `/mnt`.

Pour démonter la sauvegarde :

```
borg umount /mnt
```

# Voir les informations détaillées d'une sauvegarde

```
borg info /opt/backup::
```

## Supprimer les vieilles sauvegardes

Même si l'espace disque ne coûte aujourd'hui pas très cher, on ne va quand même pas garder 30 ans de sauvegardes

```
borg prune -w 4 --prefix='{hostname}-' /opt/backup
```

Cette commande ne gardera que 4 sauvegardes hebdomadaires. L'option `--prefix='{hostname}-'` permet de discriminer les sauvegardes à éliminer d'après un préfixe (ici le nom de la machine), ceci afin d'éviter de supprimer les sauvegardes d'une autre machine si jamais le répertoire de sauvegarde servait pour plusieurs machines.

Je vous laisse regarder les autres options de `borg prune` dans le manuel.

## Vérifier une sauvegarde (ou toutes)

```
borg check /opt/backup::
borg check /opt/backup
```

Attention ! Cela vérifie que la sauvegarde n'est pas corrompue, pas que vous avez ciblé les bons répertoires à sauvegarder ! :P

## Vérification des sauvegardes

“ « Une sauvegarde est à la fois valide et corrompue tant qu'on n'a pas essayé de la restaurer »

*La sauvegarde de Schrödinger*

Et oui, tant qu'on n'a pas essayé de restaurer des fichiers depuis la sauvegarde, comment savoir si celle-ci a fonctionné ? Les déboires de la société Gitlab qui a perdu 6 heures d'enregistrements en base de données pour cause de dysfonctionnement des 5(!) méthodes de sauvegardes nous en donnent la preuve.

Il existe cependant un outil qui permet de vérifier que vos sauvegardes respectent un certain nombre de critères : Backup Checker

Avec cet outil, on pourra s'assurer que la sauvegarde comporte bien tel ou tel fichier, soit d'une certaine taille, etc.

À défaut d'avoir le temps de mettre en place un tel outil, on pourra ponctuellement essayer de restaurer localement un fichier. Même si cela ne vérifie pas grand chose, c'est toujours mieux que pas de vérification du tout !

## Un peu de lecture

<http://sebsauvage.net/wiki/doku.php?id=borgbackup>

## Plutôt que d'écrire un script qui utilise borg...

Il y a un soft qui fait une surcouche à Borg, le rendant plus simple à utiliser : Borgmatic.

# Borgmatic

Borgmatic est un *wrapper* autour de Borg qui en simplifie infiniment l'utilisation.

## Installation

Il nous faut d'abord Borg (utilisez la version des backports Debian si vous êtes encore en *stretch*) :

```
apt install borgbackup
```

D'habitude, je préfère utiliser les paquets Debian, mais la version pip de Borgmatic apporte une option en plus que j'apprécie particulièrement.

```
apt install python3-pip
pip3 install borgmatic
```

## Configuration

C'est là que Borgmatic est pratique : il permet une configuration très simple de Borg.

Générez un fichier de configuration :

```
generate-borgmatic-config
```

Cela créera le fichier `/etc/borgmatic/config.yaml`. Si vous souhaitez que la configuration soit dans un autre fichier :

```
generate-borgmatic-config -d /etc/borgmatic/autre_config.yaml
```

La configuration générée (exemple) est très simple à comprendre et auto-documentée, je ne vais l'expliquer, je vais me contenter d'en mettre certains points en valeur

## Le dépôt



On le verra plus bas, j'utilise un serveur distant pour faire les sauvegardes. Donc :

```
location:  
  repositories:  
    - borg@server:/var/lib/borg/depot/
```

## La *passphrase*

Choisissez quelque chose de costaud et sauvegardez-là quelque part !

```
storage:  
  encryption_passphrase: "foo-bar-baz"
```

## Utilisation d'une clé SSH particulière

Je crée plus bas une clé SSH dédiée pour Borg pour faire les sauvegardes sur le serveur distant. Il faut donc que j'indique à Borgmatic que je souhaite utiliser cette clé.

```
storage:  
  ssh_command: ssh -i /root/.ssh/id_borgmatic
```

## Les hooks

Situés à la fin du fichier de configuration, il s'agit d'actions qui seront effectuées avant et après la sauvegarde, ou en cas de problème lors de l'exécution.

Personnellement, j'aime bien avoir la liste de mes sauvegardes après qu'une sauvegarde soit effectuée. Je vais donc écrire :

```
hooks:  
  after_backup:  
    - /usr/local/bin/borgmatic list
```

Comme il est conseillé d'exporter la clé du dépôt borg, je mets aussi ceci dans les `hooks` :

```
before_backup:  
  - borg key export borg@server:/var/lib/borg/depot/ /etc/ssl/private/borg.key
```

# Rétention et vérification du dépôt et des archives

Je fais ça sur le serveur, j'y reviendrai plus loin.

## Cron

On crée un petit cron (avec l'utilisateur `root`) pour sauvegarder régulièrement l'ordinateur :

```
35 0 * * * borgmatic create -c /etc/borgmatic/mon_pc.yaml --stats 2>&1
```

Si ce n'est pas un serveur allumé 24h/24, il est préférable de mettre ça dans `/etc/anacrontab` :

```
@daily 15    backup borgmatic create -c /etc/borgmatic/mon_pc.yaml --stats 2>&1
```

Cela lancera le backup tous les jours, 15 minutes après le démarrage du pc. Ou plus tard.

Attention, sur Debian, les tâches anacron ne sont lancées que si vous êtes sur secteur ! Pour changer ça, reportez-vous au fichier `/usr/share/doc/anacron/README.Debian`.

## Préparation du serveur du dépôt distant

Je préfère faire mes sauvegardes sur un disque distant : si mon disque lâche, j'aurai toujours mes sauvegardes.

Pour ce faire, je crée une clé dédiée sur l'ordinateur à sauvegarder, sans mot de passe vu que borgmatic va tourner automatiquement :

```
ssh-keygen -o -a 100 -t ed25519 -f /root/.ssh/id_borgmatic -N ''
```

**Sur le serveur de sauvegarde**, j'installe Borg et Borgmatic comme précédemment puis je crée un utilisateur `borg` :

```
adduser --system --home /var/lib/borg --shell /bin/bash --disabled-password borg
mkdir /var/lib/borg/.ssh
chmod 700 /var/lib/borg/.ssh
```

```
touch /var/lib/borg/.ssh/authorized_keys
chmod 600 /var/lib/borg/.ssh/authorized_keys
chown -R borg: /var/lib/borg/.ssh
```

Et je mets la clé publique créée précédemment dans `/var/lib/borg/.ssh/authorized_keys`, avec quelques restrictions :

```
command="/usr/bin/borg --umask=077 --info serve --append-only --restrict-to-repository
/var/lib/borg/becky2/",restrict ssh-ed25519
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX root@mon_pc
```

On génère une configuration :

```
generate-borgmatic-config -d /etc/borgmatic/mon_pc.yaml
```

Et c'est dans cette configuration qu'on va configurer les durées de rétention et les vérifications du dépôt et des archives.

## Rétention

C'est très simple. Pour garder un backup par mois sur les 6 derniers mois, un par semaine sur les 8 dernières semaines et un par jour sur les 14 derniers jours, il suffit de :

```
retention:
  keep_daily: 14
  keep_weekly: 8
  keep_monthly: 6
```

Comme Borg déduplique comme un monstre, cela ne prendra pas énormément de place (sauf si vous sauvegardez des trucs qui changent tout le temps)

## Vérification du dépôt et des archives

Cela permet de s'assurer que votre dépôt et vos archives sont utilisables et non corrompus. Je me contente de décommenter la configuration proposée :

```
consistency:
  checks:
    - repository
    - archives
  check_last: 3
```

J'ai laissé `check_last` à 3 car vérifier toutes les archives peut être fort long. Donc une vérification des 3 dernières devrait faire l'affaire, surtout si on vérifie quotidiennement.

# Dépôt

Attention, comme on est là sur le serveur distant, il faut lui donner un dossier local !

```
location:  
  repositories:  
    - /var/lib/borg/depot/
```

## Script de suppression d'anciennes sauvegardes / vérification des archives

Il y a des éléments dans ce script qui prendront tout leur sens plus tard.

**NOTA BENE** : ce script est tout à fait adaptable à un usage à Borg sans Borgmatic.

On prépare le terrain :

```
apt install libcpupanel-json-xs-perl  
mkdir -P /opt/borgmatic-stats/tmp/ /opt/borgmatic-stats/json/  
cd /opt/borgmatic-stats/  
wget https://framagit.org/snippets/3716/raw -O verify-archives.pl  
chmod 700 -R /opt/borgmatic-stats/  
chown borg: -R /opt/borgmatic-stats/
```

Puis on met ceci dans `/opt/borgmatic_prune_and_check.sh` :

```
#!/bin/bash  
cat <<EOF  
  
=====
```

== mon\_pc

```
=====
```

EOF

  

```
/usr/local/bin/borgmatic list -c /etc/borgmatic/mon_pc.yaml --json > /opt/borgmatic-stats/tmp/mon_pc.json.tmp  
CONTINUE=0  
if [[ -e /opt/borgmatic-stats/json/mon_pc.json ]]
```

```

then
    echo "Checking repository consistency."
    CONTINUE=$(/opt/borgmatic-stats/verify-archives.pl --old /opt/borgmatic-stats/json/mon_pc.json --new
/opt/borgmatic-stats/tmp/mon_pc.json.tmp)
    echo "Repository consistency checked."
else
    CONTINUE=1
fi

if [[ $CONTINUE == '1' ]]
then
    ## Check
    /usr/local/bin/borgmatic check -c /etc/borgmatic/mon_pc.yaml 2>&1 && \
    echo "Repository checked." && \
    ## Allow pruning
    borg config /var/lib/borg/depot/ append_only 0 && \
    ## Prune
    /usr/local/bin/borgmatic prune -c /etc/borgmatic/mon_pc.yaml --stats 2>&1 && \
    echo "Repository pruned."
    ## List
    echo "Borg archives of mon_pc:"
    /usr/local/bin/borgmatic list -c /etc/borgmatic/mon_pc.yaml 2>&1

    /usr/local/bin/borgmatic list -c /etc/borgmatic/mon_pc.yaml --json > /opt/borgmatic-stats/json/mon_pc.json

    ## Disallow pruning
    borg config /var/lib/borg/depot/ append_only 1

    echo "
else
    cat <<EOF

*****
* ALERT ON mon_pc! *
*****

All of the old archives can't be retrieved from the repository (/var/lib/borg/depot/)!

Someone may have deleted an archive from mon_pc.

Pruning has been aborted. Please manually review the repository.

