LVM: Utilisation avancée

Table of Contents

LVM est un outil puissant de gestion des volumes logiques. Cela remplace, d'une certaine façon, le partitionnement des disques.

Il faut bien différencier

- **le volume physique (PV)** qui est la partition où on va installer nos volumes, par exemple /dev/sda
- **le groupe de volume (VG)** qui va être un contenant des volumes proprement dit
- **le volume logique (LV)** qui est un emplacement sur la partition, à l'intérieur d'un VG.

Les Volumes logiques LVM

Dans LVM, un groupe de volumes est divisé en volumes logiques. Il existe trois types de volumes logiques LVM: les volumes linéaires, les volumes agrégés par bandes et les volumes mis en miroir.

Volumes linéaires

Un volume linéaire agrège l'espace d'un ou de plusieurs volumes physiques en un volume logique, par exemple, si on dispose de deux disques de 60 Go, on peut créer un volume logique de 120 Go. Le stockage physique est concaténé.

La création d'un volume linéaire affecte une plage d'extents physiques à une zone d'un volume logique dans l'ordre, par exemple: les extents logiques 1 à 99 peuvent correspondre à un volume physique et les extents logiques 100 à 198 peuvent correspondre à un un deuxième volume physique. Du point de vue de l'application, il existe un périphérique d'une taille de 198 extents.

Les volumes physiques qui constituent un volume logique ne doivent pas obligatoirement avoir la même taille. Par exemple un groupe de volumes VG1 composé de deux volumes physiques nommés PV1 Et PV2 (800Mo + 400mo) avec une taille d'extent physique de 4 Mo.

Les volumes physiques sont divisés en unités de 4 Mo, car il s’agit de la taille de l’extent. PV1 a une taille de 200 extents (800 Mo) et PV2 une taille de 100 extents (400 Mo). On peut donc créer un volume linéaire ayant une taille comprise entre 1 et 300 extent (4 Mo à 1200 Mo).

On peut configurer plusieurs volumes logiques linéaires de la taille souhaitée dans le pool d’extensions physiques. Par exemple dans on peut tailler dans le volume groupe VG1 deux volumes logiques: LV1, d’une taille de 250 extents (1000 Mo) et LV2, d’une taille de 50 extensions (200 Mo).

Volumes logiques en mode stripe

Lorsqu'on écrit des données sur un volume logique LVM, le système de fichiers les répartit sur les volumes physiques sous-jacents. On peut contrôler la façon dont les données sont écrites sur les volumes physiques en créant un volume logique agrégé par bandes. Cela peut améliorer l'efficacité des E/S de données.

Le striping améliore les performances en écrivant des données dans un nombre prédéterminé de

volumes physiques en alternance. Avec le striping, les entrées/sorties peuvent être effectuées en parallèle. Dans certains cas, cela peut entraîner un gain de performance quasi linéaire pour chaque volume physique supplémentaire.

Dans un volume logique agrégé par bandes, la taille de la bande ne peut pas dépasser la taille d'un extent.

Il doit y avoir suffisamment d'espace sur les volumes physiques sous-jacents constituant le groupe de volumes pour prendre en charge les volumes logiques agrégés par bande. Par exemple, si on a une bande bidirectionnelle qui utilise tout un groupe de volumes, l'ajout d'un seul volume physique au groupe de volumes ne permettra pas d'étendre la bande. Il faudra ajouter au moins deux volumes physiques.

Volumes logiques en miroir

Lorsque les données sont écrites sur un périphérique, elles le sont également sur un deuxième périphérique, ce qui les met en miroir, ce qui protège les pannes de périphérique. Le volume devient un volume linéaire et peut toujours être consulté.

LVM prend en charge les volumes en miroir. Lorsqu'on créer un volume logique en miroir, LVM garantit que les données écrites sur un volume physique sous-jacent sont mises en miroir sur un volume physique distinct. Avec LVM, on peut créer des volumes logiques en miroir avec plusieurs miroirs.

Création de volumes

les commandes doivent être passées en tant que root:

```
sudo -i
```

le prompt est maintenant en mode root « # », cela évitera d'avoir à préfixer chaque commande par sudo.

Les commandes LVM sont extrêmement simples à utiliser, et elles intègrent toutes une aide en ligne très bien conçue, claire, courte, mais suffisante. De plus, leurs noms se « devinent » assez facilement :

- toutes les commandes agissant sur les volumes physiques commencent par **pv** (pour physical volume);

- toutes les commandes agissant sur les groupes de volumes commencent par **vg** (pour volumes group);
- toutes les commandes agissant sur les volumes logiques commencent par **lv** (pour logical volume).

Création des disques

La première chose à faire est de créer et initialiser les disques.Lorsqu'il s'agit de disques de plus de 2 To, il faut du créer une table de partition GPT et pas MBR, car MBR se limite a des partition de 2 To maximum. On va donc utiliser l'utilitaire gdisk et non fdisk (ils s'utilisent pareil).

```
root@linux:~# gdisk /dev/sdc
GPT fdisk (gdisk) version 0.8.10

Partition table scan:
MBR: protective
BSD: not present
APM: not present
GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help):
```

- Tapez **n** pour créer une nouvelle partition
- Choisir le numéro de la partition (par défaut 1, c'est très bien)
- Choisir le secteur de départ (laisser le minimum par défaut)
- Choisir le dernier secteur (laisser par défaut pour utiliser le disque entier)
- Sélectionner le type **Linux LVM** en tappant **8e00** quand on vous demande le **Hex code**
- Taper **w** pour écrire les changements

Créer un volume physique

Créer un volume physique, en attribuant une partition à LVM.

```
pvcreate /dev/sdc1
```

Créer un Groupe de volumes

Il existe de nombreuses options lors de la création d'un groupe de volumes… Mais continuons de faire au plus simple. Le seul paramètre indispensable sera de lui donner un nom, nous utiliserons les valeurs par défaut pour tout le reste. Pour une raison que j'expliquerai par la suite, donnons-lui un nom très court (2 ou 3 caractères). Par exemple : « mvg » pour « mon vg ».

