Red Hat Enterprise Linux 5

Administration du gestionnaire de volumes logiques

Guide de l'administrateur LVM

Édition 3

Last Updated: 2017-10-16

Red Hat Enterprise Linux 5 Administration du gestionnaire de volumes logiques

Guide de l'administrateur LVM
Édition 3

Landmann
rlandmann@redhat.com

Notice légale

Résumé

Ce guide décrit le gestionnaire de volumes logiques (LVM de l'anglais Logical Volume Manager), y compris des informations sur l'exécution de LVM dans un environnement en clusters. Le contenu de ce document est spécifique à la version LVM2.

Table des matières

INTRODUCTION

1. À PROPOS DE CE GUIDE

Ce guide décrit le gestionnaire de volumes logiques (LVM de l'anglais Logical Volume Manager), y compris des informations sur l'exécution de LVM dans un environnement en clusters. Le contenu de ce document est spécifique à la version LVM2.

2. PUBLIC VISÉ

Ce guide est destiné aux administrateurs système gérant des systèmes Linux. Vous devez être familier avec Red Hat Enterprise Linux 5 et l'administration de systèmes de fichiers GFS.

3. VERSIONS DU LOGICIEL

Tableau 1. Versions du logiciel

Logiciel	Description
RHEL5	Se réfère à RHEL5 et les versions plus récentes
GFS	Se réfère à GFS pour RHEL5 et les versions plus récentes

4. DOCUMENTATION ANNEXE

Pour davantage d'informations à propos de l'utilisation de Red Hat Enterprise Linux, reportez-vous aux ressources suivantes :

- *Guide d'installation de Red Hat Enterprise Linux*— Fournit des informations à propos de l'installation de Red Hat Enterprise Linux 5.
- *Guide déploiement de Red Hat Enterprise Linux*— Fournit des informations à propos du déploiement, de la configuration et de l'administration de Red Hat Enterprise Linux 5.

Pour davantage d'informations à propos de Red Hat Cluster Suite pour Red Hat Enterprise Linux 5, reportez-vous aux ressources suivantes :

- *Aperçu de Red Hat Cluster Suite*— Fournit un aperçu de haut niveau à propos de Red Hat Cluster Suite.
- *Configuration et gestion d'un cluster Red Hat*— Fournit des informations à propos de l'installation, la configuration et la gestion des composants Red Hat Cluster.
- *Système de fichiers GFS : configuration et administration*— Fournit des informations à propos de l'installation, la configuration et la maintenance de Red Hat GFS (Red Hat Global File System).

- *Système de fichiers GFS : configuration et administration*— Fournit des informations à propos de l'installation, la configuration et la maintenance de Red Hat GFS2 (Red Hat Global File System 2).
- *Utilisation de "Device-Mapper Multipath"*— Fournit des informations à propos de l'utilisation de la fonctionnalité "Device-Mapper Multipath" de Red Hat Enterprise Linux 5.
- *Utilisation de GNBD avec le système de fichiers GFS*— Fournit un aperçu de l'utilisation de GNBD (de l'anglais Global Network Block Device) avec Red Hat GFS.
- *Administration du serveur virtuel Linux*— Fournit des informations à propos de la configuration de systèmes et services à hautes performances avec le serveur virtuel Linux (LVS de l'anglais Linux Virtual Server).
- *Notes de mise à jour de Red Hat Cluster Suite*— Fournit des informations à propos de la version courante de Red Hat Cluster Suite.

La documentation de Red Hat Cluster Suite et les autres documents de Red Hat sont disponibles en version HTML, PDF et RPM sur le CD-ROM de documentation de Red Hat Enterprise Linux et en ligne à l'adresse suivante : http://www.redhat.com/docs/.

5. COMMENTAIRE

Si vous trouvez des fautes de frappe ou si vous avez des suggestions pour améliorer ce manuel, n'hésitez surtout pas à nous en faire part ! Veuillez envoyer vos remarques par l'entremise de Bugzilla (http://bugzilla.redhat.com/bugzilla/) dans la rubrique **rh-cs**.

Be sure to mention the manual's identifier:

```
Bugzilla component: Documentation-cluster
Book identifier: Cluster_Logical_Volume_Manager(EN)-5 (2009-01-05T15:20)
```

By mentioning this manual's identifier, we know exactly which version of the guide you have.

Si vous avez une suggestion afin d'améliorer cette documentation, essayez d'être le plus précis possible. Si vous avez trouvé une erreur, veuillez inclure le numéro de section et quelques passages du texte pour que nous puissions retrouver l'erreur facilement.

CHAPITRE 1. LE GESTIONNAIRE DE VOLUMES LOGIQUES LVM

Ce chapitre fournit aperçu de haut niveau des composants du gestionnaire de volumes logiques (LVM)

1.1. VOLUMES LOGIQUES

La gestionnaire de volumes crée une couche d'abstraction sur le stockage physique afin que vous puissiez créer les volumes de stockage logiques. Ceci est bien plus flexible que d'utiliser directement le stockage physique.

Un volume logique fournit la virtualisation des stockages. Avec un volume logique, vous n'êtes pas restreint aux tailles des disques physiques. De plus, la configuration du stockage matériel n'est pas visible par le logiciel, il peut donc être redimensionné et déplacé sans que les applications soient arrêtées ou les systèmes de fichiers démontés. Cela peut réduire les frais d'opération.

Les volumes logiques offrent les avantages suivants par rapport à l'utilisation directe du stockage physique :

- Capacité flexible

  Lors de l'utilisation de volumes logiques, les systèmes de fichiers peuvent être étendus à travers plusieurs disques, étant donné que vous pouvez regrouper les disques et partitions dans un seul volume logique.

- Pools de stockage redimensionnables

  Vous pouvez augmenter ou réduire la taille des volumes logiques avec de simples commandes logicielles, sans reformater et repartitionner les périphériques disques sous-jacents.

- Déplacement des données en ligne

  Pour déployer de nouveaux sous-systèmes de stockage plus résistants et plus rapides, vous pouvez déplacer les données là où votre système est actif. Les données peuvent être réarrangées sur les disques pendant qu'ils sont utilisés. Vous pouvez par exemple vider un disque changeable à chaud (hot-swappable) avant de le supprimer.

- Nommage des périphériques pratique

  Les volumes de stockage logiques peuvent être gérés dans des groupes définis par les utilisateurs que vous pouvez nommer à votre convenance.

- Disques en mode stripe

  Vous pouvez créer un volume logique qui sépare les données à travers deux ou plusieurs disques en suivant un modèle round-robin. Cela peut augmenter le débit de manière significative.

- Volumes miroirs

  Les volumes logiques offrent un moyen pratique afin de configurer un miroir pour vos données.

- Instantanés de volumes

  En utilisant les volumes logiques, vous pouvez prendre des instantanés de périphériques pour des sauvegardes efficaces ou pour tester des modifications sans affecter les autres données.

L'implémentation de ces fonctionnalités dans LVM est décrite dans le reste de ce document.

1.2. APERÇU DE L'ARCHITECTURE LVM

Dans la version RHEL 4 du système d'exploitation Linux, le gestionnaire de volumes logiques LVM1 d'origine a été remplacé par LVM2, qui possède un framework noyau plus générique que LVM1. Par rapport à LVM1, LVM2 fournit les améliorations suivantes :

- Capacité flexible
- Stockage des métadonnées plus efficace
- Format de récupération amélioré
- Nouveau format de métadonnées ASCII
- Changements atomiques des métadonnées
- Copies redondantes des métadonnées

LVM2 est compatible avec LVM1, à l'exception de la prise en charge des instantanés et du cluster. Vous pouvez convertir un groupe de volumes, du format LVM1 au format LVM2, avec la commande **vgconvert**. Pour obtenir des informations à propos de la conversion du format des métadonnées LVM, reportez-vous à la page de manuel **vgconvert**(8).

L'unité de stockage physique sous-jacente d'un volume logique LVM est un périphérique bloc tel qu'une partition ou un disque entier. Ce périphérique est initialisé en tant que *volume physique* (PV).

Pour créer un volume logique LVM, les volumes physiques sont combinés dans un *groupe de volumes* (VG). Cela crée un pool d'espace disque à partir duquel les volumes logiques (LV) peuvent être assignés. Ce processus est similaire à la manière dont les disques sont divisés en partitions. Un volume logique est utilisé par des systèmes de fichiers et des applications (par exemple des bases de données).

Figure 1.1, « LVM Logical Volume Components » shows the components of a simple LVM logical volume:

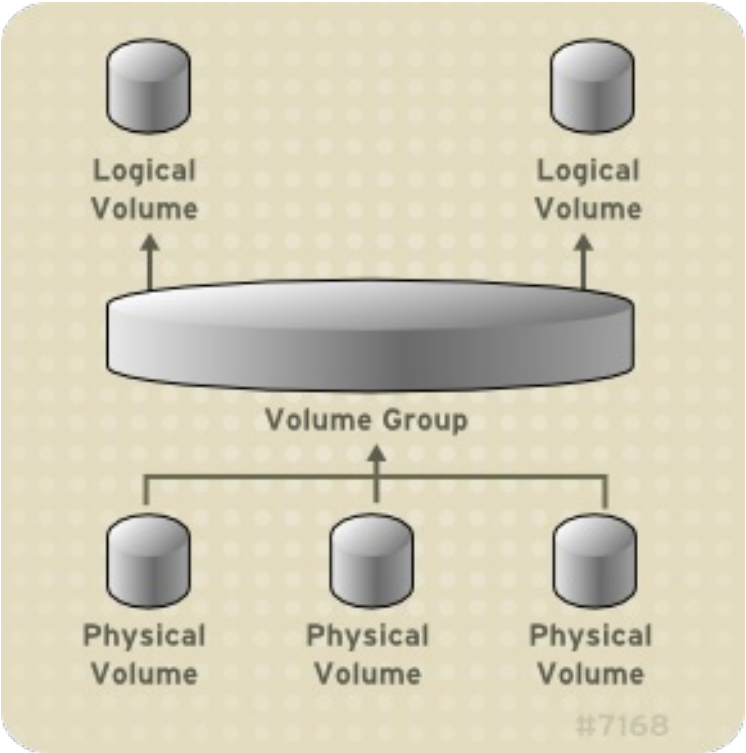


Figure 1.1. LVM Logical Volume Components

For detailed information on the components of an LVM logical volume, see Chapitre 2, *Composants LVM*.

1.3. LE GESTIONNAIRE DE VOLUMES LOGIQUES CLUSTERISÉ (CLVM)

Le gestionnaire de volumes logiques en cluster (CLVM) est un groupe d'extensions de mise en cluster pour LVM. Ces extensions permettent à un cluster d'ordinateurs de gérer la mémoire partagée (par exemple sur un SAN) en utilisant LVM.

Vous devez utiliser CLVM en fonction des besoins de votre système :

- Si un seul noeud de votre système requiert accès au stockage que vous configurez, en tant que volumes logiques, vous pourrez alors utiliser LVM sans les extensions CLVM et les volumes logiques créés avec ce noeud, sont tous locaux par rapport au noeud.
- SI vous utilisez un système clusterisé pour les failover, dans lequel juste un noeud en train d'accéder au stockage, est actif, à tout moment, vous devrez utiliser les agents HA-LVM (High Availability Logical Volume Management agents). Pour davantage d'informations sur HA-LVM, voir *Configuring and Managing a Red Hat Cluster.*
- CLVM permet à un utilisateur de configurer des volumes logiques sur une mémoire partagée en verrouillant l'accès au stockage physique pendant qu'un volume est en cours de configuration, et utilise les services de verrouillage clusterisé pour gérer la mémoire partagée.