```

```
$CONTINUE  
EOF  
fi
```

On n'oublie pas de le rendre exécutable :

```
chmod +x /opt/borgmatic_prune_and_check.sh
```

## Cron

Attention ! Il faut éditer la *crontab* de l'utilisateur `borg` :

```
crontab -e -u borg
```

Et son contenu :

```
15 3 * * * /opt/borgmatic_prune_and_check.sh
```

# Initialisation du dépôt de sauvegarde

**Sur l'ordinateur à sauvegarder :**

```
borgmatic init -c /etc/borgmatic/mon_pc.yaml --append-only -e repokey
```

Le dépôt est initialisé en **append-only** (mais de toute façon, on le force dans le `.ssh/authorized_keys` du serveur distant) et avec un chiffrement par mot de passe mais avec la clé dans le dossier du dépôt distant.

# Lancement d'une sauvegarde

**Sur l'ordinateur à sauvegarder :**

```
borgmatic create -c /etc/borgmatic/mon_pc.yaml --stats
```

Le `--stats` est l'option que j'affectionne et dont je parlais au début : cela affiche des statistiques sur la sauvegarde qui a été faite : sa taille, sa taille compressée, sa taille dédupliquée, le temps que ça a pris...

# Afficher la liste des sauvegardes

```
borgmatic list -c /etc/borgmatic/mon_pc.yaml
```

# Restauration de sauvegardes

Voir la section [kivabien](#) de l'article sur [Borg](#).

## Explication de l'option `--append-only`

Contrairement à ce qu'on pourrait croire, `--append-only` n'empêche pas de supprimer des sauvegardes ([Ce ticket Github](#) m'en a fait prendre conscience). Ainsi, un attaquant ayant pris le contrôle de l'ordinateur à sauvegarder pourra supprimer des sauvegardes. Sauf qu'elles ne seront pas vraiment supprimées : l'attaquant aura créé des transactions qui indiquent la suppression des sauvegardes mais ces transactions ne seront réellement appliquées que lors d'une action comme `prune` depuis un ordinateur qui aura le droit de faire sauter le **append-only**. C'est comme un `BEGIN TRANSACTION;` sans `COMMIT;` en SQL ☐☐

C'est le cas du script `/opt/borgmatic_prune_and_check.sh` qui fait effectivement sauter (et le remet après) le verrou avec :

```
borg config /var/lib/borg/depot/ append_only 0
```

Comme on automatise la suppression des anciennes sauvegardes avec `cron`, il nous faut un moyen de repérer de tels changements. C'est tout le but de `/opt/borgmatic-stats/verify-archives.pl` qui compare la liste des archives du dépôt avec la liste créée la dernière fois que `/opt/borgmatic_prune_and_check.sh` a supprimé les anciennes archives. Comme le serveur distant est le seul à normalement pouvoir supprimer des archives, il est logique de retrouver toutes les anciennes archives au lancement suivant.

Ainsi, en cas d'incohérence, la suppression des anciennes archives ne s'effectue pas.

**NOTA BENE** : ceci n'empêchera pas un attaquant de changer la configuration de borgmatic pour lui faire sauvegarder des trucs inutiles plutôt que ce qui vous intéresse. Ou de couper les sauvegardes. C'est un inconvénient des sauvegardes en *push* plutôt qu'en *pull*. Je n'ai pas encore trouvé de solution à ça. Peut-être un script qui analyserait la sortie de `borgmatic create --stats --json` qui me permettrait de repérer des changements importants dans la taille de la sauvegarde, le temps de sauvegarde ou le ratio de déduplication ?

# En cas d'incohérence : restaurer des sauvegardes supprimées par un attaquant

Sur le serveur distant :

```
su - borg  
cd /var/lib/borg/depot/
```

Regarder le log des transactions pour trouver les n° des transactions suspectes : `cat transactions`

Ce qui nous donne un truc du genre :

```
transaction 23, UTC time 2019-08-01T13:40:34.435019  
transaction 25, UTC time 2019-08-01T13:42:05.662188  
transaction 27, UTC time 2019-08-01T13:43:18.403771  
transaction 29, UTC time 2019-08-01T13:43:53.306636  
transaction 31, UTC time 2019-08-01T13:44:51.831937  
transaction 33, UTC time 2019-08-01T13:45:17.786886  
transaction 35, UTC time 2019-08-01T22:17:12.044179  
transaction 37, UTC time 2019-08-02T08:02:55.005430  
transaction 43, UTC time 2019-08-02T22:17:10.068843  
transaction 45, UTC time 2019-08-03T22:17:09.625745  
transaction 47, UTC time 2019-08-04T22:17:09.673447  
transaction 49, UTC time 2019-08-05T22:17:13.709208  
transaction 51, UTC time 2019-08-06T10:11:26.301496  
transaction 53, UTC time 2019-08-06T10:20:46.178014  
transaction 55, UTC time 2019-08-06T10:26:47.940003
```



Là, il faut savoir quand a eu lieu le problème. Ici, c'était le 6 août à partir de 10h. Donc les transactions 51 à 55. Mais **attention** ! Il s'agit en fait des transactions **50** à 55 : le numéro indiqué dans le fichier est celui du dernier fichier modifié par la transaction. Il faut donc partir du n+1 de la dernière transaction correcte.

Après, c'est facile :

```
rm hints.* index.* integrity.*  
rm data/**/{50..55}
```

On finit par rafraîchir le cache du dépôt :

```
borg delete --cache-only /var/lib/borg/depot/
```

Sur l'ordinateur sauvegardé, par contre, je n'arrivais pas à supprimer le cache, j'avais ce message :

```
Cache, or information obtained from the security directory is newer than repository - this is either an attack or  
unsafe (multiple repos with same ID)
```

Ceci m'a réglé le problème :

```
rm -rf ~/.cache/borg/le_dossier_avec_un_nom_monstrueux  
rm -rf ~/.config/borg/security/un_autre_dossier_avec_un_nom_monstrueux
```

# Cryptsetup

Tiré de <https://www.thegeekstuff.com/2016/03/cryptsetup-lukskey/>.

## Identifier la partition chiffrée

```
sudo lsblk -o name,size,fstype,label,mountpoint
```

Chez moi, ça donne ça :

NAME	SIZE	FSTYPE	LABEL	MOUNTPOINT
sda	238.5G			
└─sda1	512M	vfat		/boot/efi
└─sda2	244M	ext2		/boot
└─sda3	237.7G	crypto_LUKS		
└─sda3_crypt	237.7G	LVM2_member		
└─foo--vg-swap_1	6.8G	swap		[SWAP]
└─foo--vg-root	27.9G	ext4		/
└─foo--vg-home	203G	ext4		/home

La partition chiffrée est donc `/dev/sda3`.

## Identifier les slots de clés déjà utilisés

On peut avoir 8 clés de chiffrement, chacune occupant un *slot*.

```
sudo cryptsetup luksDump /dev/sda3 | grep Slot
```

Chez moi, ça donne :

Key Slot 0: ENABLED  
Key Slot 1: DISABLED  
Key Slot 2: DISABLED  
Key Slot 3: DISABLED  
Key Slot 4: DISABLED  
Key Slot 5: DISABLED  
Key Slot 6: DISABLED  
Key Slot 7: DISABLED

## Ajouter une nouvelle clé

```
sudo cryptsetup luksAddKey /dev/sda3
```

Cryptsetup vous demandera alors de taper la *passphrase* d'une des clés existantes puis de taper (deux fois) une nouvelle *passphrase* pour la nouvelle clé.

On peut forcer le slot pour la nouvelle clé :

```
sudo cryptsetup luksAddKey /dev/sda3 -S 4
```

## Supprimer une clé

```
sudo cryptsetup luksRemoveKey /dev/sda3
```

Cryptsetup vous demandera alors de taper la *passphrase* de la clé à supprimer. Je ne sais pas ce qui se passerait si la *passphrase* était utilisée pour plusieurs clés (je ne sais même pas si c'est possible).

Si on a oublié la *passphrase* de la clé à supprimer, on peut supprimer son slot :

```
cryptsetup luksKillSlot /dev/sda3 2
```

Cryptsteup vous demandera alors de taper la *passphrase* d'une des clés existantes.

# Curl

Afin d'éviter les écueils dus aux éventuels problèmes de redirection réseau des ports des machines virtuelles, nous allons utiliser la commande `curl` pour tester les sites web que nous mettrons en place au cours des TP.

Ceci constitue un petit inventaire des commandes les plus utiles de `curl` pour notre cas.

## Utilisation de base

```
curl http://example.org
```

La commande `curl` télécharge la ressource demandée (qui n'est pas nécessairement une adresse web, car `curl` est capable de télécharger des ressources d'autres protocoles, comme `ftp` par exemple) et en affiche le contenu sur la sortie standard, si ce contenu n'est pas un contenu binaire.

## Rediriger la ressource vers un fichier

```
curl http://example.org > fichier.html  
curl http://example.org --output fichier.html  
curl http://example.org -o fichier.html
```

## Réduire la verbosité de curl

Lorsque la sortie est redirigée vers un fichier, `curl` affiche une barre de progression donnant certaines indications sur le téléchargement de la ressource (le temps restant, la taille...).