La syntaxe **vgcreate** est:

```
vgcreate VolumeGroupName PhysicalVolume [optionnellement d'autres
PhysicalVolume]
```

Par exemple :

```
vgcreate mvg /dev/sdc1
```

Si tout se passe bien, on a maintenant un groupe de volumes, contenant un disque physique. On peut obtenir d'autres informations sur ce groupe de volumes en tapant **vgdisplay**:

```
vgdisplay
--- Volume group ---
VG Name               mvg
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  3
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                2
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               186,31 GiB
PE Size               4,00 MiB
Total PE              47695
Alloc PE / Size       15360 / 60,00 GiB
Free  PE / Size       32335 / 126,31 GiB
VG UUID               BaTuai-1I8o-3rkY-Ut1r-ybta-mJnl-9X0oNZ
```

Créer les volumes logiques

On va créer deux espaces que l'on pourra ensuite « formater » en ext4 par exemple.

Les options vraiment importantes sont **-n** pour son nom, et **-L** pour sa taille et le groupe de volumes dans lequel nous allons créer le volume logique. Par exemple pour faire deux volumes, 10 Gio et 50 Gio :

```
lvcreate -n Vol1 -L 10g mvg
lvcreate -n Vol2 -L 50g mvg
```

On peut vérifier avec la commande **lvdisplay**:

```
~# lvdisplay
--- Logical volume ---
LV Name                /dev/mvg/Vol1
VG Name                mvg
LV UUID                q0D6cQ-mcMP-q8sf-XTlI-DdxX-QHd1-qkaB5J
LV Write Access        read/write
LV Status              available
# open                 0
LV Size                10,00 GiB
Current LE             2560
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           252:0

--- Logical volume ---
LV Name                /dev/mvg/Vol2
VG Name                mvg
LV UUID                JZjMxI-cTAw-cbs6-02BM-4Mev-P2E7-b8JX0x
LV Write Access        read/write
LV Status              available
# open                 0
LV Size                50,00 GiB
Current LE             12800
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           252:1
```

Création du système de fichiers

Avec les partitions, on avait des noms ressemblant à `/dev/sda3`, etc. Avec LVM, on utilise aussi des périphériques dans /dev, mais le chemin est de la forme `/dev/nom_du_vg/nom_du_lv`. Autrement dit, puisqu'on a décidé d'appeler nos volumes logiques “Vol1” et “Vol2”, les noms de ces périphériques de ce volume logique sont `/dev/mvg/Vol1` et `/dev/mvg/Vol2`. À partir de maintenant, `/dev/mvg/Volx` peut être utilisé dans toutes les situations et avec toutes les commandes qui attendent quelque chose de la forme `/dev/…` Par exemple :

```
mkfs -t ext4 /dev/mvg/Vol1
mkfs -t ext4 /dev/mvg/Vol2
mkdir /Essai1
mount /dev/mvg/Vol1 /Essai1
df -h
```

Et normalement, /dev/mvg/Vol1 devrait être monté sur /Essai. Regardez bien la ligne correspondante. Si on avait choisi un nom de VG ou de LV plus long, la sortie de df aurait été modifiée, car le nom

aurait « touché » les valeurs... On aurait été obligé de passer des lignes et l'affichage aurait été plus difficile à lire. Techniquement, choisir des noms « longs » pour les VG et les LV ne pose aucun problème, mais c'est l'affichage qui sera parfois délicat. Pour cette raison uniquement, il est préférable de se limiter à 7 caractères au total (donc par exemple 3 pour le VG et 4 pour le LV, ou 2 et 5).

Pourquoi est-il écrit `/dev/mapper/mvg-Vol1` et non `/dev/mvg/Vol1` ?

Avec LVM en version 1, c'est bien `/dev/mvg/Vol1` qui aurait été affiché. Depuis la version 2, LVM utilise le périphérique `mapper`, ce qui permet pas mal de choses (comme chiffrer les volumes logiques, etc.). Pour simplifier, disons que ces deux notations « `/dev/mvg/Vol1` » et «`/dev/mapper/mvg-Vol1`» sont synonymes. Dans la pratique, il est conseillé quand même d'utiliser plutôt la forme «`/dev/mvg/Vol1`», certaines commandes ne passeront pas autrement.

Suppression

Rien de plus simple :

```
umount /Essai1  # si le volume Vol1 est monté en /Essai1
lvremove /dev/mvg/Vol1
```

une fois un volume logique effacé, il est totalement impossible de récupérer les données qu'il contenait.

Redimensionnement

Volume physique

Imaginons maintenant que le groupe de volume (mvg) n'ait plus suffisamment d'espace libre. On souhaite donc lui rajouter un volume physique afin de rajouter de l'espace libre. Ça tombe bien, on dispose d'un volume physique sdc2 que l'on va pouvoir ajouter à mvg :

On initialise le volume en vue de son utilisation dans LVM :

```
pvcreate /dev/sdc2
```

On rajoute le volume sdc2 au groupe de volume mvg :

```
vgextend mvg /dev/sdc2
```

Volume logique

Il est très facile d'augmenter ou de diminuer la taille d'un volume logique. La taille d'un LV n'a pas de lien direct avec la taille de ce qu'il contient (swap ou système de fichier). Le LV est une boîte, le système de fichier est le contenu de la boîte. Augmenter la taille de la boîte sans augmenter la taille du contenu ne pose pas de problème, mais l'inverse...

Agrandissement

Arès avoir étendu le volume group (VG), on peut étendre le volume logique, mais pour cela il faut l'identifier.

La commande **lvs** ou **lvdisplay** affiche le volume logique associé à un groupe de volumes.

```
 lvs
  LV           VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log
Cpy%Sync Convert
  root         rl   -wi-ao---- 366,99g
  swap         rl   -wi-ao----   4,00g
  vol_backups  vg00 -wi-ao----   1,42t
  vol_data     vg00 -wi-ao----  <3,44t
  vol_partimag vg00 -wi-ao---- <50,30g
```

Méthode 1 lvresize

Pour agrandir un volume il est nécessaire de démonter le système de fichier :

```
umount /Essai2
```

Maintenant pour ajouter 5Gio au volume et agrandir le système de fichier :

```
lvresize --resizefs --size +5G /dev/rl/root
```

Le paramètre `--resizefs` ne fonctionne pas avec tous les systèmes de fichiers.