Pour utiliser CLVM, le logiciel de Red Hat Cluster Suite, comprenant le démon **clmvd**, doit être en cours d'exécution. Le démon **clmvd** est l'extension de mise en cluster clé pour LVM. Le démon **clmvd** est démarré sur chaque ordinateur du cluster et distribue les mises à jour de métadonnées au sein du cluster, en présentant à chaque ordinateur du cluster la même vue des volumes logiques. Pour toute information supplémentaire sur l'installation et l'administration de Red Hat Cluster Suite, voir *Configuring and Managing a Red Hat Cluster.*

Pour s'assurer que **clmvd** soit bien démarré en cours d'initialisation, vous pouvez exécuter une commande **chkconfig ... on** sur le service **clvmd**, comme suit :

```
# chkconfig clvmd on
```

Si le démon **clvmd** n'a pas été démarré, vous pouvez exécuter une commande **service ... start** sur le service **clvmd**, comme suit :

```
# service clvmd start
```

Creating LVM logical volumes in a cluster environment is identical to creating LVM logical volumes on a single node. There is no difference in the LVM commands themselves, or in the LVM graphical user interface, as described in Chapitre 4, *Administration LVM avec les commandes CLI* and Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM* In order to enable the LVM volumes you are creating in a cluster, the cluster infrastructure must be running and the cluster must be quorate.

By default, logical volumes created with CLVM on shared storage are visible to all computers that have access to the shared storage. It is possible, however, to create logical volumes when the storage devices are visible to only one node in the cluster. It is also possible to change the status of a logical volume from a local volume to a clustered volume. For information, see Section 4.3.2, « Création de groupes de volumes au sein d'un cluster » and Section 4.3.7, « Changement des paramètres du groupe de volumes ».

Figure 1.2, « Aperçu de CLVM » shows a CLVM overview in a Red Hat cluster.

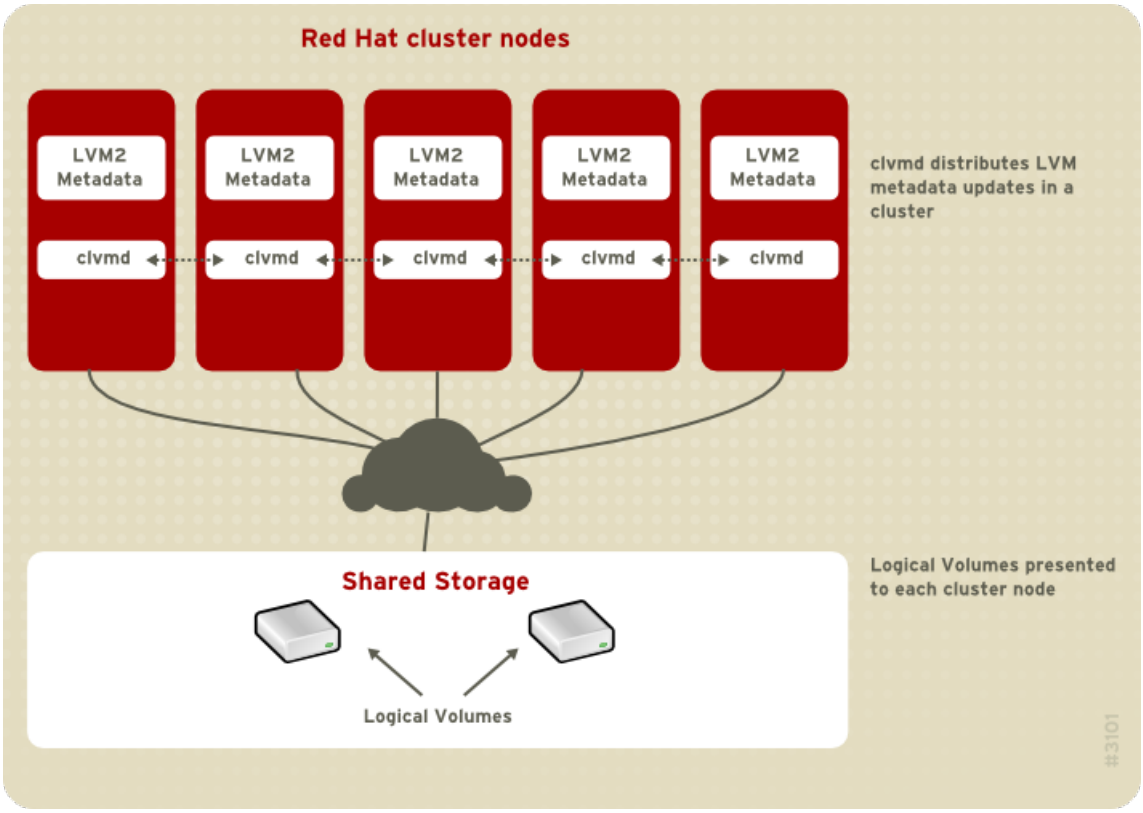
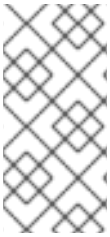


Figure 1.2. Aperçu de CLVM

NOTE

La mémoire partagée utilisée dans Red Hat Cluster Suite nécessite que vous exécutiez le démon du gestionnaire de volume logique du cluster (`clvmd`) ou les agents HA-LVM (de l'anglais High Availability Logical Volume Management). Si vous n'êtes pas en mesure d'utiliser le démon `clvmd` ou HA-LVM pour des raisons opéraitonnelles ou parce que vous ne possédez pas les privilèges d'accès, vous ne pourrez pas utiliser une instance unique LVM sur le disque partagé, car cela pourrait résulter par la corruption des données. Si vous avez des questions à ce sujet, n'hésitez pas à contacter votre représentant commercial Red Hat.



NOTE

CLVM requires changes to the `lvm.conf` file for cluster-wide locking. Information on configuring the `lvm.conf` file to support clustered locking is provided within the `lvm.conf` file itself. For information about the `lvm.conf` file, see Annexe B, *Les fichiers de configuration LVM*.

1.4. APERÇU DU DOCUMENT

Le reste de ce document inclut les chapitres suivants :

- Chapitre 2, *Composants LVM* describes the components that make up an LVM logical volume.
- Chapitre 3, *Aperçu général de l'administration LVM* provides an overview of the basic steps you perform to configure LVM logical volumes, whether you are using the LVM Command Line Interface (CLI) commands or the LVM Graphical User Interface (GUI).

- Chapitre 4, *Administration LVM avec les commandes CLI* summarizes the individual administrative tasks you can perform with the LVM CLI commands to create and maintain logical volumes.
- Chapitre 5, *Exemples de configuration LVM* provides a variety of LVM configuration examples.
- Chapitre 6, *Résolution de problèmes LVM* provides instructions for troubleshooting a variety of LVM issues.
- Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM* summarizes the operating of the LVM GUI.
- Annexe A, *Device Mapper (Mappeur de périphériques)* describes the Device Mapper that LVM uses to map logical and physical volumes.
- Annexe B, *Les fichiers de configuration LVM* describes the LVM configuration files.
- Annexe C, *Les balises des objets LVM* describes LVM object tags and host tags.
- Annexe D, *Métadonnées des groupes de volumes LVM* describes LVM volume group metadata, and includes a sample copy of metadata for an LVM volume group.

CHAPITRE 2. COMPOSANTS LVM

Ce chapitre décrit les composants d'un volume logique LVM.

2.1. LES VOLUMES PHYSIQUES

L'unité de stockage physique sous-jacente d'un volume logique LVM est un périphérique bloc tel qu'une partition ou disque entier. Afin d'utiliser le périphérique pour un volume logique LVM, celui-ci doit être initialisé en tant que volume physique (PV). Lors de l'initialisation d'un périphérique bloc en tant que volume physique, une étiquette est placée au début du périphérique.

Par défaut, l'étiquette LVM est placée dans le deuxième secteur de 512 octets. Vous pouvez surcharger cette valeur par défaut en plaçant l'étiquette sur un des 4 premiers secteurs. Cela permet, si nécessaire, aux volumes LVM de coexister avec les autres utilisateurs de ces secteurs.

Une étiquette LVM fournit un ordre de périphériques et une identification correcte à un périphérique physique, étant donné que les périphériques peuvent apparaître dans n'importe quel ordre lorsque le système est démarré. Une étiquette LVM reste persistante lors des redémarrages et tout au long du cluster.

L'étiquette LVM identifie le périphérique comme un volume physique LVM. Elle contient un identifiant unique aléatoire (UUID) pour un volume physique. Elle stocke également la taille du périphérique bloc en octets et elle enregistre l'emplacement où les métadonnées seront stockées sur le périphérique.

Les métadonnées LVM contiennent les détails de configuration des groupes de volumes LVM sur votre système. Par défaut, une copie identique des métadonnées est maintenue dans toutes les zones de métadonnées de chaque volume physique au sein du groupe de volumes. Les métadonnées LVM sont petites et stockées en ASCII.

Actuellement, LVM vous permet de stocker 0, 1 ou 2 copies identiques de ces métadonnées sur chaque volume physique. La valeur par défaut est 1. Une fois que vous aurez configuré le nombre de copies de métadonnées sur le volume physique, vous ne pourrez plus changer cette valeur. La première copie est stockée au début du périphérique, juste après l'étiquette. S'il y a une deuxième copie, elle est placée à la fin du périphérique. Si vous écrasez accidentellement la zone au début de votre disque, en écrivant sur un disque différent de celui désiré, une seconde copie de métadonnées à la fin du périphérique vous permettra de récupérer les métadonnées.

For detailed information about the LVM metadata and changing the metadata parameters, see Annexe D, *Métadonnées des groupes de volumes LVM*

2.1.1. LVM Physical Volume Layout

Figure 2.1, « Structure d'un volume physique » shows the layout of an LVM physical volume. The LVM label is on the second sector, followed by the metadata area, followed by the usable space on the device.

NOTE

Dans le noyau Linux (et tout au long de ce document), les secteurs ont une taille de 512 octets.

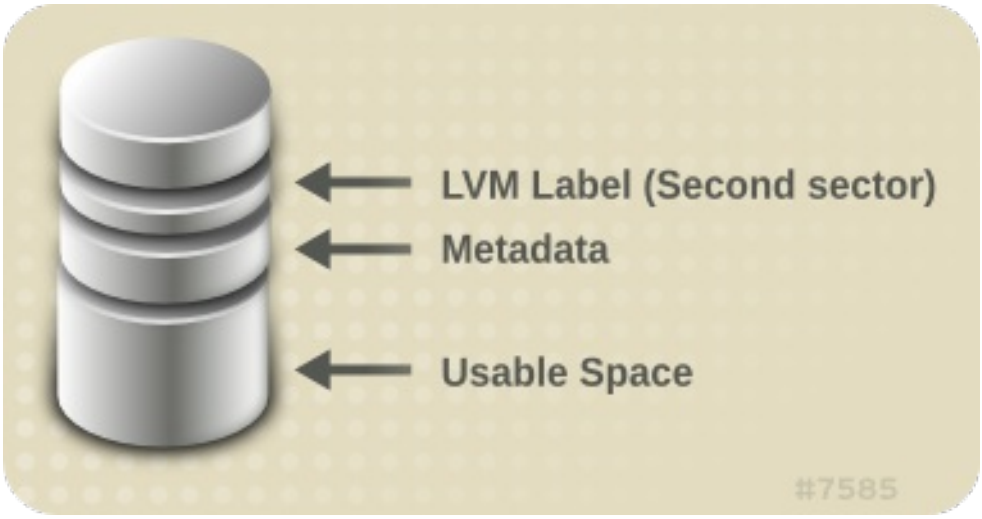


Figure 2.1. Structure d'un volume physique

2.1.2. Plusieurs partitions sur un disque

LVM vous permet de créer des volumes physiques en dehors des partitions de disques. Il est généralement recommandé, pour les raisons suivantes, de créer une partition qui couvre le disque entier afin de l'étiqueter en tant que volume physique LVM :

- Facilité d'administration

  Il est plus facile d'assurer le suivi du matériel au sein d'un système si chaque disque réel n'apparaît qu'une seule fois. Ceci est particulièrement vrai si un disque échoue. De plus, plusieurs volumes physiques sur un seul disque peuvent provoquer, au moment du démarrage, la génération d'un avertissement du noyau relatif à des types de partitions inconnus.