Pour ne pas afficher ces informations, on utilise l'option `--silent` ou son abbréviation `-s`.

```
curl --silent http://example.org -o fichier.html  
curl -s http://example.org -o fichier.html
```

## Faire autre chose qu'un GET

Pour utiliser une autre méthode HTTP que `GET`, on utilise l'option `--request` ou son abbréviation `-X`.

```
curl --request POST http://example.org  
curl -X POST http://example.org
```

## Faire une requête HEAD

Si on tente de faire une requête `HEAD` avec l'option `--request`, `curl` affichera un message d'erreur :

```
Warning: Setting custom HTTP method to HEAD with -X/--request may not work the  
Warning: way you want. Consider using -I/--head instead.
```

Il convient d'utiliser l'option `--head` ou son abbréviation `-I` :

```
curl --head http://example.org  
curl -I http://example.org
```

## Pour faire une requête avec authentification HTTP

On spécifie l'identifiant et le mot de passe avec l'option `--user` ou son abbréviation `-u`, en les séparant par un caractère `:`.

```
curl --user login:password http://example.org  
curl -u login:password http://example.org
```

## Forcer la connexion en IPv6 ou en IPv4

On utilise pour cela les options `-6` et `-4`.

```
curl -6 http://example.org  
curl -4 http://example.org
```

## Utilisation avancée

# Forcer la connexion sur une autre adresse IP

Il est possible de dire à `curl` d'utiliser une adresse IP particulière au lieu de la véritable adresse IP d'un domaine. Il faut voir cela comme une alternative à la manipulation du fichier `/etc/hosts`.

```
curl --resolve example.org:80:127.0.0.1 http://example.org
curl --resolve "example.org:80:[::1]" http://example.org
```

La syntaxe de l'option est `host:port:addr`. Le port est celui qui sera utilisé par le protocole. Spécifier le port `80` pour le protocole `https` serait inutile : il faut dans ce cas utiliser le port `443`.

# Forcer la connexion sur une autre adresse IP et un autre port

On utilise pour cela l'option `--connect-to`, relativement similaire à l'option `--resolve`. La syntaxe de l'option est `host1:port1:host2:port2`

```
curl --connect-to example.org:80:127.0.0.1:8080 http://example.org
curl --connect-to "example.org:80:[::1]:8080" http://example.org
```

# Forcer la connexion depuis une certaine interface réseau

On utilise pour cela l'option `--interface` suivi du nom d'une interface, d'une adresse IP ou d'un nom d'hôte.

```
curl --interface wlo1 http://example.org
curl --interface 203.0.113.42 http://example.org
curl --interface example.com http://example.org
```

# Ne pas vérifier la sécurité du certificat du site

Que ce soit parce qu'un certificat est expiré ou parce qu'on utilise un certificat autosigné, on peut avoir besoin que `curl` effectue bien la requête sans se préoccuper de la validité du certificat utilisé par le site. On utilise alors l'option `--insecure` ou son abbréviation `-k`.

```
curl --insecure https://example.org  
curl -k https://example.org
```

## Utiliser un fichier d'autorités de certification spécifique

Si on utilise, par exemple, un certificat autosigné, ou signé par une autorité de certification (AC) personnelle, et qu'on souhaite s'assurer que le certificat utilisé par le site est bien valide, on peut donner à `curl` un fichier contenant le certificat public de l'AC (il est possible d'y mettre plusieurs certificats) au format PEM. On utilise l'option `--cacert`.

```
curl --cacert fichier_AC.pem https://example.org
```

On peut aussi utiliser l'option `--capath` dont l'argument est un dossier contenant des fichiers de certificats d'AC.

```
curl --capath ~/ACs https://example.org
```

# Débloquer un RAID coincé en resync=PENDING

Il arrive des fois que certains disques RAID soient bloqués en `resync=PENDING` (utilisez votre supervision pour le détecter !), ce que l'on peut voir avec la commande suivante :

```
cat /proc/mdstat
```

Ça ressemble à ça :

```
md0 : active (auto-read-only) raid1 sda1[0] sdb1[1]
      16760832 blocks super 1.2 [2/2] [UU]
      resync=PENDING
```

Pour décoincer ça, il suffit de faire

```
mdadm --readwrite /dev/md0
```



# Empêcher l'activation d'un service à son installation

On peut vouloir installer un service mais éviter qu'il ne s'active à l'installation.

Par exemple, pour changer sa configuration avant utilisation, ou parce qu'un autre service écoute déjà sur le port que le service. J'ai cherché comment faire en voulant installer PostgreSQL sur un serveur où il y avait déjà un PostgreSQL installé par un autre paquet (le PostgreSQL fourni par Gitlab, en l'occurrence).

La solution est fort simple et utilise les preset d'activation de systemd.

Exemple avec MariaDB (le nom du fichier doit suivre le modèle `<priority>-<policy-name>.preset`, voir la page de manuel) :

```
mkdir -p /etc/systemd/system-preset
vi /etc/systemd/system-preset/85-systemd.preset
```

Et dedans, mettre :

```
disable mariadb.*
```

On peut aussi forcer l'activation avec `enable` à la place de `disable`.

Je vous laisse voir la page de manuel pour plus de détails.

# Exécuter une action à la mise en veille / au réveil

## Systemd

On mettra un script dans `/lib/systemd/system-sleep/` :

Exemple de script :

```
#!/bin/sh

case "${1}" in
    pre)
        echo "Suspension ou hibernation"
        ;;
    post)
        echo "Réveil ou dégel"
        ;;
    esac
```

Le 2e argument ( `$2` ) pourra être `suspend` , `hibernate` , `suspend-then-hibernate` ou `hybrid-sleep` , si vous voulez effectuer des actions différentes pour ces cas.

Pour plus d'informations, voir la [page de manuel de `systemd-sleep`](#) .

## InitV

On mettra un script dans `/etc/pm/sleep.d/` .

Exemple de script :

```
#!/bin/sh
```

```
case "${1}" in
  suspend|hibernate)
    echo "Suspension ou hibernation"
    ;;
  resume|thaw)
    echo "Réveil ou dégel"
    ;;
esac
```

# Firewalld : un firewall simple à utiliser

Firewalld est un pare-feu que je trouve très agréable à utiliser, où on peut « cacher » la complexité de certains éléments de configuration derrière des noms simples à utiliser.

Par exemple, je peux avoir un service qui n'a pas spécialement de port dédié, donc qui n'est pas proposé par firewall. Mettons un wireguard qui écoute sur le port 9879. Plutôt que d'utiliser `9879/udp` dans ma configuration, je vais créer un service `wireguard`, et c'est ce service que j'autoriserai.

Ce sera bien plus parlant quand je relirai la configuration.

Liens :

- Documentation officielle : <https://firewalld.org/documentation/>
- <https://www.linuxtricks.fr/wiki/firewalld-le-pare-feu-facile-sous-linux>
- <https://www.rootusers.com/how-to-use-firewalld-rich-rules-and-zones-for-filtering-and-nat/>
- <https://kb.vander.host/security/firewalld-cheat-sheet/>

## Principes

En très gros et en très résumé, on va avoir des zones, qui sont des ensembles d'adresses IP et la zone `public` qui concerne toutes les IPs, sauf celles qui sont dans d'autres zones.

On va aussi avoir des services, qui décrivent... des services : port et protocole (ex: `5666` et `tcp` pour nrpe).

On va aussi avoir des « rich rules » dans les zones, des exceptions aux règles appliquées dans la zone.

Toute la configuration est dans des fichiers XML très simples à lire, c'est très agréable. Tant qu'on ne surcharge pas la configuration par défaut, les fichiers sont dans `/usr/lib/firewalld/` mais dès qu'on modifie un élément de configuration, celui-ci se retrouvera copié dans `/etc/firewalld/` et modifié.

# Test de configuration

Si on modifie de la configuration à la main (en écrivant dans `/etc/firewalld`, on prendra soin de tester la configuration avec les commandes suivantes :

Si `firewalld` est coupé :

```
firewall-offline-cmd --check-config
```

Si `firewalld` est lancé :

```
firewall-cmd --check-config
```

## Attention

Quand on installe `firewalld`, le firewall démarre de suite en n'autorisant en public que `ssh` (et `dhcpv6-client`).

Il est donc préférable de l'installer et de le couper directement après :

```
apt install firewalld &&  
systemctl stop firewalld
```

On pourra créer la configuration tranquillement avec la commande `firewall-offline-cmd` et lancer le service une fois la configuration terminée.

## Changements permanents

Si on veut rendre un changement permanent (c-à-d qu'il soit écrit dans la config au lieu d'être juste appliqué jusqu'au redémarrage), il faut ajouter ça aux commandes :

```
--permanent
```

Par contre, avec `--permanent`, il faut recharger la configuration pour appliquer les modifications (ou alors on applique une fois avec `--permanent` et une fois sans) :

```
firewall-cmd --reload
```

À l'inverse, on peut créer des règles sans le `--permanent` et ensuite écrire ces règles dans la configuration permanent avec la commande suivante :

```
firewall-cmd --runtime-to-permanent
```

**NB** : certaines commandes nécessitent forcément le `--permanent` .

# Bloquer une adresse IP

On peut soit ajouter les adresses aux zones `drop` ou `block` :

```
firewall-cmd --zone=drop --add-source 192.0.2.0/24
firewall-cmd --zone=drop --add-source 192.0.2.0/24 --permanent
```

Soit ajouter une `rich-rule` ( `man firewalld.richlanguage` ) à la zone `public` :

```
firewall-cmd --zone public --add-rich-rule "rule family=ipv4 source address=192.0.2.0/24 reject"
firewall-cmd --zone public --add-rich-rule "rule family=ipv4 source address=192.0.2.0/24 reject" --permanent
```

Pour enlever un blocage :

```
firewall-cmd --zone drop --remove-source 51.159.0.0/16
firewall-cmd --zone drop --remove-source 51.159.0.0/16 --permanent
```

```
firewall-cmd --zone public --remove-rich-rule "rule family=ipv4 source address=51.159.0.0/16 reject"
firewall-cmd --zone public --remove-rich-rule "rule family=ipv4 source address=51.159.0.0/16 reject" --
permanent
```

Pour voir les blocages par `rich-rule` (la 1ère commande donne les blocages actuellement activés, l'autre ceux qui sont dans les fichiers de configuration. Il peut y avoir une différence... ou pas !) :

```
firewall-cmd --list-rich-rules
firewall-cmd --list-rich-rules --permanent
```

Pour bloquer un ipset (voir plus bas) :

```
firewall-cmd --zone=drop --add-source ipset:le_nom_de_l_ipset
firewall-cmd --zone=drop --add-source ipset:le_nom_de_l_ipset --permanent
```

# Zones

Voir les zones disponibles :

```
firewall-cmd --get-zones
```

NB : la zone `public`, par défaut, n'autorise que le SSH et dhcpv6-client. L'installation de firewalld sur une machine va donc couper l'accès aux services. Il faut donc stopper firewalld juste après son installation, regarder les ports utilisés sur la machine et modifier la zone `public` soit en copiant `/usr/lib/firewalld/zones/public.xml` dans `/etc/firewalld/zones/`, soit en préparant une ligne de commande à lancer juste après le démarrage de firewalld.