Une fois l'opération terminée, le volume une fois monté a gagné 5Gio.

Méthode 2 lvextend

Bien qu'il soit évidemment moins risqué d'agrandir ou de diminuer la taille d'un système de fichiers après l'avoir démonté, la plupart des formats (ext3, reisersfs, ext4…) supportent désormais cette modification “à chaud” (avec des données qui restent donc accessibles en lecture/écriture durant toute l'opération).

Etendre la taille du volume logique:

```
lvextend -l +100%FREE /dev/mvg/Vol2
```

Étendre le système de fichiers

Il faut confirmer le type de système de fichiers utilisé. On peut vérifier le système de fichiers avec `lsblk -f` ou `df -Th`.

Redimensionner le système de fichiers sur le volume logique après son extension pour afficher les modifications. Par exemple pour redimensionner le système de fichiers XFS à l'aide de la commande xfs_growfs.

```
xfs_growfs /dev/rl/root
```

Rétrécissement

Diminuer la taille d'un système de fichier est un peu plus délicat. En effet, il faut dans un premier temps s'assurer de pouvoir réduire d'autant qu'on le souhaite.

Tous les systèmes de fichiers ne supportent pas d'être redimensionnés

D'abord vérifier l'espace du système de fichier :

```
df -h | grep ca
/dev/mapper/svg-ca    512M  230M  283M  45% /home/ca
```

Les valeurs qui nous intéresses sont la deuxième et la quatrième, à savoir :

- 512Mio d'espace total
- 283Mio d'espace libre

L'espace libre étant de 283Mio, on peut réduire l'espace de 256Mio.

démonter le volume :

```

```

```
umount /dev/mapper/svg-ca
```

retirer 256Mio :

```
lvresize --resizefs --size -256M /dev/mapper/svg-ca
```

Si la partition n'est pas démonté, la commande propose de la démonter et s'occupera de la remonter une fois le redimensionnement terminé.

Le volume peut maintenant être monté :

```
mount /dev/mapper/svg-ca /home/ca
```

Et on peut alors afficher sa nouvelle taille :

```
df -h | grep ca
/dev/mapper/svg-ca    256M  230M   27M  90% /home/ca
```

Rétrécissement lorsque le disque est chiffré

Lorsque le disque est chiffré le principe retenu est de diminuer la taille du répertoire ROOT tel que décrit au-dessus, de laisser le répertoire swap en état et d'utiliser gparted pour faire le rétrécissement physique afin de libérer un espace disque destiné à devenir une nouvelle partition ou être ajouté à la partition de boot existante. Tout ce traitement se fera quasiment en ligne de commande et surtout à partir du support d'installation car certaines séquences obligent à démonter la partition.

Récupérer les informations

- Récupérerer les informations de la partition physique avec la commande

```
sudo pvdisplay
```

qui indiquera aussi la valeur de l'unité d'allocation de l'espace logique. Il y a toutes les chances qu'elle soit de 4 Mo. Dans ce chapitre, il est nommé sda35 car c'est une fabrication et pas le vrai volume.

- Récupérer le nom du conteneur avec la commande

```
sudo vgdisplay | grep Name
```

- Récupérer la liste des volumes logiques contenus dans la partition est disponible avec la commande suivante

```

```

```
sudo lvdisplay | grep "LV Name"
```

On y trouvera certainement la partition root ainsi que la partition swap.

Pour libérer de l'espace dans la partition physique, il faut d'abord qu'il y ait de l'espace libre dans les partitions logiques. S'il n'y en a pas suffisamment, cela doit se faire en amont avec l'instance UBUNTU opérationnelle. Normalement l'espace à libérer se trouve dans la structure root.

Libération de l'espace dans la partition logique.

Cette opération peut inquiéter mais elle est sans risque car le logiciel contrôle que la valeur ne dépasse pas le maxima de l'espace alloué pour la partition et n'est jamais inférieure au minima nécessaire déjà utilisé par le stockage des données.

Elle peut être lancée plusieurs fois avec un paramétrage pouvant dire en valeur absolue ou en valeur relative.

Il est possible de faire un agrandissement avec une partition utilisée. Mais il faut obligatoirement que la partition soit démontée pour une diminution de taille. Cela nécessite d'opérer à partir d'un support d'installation.

Il est nécessaire de commencer par un contrôle de la qualité de la partition

```
sudo e2fsck  -f /dev/ubuntu-vg/root
```

Puis rétrécir avec cette commande

```
sudo resize2fs -p    /dev/ubuntu-vg/root  1g
            resize2fs 1.44.1 (24-Mar-2018)
            resize2fs: La nouvelle taille est plus petite que le minimum
(1234929)
```

Il reste alors à faire le calcul 1234929*4/1024 et lancer la nouvelle commande

```
sudo resize2fs -p    /dev/ubuntu-vg/root 4824g
            resize2fs 1.44.1 (24-Mar-2018)
            La partition (ou le périphérique) contenante n'a que 1234944
(4k) blocs.
            Vous avez demandé une nouvelle taille de 1264582656 blocs.
```

Sans se tromper d'unités. On voit alors le recadrage s'exécuter et les X s'ajouter au fur et à mesure

```
      sudo resize2fs -p    /dev/ubuntu-vg/root  4824m
            resize2fs 1.44.1 (24-Mar-2018)
            En train de redimensionner le système de fichiers sur
/dev/ubuntu-vg/root à 1234944 (4k) blocs.
            Début de la passe 2 (max = 336431)
```

```
            Relocalisation de blocs
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            Début de la passe 3 (max = 48)
            Examen de la table d'i-noeuds
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            Le système de fichiers sur /dev/ubuntu-vg/root a maintenant une
taille de 1234944 blocs (4k).
```