- Performances en mode stripe

  LVM ne peut pas savoir si deux volumes physiques sont sur le même disque physique. Si vous créez un volume logique en mode stripe lorsque deux volumes physiques sont sur le même disque, les stripes pourraient se trouver sur des partitions différentes. Cela causerait une diminution des performances plutôt qu'une augmentation.

Bien que cela ne soit pas recommandé, il peut y avoir des circonstances particulières pour lesquelles vous devriez diviser un disque en plusieurs volumes physiques LVM différents. Par exemple, sur un système avec peu de disques, il pourrait être nécessaire de déplacer les données autour des partitions lorsque vous migrez un système existant vers des volumes LVM. De plus, si vous avez un disque très volumineux et que vous voulez, à des fins d'administration, plus d'un groupe de volumes, il est alors nécessaire de partitionner le disque. Si vous disposez d'un disque avec plus d'une partition et que deux de ces partitions sont dans le même groupe de volumes, n'oubliez pas de spécifier celles qui doivent être incluses dans un volume logique lors de la création des volumes en mode stripe.

2.2. LES GROUPES DE VOLUMES

Les volumes physiques sont combinés en des groupes de volumes (VG). Cela crée un pool d'espace disque à partir duquel les volumes logiques peuvent être assignés.

À l'intérieur d'un groupe de volumes, l'espace disque disponible pour l'allocation est divisé en des unités de taille fixe appelées des extensions. Une extension est la plus petite unité d'espace pouvant être allouée. Au l'intérieur d'un volume physique, les extensions sont appelées des extensions physiques.

Un volume logique est divisé en extensions logiques de même taille que les extensions physiques. La taille des extensions est donc la même pour tous les volumes logiques du groupe de volumes. Le groupe de volumes mappe les extensions logiques avec les extensions physiques.

2.3. LES VOLUMES LOGIQUES LVM

Dans LVM, un groupe de volumes est divisé en volumes logiques. Il y a trois types de volumes logiques LVM : les volumes *linéaires*, les volumes *en mode stripe*et les volumes *en miroir*. Ils sont décrits dans les sections suivantes.

2.3.1. Les volumes linéaires

Un volume linéaire regroupe plusieurs volumes physiques dans un volume logique. Si vous avez par exemple deux disques de 60Go, vous pouvez créer un volume logique de 120Go. Le stockage physique est concaténé.

Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, as shown in Figure 2.2, « Mappage des extensions » logical extents 1 to 99 could map to one physical volume and logical extents 100 to 198 could map to a second physical volume. From the point of view of the application, there is one device that is 198 extents in size.

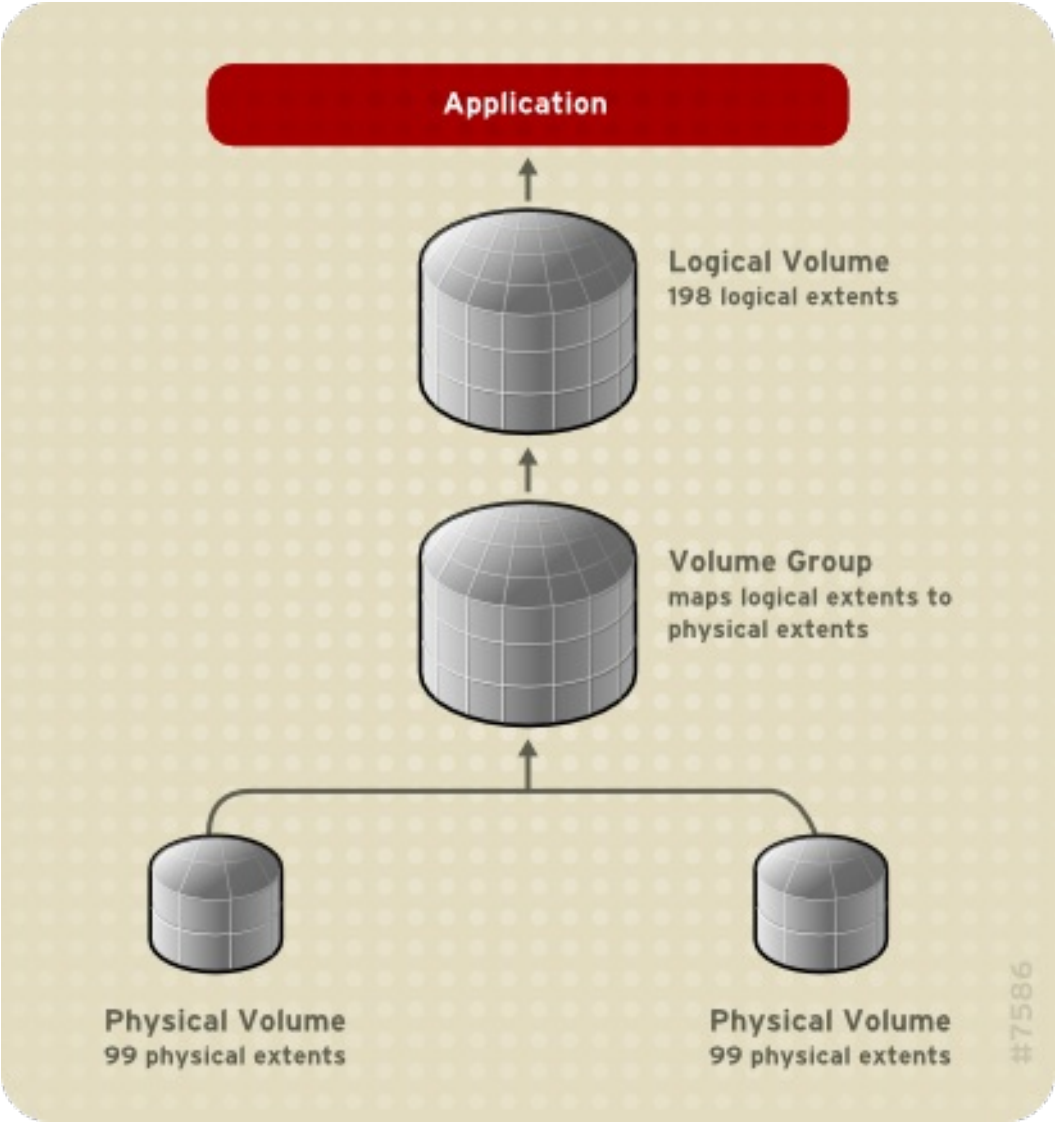


Figure 2.2. Mappage des extensions

The physical volumes that make up a logical volume do not have to be the same size. Figure 2.3, « Volume linéaire avec des volumes physiques différents » shows volume group **VG1** with a physical extent size of 4MB. This volume group includes 2 physical volumes named **PV1** and **PV2**. The physical volumes are divided into 4MB units, since that is the extent size. In this example, **PV1** is 100 extents in size (400MB) and **PV2** is 200 extents in size (800MB). You can create a linear volume any size between 1 and 300 extents (4MB to 1200MB). In this example, the linear volume named **LV1** is 300 extents in size.

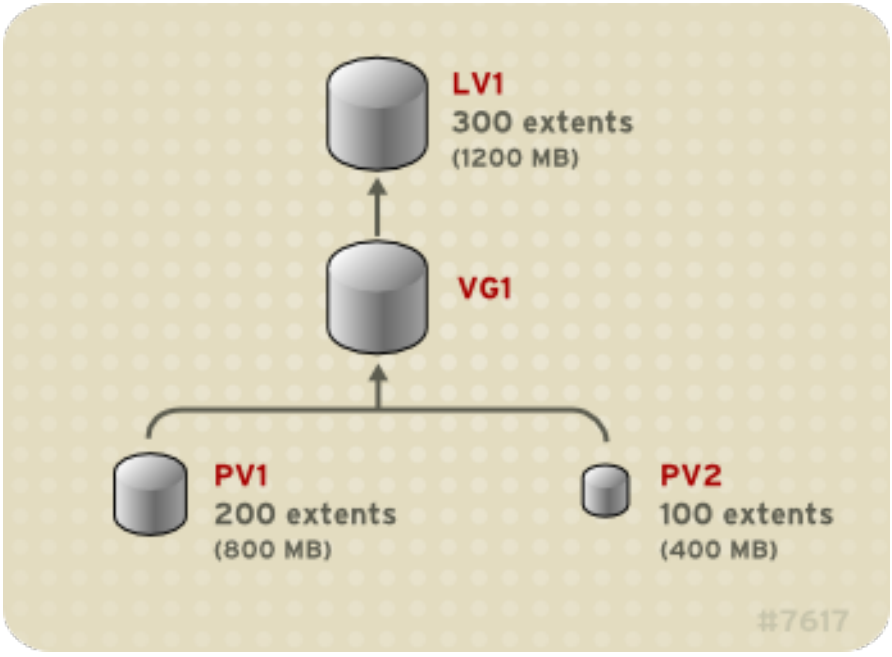


Figure 2.3. Volume linéaire avec des volumes physiques différents

You can configure more than one linear logical volume of whatever size you desire from the pool of physical extents. Figure 2.4, « Plusieurs volumes logiques » shows the same volume group as in Figure 2.3, « Volume linéaire avec des volumes physiques différents » , but in this case two logical volumes have been carved out of the volume group: **LV1**, which is 250 extents in size (1000MB) and **LV2** which is 50 extents in size (200MB).

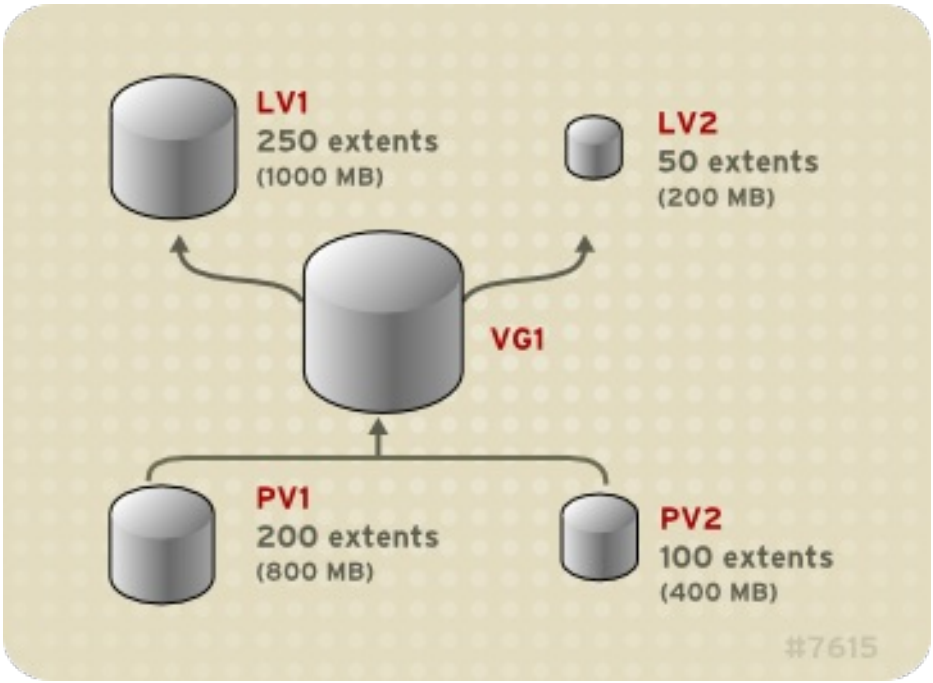


Figure 2.4. Plusieurs volumes logiques

2.3.2. Les volumes logiques en mode stripe

Lorsque vous écrivez des données sur un volume logique LVM, le système de fichiers disperse les données sur les volumes physiques sous-jacents. Vous pouvez contrôler la façon dont les données sont écrites sur les volumes physiques en créant un volume logique en mode stripe. Pour les lectures et écritures séquentielles volumineuses, cela peut améliorer l'efficacité des E/S de données.