NB : les zones peuvent avoir une `target`, l'action à appliquer aux connexions qui correspondent à la zone. Voir [la doc](#).

NB : Une adresse IP ne peut se trouver que dans une seule zone mais on peut ajouter dans une zone un réseau qui contient une adresse IP déjà présente dans une autre zone. Cependant, le comportement peut ne pas être celui attendu. Il vaut mieux ajouter une `rich-rule` à la zone pour faire une exception aux règles de la zone.

Voir la zone par défaut (celle sur laquelle s'appliqueront les modifications si on ne spécifie pas la zone) :

```
firewall-cmd --get-default-zone
```

Définir la zone par défaut :

```
firewall-cmd --set-default-zone work
```

Voir la configuration de la zone :

```
firewall-cmd --info-zone lazone
```

Voir la configuration de toutes les zones :

```
firewall-cmd --list-all-zones
```

Créer une zone :

```
firewall-cmd --permanent --new-zone mazonne  
firewall-cmd --reload
```

Supprimer une zone :

```
firewall-cmd --permanent --delete-zone mazonne  
firewall-cmd --reload
```

Chaque interface du système peut être attribuée à une zone. Pour ajouter l'interface ens192 à la zone work en l'enlevant de sa précédente zone :

```
firewall-cmd --change-interface ens192 --zone work [--permanent]
```

Pour retirer l'interface ens192 de la zone work :

```
firewall-cmd --remove-interface ens192 --zone work [--permanent]
```

Pour ajouter l'interface ens192 à la zone work (interface qui ne soit pas être affectée à une zone) :

```
firewall-cmd --add-interface ens192 --zone work [--permanent]
```

Ajouter des adresses IP ou un réseau à une zone :

```
firewall-cmd --zone work --add-source 192.0.2.0/24 [--permanent]  
firewall-cmd --zone work --add-source 192.0.2.200 [--permanent]
```

Retirer des adresses IP ou un réseau d'une zone :

```
firewall-cmd --zone work --remove-source 192.0.2.0/24 [--permanent]  
firewall-cmd --zone work --remove-source 192.0.2.200 [--permanent]
```

Pour basculer une adresse IP ou un réseau d'une zone à une autre :

```
firewall-cmd --zone l_autre_zone --change-source 192.0.2.0/24 [--permanent]  
firewall-cmd --zone l_autre_zone --change-source 192.0.2.200 [--permanent]
```

Si l'adresse IP / le réseau était dans une autre zone, ça équivaut à un `--remove-source` suivi d'un `--add-source`, si ce n'était pas le cas, ça fait juste comme un `--add-source`.

Voir la `target` d'une zone :

```
firewall-cmd --permanent --get-target --zone drop
```

Définir la `target` d'une zone :

```
firewall-cmd --permanent --set-target [default|ACCEPT|DROP|REJECT] --zone drop
```



Pour voir dans quelle zone est une adresse IP :

```
firewall-cmd --get-zone-of-source=<adresse IP ou réseau en notation CIDR ou adresse MAC ou ipset>
```

Si ça répond `no zone`, c'est que l'IP ou le réseau n'est pas explicitement associé à une zone.

**Attention** : si un réseau est enregistré dans une zone, lancer la commande sur une IP du réseau ne renverra pas la zone en question !

# Services

Voir les services existants :

```
firewall-cmd --get-services
```

Voir le détail d'un service :

```
firewall-cmd --info-service ssh
```

Créer un nouveau service :

```
firewall-cmd --permanent --new-service influxdb  
firewall-cmd --permanent --service influxdb --set-description InfluxDB  
firewall-cmd --permanent --service influxdb --add-port 8086/tcp
```

Ajouter un service à une zone :

```
firewall-cmd --zone public --add-service nrpe [--permanent]
```

Retirer un service d'une zone :

```
firewall-cmd --zone public --remove-service nrpe [--permanent]
```

Voir les services d'une zone :

```
firewall-cmd --list-services --zone work
```

Si on ne souhaite pas créer de service mais autoriser un certain port et protocole, on peut les ajouter directement à la zone :

```
firewall-cmd --zone work --add-port 1234/udp [--permanent]
```

Et pour les supprimer :

```
firewall-cmd --zone work --remove-port 1234/udp [--permanent]
```

Pour voir les ports/protocoles d'une zone (ça ne listera pas les services !) :

```
firewall-cmd --list-ports --zone work
```

# IPSet : groupes d'adresses

Doc : [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/security\\_guide/sec-setting\\_and\\_controlling\\_ip\\_sets\\_using\\_firewalld](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sec-setting_and_controlling_ip_sets_using_firewalld)

Cas d'usage : on voit plein de spammeurs venant du VPN d'Avast. On fait un `whois`, on voit le numéro d'**AS** des serveurs d'Avast, on va sur <https://asnlookup.com/> pour choper leur liste d'adresses IP et on en fait un ipset pour les bloquer.

NB : un ipset ne peut contenir qu'un type d'adresses, IPv4 ou IPv6, pas les deux.

NB : **Fail2ban**, lorsqu'il utilise firewalld, utilise des ipset, mais à un niveau un peu plus bas (nftables). Ces ipset ne sont pas vus par firewalld. On peut voir tous les ipset, même bas niveau avec la commande `ipset list -name` ( `ipset list le_nom_de_l_ipset` pour voir les adresses et le détail de l'ipset).

Un ipset sert à regrouper des adresses pour leur appliquer des règles facilement.

Voir les types d'ipset disponibles :

```
firewall-cmd --get-ipset-types
```

Voir les ipsets existants :

```
firewall-cmd --permanent --get-ipsets
```

Créer un ipset :

```
firewall-cmd --permanent --type hash:net --new-ipset test
```

Pour un ipset IPv6 :

```
firewall-cmd --permanent --type hash:net --option "family=inet6" --new-ipset test-v6
```

Supprimer un ipset :

```
firewall-cmd --permanent --delete-ipset test
```

Voir les infos d'un ipset :

```
firewall-cmd --info-ipset test [--permanent]
```

Ajouter une adresse IP à un ipset :

```
firewall-cmd --ipset test --add-entry 192.0.2.1 [--permanent]
```

Supprimer une adresse IP d'un ipset :

```
firewall-cmd --ipset test --remove-entry 192.0.2.1 [--permanent]
```

Voir les adresses IP d'un ipset :

```
firewall-cmd --ipset test --get-entries [--permanent]
```

Ajouter un paquet d'IP d'après un fichier :

```
cat > iplist.txt <<EOL
192.0.2.2
192.0.2.3
198.51.100.0/24
203.0.113.254
EOL
firewall-cmd --ipset test --add-entries-from-file iplist.txt [--permanent]
```

Supprimer un paquet d'IP d'après un fichier :

```
firewall-cmd --ipset test --remove-entries-from-file iplist.txt [--permanent]
```

Ajouter un ipset dans une zone :

```
firewall-cmd --zone drop --add-source ipset:test [--permanent]
```

Supprimer un ipset d'une zone :

```
firewall-cmd --zone drop --remove-source ipset:test [--permanent]
```

# Créer des exceptions avec des règles riches

Cas classique : on bloque un pays avec des ipset et quelqu'un a besoin d'accéder à nos services depuis là-bas.

L'ipset est dans la zone `drop`. On va ajouter une `rich-rule` pour faire une exception :

```
firewall-cmd --zone drop --add-rich-rule='rule family="ipv4" source address="192.0.2.29" port
port="443" protocol="tcp" accept'
firewall-cmd --zone drop --add-rich-rule='rule family="ipv4" source address="192.0.2.29" port port="80"
protocol="tcp" accept'
firewall-cmd --zone drop --permanent --add-rich-rule='rule family="ipv4" source address="192.0.2.29" port
port="443" protocol="tcp" accept'
firewall-cmd --zone drop --permanent --add-rich-rule='rule family="ipv4" source address="192.0.2.29" port
port="80" protocol="tcp" accept'
```

Pour supprimer l'exception :

```
firewall-cmd --zone drop --remove-rich-rule 'rule family="ipv4" source address="192.0.2.29" port
port="80" protocol="tcp" accept'
firewall-cmd --zone drop --remove-rich-rule 'rule family="ipv4" source address="192.0.2.29" port
port="443" protocol="tcp" accept'
firewall-cmd --zone drop --permanent --remove-rich-rule 'rule family="ipv4" source address="192.0.2.29" port
port="80" protocol="tcp" accept'
firewall-cmd --zone drop --permanent --remove-rich-rule 'rule family="ipv4" source address="192.0.2.29" port
port="443" protocol="tcp" accept'
```

On peut utiliser des services dans les règles riches :

```
firewall-cmd --zone drop --add-rich-rule='rule family="ipv4" source address="192.0.2.29" service name=https
accept'
```

On peut utiliser des ipset dans les règles riches :

```
firewall-cmd --zone drop --add-rich-rule='rule family="ipv4" source ipset="test-v6" service name=https accept'
```

**NB** : ajouter l'adresse IP en source dans la zone `public` ne servirait strictement à rien.

# Blocage geoIP

On peut se baser sur le script de <https://github.com/simonbouchard/geoip-blocking-w-firewalld> (le fork de Framasoft).

On modifie les pays à bloquer dans `/etc/default/firewalld-geoip` et on lance le script. À mettre dans un cron pour mettre à jour les adresses.

# Faire du NAT

Voir la partie `Network Address Translation (NAT)` de <https://www.rootusers.com/how-to-use-firewalld-rich-rules-and-zones-for-filtering-and-nat/>. Je n'ai pas eu l'occasion de tester ça.