Dans cet exemple, le système de fichier a été rétréci au maxima. En pratique, il faut beaucoup moins rétrécir. Lorsque ce rétrécissement est fait, il ne faut surtout pas oublier d'en informer l'enveloppe logique en mettant de préférence la même valeur.

```
    sudo lvresize  -v    /dev/ubuntu-vg/root --resizefs --size 4824m
            Executing: /sbin/fsadm --verbose check /dev/ubuntu-vg/root
            fsadm: "ext4" filesystem found on "/dev/mapper/ubuntu--vg-root".
            fsadm: Skipping filesystem check for device "/dev/mapper/ubuntu-
-vg-root" as the filesystem is mounted on /mnt/ROOT
            /sbin/fsadm failed: 3
            Executing: /sbin/fsadm --verbose resize /dev/ubuntu-vg/root
4939776K
            fsadm: "ext4" filesystem found on "/dev/mapper/ubuntu--vg-root".
            fsadm: Device "/dev/mapper/ubuntu--vg-root" size is 6442450944
bytes
            fsadm: Parsing tune2fs -l "/dev/mapper/ubuntu--vg-root"
            fsadm: Resizing filesystem on device "/dev/mapper/ubuntu--vg-
root" to 5058330624 bytes (1234944 -> 1234944 blocks of 4096 bytes)
            fsadm: Executing resize2fs /dev/mapper/ubuntu--vg-root 1234944
            resize2fs 1.44.1 (24-Mar-2018)
            Le système de fichiers a déjà 1234944 blocs (4k). Rien à faire !
            Archiving volume group "ubuntu-vg" metadata (seqno 6).
            Reducing logical volume ubuntu-vg/root to 4,71 GiB
            Size of logical volume ubuntu-vg/root changed from 6,00 GiB
(1536 extents) to 4,71 GiB (1206 extents).
            Loading ubuntu--vg-root table (253:1)
            Suspending ubuntu--vg-root (253:1) with device flush
            Resuming ubuntu--vg-root (253:1)
            Creating volume group backup "/etc/lvm/backup/ubuntu-vg" (seqno
7).
            Logical volume ubuntu-vg/root successfully resized.
```

Dans cet exemple, on voit que la commande inclut la fonction de retaillage du système de fichier.

reboucher les trous

Si on lance **gparted** pour rétrécir la partition physique, Cela ne fonctionnera pas très bien sauf exception. Son message sera du style

```
 lvm pvresize -v --yes --setphysicalvolumesize 10711040K '/dev/sda35'
```

```
00:00:01    ( ERREUR )     |
       0 physical volume(s) resized / 1 physical volume(s) not resized
       Wiping internal VG cache
       Wiping cache of LVM-capable devices
       Archiving volume group "ubuntu-vg" metadata (seqno 9).
        WARNING: /dev/sda35: Pretending size is 21422080 not 31457280
sectors.
       Resizing volume "/dev/sda35" to 21422080 sectors.
       Resizing physical volume /dev/sda35 from 3839 to 2614 extents.
       /dev/sda35: cannot resize to 2614 extents as later ones are
allocated.
```

Pour éviter cette erreur, il est nécessaire de vérifier que toutes les données sont bien tassées dans la partition physique avec cette commande qui est la clé de voute de l'opération.

```
  sudo pvs -v --segments /dev/sda35
       Wiping internal VG cache
       Wiping cache of LVM-capable devices
       PV         VG        Fmt  Attr PSize   PFree  Start SSize LV
Start Type   PE Ranges
      /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g     0   385 swap_1     0
linear /dev/sda35:0-384
      /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g   385   895            0
free
      /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  1280  1206 root       0
linear /dev/sda35:1280-2485
      /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  2486   330            0
free
      /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  2816  1023 swap_2     0
linear /dev/sda35:2816-3838
```

On constate qu'il y a des trous au milieu. il n'en faut absolument pas. Il est nécessaire que toute la fin soit en état **free**. La commande suivante est à faire

```
 sudo pvmove -v  --alloc anywhere /dev/sda35:3327-3838
      Executing: /sbin/modprobe dm-mirror
      Cluster mirror log daemon is not running.
      Wiping internal VG cache
      Wiping cache of LVM-capable devices
      Archiving volume group "ubuntu-vg" metadata (seqno 9).
      Creating logical volume pvmove0
      Moving 1023 extents of logical volume ubuntu-vg/swap_2.
      activation/volume_list configuration setting not defined: Checking
only host tags for ubuntu-vg/swap_2.
      Creating ubuntu--vg-swap_2
      Loading ubuntu--vg-swap_2 table (253:0)
      Resuming ubuntu--vg-swap_2 (253:0)
      Setting up pvmove in on-disk volume group metadata.
      Creating ubuntu--vg-pvmove0
```

```
      Loading ubuntu--vg-pvmove0 table (253:1)
      Loading ubuntu--vg-swap_2 table (253:0)
      Suspending ubuntu--vg-swap_2 (253:0) with device flush
      activation/volume_list configuration setting not defined: Checking
only host tags for ubuntu-vg/pvmove0.
      Resuming ubuntu--vg-pvmove0 (253:1)
      Loading ubuntu--vg-pvmove0 table (253:1)
      Suppressed ubuntu--vg-pvmove0 (253:1) identical table reload.
      Resuming ubuntu--vg-swap_2 (253:0)
      Creating volume group backup "/etc/lvm/backup/ubuntu-vg" (seqno 10).
      Checking progress before waiting every 15 seconds.
      /dev/sda35: Moved: 0,10%
      /dev/sda35: Moved: 8,31%
      /dev/sda35: Moved: 16,42%
      /dev/sda35: Moved: 24,73%
      /dev/sda35: Moved: 32,94%
      /dev/sda35: Moved: 41,15%
      /dev/sda35: Moved: 49,46%
      /dev/sda35: Moved: 57,87%
      /dev/sda35: Moved: 66,18%
      /dev/sda35: Moved: 74,58%
      /dev/sda35: Moved: 82,99%
      /dev/sda35: Moved: 87,49%
      /dev/sda35: Moved: 95,60%
      /dev/sda35: Moved: 100,00%
      Polling finished successfully.
```