Le striping améliore les performances en écrivant des données sur un nombre prédéterminé de volumes physiques avec une technique round-round. Avec le striping, les E/S peuvent être faites en parallèle. Dans certaines situations, cela peut résulter en un gain de performances quasiment linéaire pour chaque volume physique supplémentaire dans le stripe.

La figure suivante illustre des données séparées à travers 3 volumes physiques. Dans cette figure :

- le premier stripe de données est écrit sur PV1
- le second stripe de données est écrit sur PV2
- le troisième stripe de données est écrit sur PV3
- le quatrième stripe de données est écrit sur PV1

Dans un volume logique en mode stripe, la taille du stripe ne peut pas excéder la taille d'une extension.

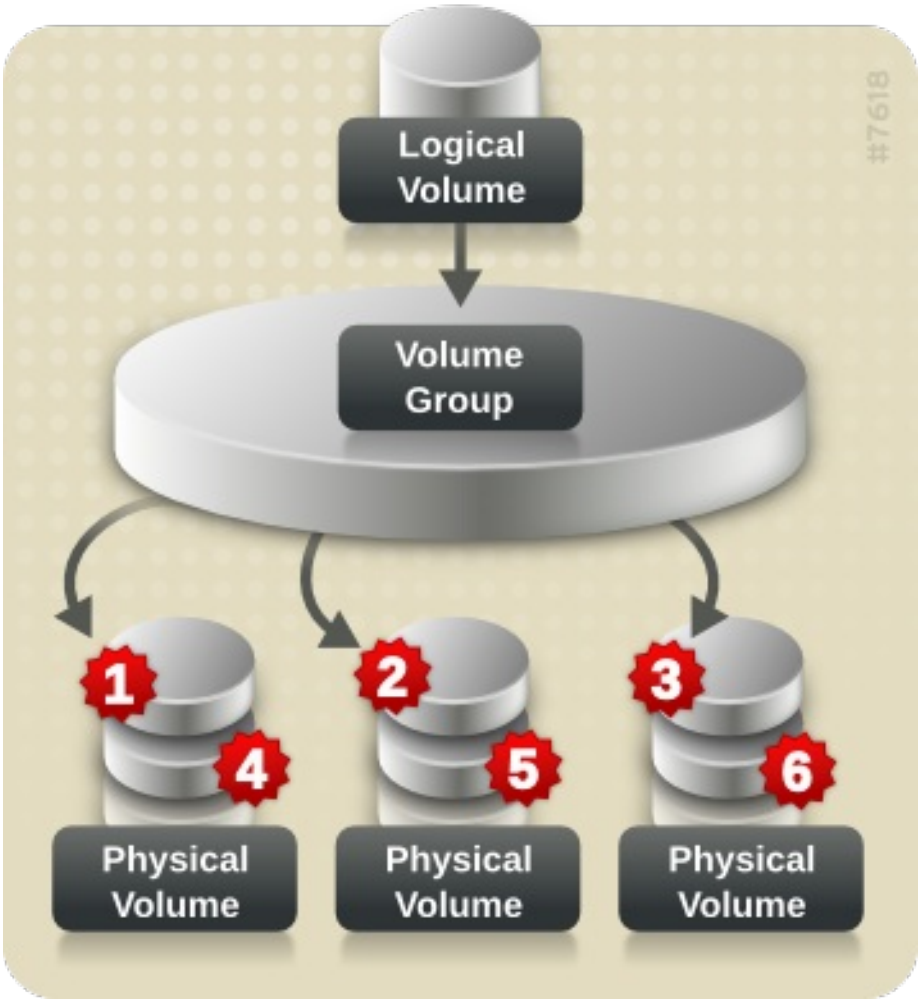


Figure 2.5. Striping des données à travers 3 PV

Striped logical volumes can be extended by concatenating another set of devices onto the end of the first set. In order extend a striped logical volume, however, there must be enough free space on the

underlying physical volumes that make up the volume group to support the stripe. For example, if you have a two-way stripe that uses up an entire volume group, adding a single physical volume to the volume group will not enable you to extend the stripe. Instead, you must add at least two physical volumes to the volume group. For more information on extending a striped volume, see Section 4.4.9, « Augmenter la taille d'un volume en mode stripe ».

2.3.3. Les volumes logiques en miroir

Un miroir gère des copies identiques de données sur différents périphériques. Lorsque des données sont écrites sur un périphérique, elles sont répliquées sur un deuxième périphérique, créant ainsi un miroir. Cela assure la protection des échecs de périphériques. Lorsqu'une section du miroir échoue, le volume logique devient un volume linéaire et reste accessible.

LVM supporte les volumes en miroir. Lorsque vous créez un volume logique en miroir, LVM s'assure que les données écrites sur un volume physique sous-jacent soient répliquées sur un volume physique séparé. Avec LVM, vous pouvez créer des volumes logiques en miroir avec plusieurs miroirs.

Un miroir LVM divise le périphérique qui est copié en zones qui ont généralement une taille de 512Mo. LVM maintient un petit fichier journal afin d'assurer le suivi des zones synchronisées avec le ou les mirois. Ce fichier journal peut être stocké en mémoire ou enregistré sur le disque afin d'être persistant entre les redémarrages.

Figure 2.6, « Mirrored Logical Volume » shows a mirrored logical volume with one mirror. In this configuration, the log is maintained on disk.

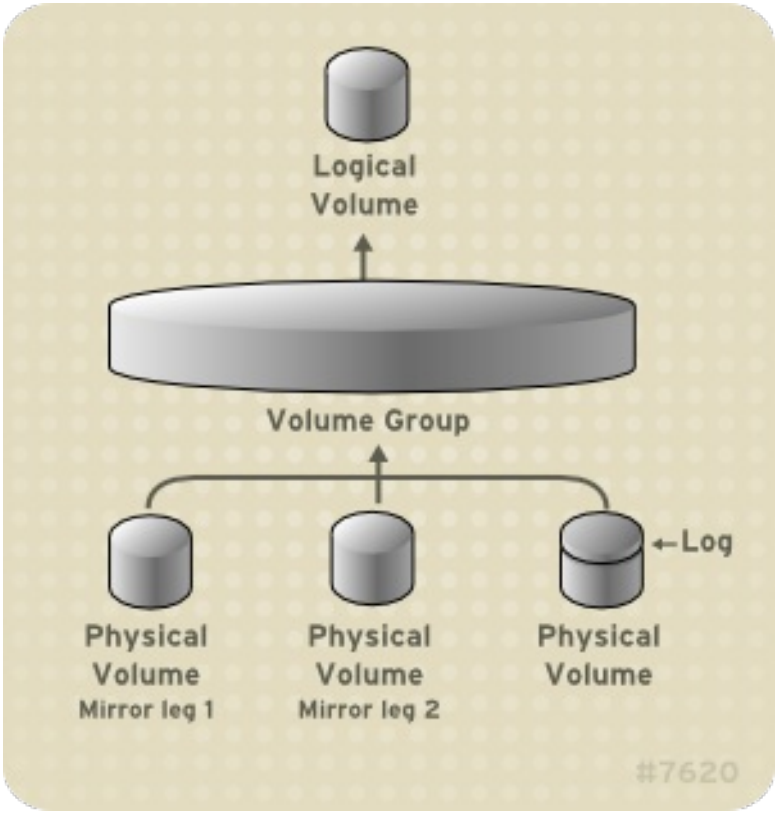
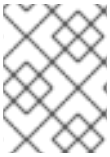


Figure 2.6. Mirrored Logical Volume



NOTE

Au moment de la sortie des notes de mise à jour de RHEL 5.3, les volumes logiques en miroir ne sont pas supportés dans un cluster.

For information on creating and modifying mirrors, see Section 4.4.1.3, « Création de volumes en miroir ».

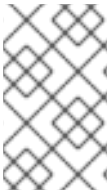
2.3.4. Les volumes d'instantanés

La fonctionnalité relative aux instantanés LVM permet de créer des images virtuelles de périphériques à un moment donné sans provoquer l'interruption d'un service. Lorsqu'un changement est effectué sur le périphérique d'origine après qu'un instantané ait été pris, la fonctionnalité relative aux instantanés LVM effectue une copie de la zone de données modifiée, telle qu'elle était avant son changement, afin de pouvoir reconstruire l'état du périphérique.

NOTE

Les instantanés LVM ne sont pas supportés à travers les noeuds d'un cluster.

Because a snapshot copies only the data areas that change after the snapshot is created, the snapshot feature requires a minimal amount of storage. For example, with a rarely updated origin, 3-5 % of the origin's capacity is sufficient to maintain the snapshot.



NOTE

Les copies d'instantanés pour un système de fichiers sont des copies virtuelles, et non pas un média de sauvegarde pour un système de fichiers. Les instantanés ne remplacent pas une procédure de sauvegarde.

Si un instantané dépasse sa capacité de stockage, ce dernier est ignoré. Cela permet de s'assurer qu'il y a suffisamment d'espace pour le périphérique d'origine. Vous devriez régulièrement contrôler la taille d'un instantané. Les instantanés sont pleinement redimensionnables. Ainsi, si vous avez une capacité de stockage adéquate, vous pouvez augmenter la taille du volume d'instantané afin d'éviter que ce dernier soit ignoré. À l'inverse, si vous trouvez que le volume d'instantané est plus grand qu'il ne devrait l'être, vous pouvez réduire la taille du volume pour libérer de l'espace requis par d'autres volumes logiques.

Lorsque vous créez l'instantané d'un système de fichiers, un accès au périphérique d'origine en lecture-écriture reste possible. Si une partie de l'instantané est changée, elle est marquée et ne sera jamais copiée à partir du volume d'origine.

Vous pouvez utiliser la fonctionnalité relative aux instantanés de différentes manières :

- Généralement, un instantané est pris afin d'effectuer une sauvegarde sur un volume logique sans arrêter le système en cours qui met à jour les données de façon continue.
- Vous pouvez exécuter la commande **fsck** sur l'instantané d'un système de fichiers pour vérifier l'intégrité du système de fichiers et déterminer si le système de fichiers d'origine doit être réparé.
- Étant donné que l'instantané est en lecture-écriture, vous pouvez tester les applications avec des données de production en prenant un instantané et en exécutant des tests sur cet instantané, sans modifier les données réelles.
- Vous pouvez créer des volumes pour une utilisation avec le contrôleur de machines virtuelles Xen. Vous pouvez utiliser la fonctionnalité relative aux instantanés afin de créer une image disque, prendre un instantané de cette image et le modifier pour une instance domU

particulière. Vous pouvez ensuite créer un autre instantané et le modifier pour une autre instance domU. Dans la mesure où le seul stockage utilisé est composé de sections qui ont été modifiées sur le périphérique d'origine ou l'instantané, la majorité du volume est partagée.

CHAPITRE 3. APERÇU GÉNÉRAL DE L'ADMINISTRATION LVM

This chapter provides an overview of the administrative procedures you use to configure LVM logical volumes. This chapter is intended to provide a general understanding of the steps involved. For specific step-by-step examples of common LVM configuration procedures, see Chapitre 5, *Exemples de configuration LVM*.