# Lancer des commandes sudo avec authentification par agent SSH

## Installation du paquet nécessaire

```
apt install libpam-ssh-agent-auth
```

## Configuration

Modifier la configuration sudo pour y ajouter cette ligne :

```
Defaults:%sudo env_keep += "SSH_AUTH_SOCK"
```

On peut éditer le fichier `/etc/sudoers` avec la commande `visudo` ou mettre ceci dans un fichier du dossier `/etc/sudoers.d` avec `visudo -f /etc/sudoers.d/keep-ssh-auth-sock`

Mettre cette ligne dans le fichier `/etc/pam.d/sudo`, au-dessus de la ligne `@include common-auth` :

```
auth sufficient pam_ssh_agent_auth.so file=/etc/security/sudo_authorized_keys
```

Enfin, mettre les clés publiques des clés SSH qui peuvent être utilisées pour cette authentification dans le fichier `/etc/security/sudo_authorized_keys`.

# LVM

LVM permet, à partir de plusieurs disques physiques, de créer des partitions qui utiliseront plusieurs disques de manière invisible.

L'autre avantage est de pouvoir rajouter du disque sans arrêter la machine ou démonter la partition.

## Bases

Le LVM est composé de plusieurs morceaux :

- les volumes physiques, listables avec `pvs`, correspondants aux partitions physiques des disques utilisés pour le LVM
- les groupes de volumes, listables avec `vgs`, qui sont des agrégats de volumes physiques
- les volumes logiques, listables avec `lvs`, qui sont des partitions utilisant des portions de groupes de volumes

On n'est pas obligé d'utiliser toute la place disponible dans un groupe de volume lorsqu'on crée un volume logique, on a tout à fait le droit de se garder de la place.

On peut augmenter la taille d'un volume physique à chaud, sans démonter la partition, mais on doit la démonter si on veut réduire la taille ! C'est pourquoi il vaut mieux mettre juste ce qu'il faut comme taille (avec une marge de sécurité, bien sûr), quitte à augmenter plus tard la taille de la partition, plutôt que de tout mettre et ne plus avoir de marge pour une autre partition.

## Créer une nouvelle partition

```
lvcreate -L 30G -n nom-partition xenvg  
mkfs.ext4 /dev/mapper/xenvg-nom-partition
```

## Augmentation de la taille d'une partition

Ajout d'un nouveau disque au groupe de volumes `xenvg` :

```
pvcreate /dev/sdc1  
vgextend xenvg /dev/sdc1
```

Augmentation de la taille de la partition `data`, appartenant au groupe `xenvg` :

```
lvextend -L +6.5G /dev/xenvg/data  
resize2fs /dev/mapper/xenvg-data
```

Pour prendre toute la place disponible :

```
lvextend -l +100%FREE /dev/mapper/xenvg-data  
resize2fs /dev/mapper/xenvg-data
```

Si c'est une partition de "swap" qui a été étendue :

```
swapoff -a  
mkswap /dev/mapper/beta--vg-swap_1  
swapon -a
```



# Monter une ou des partitions contenues dans un fichier qcow2

J'ai des images disques qcow2 sur ma machine, qui servent à mes machines virtuelles que j'utilise pour développer.

Pour modifier des fichiers dessus sans avoir besoin de démarrer les VMs, on peut monter les images disques sur le système hôte.

## Installer l'outil nécessaire

```
sudo apt install libguestfs-tools
```

## Monter l'image disque

### Monter tout le système de fichier

L'outil `guestmount` va inspecter les disques à la recherche d'un système d'exploitation et va monter toutes les partitions comme elles seraient montées sur la machine virtuelle.

```
sudo guestmount -a /path/to/qcow2/image -i /path/to/mount/point
```

### Monter une partition en particulier

```
sudo guestmount -a /path/to/qcow2/image -m <device> /path/to/mount/point
```

Exemple réel :

```
guestmount -a ~luc/.vms/sympa.qcow2 -m /dev/sda1 /mnt/
```

- `~luc/.vms/sympa.qcow2` : le chemin vers l'image
- `/dev/sda1` : la partition de la VM à monter
- `/mnt/` : l'endroit où monter la partition

Si vous ne connaissez pas l'identifiant de la partition que vous souhaitez monter, vous pouvez mettre une partition fantaisiste (exemple: `/dev/trs`) et le message d'erreur vous indiquera les partitions existantes :

```
libguestfs : erreur : mount_options: mount_options_stub: /dev/trs: No such file or directory
guestmount: '/dev/trs' could not be mounted.
guestmount : Voulez-vous monter l'un de ces systèmes de fichiers ?
guestmount:   /dev/sda1 (ext4)
guestmount:   /dev/sda5 (swap)
```

# Démonter l'image disque

```
umount /mnt/
```

---

Tiré de <https://www.xmodulo.com/mount-qcow2-disk-image-linux.html>.

# Réinstaller les modules Perl installés avec la version précédente de Perl

Un truc con quand on installe des modules Perl avec `cpan` ou `cpanm`, c'est qu'ils sont installés dans un répertoire dont le nom est la version de Perl utilisée.

Donc quand on installe un module sur une Debian Stretch et qu'on met à jour la machine vers Debian Buster, les modules installés via `cpan` ou `cpanm` ne sont plus disponibles vu qu'on change de version de Perl.

Heureusement que les modules installés sont listés dans un coin ☐

Pour trouver les modules qu'on avait installé en Stretch :

```
cat /usr/local/lib/x86_64-linux-gnu/perl/5.24*/perllocal.pod | grep "C<Module>" | sed -e 's/*C<Module>L<\\(.*)\\|.*/>\\1/' | tr '\\n' ' '
```

Pour tous les réinstaller en un tour de main :

```
cpanm $(cat /usr/local/lib/x86_64-linux-gnu/perl/5.24*/perllocal.pod | grep "C<Module>" | sed -e 's/*C<Module>L<\\(.*)\\|.*/>\\1/' | tr '\\n' ' ')
```

Bien évidemment, ça marche avec les précédentes versions de Debian, et ça devrait aussi fonctionner avec les suivantes, il n'y a qu'à changer le numéro de version ☐

**NB** : ça ne concerne pas les modules Perl installés via les paquets Debian.

# Salt

Salt est un logiciel de gestion de configuration comme Puppet ou Ansible.

Je l'utilise chez Framasoft et sur mon infra personnelle parce que je l'aime bien :

- rapide ;
- très bien documenté ;
- syntaxe claire, accessible mais néanmoins flexible et puissante.

## Changer ses mots de passe en masse

Je change mes mots de passe régulièrement (une fois par an environ). C'est toujours galère à faire quand on gère une tripotée de serveurs (entre les serveurs physiques et les VMs, on en est à plus de 100 serveurs chez Framasoft).

Avant, je faisais ça à la main : je lançais mssh sur 4, 6 ou 8 serveurs à la fois, et je modifiais mon mot de passe à la main. Mais ça, c'était avant.

### Salt à la rescousse

Pour changer le mot de passe de l'utilisateur `bar` sur le serveur `foo` avec salt, on fait :

```
salt foo shadow.set_password bar "$6$selselse$HASHEDPASSWORD"
```

`$6$selselse$HASHEDPASSWORD` correspond à votre mot de passe salé et hashé. Vous retrouvez un brol du genre dans votre `/etc/shadow` (Oh ! Vous avez remarqué ? C'est le nom du module salt qui permet de modifier votre mot de passe ! C'est bien fait quand même ☺)

Pour créer l'ensemble `$6$selselse$HASHEDPASSWORD`, vous pouvez utiliser python (nécessite le module passlib, fourni par le paquet Debian `python3-passlib`) :

```
python3 -c "from passlib.hash import sha512_crypt; print(sha512_crypt.hash('LE_PASSWORD', rounds=5000))"
```

Bon, on sait comment faire, mais on ne va pas s'amuser à taper 100 fois ces commandes !

Salt permet de vérifier que les minions (les agents Salt) répondent bien avec cette commande :

```
salt foo test.ping
```

Ce qui donne :

```
foo:  
True
```

On va changer le format de sortie :

```
salt foo --out text test.ping
```

Ce qui nous donne :

```
foo: True
```

Bien ! On peut pinguer d'un coup tous les minions avec :

```
salt \* --out text test.ping
```

On a donc la liste des minions, la commande pour changer le mot de passe... on va mixer tout ça :

```
salt \* --out text test.ping | \  
sed -e "s@([^\:]*):.*@echo salt \1 shadow.set_password bar \\\\"$(python3 -c \"from passlib.hash import  
sha512_crypt; print(sha512_crypt.hash('LE_PASSWORD', rounds=5000))\")\\\\"@"
```

1ère regex : on dégage les `: True`, et la deuxième, on enrobe le nom du minion pour que ça nous donne un truc comme :

```
echo salt foo shadow.set_password bar \"$(python3 -c \"from passlib.hash import sha512_crypt;  
print(sha512_crypt.hash('LE_PASSWORD', rounds=5000))\")\"
```

Quand on exécute ça, ça donne un truc genre :

```
salt foo shadow.set_password bar  
"$6$8qJhzAi6$.O8bOisJaM9fH05aXx7xnKXOVFoI9CRzjORFWDqoPR/TBOiYVZUEJKtUKirNMyaZJvJMPVUMhNry9QP  
JgHK/"
```

Bien évidemment, on va mettre ça dans un fichier qu'on va éditer pour modifier le mot de passe (bah oui, on va quand même pas mettre le même mot de passe sur tous les serveurs).

```
salt \* --out text test.ping | \  
sed -e "s@\([^\:]*\)::.*@echo salt \1 shadow.set_password bar \\\\"$(python3 -c \"from passlib.hash import  
sha512_crypt; print(sha512_crypt.hash('LE_PASSWORD', rounds=5000))\\\"\\\\\\\"@\" > /tmp/chpasswd.txt
```

On édite `/tmp/chpasswd.txt` pour mettre ses mots de passe bien comme il faut puis :

```
bash /tmp/chpasswd.txt | bash
```

Le `echo` va nous sortir la commande kivabien qui sera interprétée par bash. Le bout de python transformera le mot de passe en hash dans le format kivabien pour le fichier `/etc/shadow` et la commande salt sera lancée sur chaque minion.