On vérifie quand même que tout est enfin correct

```
 sudo pvs -v --segments /dev/sda35
      Wiping internal VG cache
      Wiping cache of LVM-capable devices
      PV         VG        Fmt  Attr PSize   PFree  Start SSize LV     Start
Type   PE Ranges
     /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g     0   385 swap_1     0
linear /dev/sda35:0-384
     /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g   385   895 swap_2     0
linear /dev/sda35:385-1279
     /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  1280  1206 root       0
linear /dev/sda35:1280-2485
     /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  2486   128 swap_2   895
linear /dev/sda35:2486-2613
     /dev/sda35 ubuntu-vg lvm2 a--  <15,00g <4,79g  2614  1225            0
free
```

C'est bien le cas, on pourra continuer avec gparted.

Lorsqu'on libère très peu de place, par exemple 100 unités et que la dernière partition logique est de taille plus importante, dans ce cas, il ne faut prendre que les 100

dernières unités de la partition. La codification de la commande devient alors:

```
sudo pvmove -v --alloc anywhere /dev/sda35:3739-3838
```

Finalisation avec gparted

Il faut penser à démonter les partitions logiques au cas où elles auraient été montées pour vérifier quelques informations.

Il faut libérer la structure LVM que gparted gère mal. C'est à faire pour la totalité des partitions logiques de la partition physique. La déactivation est à faire avec ces commandes:

```
 sudo lvchange -v /dev/ubuntu-vg/root     --activate n
 sudo lvchange -v /dev/ubuntu-vg/swap_1   --activate n
 sudo lvchange -v /dev/ubuntu-vg/swap_2   --activate n
```

Gparted pourra alors modifier la taille de l'enveloppe physique. Il propose une valeur minima et une valeur maxima. Voici un exemple de réalisation:

```
  GParted 0.30.0 --enable-libparted-dmraid --enable-online-resize
  Libparted 3.2
  Réduire /dev/sda35 de 15.00 Gio à 10.21 Gio  00:00:36    ( SUCCÈS )  |
  calibrer /dev/sda35  00:00:12    ( SUCCÈS )   |
  chemin : /dev/sda35 (partition)
  début : 584894464
  fin : 616351743
  taille : 31457280 (15.00 Gio)
  vérifier le système de fichiers sur /dev/sda35 et corriger les problèmes
(si possible)  00:00:00    ( SUCCÈS ) |
  lvm pvck -v '/dev/sda35'  00:00:00    ( SUCCÈS )     |
  Found label on /dev/sda35, sector 1, type=LVM2 001
  Found text metadata area: offset=4096, size=1044480
  Scanning /dev/sda35
  Found LVM2 metadata record at offset=24064, size=2048, offset2=0 size2=0
  Found LVM2 metadata record at offset=21504, size=2560, offset2=0 size2=0
  Found LVM2 metadata record at offset=18944, size=2560, offset2=0 size2=0
  Found LVM2 metadata record at offset=12288, size=6656, offset2=0 size2=0
  Found LVM2 metadata record at offset=10752, size=1536, offset2=0 size2=0
  réduire le système de fichiers  00:00:01    ( SUCCÈS )   |
  lvm pvresize -v --yes --setphysicalvolumesize 10711040K '/dev/sda35'
00:00:01    ( SUCCÈS )    |
  Physical volume "/dev/sda35" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
  Wiping internal VG cache
  Wiping cache of LVM-capable devices
  Archiving volume group "ubuntu-vg" metadata (seqno 13).
  WARNING: /dev/sda35: Pretending size is 21422080 not 31457280 sectors.
```

```
  Resizing volume "/dev/sda35" to 21422080 sectors.
  Resizing physical volume /dev/sda35 from 3839 to 2614 extents.
  Updating physical volume "/dev/sda35"
  Creating volume group backup "/etc/lvm/backup/ubuntu-vg" (seqno 14).
  réduire la partition de 15.00 Gio à 10.21 Gio  00:00:23    ( SUCCÈS )  |
  ancien début : 584894464
  ancienne fin : 616351743
  ancienne taille : 31457280 (15.00 Gio)
  nouveau début : 584894464
  nouvelle fin : 606316543
  nouvelle taille : 21422080 (10.21 Gio)
  =======================================
```

Lorsque cette opération est faite, La partition peut être manipulée avec gparted comme toutes les autres.

Décalage vers la droite, vers la gauche, agrandissement et pourquoi pas un copier/coller (cette dernière action n'a pas été vérifiée.

Lorsque le travail de gparted est fini, il faut réactiver les volumes logiques avec ces commandes

```
sudo lvchange -v /dev/ubuntu-vg/root     --activate a
sudo lvchange -v /dev/ubuntu-vg/swap_1   --activate a
sudo lvchange -v /dev/ubuntu-vg/swap_2   --activate a
```

Snapshot

Avec LVM 1, les instantanés sont en lecture seule. Ils fonctionnent par l'utilisation d'une table d'exception qui trace les blocs modifiés : lorsqu'un bloc est modifié sur la source, il est d'abord copié dans l'instantané, marqué comme modifié dans la table d'exceptions et ensuite modifié sur le volume source avec les nouvelles données.

Avec LVM 2, les instantanés sont par défaut en lecture/écriture. Le fonctionnement est similaire aux instantanés en lecture seule avec la possibilité supplémentaire d'écrire sur l'instantané : le bloc est alors marqué comme utilisé dans la table d'exceptions et ne sera plus récupéré du volume source. Cela ouvre de nouvelles perspectives par rapport au fonctionnement en lecture seule de LVM 1. Par exemple, on peut faire l'instantané d'un volume, le monter et tester un programme expérimental qui modifie les fichiers dessus. Si le résultat n'est pas satisfaisant, on peut le démonter, le supprimer et remonter le système de fichiers originel à la place. C'est aussi utile pour créer des volumes utilisés avec Xen. Vous pouvez créer une image disque et en faire un instantané que vous pourrez modifier avec une instance spécifique de domU. Vous pourrez ensuite créer un autre instantané de l'image originale et le modifier avec une autre instance de domU. Comme les instantanés ne stockent que les blocs modifiés, la majeure partie du volume sera partagée entre les domUs.