For descriptions of the CLI commands you can use to perform LVM administration, see Chapitre 4, *Administration LVM avec les commandes CLI*. Alternately, you can use the LVM GUI, which is described in Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM*.

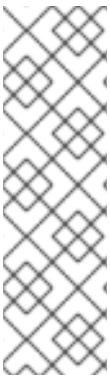
3.1. CRÉATION DE VOLUMES LVM DANS UN CLUSTER

To create logical volumes in a cluster environment, you use the Clustered Logical Volume Manager (CLVM), which is a set of clustering extensions to LVM. These extensions allow a cluster of computers to manage shared storage (for example, on a SAN) using LVM. In order to use CLVM, the Red Hat Cluster Suite software, including the **clmvd** daemon, must be started at boot time, as described in Section 1.3, « Le gestionnaire de volumes logiques clusterisé (CLVM) » .

La création de volumes logiques LVM dans un environnement en clusters est identique à la création de volumes logiques LVM sur un noeud unique. Il n'y aucune différence dans les commandes LVM elles-mêmes ou dans l'interface utilisateur graphique LVM. Afin d'activer les volumes LVM dans un cluster, l'infrastructure de cluster doit être en cours d'exécution et le cluster doit avoir le quorum.

CLVM requires changes to the **lvm.conf** file for cluster-wide locking. Information on configuring the **lvm.conf** file to support clustered locking is provided within the **lvm.conf** file itself. For information about the **lvm.conf** file, see Annexe B, *Les fichiers de configuration LVM*.

By default, logical volumes created with CLVM on shared storage are visible to all computers that have access to the shared storage. It is possible, however, to create logical volumes when the storage devices are visible to only one node in the cluster. It is also possible to change the status of a logical volume from a local volume to a clustered volume. For information, see Section 4.3.2, « Création de groupes de volumes au sein d'un cluster » and Section 4.3.7, « Changement des paramètres du groupe de volumes »



NOTE

Le stockage partagé utilisable dans Red Hat CLuster Suite demande de votre part d'exécuter le démon du gestionnaire de volumes logiques en clusters (**clvmd**) ou bien les agents HA-LVM (High Availability Logical Volume Management agents). Si vous n'êtes pas en position d'utiliser le démon **clvmd** ou HA-LVM pour des raisons opérationnelles, ou parce que vous n'avez pas les privilèges requis, vous ne pourrez pas utiliser d'instance-simple sur le disque partagé car cela risquerait de polluer les données. Si vous avez des questions à ce sujet, n'hésitez pas à contacter votre représentant commercial Red Hat.

Pour davantage d'informations sur la façon d'installer Red Hat Cluster Suite et de configurer une infrastructure de cluster, reportez-vous au guide de *Configuration et gestion d'un cluster Red Hat*.

3.2. APERÇU DE LA CRÉATION D'UN VOLUME LOGIQUE

Ce qui suit est un résumé des étapes à suivre afin de créer un volume logique LVM.

1. Initialisez les partitions que vous utiliserez pour le volume LVM en tant que volumes physiques (cela permet de les étiqueter).

2. Créez un groupe de volumes.

3. Créez un volume logique.

Après avoir créé le volume logique, vous pouvez créer et monter le système de fichiers. Les exemples dans ce document utilisent les systèmes de fichiers GFS.

1. Créez un système de fichiers GFS sur le volume logique avec la commande **gfs_mkfs**.

2. Créez un nouveau point de montage avec la commande **mkdir**. Dans un système clusterisé, créez le point de montage sur tous les noeuds du cluster.

3. Montez le système de fichiers. Vous pouvez ajouter une ligne au fichier **fstab** pour chaque noeud du système.

Alternativement, vous pouvez créer et monter le système de fichiers GFS avec l'interface utilisateur graphique LVM.

La création d'un volume LVM ne dépend pas de la machine, étant donné que la zone de stockage pour les informations d'installation LVM se trouve sur les volumes physiques et non pas sur la machine où le volume a été créé. Les serveurs qui utilisent le stockage ont des copies locales mais ils peuvent en recréer à partir du contenu des volumes physiques. Vous pouvez attacher des volumes physiques à un autre serveur si les versions LVM sont compatibles.

3.3. AUGMENTATION DE LA TAILLE D'UN SYSTÈME DE FICHIERS SUR UN VOLUME LOGIQUE

Pour augmenter la taille d'un système de fichiers sur un volume logique, effectuez les étapes suivantes :

1. Créez un nouveau volume physique.

2. Étendez le groupe de volumes contenant le volume logique avec le système de fichiers que vous voulez augmenter afin d'inclure le nouveau volume physique.

3. Étendez le volume logique pour inclure le nouveau volume physique.

4. Augmentez la taille du système de fichiers.

Si vous avez suffisamment d'espace non alloué dans le groupe de volumes, vous pouvez utiliser cet espace afin d'étendre le volume logique plutôt que d'effectuer les étapes 1 et 2.

3.4. SAUVEGARDE D'UN VOLUME LOGIQUE

À moins qu'elles soient désactivées dans le fichier **lvm.conf**, les sauvegardes de métadonnées et d'archives sont automatiquement créées lorsque la configuration d'un groupe de volumes ou d'un volume logique change. Par défaut, les sauvegardes de métadonnées sont stockées dans le fichier **/etc/lvm/backup** et les sauvegardes d'archives de métadonnées sont stockées dans **/etc/lvm/archive**. La durée pendant laquelle les archives de métadonnées sont stockées dans **/etc/lvm/archive** ainsi que le nombre de fichiers archives stockés sont spécifiés dans le fichier **lvm.conf**. Une sauvegarde système journalière devrait inclure le contenu du répertoire **/etc/lvm** dans la sauvegarde.

Notez que les sauvegardes de métadonnées n'enregistrent pas les données utilisateur et système contenues dans les volumes logiques.

You can manually back up the metadata to the **/etc/lvm/backup** file with the **vgcfgbackup** command. You can restore metadata with the **vgcfgrestore** command. The **vgcfgbackup** and **vgcfgrestore** commands are described in Section 4.3.12, « Sauvegarde des métadonnées d'un groupe de volumes ».

3.5. JOURNALISATION

Toutes les sorties de messages passent à travers un module de journalisation avec des options de niveaux d'enregistrement indépendantes pour :

- sortie/erreur standard
- syslog
- fichier journal
- fonction de journalisation externe

The logging levels are set in the **/etc/lvm/lvm.conf** file, which is described in Annexe B, *Les fichiers de configuration LVM*.

CHAPITRE 4. ADMINISTRATION LVM AVEC LES COMMANDES CLI

Ce chapitre résume les tâches d'administration individuelles que vous pouvez effectuer avec les commandes CLI (de l'anglais Command Line Interface) LVM afin de créer et maintenir les volumes logiques.

NOTE

If you are creating or modifying an LVM volume for a clustered environment, you must ensure that you are running the **clvmd** daemon. For information, see see Section 3.1, « Création de volumes LVM dans un cluster ».

4.1. UTILISATION DES COMMANDES CLI

Il y a plusieurs fonctionnalités générales à toutes les commandes CLI LVM.

Lorsque des tailles sont requises pour l'argument d'une ligne de commande, les unités peuvent toujours être spécifiées explicitement. Si vous ne spécifiez pas d'unité, une valeur par défaut est définie, habituellement en Ko ou Mo. Les commandes CLI LVM n'acceptent pas les fractions.

Lorsque vous spécifiez des unités via un argument de ligne de commande, LVM n'est pas sensible à la casse ; spécifier M ou m est par exemple équivalent et les puissances de 2 (multiples de 1024) sont utilisées. Cependant, lors de la spécification de l'argument **--units** dans une commande, les lettres minuscules représentent les unités en multiples de 1024 alors que les lettres majuscules représentent les unités en multiples de 1000.

Lorsque les commandes utilisent les noms de groupes de volumes et de volumes logiques en tant qu'arguments, le nom complet du chemin d'accès est optionnel. Un volume logique appelé **lvol0** dans un groupe de volumes appelé **vg0** peut être spécifié comme **vg0/lvol0**. Si une liste de groupes de volumes est laissée vide, une liste de tous les groupes de volumes sera substituée. Si, à la place d'une liste de volumes logiques, un groupe de volumes est spécifié, une liste de tous les volumes logique du groupe de volumes sera substituée. Par exemple, la commande **lvdisplay vg0** affichera tous les volumes logiques du groupe de volumes **vg0**.

Toutes les commandes LVM acceptent un argument **-v** qui peut être saisi plusieurs fois afin d'améliorer la verbosité d'une sortie. Les exemples suivants illustrent la sortie par défaut de la commande **lvcreate**.

```
# lvcreate -L 50MB new_vg
  Rounding up size to full physical extent 52.00 MB
  Logical volume "lvol0" created
```

La commande suivante illustre la sortie de la commande **lvcreate** avec l'argument **-v**.

```
# lvcreate -v -L 50MB new_vg
    Finding volume group "new_vg"
  Rounding up size to full physical extent 52.00 MB
    Archiving volume group "new_vg" metadata (seqno 4).
    Creating logical volume lvol0
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
    Found volume group "new_vg"
    Creating new_vg-lvol0
```

```
    Loading new_vg-lvol0 table
    Resuming new_vg-lvol0 (253:2)
    Clearing start of logical volume "lvol0"
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
  Logical volume "lvol0" created
```

Vous pourriez également utiliser l'argument **-vv**, **-vvv** ou **-vvvv** pour afficher davantage de détails à propos de l'exécution de cette commande. L'argument **-vvvv** fournit actuellement la plus grande quantité d'informations. L'exemple suivant illustre uniquement les premières lignes de la sortie pour la commande **lvcreate** avec l'argument **-vvvv** spécifié.

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913         Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:916         O_DIRECT will be used
#config/config.c:864       Setting global/locking_type to 1
#locking/locking.c:138       File-based locking selected.
#config/config.c:841       Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358       Getting target version for linear
#ioctl/libdm-iface.c:1569        dm version   OF   [16384]
#ioctl/libdm-iface.c:1569        dm versions   OF   [16384]
#activate/activate.c:358       Getting target version for striped
#ioctl/libdm-iface.c:1569        dm versions   OF   [16384]
#config/config.c:864       Setting activation/mirror_region_size to 512
...
```

Vous pouvez afficher l'aide de n'importe quelle commande CLI en utilisant l'argument **--help**.

```
commandname --help
```

Pour afficher la page de manuel d'une commande, exécutez la commande **man** :

```
man commandname
```

La commande **man lvm** fournit des informations générales en ligne à propos de LVM.

All LVM objects are referenced internally by a UUID, which is assigned when you create the object. This can be useful in a situation where you remove a physical volume called **/dev/sdf** which is part of a volume group and, when you plug it back in, you find that it is now **/dev/sdk**. LVM will still find the physical volume because it identifies the physical volume by its UUID and not its device name. For information on specifying the UUID of a physical volume when creating a physical volume, see see Section 6.4, « Recupération des métadonnées du volume physique ».

4.2. ADMINISTRATION DE VOLUMES PHYSIQUES

Cette section décrit les commandes qui permettent d'administrer les volumes physiques.

4.2.1. Création de volumes physiques

Les sous-sections suivantes décrivent les commandes utilisées pour la création de volumes physiques.

4.2.1.1. Paramétrage du type de partition

Si vous utilisez un périphérique disque entier pour votre volume physique, le disque ne doit pas avoir

de table de partitions. Pour les partitions de disque DOS, l'id de la partition doit être configuré sur 0x8e en utilisant la commande **fdisk** ou **cfdisk** ou un équivalent. Pour les périphériques disque entiers, seule la table de partition doit être supprimée, ce qui détruira efficacement toutes les données sur ce disque. Vous pouvez supprimer une table de partitions existante en remettant à zéro le premier secteur avec la commande suivante :

```
dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

4.2.1.2. Initialisation des volumes physiques

Utilisez la commande **pvcreate** pour initialiser un périphérique bloc à utiliser en tant que volume physique. L'initialisation est analogue au formatage d'un système de fichiers.