# Sed

C'est l'outil absolu pour modifier du texte en le passant par un pipe ! Ou pour effectuer des changements en masses sur un fichier sans l'ouvrir. Bref, comme le dit l'adage : « Sed, c'est bien »



Il est possible de faire des trucs de tarés avec (hey, c'est pas juste un truc pour faire des substitutions à coup d'expressions rationnelles, c'est un vrai éditeur de texte, on peut se balader dans le texte, faire des copier/coller, etc).

## Syntaxe de base

Avec un fichier :

```
sed <commande> fichier.txt [fichier2.txt] [fichier3.txt]
```

En utilisant la sortie d'une autre commande :

```
find . -name \*.txt | sed <commande>
```

**NB :** `sed`, par défaut, ne modifie pas le fichier utilisé. Il affichera sur la sortie standard le fichier modifié par la commande passée à `sed`. Si on souhaite que le fichier soit modifié, on utilise l'option `--in-place` ou son abbréviation `-i` (on peut indifféremment placer l'option avant ou après la commande) :

```
sed <commande> --in-place fichier.txt  
sed -i <commande> fichier.txt
```

## Expression régulières

Si `sed` est un éditeur de texte (son nom veut dire Stream EDitor) et qu'il est utilisable en lui donnant des commandes équivalentes à « Va à ligne 3, supprime 4 caractères, descend à la ligne suivante... », on l'utilise souvent avec des expressions régulières.

## Rappel sur les expressions régulières

Les expressions régulières permettent de rechercher des correspondances avec un motif, écrit avec une syntaxe spéciale.

Les éléments de syntaxe suivants appartiennent aux *Perl Compatible Regular Expression* (PCRE), qui sont le standard de la très grande majorité des langages de programmation. `sed` utilisant une syntaxe légèrement différente, certains caractères devront être échappés (voir plus bas).

**NB** : j'ai placé les expressions régulières de cette section entre des `/` pour les distinguer des chaînes de caractères simples.

- `.` : correspond à un caractère, n'importe lequel. `/./` correspondra à tout sauf à une chaîne vide
- `?` : quantificateur, modifie la correspondance du caractère qui le précède : celui ci peut être présent zéro ou une fois. `/a?/` correspondra à une chaîne vide ou à `a`
- `+` : quantificateur : le caractère précédent sera présent une ou plusieurs fois. `/a+/` correspondra à `a`, `aa`, `aaa`...
- `*` : quantificateur : le caractère précédent sera présent zéro ou plusieurs fois. `/a*/` correspondra à une chaîne vide, à `a`, `aa`, `aaa`...
- `{n}` : quantificateur : le caractère précédent sera présent `n` fois. `/a{3}/` correspondra à `aaa`
- `{n,m}` : quantificateur : le caractère précédent sera présent de `n` à `m` fois. `/a{3,5}/` correspondra à `aaa`, `aaaa` ou `aaaaa`
- `{n,}` : quantificateur : le caractère précédent sera présent au moins `n` fois. `/a{3,}/` correspondra à `aaa`, `aaaa`, `aaaaa`, `aaaaaa`...
- `|` : séparateur d'expression. `/bonjour|hello/` correspondra à `bonjour` ou à `hello`
- `[^liste]` : correspond aux caractères n'étant pas entre crochets. `/[^ae]/` correspondra à n'importe quel caractère sauf à `a` et à `e`
- `[liste]` : correspond à un des caractères entre crochets. `/[ae]/` correspondra à `a` ou à `e`. Pour que le caractère `^` soit un choix possible, il faut le placer à une autre place que la première place : `[liste^]`, `[lis^te]`... On peut aussi spécifier des plages de caractères : `[0-9a-zA-Z]`
- `^` : ancre, correspond au début de la ligne. `/^a/` correspondra à `a` si celui-ci est le premier caractère de la ligne
- `$` : ancre, correspond à la fin de la ligne. `/a$/` correspondra à `a` si celui-ci est le dernier caractère de la ligne
- `(...)` : groupe l'expression. On peut s'en servir, par exemple, pour capturer des éléments (ce qui permet de les réutiliser plus tard) ou faire des sous-expressions. `/Bonjour (foo|bar), ça va \?/` correspondra à `Bonjour foo, ça va ?` et à `Bonjour bar, ça va ?`

Pour utiliser les caractères de manière littérale (exemple : pour correspondre à un point), on les échappera avec un `\`. `/\./` correspondra au caractère point (`.`).

## Caractères à échapper dans sed



La version GNU de `sed` utilise les *Basic Regular Expression* (BRE), qui ont une syntaxe légèrement différentes des PCRE.

Certains caractères doivent donc être échappés dans les BRE, qui sont utilisables tels quels pour des PCRE :

- le quantificateur `+`. Exemple : `/a\+/`
- le quantificateur `?`. Exemple : `/a\?/`
- le quantificateur `{i}`. Exemple : `/a\{5\}/`
- les parenthèses `(...)`. Exemple : `/\a\(/`
- le séparateur d'expressions régulières `|`. Exemple : `/a\|b/`

# Effectuer une substitution de texte

La commande à utiliser est `'s/expression régulière/substitution/'` :

```
sed 's/foo/bar/' foo.txt
```

Cette commande remplacera la **1ère occurrence** de `foo` de **chaque ligne** du fichier par `bar`.

Si on souhaite remplacer toutes les occurrences de chaque ligne, on emploie le modificateur `g` :

```
sed 's/foo/bar/g' foo.txt
```

Si on ne souhaite remplacer que l'occurrence n°X de chaque ligne :

```
sed 's/foo/bar/X' foo.txt
## Exemple avec la 2e occurrence :
sed 's/foo/bar/2' foo.txt
```

Si on ne souhaite remplacer l'occurrence n°X de chaque ligne, ainsi que les suivantes :

```
sed 's/foo/bar/gX' foo.txt
## Exemple avec la 2e occurrence et les suivantes :
sed 's/foo/bar/g2' foo.txt
```

Ajouter quelque chose au début de chaque ligne :

```
sed 's/^/FooBar /' foo.txt
```

Ajouter quelque chose à la fin de chaque ligne :

```
sed 's/$/ BazQux/' foo.txt
```

Si on souhaite que l'expression régulière soit insensible à la casse (ex : `s` qui correspond aussi à `S`), on emploie le modificateur `i` :

```
sed 's/foo/bar/i' foo.txt
```

On peut utiliser plusieurs modificateurs en même temps :

```
sed 's/foo/bar/gi' foo.txt
```

Pour réutiliser ce qui a correspondu à l'expression régulière dans la chaîne de substitution, on utilise le caractère `&` :

```
sed 's/foo/& et bar/' foo.txt
```

Pour réutiliser ce qui a correspondu à un groupe d'expression, on utilise `\1` pour le 1er groupe, `\2` pour le 2e groupe... :

```
sed 's/Bonjour (foo|bar). Il fait beau./Bonsoir \1. À demain./' foo.txt
```

**NB** : on peut utiliser d'autres séparateurs que le caractère `/`, ce qui rend l'écriture d'une commande plus simple si l'expression régulière ou la substitution comportent des `/` : plus besoin de les échapper. Exemple :

```
sed 's@/home/foo@/var/bar@' foo.txt
```

# Supprimer des lignes

Supprimer la ligne `n` :

```
sed 'nd' foo.txt
```

## Exemple avec la 3e ligne :

```
sed '3d' foo.txt
```

Supprimer les lignes `n` à `m` :

```
sed 'n,md' foo.txt
```

## Exemple :

```
sed '3,5d' foo.txt
```

Supprimer toutes les lignes sauf la ligne `n` :

```
sed 'n!d' foo.txt  
  
## Exemple :  
  
sed '6!d' foo.txt
```

**NB** : attention, le caractère `!` est un caractère spécial pour `bash` (et les autres shells). Si vous utilisez des apostrophes (`'`) pour votre commande `sed`, tout ira bien, mais si vous utilisez des guillemets doubles (`"`), il faut l'échapper avec un `\`.

Supprimer toutes les lignes sauf les lignes `n` à `m` :

```
sed 'n,m!d' foo.txt  
  
## Exemple :  
  
sed '6,8!d' foo.txt
```

Supprimer les lignes qui correspondent à une expression régulière :

```
sed '/regex/d' foo.txt  
  
## Exemple :  
  
sed '/foobar/d' foo.txt
```

Pour supprimer les lignes vides, d'un fichier, il suffit d'utiliser l'expression régulière qui signifie que la ligne est vide :

```
sed '/^$/d' foo.txt
```

Pour supprimer la première ligne correspondant à une expression régulière, ainsi que toutes les lignes suivantes :

```
sed '/regex/, $d' foo.txt  
  
## Exemple :  
  
sed '/foobar/, $d' foo.txt
```

**NB** : attention encore une fois aux guillemets doubles, il faudrait échapper `$` dans cette commande car le shell essaierait d'interpréter `$d` comme une variable.

# Tmux

Tmux est un multiplexeur de terminal. Il permet d'utiliser plusieurs terminaux virtuels dans une seule fenêtre de terminal ou une session sur un terminal distant.

## Ligne de commande

### Lancement

Rien de plus simple :

```
tmux
```

Pour nommer une session :

```
tmux new-session -s <nom de la session>
```

### Voir les sessions tmux existantes

```
tmux ls
```

### Se rattacher à une session tmux existante

S'il n'y a qu'une seule session, ou si vous voulez vous rattacher à la dernière auquel vous étiez rattaché :

```
tmux at
```

Si vous souhaitez cibler une autre session :

```
tmux at -t <nom de la session>
```

Bonus : vous pouvez être plusieurs personnes sur une même session. C'est très pratique quand on doit faire des manipulations à plusieurs, ou dans un but didactique.

# Couper une session tmux existante

S'il n'y a qu'une seule session, ou si vous voulez couper la dernière auquel vous étiez rattaché :

```
tmux kill-session
```

Si vous souhaitez cibler une autre session :

```
tmux kill-session -t <nom de la session>
```

## Aide

Faites `Ctrl + b` puis `?` et tmux vous affichera une liste de commande accessibles avec des raccourcis claviers

## Raccourcis claviers usuels

### Se détacher de la session tmux

Tmux peut continuer à fonctionner même quand on est déconnecté ou qu'on se détache de la session, ce qui est très pratique pour :

- les connexions réseaux moisiées qui vous déconnectent de votre session SSH
- lancer une commande qui va durer longtemps

`Ctrl + b` puis `d`

### Ouvrir une nouvelle fenêtre

`Ctrl + b` puis `c`

### Naviguer entre les fenêtres

`Ctrl + b` puis `n` (-> `next`) pour passer à la fenêtre suivante.