Création d'un snapshot LVM

```
lvcreate -L 10g -s -n lv_test_20110617 /dev/vg_data/lv_test
```

Va créer un snapshot du LV “lv*test” à la taille de 10Go qui va avoir comme nom “lv*test_20110617”. La taille d'utilisation du snapshot évolue avec l'utilisation. Si ce snapshot se retrouve rempli à 100%, il devient alors inutilisable (état “INACTIVE”) mais pas d’inquiétude car il n'y a pas d’impact pour le LV d'origine.

Pourquoi donner une taille au snapshot ? Tout simplement parce que celui-ci est intelligent, donc il ne va pas copier l'intégralité du LV original. Au contraire, il ne va stocker que les différences. C'est pourquoi il est instantané et commence avec une occupation taille nulle. La commande lvdisplay permet de voir l'évolution de la taille.

Redimensionnement du snapshot

La taille du snapshot est trop petite et elle arrive bientôt à 100%, pourtantlorqu'on a encore besoin d'utiliser ce snap ? Il faut donc redimensionner ! Vérifier avec vgdisplay que le VG dispose encore d'assez d'espace libre (Free PE / Size) puis effectuer :

```
lvresize -L +3GB /dev/vg_data/lv_test_20110617
```

Va ajouter 3Go au snap `lv_test_20110617` qui est présent dans le VG `vg_data`.

Fusionner un snapshot

Le but ici est de fusionner un snapshot modifié vers le LV d'origine. Pour ainsi dire, “faire que les modifications apportées sur le snapshot se retrouvent sur le LV d'origine”.

```
lvconvert --merge /path/to/dev/snap
```

Il faut un kernel (>=2.6.33)

Changement d'un disque défectueux

Le disque /dev/sda présente des signes de faiblesse (signalés par SMART, par la présence de nombreux fichiers dans les dossiers “lost + found” des partitions). On veut le remplacer par un disque neuf, de taille plus importante (surtout pas plus petite !), qu'on a installé dans la machine (ou sur un port USB) et qui est reconnu par le système comme étant /dev/sdb.

Principe

Supposons que le disque initial (/dev/sda) ait été formaté ainsi :

1. `/dev/sda1` est une partition primaire, de type bootable, montée sur /boot.
2. `/dev/sda2` est une partition étendue, contenant la partition logique `/dev/sda5` de type lvm2.

On a besoin de copier `/dev/sda1` sur une partition `/dev/sdb1`, et `/dev/sda5` sur une partition `/dev/sdb5`.

utiliser l'outil GParted pour préparer le disque `/dev/sdb` et copier la partition de boot. Gparted ne gérant pas lvm2, il faut utiliser la ligne de commande pour la copier de `/dev/sda5.`

Avec GParted

Lancer Gparted (Système → Administration → Editeur de partitions GParted).

- Les partitions du disque `/dev/sda` s'affichent. Noter la taille de `/dev/sda1`, ainsi que son filesystem (ext2/ext3/ext4).
- Passer au disque `/dev/sdb`:
  - Créer une nouvelle partition primaire `/dev/sdb1`, de taille légèrement supérieure à celle de `/dev/sda1`.
  - **Appliquer** pour que la création soit effective, puis modifier (par clic droit) les drapeaux de `/dev/sdb1` pour rendre cette partition bootable.
  - Créer une partition étendue `/dev/sdb2`, occupant le reste du disque. Sur cette partition, créer une partition logique `/dev/sdb5` non formatée.
  - **Appliquer** pour que les créations soient effectives.
- Repasser au disque `/dev/sda`. Cliquer-droit sur `/dev/sda1` et choisir **Démonter** puis **Copier**.
- Repasser au disque `/dev/sdb`. Cliquer-droit sur `/dev/sdb1` et choisir **Coller** (ou **Paste**). “Appliquer” à nouveau.
- Fermer GParted.

En ligne de commande

- Remonter la partition de boot :

```
sudo mount /boot
```

- Faire un scan des volumes physiques de LVM :

```
sudo pvscan
  PV /dev/sda5   VG delphy   lvm2 [148,81 GiB / 4,87 GiB free]
  Total: 1 [148,81 GiB] / in use: 1 [148,81 GiB] / in no VG: 0 [0   ]
```

Cela signifie que le volume physique (PV) `/dev/sda5` est inclus dans le groupe de volumes (VG) nommé ici delphy.

- Déclarer `/dev/sdb5` comme volume physique :

```
sudo pvcreate /dev/sdb5
  Physical volume "/dev/sdb5" successfully created
```

Vérifier qu'il existe bien, mais n'est pas encore attribué à un groupe de volumes :

```
sudo pvscan
  PV /dev/sda5   VG delphy          lvm2 [148,81 GiB / 4,87 GiB free]
  PV /dev/sdb5                      lvm2 [465,47 GiB]
  Total: 2 [614,28 GiB] / in use: 1 [148,81 GiB] / in no VG: 1 [465,47 GiB]
```

- Attribuer `/dev/sdb5` au groupe de volumes (ici delphy). Ce groupe de volumes est “étendu” à `/dev/sdb5` :

```
sudo vgextend delphy /dev/sdb5
  Volume group "delphy" successfully extended
```

- Vérification :

```
sudo pvscan
  PV /dev/sda5   VG delphy   lvm2 [148,81 GiB / 4,87 GiB free]
  PV /dev/sdb5   VG delphy   lvm2 [465,46 GiB / 465,46 GiB free]
  Total: 2 [614,27 GiB] / in use: 2 [614,27 GiB] / in no VG: 0 [0   ]
```

Lancer enfin le déplacement des données, du volume physique /dev/sda5 vers le volume physique /dev/sdb5 :

```
sudo pvmove /dev/sda5 /dev/sdb5
  /dev/sda5: Moved: 0,3%
  /dev/sda5: Moved: 0,7%
  /dev/sda5: Moved: 1,0%
  /dev/sda5: Moved: 1,3%
  ...
  /dev/sda5: Moved: 99,8%
  /dev/sda5: Moved: 100,0%
```

'opération peut prendre du temps (plusieurs heures pour les grosses partitions), suivant la taille des données à transférer, la rapidité des disques, etc.