La commande suivante initialise **/dev/sdd1**, **/dev/sde1** et **/dev/sdf1** pour l'utilisation en tant que volumes physiques LVM.

```
pvcreate /dev/sdd1 /dev/sde1 /dev/sdf1
```

Pour initialiser les partitions plutôt que les disques entiers, exécutez la commande **pvcreate** sur la partition. L'exemple suivant initialise **/dev/hdb1** comme un volume physique LVM pour l'utiliser par la suite dans le cadre d'un volume logique LVM.

```
pvcreate /dev/hdb1
```

4.2.1.3. Recherche de périphériques blocs

Vous pouvez rechercher des périphériques blocs qui peuvent être utilisés comme des volumes physiques avec la commande **lvmdiskscan**, comme illustré dans l'exemple suivant.

```
# lvmdiskscan
/dev/ram0                     [       16.00 MB]
  /dev/sda                    [       17.15 GB]
  /dev/root                   [       13.69 GB]
  /dev/ram                    [       16.00 MB]
  /dev/sda1                   [       17.14 GB] LVM physical volume
  /dev/VolGroup00/LogVol01    [      512.00 MB]
  /dev/ram2                   [       16.00 MB]
  /dev/new_vg/lvol0           [       52.00 MB]
  /dev/ram3                   [       16.00 MB]
  /dev/pkl_new_vg/sparkie_lv  [        7.14 GB]
  /dev/ram4                   [       16.00 MB]
  /dev/ram5                   [       16.00 MB]
  /dev/ram6                   [       16.00 MB]
  /dev/ram7                   [       16.00 MB]
  /dev/ram8                   [       16.00 MB]
  /dev/ram9                   [       16.00 MB]
  /dev/ram10                  [       16.00 MB]
  /dev/ram11                  [       16.00 MB]
  /dev/ram12                  [       16.00 MB]
  /dev/ram13                  [       16.00 MB]
  /dev/ram14                  [       16.00 MB]
  /dev/ram15                  [       16.00 MB]
  /dev/sdb                    [       17.15 GB]
```

```
  /dev/sdb1                     [       17.14 GB] LVM physical volume
  /dev/sdc                      [       17.15 GB]
  /dev/sdc1                     [       17.14 GB] LVM physical volume
  /dev/sdd                      [       17.15 GB]
  /dev/sdd1                     [       17.14 GB] LVM physical volume
  7 disks
  17 partitions
  0 LVM physical volume whole disks
  4 LVM physical volumes
```

4.2.2. Affichage des volumes physiques

Vous pouvez utiliser trois commandes afin d'afficher les propriétés des volumes physiques LVM : **pvs**, **pvdisplay** et **pvscan**.

The **pvs** command provides physical volume information in a configurable form, displaying one line per physical volume. The **pvs** command provides a great deal of format control, and is useful for scripting. For information on using the **pvs** command to customize your output, see Section 4.9, « Rapport personnalisé pour LVM ».

La commande **pvdisplay** fournit une sortie verbeuse sur plusieurs lignes pour chaque volume physique. Elle affiche des propriétés physiques (taille, extensions, groupe de volumes, etc.) dans un format fixe.

Les exemples suivants illustrent la sortie de la commande **pvdisplay** pour un volume physique unique.

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               new_vg
  PV Size               17.14 GB / not usable 3.40 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              4388
  Free PE               4375
  Allocated PE          13
  PV UUID               Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
```

La commande **pvscan** recherche les volumes physiques à travers tous les périphériques blocs LVM supportés dans le système.

La commande suivante illustre tous les périphériques physiques trouvés :

```
# pvscan
 PV /dev/sdb2   VG vg0   lvm2 [964.00 MB / 0   free]
 PV /dev/sdc1   VG vg0   lvm2 [964.00 MB / 428.00 MB free]
 PV /dev/sdc2            lvm2 [964.84 MB]
 Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

You can define a filter in the **lvm.conf** so that this command will avoid scanning specific physical volumes. For information on using filters to control which devices are scanned, see Section 4.6, « Contrôler l'analyse des périphériques LVM avec les filtres ».

4.2.3. Empêcher l'allocation sur un volume physique

Vous pouvez empêcher l'allocation des extensions physiques sur l'espace libre d'un ou plusieurs volumes physiques avec la commande **pvchange**. Cela peut être nécessaire s'il y a des erreurs de disque ou si vous prévoyez de supprimer le volume physique.

La commande suivante interdit l'allocation des extensions physiques sur **/dev/sdk1**.

```
pvchange -x n /dev/sdk1
```

Vous pouvez également utiliser les arguments **-xy** de la commande **pvchange** pour autoriser l'allocation là où elle a été précédemment interdite.

4.2.4. Redimensionnement de volumes physiques

Si vous devez changer la taille d'un périphérique bloc sous-jacent pour une raison ou pour une autre, utilisez la commande **pvresize** pour mettre à jour LVM avec la nouvelle taille. Vous pouvez exécuter la commande pendant que LVM utilise le volume physique.

4.2.5. Suppression de volumes physiques

Si un périphérique n'est plus utilisé par LVM, vous pouvez supprimer l'étiquette LVM avec la commande **pvremove**. L'exécution de la commande **pvremove** remet à zéro les métadonnées LVM sur un volume physique vide.

If the physical volume you want to remove is currently part of a volume group, you must remove it from the volume group with the **vgreduce** command, as described in Section 4.3.6, « Suppression de volumes physiques à partir d'un groupe de volumes ».

```
# pvremove /dev/ram15
  Labels on physical volume "/dev/ram15" successfully wiped
```

4.3. ADMINISTRATION D'UN GROUPE DE VOLUMES

Cette section décrit les commandes qui permettent d'administrer les groupes de volumes.

4.3.1. Création de groupes de volumes

To create a volume group from one or more physical volumes, use the **vgcreate** command. The **vgcreate** command creates a new volume group by name and adds at least one physical volume to it.

La commande suivante crée un groupe de volumes appelé **vg1** qui contient les volumes physiques **/dev/sdd1** et **/dev/sde1**.

```
vgcreate vg1 /dev/sdd1 /dev/sde1
```

Lorsque des volumes physiques sont utilisés pour créer un groupe de volumes, son espace disque est divisé par défaut en des extensions de 4Mo. Cette extension correspond à la taille minimum par laquelle le volume logique peut être augmenté ou diminué. Les extensions avec une taille plus importante n'auront pas d'impact sur les performances des E/S du volume logique.

Si la valeur par défaut ne correspond pas à vos besoins, vous pouvez spécifier la taille d'extension avec la commande **vgcreate** et l'argument **-s** . Vous pouvez spécifier des limites sur le nombre de volumes

physiques ou logiques que le groupe de volumes peut avoir en utilisant les arguments **-p** et **-l** de la commande **vgcreate**.

Par défaut, un groupe de volumes alloue des extensions physiques conformément à des règles communes. Par exemple ne pas placer de stripes parallèles sur le même volume physique. Il s'agit de la politique d'allocation **normal**. Vous pouvez utiliser l'argument **--alloc** de la commande **vgcreate** pour spécifier une politique d'allocation **contiguous**, **anywhere** ou **cling**.

La politique **contiguous** requiert que les nouvelles extensions soient adjacentes aux extensions existantes. S'il y a suffisamment d'extensions libres pour satisfaire une requête d'allocation mais qu'une politique d'allocation **normal** ne les utilise pas , la politique d'allocation **anywhere** les utilisera, même si le fait de placer deux stripes sur le même volume physique réduit les performances. La politique **cling** place de nouvelles extensions sur le même volume physique que les extensions existantes dans le même stripe du volume logique. Ces politiques peuvent être changées en utilisant la commande **vgchange**.

En général, les politiques d'allocation autres que **normal** sont requises uniquement dans des situations spéciales où vous devez spécifier une allocation d'extension inhabituelle ou non standard.

Les groupes LVM et les volumes logiques sous-jacents sont inclus dans le répertoire des fichiers spéciaux de périphériques **/dev** avec la structure suivante :

```
/dev/vg/lv/
```

Par exemple, si vous créez deux groupes de volumes **myvg1** et **myvg2**, avec pour chacun d'entre eux trois volumes logiques appelés **lvo1**, **lvo2** et **lvo3**, 6 fichiers spéciaux de périphériques seront créés :

```
/dev/myvg1/lv01
/dev/myvg1/lv02
/dev/myvg1/lv03
/dev/myvg2/lv01
/dev/myvg2/lv02
/dev/myvg2/lv03
```

La taille maximum d'un périphérique avec LVM est de 8 Exaoctets sur les CPU 64 bits.

4.3.2. Création de groupes de volumes au sein d'un cluster

Vous pouvez créer des groupes de volumes dans un environnement clusterisé grâce à la commande **vgchange**, de la même manière que vous les créez sur un noeud simple.

Par défaut, les groupes de volumes créés avec CLVM sur la mémoire partagée, sont visibles par tous les ordinateurs qui ont accès à la mémoire partagée. Il est possible, cependant, de créer des groupes de volumes locaux, visibles que par un seul noeud du cluster, en utilisant l'option **-c n** de la commande **vgcreate**.

La commande suivante, lorsqu'elle est exécutée dans un environnement clusterisé, crée un groupe de volumes local par rapport au noeud à partir duquel la commande a été exécutée. La commande crée un volume local appelé **vg1** qui contient les volumes physiques **/dev/sdd1** et **/dev/sde1**.

```
vgcreate -c n vg1 /dev/sdd1 /dev/sde1
```

You can change whether an existing volume group is local or clustered with the **-c** option of the **vgchange** command, which is described in Section 4.3.7, « Changement des paramètres du groupe de volumes ».

Vous pouvez vérifier si un groupe de volumes existant est clusterisé par la commande **vgs**, qui affiche l'attribut **c** quand le volume est clusterisé. La commande suivante affiche les attributs des groupes de volumes **VolGroup00** et **testvg1**. Dans cet exemple, **VolGroup00** n'est pas clusterisé, alors que **testvg1** l'est, comme indiqué par l'attribut **c** sous l'en-tête **Attr**.

```
[root@doc-07]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  VolGroup00      1   2   0 wz--n- 19.88G    0
  testvg1         1   1   0 wz--nc 46.00G 8.00M
```

For more information on the **vgs** command, see Section 4.3.4, « Affichage des groupes de volumes »Section 4.9, « Rapport personnalisé pour LVM » , and the **vgs** man page.

4.3.3. Ajout de volumes physiques à un groupe de volumes

To add additional physical volumes to an existing volume group, use the **vgextend** command. The **vgextend** command increases a volume group's capacity by adding one or more free physical volumes.

La commande suivante ajoute le volume physique **/dev/sdf1** au groupe de volumes **vg1**.

```
vgextend vg1 /dev/sdf1
```

4.3.4. Affichage des groupes de volumes

Vous pouvez utiliser deux commandes pour afficher les propriétés d'un groupe de volumes : **vgs** et **vgdisplay**.

The **vgscan** command will also display the volume groups, although its primary purpose is to scan all the disks for volume groups and rebuild the LVM cache file. For information on the **vgscan** command, see Section 4.3.5, « Analyse des disques pour les groupes de volumes afin de construire le fichier de cache ».

The **vgs** command provides volume group information in a configurable form, displaying one line per volume group. The **vgs** command provides a great deal of format control, and is useful for scripting. For information on using the **vgs** command to customize your output, see Section 4.9, « Rapport personnalisé pour LVM ».