`Ctrl + b` puis `p` (-> `previous`) pour passer à la fenêtre précédente

# Renommer une fenêtre

`Ctrl` + `b` puis `,`, vous aurez un prompt dans la barre en bas, tapez ce que vous voulez et appuyez sur la touche `Entrée`.

Pour sortir du prompt sans valider, appuyez sur la touche `Esc`.

# Créer un nouveau panneau

## En coupant la fenêtre horizontalement

Le nouveau panneau sera créé à droite du panneau courant.

`Ctrl` + `b` puis `%`

## En coupant la fenêtre verticalement

Le nouveau panneau sera créé en dessous du panneau courant.

`Ctrl` + `b` puis `"`

# Zoomer sur un panneau

Si les panneaux sont pratiques, on a parfois envie d'en prendre un et de l'avoir temporairement en plein écran. Pour ça :

`Ctrl` + `b` puis `z`

Et on utilise le même raccourci pour remettre le panneau en petit comme avant.

# Naviguer entre les panneaux

`Ctrl` + `b` puis utiliser les flèches du clavier

# Commandes

Pour entrer en mode commande, faites `Ctrl` + `b` puis `:`. Vous aurez un prompt dans la barre en bas.

# Afficher toutes les commandes disponibles

Utiliser la commande `list-commands`.

## Changer le répertoire de départ

Quand on ouvre un tmux, chaque nouvelle fenêtre ou nouveau panneau s'ouvrira dans le répertoire depuis lequel vous avez créé le tmux.

Pour changer ce répertoire de départ, utiliser la commande `attach -c /le/dossier/que/vous/voulez`.

## Afficher les variables d'environnement de la session

Utilisez la commande `show-environnement` ou son alias `showenv`.

Ce sont les variables d'environnement de tmux : il y a aussi les variables d'environnement du système, qui sont copiées par tmux au démarrage d'une session et qui sont fusionnées avec les variables d'environnement de tmux, celles-ci ayant la priorité si une variable existe dans les deux environnements.

## Paramétrer une variable d'environnement pour la session

Pour modifier ou ajouter une variable d'environnement à la session, utilisez la commande `set-environnement NOM_VAR valeur` ou son alias `setenv NOM_VAR valeur`.

# Trucs et astuces

Il est un certain nombre de logiciels qu'un administrateur systèmes rencontrera au cours du temps. Ceux-ci lui permettront d'acquérir des automatismes et de gagner un temps considérable dans l'accomplissement de ses tâches.

Page aussi disponible sur <https://luc.frama.io/cours-asrall/tips/>.

## Éditer un fichier

Que votre éditeur favori soit *vim*, *nano*, *emacs* ou autre, il faut vous assurer :

- de connaître ses commandes (aller à la ligne X, faire un chercher/remplacer, réindenter)
- de savoir activer la coloration syntaxique. Ça paraît superfétatoire, mais c'est essentiel car cela vous permet de vous repérer plus rapidement dans le fichier voire de détecter des erreurs de syntaxe.

Il est nécessaire de connaître son éditeur pour être efficace.

Pour *vim*, vous pouvez lancer *vimtutor* qui vous fera passer par des exercices pour vous familiariser avec *vim*. Le site [VimCasts](#) regorge de tutoriaux et d'astuces.

**Attention** : quand bien même vous ne choisiriez pas *vim*, il vous faut en connaître les commandes de base. En effet, *emacs* n'est que rarement installé sur un serveur, et *nano* est parfois (souvent ?) trop limité pour travailler vite et bien.

## Les processus

### htop

*htop* permet de lister les processus, rechercher un processus, tuer des processus, trier les processus selon différents critères...

Il affiche également des informations sur le système : occupation mémoire, utilisation des processeurs, charge du système, etc.



Pour n'afficher que les processus de l'utilisateur `foo` :

```
htop -u foo
```

Voir <https://peteris.rocks/blog/htop/> pour comprendre les informations fournies par *htop*.

[Carl Chenet](#) a traduit ces articles en français dans une série d'article disponible sur <https://carlchenet.com/category/htop-explique/>.

## kill

La commande *kill* permet d'envoyer un signal à un processus. On peut indifféremment utiliser le n° ou le nom d'un signal pour l'utiliser. Ainsi `kill -9 <PID>` est normalement équivalent à `kill -KILL <PID>`.

Pour être bien certain du signal envoyé, il est préférable d'utiliser son nom : tous les signaux n'ont pas un n° attribué de façon certaine.

Voir [https://en.wikipedia.org/wiki/Unix\\_signal#POSIX\\_signals](https://en.wikipedia.org/wiki/Unix_signal#POSIX_signals) pour la liste des signaux POSIX.

## killall

*killall* est le petit frère de *kill*. Il permet d'envoyer des signaux aux processus sans connaître leur PID, juste avec leur nom.

Comme *killall* peut ratisser large, il vaut mieux lui préférer le couple *pgrep* / *pkill*.

## pgrep / pkill

*pgrep* permet de rechercher parmi les processus, *pkill* permet d'envoyer un signal aux processus avec la même syntaxe de recherche que *pgrep*.

Rechercher un processus par son nom :

```
pgrep nom
```

Rechercher un processus par l'intégralité de sa ligne de commande :

```
pgrep -f nom
```

Rechercher un processus par son nom, appartenant à l'utilisateur `foo` :

```
pgrep -u foo nom
```

Afficher le nom du processus en plus de son PID :

```
pgrep -l nom
```

Afficher la ligne de commande complète en plus de son PID :

```
pgrep -a nom
```

Envoyer le signal SIGTERM aux processus correspondants à la recherche :

```
pkill SIGTERM nom
```

## Isof

*Isof* permet de connaître le ou les processus utilisant une ressource.

Qui utilise `/home/foo` ?

```
Isof /home/foo
```

Qui utilise `/dev/sda` ?

```
Isof /dev/sda
```

Qui utilise le port 80 ?

```
Isof -i :80
```

## Les logs

### multitail

*multitail* permet de surveiller en temps réel les modifications d'un ou plusieurs fichiers à la manière d'un `tail -f` mais est bien plus souple d'usage.

Lire plusieurs fichiers :

```
multitail mail.log kern.log
```

Filtrer les lignes affichées d'un fichier selon une regex :

```
multitail -e regex mail.log kern.log
```

Filtrer les lignes affichées de *tous* les fichiers selon une regex :

```
multitail -E regex mail.log kern.log
```

Pour les données depuis l'entrée standard :

```
commande_qui_fait_des_logs | multitail -j
```

Une fois *multitail* lancé, un grand nombre de raccourcis claviers permet de le manipuler :

- **Entrée** : Affiche une ligne rouge avec l'heure et la date sur chaque fenêtre d'affichage de fichier (utile pour se donner un repère avant un test générant des logs)
- **O** (la lettre o en majuscule) : Efface l'affichage de toutes les fenêtres
- **/** : Effectue une recherche dans toutes les fenêtres
- **b** : Permet de revenir en arrière sur une fenêtre
- **F1** : affiche l'aide, avec tous les raccourcis claviers

## goaccess

*goaccess* va analyser en temps réel les logs d'un serveur pour fournir des statistiques.

On pourra alors voir rapidement quelle est l'adresse IP qui se connecte le plus, quelle est la page la plus visitée, etc.

## Veille technologique

Non, passer du temps sur [LinuxFR](#) ou sur le [Journal du hacker](#) n'est pas du temps perdu, quoi qu'on en dise. Il est en effet important d'effectuer une veille technologique régulière afin de découvrir de nouvelles technologies, de nouvelles astuces ou d'être averti de nouvelles failles de sécurité.

Votre meilleur ami pour cette veille sera un lecteur de flux RSS. En effet, un lecteur de flux a cet immense avantage sur les réseaux sociaux d'être asynchrone : partez en vacances deux semaines, revenez, et lisez tout ce que vous avez loupé (essayez un peu de faire cela avec Twitter : impossible). Vous pouvez aussi généralement le configurer pour qu'il vous envoie un résumé par mail de vos flux RSS... parfait quand on le couple à la liste de discussion des autres administrateurs

systèmes !

Attention : les réseaux sociaux comme Twitter peuvent aussi être utiles, de par leur propension à propager (très) rapidement l'information. Le revers de la médaille est qu'il faudra bien vérifier la véracité de la-dite information.

# SSH

## Concierge

SSH fonctionne bien de base, mais avoir un fichier de configuration SSH améliore grandement les choses.

Exemple : votre identifiant sur votre machine locale est *rim*, mais *rimd* sur la machine *mavrick.chatons.org*. Pour vous connecter, vous lancez la commande `ssh rimd@mavrick.chatons.org`

Avec un fichier de configuration ssh (*~/.ssh/config*) contenant

```
Host mavrick
  HostName mavrick.chatons.org
  User rimd
```

vous pourrez vous connecter avec un simple `ssh mavrick`,

Avec quelques serveurs, la gestion de ce fichier ne pose pas de problème, mais on s'aperçoit, au fur et à mesure que l'on a plus de serveurs à gérer que cela devient une plaie. C'est là qu'intervient *concierge*.

*concierge* permet de gérer son fichier de configuration avec un langage de *template*.

On pourra donc écrire

```
{% for i in ('dorone', 'khais') %}
Host {{i}}
  HostName {{i}}.chatons.org
  User rimd
  IdentitiesOnly yes
  IdentityFile /home/%u/.ssh/id_chatons
{% endfor %}
```

```
{% for i in ('gohan', 'diren') %}  
Host {{i}}  
    HostName {{i}}.perso.org  
    User rim  
    IdentitiesOnly yes  
    IdentityFile /home/%u/.ssh/id_perso  
{% endfor %}
```

Ce qui créera des entrées dans le fichier de configuration SSH pour les serveurs *dorone*, *khais*, *gohan* et *diren*.

Voir <https://github.com/9seconds/concierge> pour l'installation de *concierge*.