- Vérifier que le contenu de `/dev/sda5` a bien été transféré sur `/dev/sdb5` :

```
sudo pvscan
  PV /dev/sda5   VG delphy   lvm2 [148,81 GiB / 148,81 GiB free]
  PV /dev/sdb5   VG delphy   lvm2 [465,46 GiB / 321,53 GiB free]
```

En effet, la totalité de `/dev/sda5` est libre, et `/dev/sdb5` est occupée par les données transférées.

- Supprimer `/dev/sda5` du groupe de volumes delphy :

```
sudo vgreduce delphy /dev/sda5
  Removed "/dev/sda5" from volume group "delphy"
```

- Vérifier :

```
sudo pvscan
  PV /dev/sdb5   VG delphy          lvm2 [465,46 GiB / 321,53 GiB free]
  PV /dev/sda5                      lvm2 [148,81 GiB]
  Total: 2 [614,28 GiB] / in use: 1 [465,46 GiB] / in no VG: 1 [148,81 GiB]
```

- Enlever le disque des volumes physiques :

```
sudo pvremove /dev/sda5
  Labels on physical volume "/dev/sda5" successfully wiped
```

- On peut désormais enlever le disque.

Finalisation

Réinstaller GRUB sur le MBR du disque dur :

```
sudo grub-install /dev/sdb
```

Eteigner l'ordinateur, enlever l'ancien disque et le remplacer par le nouveau, au niveau des branchements.

Volume LVM en miroir

Un volume en miroir consiste en un volume logique linéaire (LV) et une copie de celui-ci. Pour créer un volume en miroir, vous spécifier le nombre de copies de données à effectuer avec l'argument **-m** de la commande **lvcreate**:

- **-m1**, permet de créer un miroir qui produira deux copies du système de fichiers : un volume logique linéaire plus une copie. De façon similaire
- **-m2** permet de créer deux miroirs qui produiront trois copies du système de fichiers.

Création de LVM en miroir

Au début, 2 nouveaux PV sont générés et résumés dans un VG:

```
# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
# pvcreate /dev/sde1
  Physical volume "/dev/sde1" successfully created
# vgcreate mirrdata /dev/sdd1 /dev/sde1
  Volume group "mirrdata" successfully created
```

Maintenant, dans ce VG, un LV en miroir est créé, qui utilise `/dev/sdd1` et `/dev/sde1`. Le journal de synchronisation est ensuite transféré dans la mémoire principale avec l'option `--corelog`. Cette procédure est uniquement recommandée à des fins de test, car le journal est perdu après un redémarrage et, par conséquent, aucune information n'est disponible sur les composants synchronisés. La synchronisation commencerait donc depuis le début. Par conséquent, une troisième PV pour les données de journal serait recommandée. La commande suivante génère le LV en miroir:

```
lvcreate -l50%FREE -m1 -n mirrorlv mirrdata /dev/sdd1 /dev/sde1 --corelog
```

Si un segment échoue maintenant, le second est automatiquement fourni en tant que LV linéaire et le segment défaillant peut être remplacé. Le LV désormais fourni en linéaire peut ensuite être reconverti en LV symétrisé, s'il y a de nouveau suffisamment d'espace dans le VG. Il existe une commande lvconvert. Voici un exemple simple: la conversion d'une "donnée" LV du VG "vg00" - il doit y avoir suffisamment d'espace - ressemble à ceci:

```
# lvconvert -m1 vg00/data --corelog
  vg00/data: Converted: 70.1%
  vg00/data: Converted: 100.0%
  Logical volume data converted
```

Pour vérifier si les données sont vraiment mises en miroir et sur quels périphériques se trouvent le mirroir, vous pouvez les filtrer avec la commande lvs:

```
# lvs -a -o +devices
```

```
   LV                 VG       Attr   LSize    Origin Snap%  Move Log
Copy%  Convert Devices
mirrorlv           mirrdata mwi-a- 1020.00m                      100.00
mirrorlv_mimage_0(0),mirrorlv_mimage_1(0)
 [mirrorlv_mimage_0] mirrdata iwi-ao 1020.00m
/dev/sdd1(0)
 [mirrorlv_mimage_1] mirrdata iwi-ao 1020.00m
/dev/sde1(0)
```

Le régions

Un miroir LVM divise le périphérique en train d'être copié en régions faisant une taille de 512 Ko par défaut. On utilise l'argument **-R** de la commande **lvcreate** pour spécifier la taille des régions en Mo. On peut aussi changer la taille des régions par défaut en modifiant le paramètre **mirror*region*size** dans le fichier `lvm.conf`.

En raison des limitations dans l'infrastructure du cluster, les miroirs de clusters de plus de 1,5 To ne peuvent pas être créés avec une taille de région par défaut de 512 Ko. Les utilisateurs nécessitant de plus grands miroirs doivent augmenter la taille des régions par défaut. L'omission d'effectuer cette augmentation de taille des régions aura pour résultat de suspendre la création LVM et pourrait aussi suspendre d'autres commandes LVM.

En règle générale pour spécifier la taille des régions pour des miroirs faisant plus de 1,5 To, on peut prendre la taille du miroir en téraoctets et arrondir ce nombre à la puissance de 2 suivante, en utilisant ce nombre en tant qu'argument **-R** de la commande **lvcreate**. Par exemple, spécifier:

- **-R 2** si la taille du miroir est de 1,5 To,
- **-R 4** si la taille du miroir est de 3 To,
- **-R 8** si la taille du miroir est de 5 To.