La commande **vgdisplay** affiche les propriétés des groupes de volumes (la taille, les extensions, le nombre de volumes physiques, etc.) dans un format fixe. L'exemple suivant illustre la sortie d'une commande **vgdisplay** pour le groupe de volumes **new_vg**. Si vous ne spécifiez pas un groupe de volumes, tous les groupes de volumes existants sont affichés.

```
# vgdisplay new_vg
  --- Volume group ---
  VG Name               new_vg
  System ID
  Format                lvm2
  Metadata Areas        3
```

```
  Metadata Sequence No  11
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               51.42 GB
  PE Size               4.00 MB
  Total PE              13164
  Alloc PE / Size       13 / 52.00 MB
  Free  PE / Size       13151 / 51.37 GB
  VG UUID               jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

4.3.5. Analyse des disques pour les groupes de volumes afin de construire le fichier de cache

La commande **vgscan** analyse tous les périphériques disques supportés dans le système afin de rechercher les volumes physiques LVM et les groupes de volumes. Cela permet de construire le fichier de cache LVM dans **/etc/lvm/.cache**, qui maintient une liste des périphériques LVM en cours.

LVM exécute automatiquement la commande **vgscan** au démarrage du système et à d'autres moments pendant les opérations LVM, par exemple lorsque vous exécutez la commande **vgcreate** ou lorsque LVM détecte une incohérence. Vous pourriez devoir exécuter la commande **vgscan** manuellement lorsque vous changez la configuration de votre matériel afin que les périphériques qui n'étaient pas présents au démarrage puissent être visibles par le système. Cela peut être nécessaire, par exemple, lorsque vous rajoutez de nouveaux disques au système sur un SAN ou si vous branchez "à chaud" un nouveau disque qui a été étiqueté en tant que volume physique.

You can define a filter in the **lvm.conf** file to restrict the scan to avoid specific devices. For information on using filters to control which devices are scanned, see Section 4.6, « Contrôler l'analyse des périphériques LVM avec les filtres ».

L'exemple suivant illustre la sortie de la commande **vgscan**.

```
# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "new_vg" using metadata type lvm2
  Found volume group "officevg" using metadata type lvm2
```

4.3.6. Suppression de volumes physiques à partir d'un groupe de volumes

To remove unused physical volumes from a volume group, use the **vgreduce** command. The **vgreduce** command shrinks a volume group's capacity by removing one or more empty physical volumes. This frees those physical volumes to be used in different volume groups or to be removed from the system.

Avant de supprimer un volume physique d'un groupe de volumes, vous pouvez utiliser la commande **pvdisplay** afin de vous assurer que le volume physique ne soit pas utilisé par un volume logique.

```
# pvdisplay /dev/hda1
```

```
-- Physical volume ---
PV Name               /dev/hda1
VG Name               myvg
PV Size               1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#                   1
PV Status             available
Allocatable           yes (but full)
Cur LV                1
PE Size (KByte)       4096
Total PE              499
Free PE               0
Allocated PE          499
PV UUID               Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

Si le volume physique est encore utilisé, vous devrez migrer les données sur un autre volume physique en utilisant la commande **pvmove**. Utilisez ensuite la commande **vgreduce** pour supprimer le volume physique :

La commande suivante supprime le volume physique **/dev/hda1** du groupe de volumes **my_volume_group**.

```
# vgreduce my_volume_group /dev/hda1
```

4.3.7. Changement des paramètres du groupe de volumes

There are several volume group parameters that you can change for an existing volume group with the **vgchange** command. Primarily, however, this command is used to deactivate and activate volume groups, as described in Section 4.3.8, « Activation et désactivation des groupes de volumes » ,

La commande suivante change le nombre maximum de volumes logiques du groupe de volumes **vg00** à 128.

```
vgchange -l 128 /dev/vg00
```

Pour obtenir une description des paramètres du groupe de volumes vous pouvez changer la commande **vgchange**, reportez-vous à la page de manuel **vgchange**(8).

4.3.8. Activation et désactivation des groupes de volumes

Lorsque vous créez un groupe de volumes, il est par défaut activé. Cela signifie que les volumes logiques de ce groupe sont accessibles et sujets au changement.

Il y a plusieurs circonstances pour lesquelles vous devriez rendre un groupe de volumes inactif et ainsi non perceptible par le noyau. Pour activer ou désactiver un groupe de volumes, utilisez l'argument **-a** (**--available**) de la commande **vgchange**.

L'exemple suivant désactive le groupe de volumes **my_volume_group**.

```
vgchange -a n my_volume_group
```

Si le verrouillage en cluster est activé, ajoutez "e" afin d'activer ou désactiver un groupe de volumes exclusivement sur un noeud ou "l" pour activer ou désactiver un groupe de volumes sur le noeud local uniquement. Les volumes logiques avec des instantanés d'hôte unique sont toujours activés exclusivement car ils peuvent être utilisés sur un seul noeud à la fois.

You can deactivate individual logical volumes with the **lvchange** command, as described in Section 4.4.4, « Changement des paramètres d'un groupe de volumes logiques » , For information on activating logical volumes on individual nodes in a cluster, see Section 4.8, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

4.3.9. Suppression de groupes de volumes

Pour supprimer un groupe de volumes qui ne contient pas de volumes logiques, utilisez la commande **vgremove**.

```
# vgremove officevg
  Volume group "officevg" successfully removed
```

4.3.10. Fractionnement d'un groupe de volumes

Pour fractionner les volumes physiques d'un groupe de volumes et créer un nouveau groupe de volumes, utilisez la commande **vgsplit**.

Les volumes logiques ne peuvent pas être séparés entre les groupes de volumes. Chaque volume logique existant doit être entièrement sur les volumes physiques formant l'ancien ou le nouveau groupe de volumes. Si nécessaire, vous pouvez cependant utiliser la commande **pvmove** pour forcer le fractionnement.

L'exemple suivant sépare le nouveau groupe de volumes **smallvg** du groupe de volumes d'origine **bigvg**.

```
# vgsplit bigvg smallvg /dev/ram15
  Volume group "smallvg" successfully split from "bigvg"
```

4.3.11. Combinaison de groupes de volumes

Two combine two volume groups into a single volume group, use the **vgmerge** command. You can merge an inactive "source" volume with an active or an inactive "destination" volume if the physical extent sizes of the volume are equal and the physical and logical volume summaries of both volume groups fit into the destination volume groups limits.

La commande suivante fusionne le groupe de volumes inactif **my_vg** avec le groupe de volumes actif ou inactif **databases** en rapportant des informations d'exécution détaillées.

```
vgmerge -v databases my_vg
```

4.3.12. Sauvegarde des métadonnées d'un groupe de volumes

Les sauvegardes de métadonnées et d'archives sont automatiquement créées à chaque changement de configuration d'un groupe de volumes ou d'un volume logique, à moins que cela soit désactivé dans le fichier **lvm.conf**. Par défaut, la sauvegarde de métadonnées est stockée dans **/etc/lvm/backup** et les archives de métadonnées sont stockées dans **/etc/lvm/archive**. Vous pouvez manuellement sauvegarder les métadonnées dans **/etc/lvm/backup** avec la commande **vgcfgbackup**.

La commande **vgcfrestore** restaure les métadonnées d'un groupe de volumes à partir de l'archive sur tous les volumes physiques du groupe de volumes.

For an example of using the **vgcfgrestore** command to recover physical volume metadata, see Section 6.4, « Recupération des métadonnées du volume physique ».

4.3.13. Renommer un groupe de volumes

Utilisez la commande **vgrename** pour renommer un groupe de volumes existant.

Les deux commandes suivantes renomment le groupe de volumes existant de **vg02** à **my_volume_group**.

```
vgrename /dev/vg02 /dev/my_volume_group
```

```
vgrename vg02 my_volume_group
```

4.3.14. Déplacer un groupe de volumes sur un autre système

Vous pouvez déplacer un groupe de volumes LVM sur un autre système. Pour ce faire, il est recommandé d'utiliser les commandes **vgexport** et **vgimport**.

La commande **vgexport** rend un groupe de volumes inactif inaccessible au système afin que vous puissiez détacher ses volumes physiques. La commande **vgimport** rend un groupe de volumes accessible à une machine après que la commande **vgexport** l'ait rendu inactif.

Pour déplacer un groupe de volumes d'un système à un autre, effectuez les étapes suivantes :

1. Assurez-vous qu'aucun utilisateur ne soit en train d'accéder aux fichiers sur les volumes actifs du groupe de volumes, puis démontez les volumes logiques.

2. Utilisez l'argument **-a n** de la commande **vgchange** pour marquer le groupe de volumes comme étant inactif, afin d'empêcher toute activité sur le groupe de volumes.

3. Utilisez la commande **vgexport** afin d'exporter le groupe de volumes. Cela l'empêche d'être accédé par le système à partir duquel vous le supprimez.

   Après avoir exporté le groupe de volumes, le volume physique apparaîtra comme faisant partie d'un groupe de volumes exporté lorsque vous exécuterez la commande **pvscan**, comme dans l'exemple suivant.

   ```
   [root@tng3-1]# pvscan
     PV /dev/sda1    is in exported VG myvg [17.15 GB / 7.15 GB free]
     PV /dev/sdc1    is in exported VG myvg [17.15 GB / 15.15 GB free]
     PV /dev/sdd1   is in exported VG myvg [17.15 GB / 15.15 GB free]
     ...
   ```

   Lorsque le système est éteint, vous pouvez débrancher les disques qui composent le groupe de volumes et les connecter au nouveau système.

4. Lorsque les disques sont connectés au nouveau système, utilisez la commande **vgimport** pour importer le groupe de volumes et le rendre ainsi accessible au nouveau système.

5. Activez le groupe de volumes avec l'argument **-a y** de la commande **vgchange**.

6. Montez le système de fichiers afin qu'il puisse être utilisé.

4.3.15. Recréation du répertoire d'un groupe de volumes

Pour recréer le répertoire d'un groupe de volumes et les fichiers spéciaux d'un volume logique, utilisez la commande **vgmknodes**. Cette commande vérifie les fichiers spéciaux LVM2 du répertoire **/dev** qui sont requis par les volumes logiques actifs. Elle crée tout fichier spécial manquant et supprime ceux qui ne sont plus utilisés.

Vous pouvez incorporer la commande **vgmknodes** dans la commande **vgscan** en spécifiant l'argument **--mknodes**.

4.4. ADMINISTRATION DE VOLUMES LOGIQUES

Cette section décrit le commandes permettant de mettre en oeuvre les différents aspects de l'administration de volumes logiques.

4.4.1. Création de volumes logiques

Pour créer un volume logique, utilisez la commande **lvcreate**. Vous pouvez créer des volumes linéaires, des volumes en mode stripe et des volumes en miroir, comme décrit dans les sous-sections suivantes.

Si vous ne spécifiez pas de nom pour le volume logique, le nom par défaut **lvol#** est utilisé où **#** correspond au numéro interne du volume logique.

Les sections suivantes fournissent des exemples de création de volumes logiques pour les trois types de volumes logiques que vous pouvez créer avec LVM.

4.4.1.1. Création de volumes linéaires

Lorsque vous créez un volume logique, celui-ci est issu d'un groupe de volumes en utilisant les extensions libres sur les volumes physiques qui composent le groupe de volumes. Habituellement, les volumes logiques utilisent tout l'espace disponible sur les volumes physiques sous-jacents. La modification du volume logique libère et réalloue l'espace dans les volumes physiques.