## Mssh

*mssh*, disponible habituellement dans les dépôts de votre distribution préférée, vous permettra de lancer plusieurs connexions SSH en même temps. La fenêtre contiendra autant de terminaux que de connexions SSH. Les commandes tapées seront envoyées à tous les terminaux en même temps (il est possible de n'envoyer la commande que sur un seul serveur ou de "désactiver" certains serveurs pour que les commandes ne leur soient pas envoyées).

*mssh* est très utile pour effectuer des tâches simultanément.

On lance *mssh* ainsi : `mssh gohan diren`

## Confort visuel

### redshift

*redshift*, lui aussi généralement dans les dépôts, ajuste la température de votre écran en fonction de l'heure. L'idée est de rougir graduellement l'écran afin d'éviter la fatigue visuelle due à la lumière bleue de votre écran.

## Confort dans le terminal

### bash-completion

Activer l'utilisation d'une complétion avancée des commandes se fait dans Debian en décommentant les lignes suivantes du fichier */etc/bash.bashrc* :

```
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
```

Cela permettra, par exemple, de compléter les options d'un logiciel, le nom d'un paquet à installer, etc. Sans cela, vous n'aurez que la complétion du logiciel que vous voulez utiliser et des chemins de votre système de fichiers.

## tree

*tree* affichera l'ensemble d'une arborescence... sous forme arborescente. Ce qui permet de parcourir un dossier très vite.

```
% tree foo
foo
├── bar
│   └── baz.txt

1 directory, 1 file
```

## Sécurité

### mkpasswd.pl

Fourni par le paquet *libstring-mkpasswd-perl* dans Debian, *mkpasswd.pl* permet de générer des mots de passe aléatoires, éventuellement en forçant quelques paramètres.

```
% mkpasswd.pl -l 20 -s 4
kta*vvN:g7bxM/se8a-b
```

- `-l 20` : 20 caractères

- `-s 4` : avec 4 caractères spéciaux (ponctuation, pourcent, etc)

# Manipulation de données

## Sort

Le vénérable `sort` permet de trier des données.

```
sort < fichier.txt
```

Parmi les options intéressantes :

- `-u` supprime les doublons
- `-n` fait un tri numérique, c'est-à-dire que `2` vient avant `10`, là une machine dira que `10` vient avant `2` car le premier caractère `1` vient avant `2`
- `-h` fait un tri « humain » des nombres, c'est-à-dire qu'il va lire `2K` comme `2000`, et le triera après `1G`
- `-r` trie en sens inverse
- `-k` trie selon une clé. Par exemple, avec `foo bar`, `bar` est la clé n°2
- `-t` spécifie le séparateur des clés. Par défaut, la séparation est définie par une transition entre caractère vide (espace, tabulation, etc.) en non-vide

## Jq

Le logiciel `jq` permet de manipuler du JSON directement depuis la ligne de commande.

J'avoue, je l'utilise surtout pour récupérer juste l'info qu'il me faut, je modifie très rarement du JSON avec (mais on peut !)

```
jq '.job.status' < foo.json
```

## Yq

Le logiciel `yq` est au YAML ce que `jq` est au JSON. D'ailleurs, `yq` se sert de `jq` en interne.

```
yq '.job.status' < foo.yml
```

Note : `yq` n'aime pas les clés avec un `-`. Il faut alors utiliser une autre syntaxe que la classique `.key` :

```
yq '.job["foo-bar"]' < foo.yml
```

# Divers

## ncdu

*ncdu* va regarder la taille du répertoire ciblé (celui où on se trouve par défaut) et afficher les fichiers/dossiers contenus, triés par taille. Très utile pour trouver ce qui bouffe de l'espace disque.

Petit bonus : pour avoir un *ncdu* sur un *bucket* S3, on peut utiliser le logiciel *rclone* ainsi :

```
rclone ncdu remote:path
```

## watch

*watch* permet de lancer une commande à intervalle régulier. Après une modification DNS, *watch dig chatons.org* pourra par exemple vous permettre de surveiller la prise en compte de cette modification sur votre résolveur.

## truncate

*truncate* permet de réduire ou étendre la taille d'un fichier à la taille indiquée.

```
truncate -s 1M fichier_trop_gros
```

## split

*split* permet de découper un fichier en plusieurs parties.

## wall

*wall* permet d'envoyer un message à tous les utilisateurs connectés à la machine.





# Supprimer une clé SSH stockée par gpg-agent

Gpg-agent peut servir d'agent SSH (il va donc conserver les clés en mémoire) mais si l'ajout d'une clé se fait bien classiquement avec `ssh-add ...`, la suppression d'une clé est plus complexe, `ssh-agent -d ...` ne fonctionnant pas.

## Repérer la somme MD5 de la clé à supprimer

Pour afficher les sommes MD5 des clés stockées par l'agent :

```
ssh-add -l -E md5
```

Ça donne un truc comme :

```
256 MD5:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa foo@bar (ED25519)
256 MD5:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb baz@qux (ED25519)
```

Si malheureusement vous n'avez pas spécifié le commentaire lors de la création de votre clé SSH, il y a des chances que vous vous retrouviez avec un truc comme :

```
256 MD5:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa foo@bar (ED25519)
256 MD5:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb foo@bar (ED25519)
```

Regardez bien la fin des lignes : vous ne savez pas quelle clé vous souhaitez supprimer !

Pour retrouver quelle somme MD5 correspond à quelle clé :

```
ssh-add -L
```

Ce qui affiche les clés publiques des clés SSH stockées, à savoir un truc comme :

```
ssh-ed25519 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
foo@bar  
ssh-ed25519 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY foo@bar
```

Un coup de `grep -RF XX`  
`~/ssh` vous permettra d'identifier les clés. L'ordre entre `ssh-add -l -E md5` et `ssh-add -l` reste le  
 même (heureusement).

# Afficher le grip des clés stockées par gpg-agent

```
gpg-connect-agent 'KEYINFO --ssh-list --ssh-fpr' /bye
```

Ce qui donne un truc comme :

```
S KEYINFO VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV D - - P
MD5:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa - S
S KEYINFO ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ D - - P
MD5:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb:bb - S
```

Le texte après le `KEYINFO` est le grip de la clé, qui va nous servir pour supprimer la clé :

```
gpg-connect-agent 'DELETE_KEY ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ' /bye
```

Et c'est fini 

# Swap

Le swap est un espace d'échange qui recueille des données normalement en RAM lorsque l'utilisation de celle-ci dépasse un certain point.

## Gérer les espaces d'échange

### Voir l'utilisation des espaces d'échanges

```
cat /proc/swaps
```

Cela va donner quelque chose comme :

Filename	Type	Size	Used	Priority
/dev/dm-3	partition	3911676	3906776	-2
/var/swap	file	5242876	310324	-3

## Monter et démonter un espace d'échange

Les termes de montage/démontage ne sont pas corrects car les espaces d'échange ne sont pas montés sur le système. Vous ne les verrez pas avec la commande `mount`.

Pour ne plus utiliser un espace d'échange :

```
swapoff /dev/dm-3
```

Cette commande peut prendre un peu de temps car le contenu de l'espace d'échange va être déplacé dans la RAM ou dans un autre espace d'échange, ou oublié par le système s'il n'y a plus de place disponible.

Pour le réutiliser :

```
swapon /dev/dm-3
```

L'option `-a` permet d'agir sur tous les espaces d'échanges connus du système (dans `/etc/fstab` la plupart du temps. Systemd a un truc pour ça aussi, mais je ne l'ai encore jamais rencontré).

Exemple :

```
swapoff -a  
swapon -a
```

## Modifier le recours aux espaces d'échange

Les espaces d'échanges vont être utilisés avec plus ou moins d'agressivité selon la valeur de `vm.swappiness` de votre système (pour voir cette valeur : `sysctl vm.swappiness`). Cette valeur peut être comprise entre 0 et 100.

Un nombre élevé veut dire que le noyau va avoir plus tendance à décharger la RAM dans les espaces d'échanges que dans un système avec un nombre bas.

Pour modifier temporairement la valeur :

```
sysctl -w vm.swappiness=10
```

Pour la modifier de façon permanente :

```
echo "vm.swappiness = 10" > /etc/sysctl.d/99-swappiness.conf  
sysctl -p /etc/sysctl.d/99-swappiness.conf
```

(le `sysctl -p` est là pour appliquer la valeur que vous venez de mettre dans le fichier)

## Les différents supports d'espaces d'échanges

On peut avoir du *swap* avec une partition comme le propose Debian lors de l'installation ou via un fichier *swap*, comme le fait Ubuntu. On peut aussi avoir du *swap*... sur la RAM ! (voir plus bas)

L'avantage du fichier sur la partition est sa manipulation plus facile. Je pense en particulier à la modification de la taille du *swap*.

## Créer un fichier *swap*

C'est excessivement simple : on crée un fichier, on le prépare comme il faut, on le déclare dans `/etc/fstab` et on l'utilise.

```
fallocate -l 2G /var/swap  
mkswap /var/swap  
chmod 600 /var/swap  
echo "/var/swap none swap sw 0 0" >> /etc/fstab  
swapon /var/swap
```

# Utiliser de la RAM pour l'espace d'échange

Cela paraît contre-intuitif, mais c'est très simple : l'espace d'échange sera compressé et conservé en RAM. Le coût en performances de la compression/décompression des données est, avec nos processeurs actuels, généralement moindre que celui de l'utilisation d'un disque, fut-il SSD : la RAM permet des accès beaucoup, beaucoup plus rapides que n'importe quel disque.

Comme les espaces d'échanges sont utilisés comme de la RAM supplémentaire, mais lente, avoir ceux-ci sur la RAM, mais compressés équivaut plus ou moins à une augmentation de taille de RAM au prix de quelques cycles CPU.

Pour utiliser ce mécanisme, il suffit, sur Debian, d'installer le paquet `zram-tools`, de modifier `/etc/default/zramswap` à son goût et de relancer le service `zramswap`.

# Systemd

## Créer un service utilisateur

Mettre le service dans le dossier `~/.config/systemd/user/` puis :

```
systemctl --user daemon-reload  
systemctl --user enable --now monscript.service
```

## Manipuler un service utilisateur depuis le compte root

```
systemctl --user --machine=le_user@ stop monscript.service
```