La commande suivante crée un volume logique en miroir avec une taille de région de 2 Mo :

```
lvcreate -m1 -L 2T -R 2 -n mirror vol_group
```

Le fichier journal

LVM maintient un petit fichier journal afin de garder une trace des régions synchronisées avec le ou les miroirs. Par défaut, ce fichier journal est stocké sur le disque, il est donc persistant à travers les redémarrages et assure ainsi que le miroir ne doit pas être resynchronisé à chaque fois qu'une machine redémarre ou se bloque.

Le fichier journal du miroir est créé sur un périphérique séparé à partir des périphériques sur lesquels n'importe quelle branche du miroir peut être créée.

Avec des miroirs de clusters, la gestion du journal des miroirs est sous l'entière responsabilité du noeud du cluster avec l'ID actuel du cluster le plus bas. Ainsi, lorsque le périphérique possédant le journal des miroirs de cluster se retrouve indisponible sur un sous-ensemble du cluster, le miroir clusterisé peut continuer à fonctionner sans le moindre impact tant que le noeud du cluster avec l'ID le plus bas conserve l'accès au journal miroir. Comme le miroir n'est pas perturbé, aucune action corrective automatique (réparation) n'est effectuée. Lorsque le noeud de cluster avec l'ID le plus bas perd son accès au journal miroir , une action automatique sera alors lancée (peu importe le niveau d'accessibilité du journal depuis d'autres noeuds).

Lorsqu'un miroir est créé, les régions du miroir sont synchronisées. Pour les composants miroir volumineux, le processus de synchronisation peut prendre un moment. Lorsqu'on crée un nouveau miroir qui n'a pas besoin d'être réactivé, vous pouvez spécifier l'argument `nosync` pour indiquer qu'une synchronisation initiale à partir du premier périphérique n'est pas requise.

Fichier journal du miroir en mémoire.

On peut spécifier que ce fichier journal soit stocké en mémoire avec l'argument **–mirrorlog core** ; ceci élimine le besoin de posséder un périphérique de fichiers journaux supplémentaire, mais le miroir entier devra être resynchronisé à chaque redémarrage.

La commande suivante crée un volume logique en miroir à partir du groupe de volumes **bigvg**. Le volume logique est appelé **ondiskmirvol** et a un seul miroir. Le volume a une taille de 12Mo et garde le fichier journal du miroir en mémoire.

```
# lvcreate -L 12MB -m1 --mirrorlog core -n ondiskmirvol bigvg
  Logical volume "ondiskmirvol" created
```

Fichier journal du miroir sur le même périphérique

Il possible de créer un fichier journal sur le même périphérique que celui des branches du miroir à l'aide de l'argument **–alloc anywhere** de la commande **vgcreate**. Ceci peut dégrader la performance, mais permet aussi de créer un miroir, même si on ne possède que deux périphériques sous-jacents.

La commande suivante crée un volume logique en miroir avec un seul miroir dans lequel le fichier journal du miroir se trouve sur le même périphérique que l'une des branches du miroir. Dans cet exemple, le groupe de volumes vg0 est composé de deux périphériques. Cette commande crée un volume de 500 méga-octets nommé mirrorlv dans le groupe de volumes vg0.

```
lvcreate -L 500M -m1 -n mirrorlv -alloc anywhere vg0
```

Tout comme avec un fichier journal de miroir standard, il est possible de créer les fichiers journaux de miroirs redondants sur le même périphérique que les branches du miroir en utilisant l'argument **–alloc anywhere** de la commande **vgcreate**. Ceci peut dégrader la performance, mais permet de créer un fichier journal de miroir redondant même si on ne possède pas suffisamment de périphériques sous-jacents pour que chaque fichier journal soit stocké sur un périphérique autre que celui des branches du miroir.

Fichier journal du miroir en miroir

Pour créer un fichier journal miroir qui est lui-même mis en miroir, spécifier l'argument `--mirrorlog mirrored`. La commande suivante crée un volume logique en miroir à partir du groupe de volumes bigvg. Le volume logique est appelé twologvol et a un seul miroir. Le volume a une taille de 12Mo et le fichier journal du miroir est en miroir, avec chaque journal stocké sur un périphérique séparé.

```
# lvcreate -L 12MB -m1 --mirrorlog mirrored -n twologvol bigvg
  Logical volume "twologvol" created
```

Fichier journal du miroir sur un disque particulier

On peut spécifier les périphériques à utiliser pour les fichiers journaux et les branches du miroir, ainsi que les extensions des périphériques à utiliser. Pour forcer un fichier journal sur un disque particulier, spécifier une seule extension sur le disque où il sera placé. LVM ne respecte pas forcément l'ordre dans lequel les périphériques sont listés dans la ligne de commande. Si des volumes physiques sont listés, il s'agit du seul emplacement sur lequel l'allocation aura lieu. Toute extension physique inclue dans la liste qui est déjà allouée sera ignorée.

La commande suivante crée un volume logique en miroir avec un seul miroir et un seul fichier journal qui n'est pas en miroir. Le volume a une taille de 500 méga-octets, il s'appelle mirrorlv et est issu du groupe de volumes vg0. La première branche du miroir se trouve sur le périphérique `/dev/sda1`, la seconde sur `/dev/sdb1` et le fichier journal du miroir se trouve sur `/dev/sdc1`.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1 /dev/sdb1 /dev/sdc1
```

La commande suivante crée un volume logique en miroir avec un seul miroir. Le volume a une taille de 500 méga-octets, il s'appelle mirrorlv et est issu du groupe de volumes `vg0`. La première branche du miroir se trouve sur les extensions 0-499 du périphérique `/dev/sda1`, la seconde sur les extensions 0-499 du périphérique `/dev/sdb1`, et le fichier journal du miroir commence à l'extension 0 du périphérique `/dev/sdc1`. Il y a 1Mo d'extensions. Si une des extensions spécifiées a déjà été allouée, elle sera ignorée.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499
```

```
/dev/sdc1:0
```