La commande suivante crée un volume logique de 10 giga-octets dans le groupe de volumes **vg1**.

```
lvcreate -L 10G vg1
```

La commande suivante crée un volume logique linéaire de 1500 méga-octets appelé **testlv** dans le groupe de volumes **testvg**, créant ainsi le périphérique bloc **/dev/testvg/testlv**.

```
lvcreate -L1500 -n testlv testvg
```

La commande suivante crée un volume logique de 50 giga-octets appelé **gfslv** à partir des extensions libres du groupe de volumes **vg0**.

```
lvcreate -L 50G -n gfslv vg0
```

Vous pouvez utiliser l'argument **-l** de la commande **lvcreate** pour spécifier la taille, en extensions, du volume logique. Vous pouvez également utiliser cet argument pour spécifier le pourcentage du groupe de volumes à utiliser pour le volume logique. La commande suivante crée un volume logique appellé **mylv** qui utilise 60% de l'espace total dans le groupe de volumes **testvol**.

```
lvcreate -l 60%VG -n mylv testvg
```

Vous pouvez également utiliser l'argument **-l** de la commande **lvcreate** pour spécifier le pourcentage de l'espace libre restant dans le groupe de volumes comme la taille du volume logique. La commande suivante crée un volume logique appelé **yourlv** qui utilise tout l'espace non alloué dans le groupe de volumes **testvol**.

```
lvcreate -l 100%FREE -n yourlv testvg
```

You can use **-l** argument of the **lvcreate** command to create a logical volume that uses the entire volume group. Another way to create a logical volume that uses the entire volume group is to use the **vgdisplay** command to find the "Total PE" size and to use those results as input to the the **lvcreate** command.

Les commandes suivantes créent un volume logique appelé **mylv** qui remplit le groupe de volumes **testvg**.

```
# vgdisplay testvg | grep "Total PE"
Total PE              10230
# lvcreate -l 10230 testvg -n mylv
```

The underlying physical volumes used to create a logical volume can be important if the physical volume needs to be removed, so you may need to consider this possibility when you create the logical volume. For information on removing a physical volume from a volume group, see Section 4.3.6, « Suppression de volumes physiques à partir d'un groupe de volumes ».

Pour créer un volume logique alloué à partir d'un volume physique spécifique du groupe de volumes, spécifiez le ou les volumes physiques à la fin de la ligne de commande **lvcreate**. La commande suivante crée un volume logique appelé **testlv** dans le groupe de volumes **testvg**, alloué à partir du volume physique **/dev/sdg1**.

```
lvcreate -L 1500 -ntestlv testvg /dev/sdg1
```

Vous pouvez spécifier les extensions d'un volume physique devant être utilisées pour le volume physique. La commande suivante crée un volume logique linéaire à partir des extensions 0 à 25 du volume physique **/dev/sda1** et à partir des extensions 50 à 125 du volume physique **/dev/sdb1** dans le groupe de volumes **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25 /dev/sdb1:50-125
```

L'exemple suivant crée un volume logique linéaire à partir des extensions 0 à 25 du volume physique **/dev/sda1** et continue ensuite à étendre le volume logique à l'extension 100.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25:100-
```

The default policy for how the extents of a logical volume are allocated is **inherit**, which applies the same policy as for the volume group. These policies can be changed using the **lvchange** command. For information on allocation policies, see Section 4.3.1, « Création de groupes de volumes ».

4.4.1.2. Création de volumes en mode stripe

For large sequential reads and writes, creating a striped logical volume can improve the efficiency of the data I/O. For general information about striped volumes, see Section 2.3.2, « Les volumes logiques en mode stripe ».

Lorsque vous créez un volume logique en mode stripe, vous pouvez spécifier le nombre de stripes avec l'argument **-i** de la commande **lvcreate**. Cela permet de déterminer le nombre de volumes physiques sur lesquels le volume logique sera "stripé". Le nombre de stripes ne peut pas être supérieur au nombre de volumes physiques dans le groupe de volumes (à moins que l'argument **--alloc anywhere** soit utilisé).

Si les périphériques physiques sous-jacents qui composent un volume logique en mode stripe ont différentes tailles, la taille maximale du volume en mode stripe est déterminée par le plus petit périphérique sous-jacent. Par exemple, dans un stripe à deux branches, la taille maximale correspond à deux fois la taille du plus petit périphérique. Dans un stripe à trois branches, la taille maximale correspond à trois fois la taille du plus petit périphérique.

La commande suivante crée un volume logique en mode stripe à travers deux volumes physiques avec un stripe de 64Ko. Le volume logique a une taille de 50 giga-octets, il s'appelle **gfslv** et est issu du groupe de volumes **vg0**.

```
lvcreate -L 50G -i2 -I64 -n gfslv vg0
```

Comme avec les volumes linéaires, vous pouvez spécifier les extensions du volume physique que vous utilisez pour le stripe. La commande suivante crée un volume en mode stripe avec une taille de 100 extensions qui se trouve sur deux volumes physiques. Il s'appelle **stripelv** et fait partie du groupe de volumes **testvg**. Le stripe utilisera les secteurs 0-50 de **/dev/sda1** et les secteurs 50-100 de **/dev/sdb1**.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-50 /dev/sdb1:50-100
  Using default stripesize 64.00 KB
  Logical volume "stripelv" created
```

4.4.1.3. Création de volumes en miroir

Lorsque vous créez un volume en miroir, vous spécifiez le nombre de copies de données à effectuer avec l'argument **-m** de la commande **lvcreate**. Si vous spécifiez l'argument **-m1**, vous créez un miroir qui produira deux copies du système de fichiers : un volume logique linéaire plus une copie. De façon similaire, spécifiez l'argument **-m2** pour créer deux miroirs qui produiront trois copies du système de fichiers.

La commande suivante crée un volume logique en miroir avec un seul miroir. Le volume a une taille de 50 giga-octets, s'appelle **mirrorlv** et est issu du groupe de volumes **vg0** :

```
lvcreate -L 50G -m1 -n gfslv vg0
```

Un miroir LVM divise le périphérique copié en plusieurs régions qui, par défaut, ont une taille de 512Ko. Vous pouvez utiliser l'argument **-R** pour spécifier la taille des régions en Mo. LVM maintient un 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. Cependant, vous pouvez utiliser l'argument **--corelog** afin que ce fichier soit stocké en mémoire ; vous n'aurez pas besoin de 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 --corelog -n ondiskmirvol bigvg
  Logical volume "ondiskmirvol" created
```

Le journal miroir est créé sur un dispositif séparé des autres dispositifs sur lesquelles toutes les branches du miroir sont créées. Il est possible, cependant, de créer un journal miroir sur le même dispositif, en tant qu'une des branches de miroir en utilisant l'argument **--alloc anywhere** de la commande **vgcreate**. Cela risque de dégrader la performance, mais vous permet de créer un miroir, même si vous n'avez que deux dispositifs sous-jacents.

La commande suivante crée un volume logique en miroir avec un seul miroir dont le journal est sur le même dispositif qu'une des branches du miroir. Dans cet exemple, le groupe de volume consiste en deux dispositifs uniquement. Le volume en miroir créé par cette commande, d'une taille de 500 megaoctets, s'appelle **mirrorlv**, et est issu du groupe de volumes **vg0**.

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

Lorsqu'un miroir est créé, les régions du miroir sont synchronisées. Pour les composants miroir volumineux, le processus de synchronisation peut prendre un moment. Lorsque vous créez 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.

Vous pouvez spécifier les périphériques à utiliser pour les fichiers journaux du miroir ainsi que les extensions des périphériques. Pour forcer un fichier journal sur un disque particulier, spécifiez 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. Le volume a une taille de 500 mega-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 mega-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 de **/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
```

NOTE

Comme dans la version RHEL 5.3, les volumes logiques en miroir sont supportés dans le cluster.

4.4.1.4. Changement de la configuration du volume en miroir

Vous pouvez convertir un volume logique à partir d'un volume en miroir vers un volume linéaire ou à partir d'un volume linéaire vers un volume en miroir avec la commande **lvconvert**. Vous pouvez également utiliser cette commande pour reconfigurer les autres paramètres d'un volume logique existant, tels que **corelog**.

Lorsque vous convertissez un volume logique en un volume en miroir, vous créez des branches de miroir pour un volume existant. Cela signifie que votre groupe de volumes doit contenir les périphériques et l'espace nécessaires pour les branches et le fichier journal du miroir.

If you lose a leg of a mirror, LVM converts the volume to a linear volume so that you still have access to the volume, without the mirror redundancy. After you replace the leg, you can use the **lvconvert** command to restore the mirror. This procedure is provided in Section 6.3, « Récupération suite à un échec miroir LVM ».

La commande suivante convertit le volume logique linéaire **vg00/lvol1** en un volume logique en miroir.

```
lvconvert -m1 vg00/lvol1
```

La commande suivante convertit le volume logique en miroir **vg00/lvol1** en un volume logique linéaire, en supprimant la branche du miroir.

```
lvconvert -m0 vg00/lvol1
```

4.4.2. Numéros de périphérique persistants

Les numéros de périphérique majeur et mineur sont alloués dynamiquement au chargement du module. Certaines applications fonctionnent mieux lorsque le périphérique bloc est toujours activé avec le même numéro de périphérique (majeur ou mineur). Vous pouvez spécifier ces numéros avec les commandes **lvcreate** et **lvchange** en utilisant les arguments suivants :

```
--persistent y --major major --minor minor
```

Use a large minor number to be sure that it hasn't already been allocated to another device dynamically.

Si vous exportez un système de fichiers en utilisant NFS, la spécification du paramètre **fsid** dans le fichier d'exports pourrait vous empêcher de définir un numéro de périphérique persistant dans LMV.

4.4.3. Redimensionnement des volumes logiques

Pour modifier la taille d'un volume logique, utilisez la commande **lvreduce**. Si le volume logique contient un système de fichiers, commencez par réduire la taille du système de fichiers (ou utilisez l'interface utilisateur graphique LVM) afin que le volume logique ait un taille identique ou supérieur à ce dernier.

La commande suivante réduit la taille du volume logique **lvol1** de 3 extensions logiques dans le groupe de volumes **vg00**.

```
lvreduce -l -3 vg00/lvol1
```

4.4.4. Changement des paramètres d'un groupe de volumes logiques

Pour changer les paramètres d'un volume logique, utilisez la commande **lvchange**. Pour obtenir une liste des paramètres que vous pouvez changer, reportez-vous à la page de manuel **lvchange**(8).

You can use the **lvchange** command to activate and deactivate logical volumes. To activate and deactivate all the logical volumes in a volume group at the same time, use the **vgchange** command, as described in Section 4.3.7, « Changement des paramètres du groupe de volumes » .

La commande suivante change les permissions du volume **lvol1** du groupe de volumes **vg00** en lecture seulement.

```
lvchange -pr vg00/lvol1
```

4.4.5. Renommer les volumes logiques

Pour renommer un volume logique existant, utilisez la commande **lvrename**.

Les deux commandes suivantes renomment le volume logique **lvold** du groupe de volumes de **vg02** à **lvnew**.

```
lvrename /dev/vg02/lvold /dev/vg02/lvnew
```

```
lvrename vg02 lvold lvnew
```

For more information on activating logical volumes on individual nodes in a cluster, see Section 4.8, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

4.4.6. Suppression de volumes logiques

Pour supprimer un volume logique inactif, utilisez la commande **lvremove**. Vous devez fermer un volume logique avec la commande **umount** avant qu'il puisse être supprimé. De plus, dans un environnement en clusters, vous devez désactiver un volume logique avant qu'il puisse être supprimé.

Si le volume logique est monté, démontez-le avant de le supprimer.

La commande suivante supprime le volume logique **/dev/testvg/testlv** du groupe de volumes **testvg**. Notez que, dans ce cas, le volume logique n'a pas été désactivé.

```
[root@tng3-1 lvm]# lvremove /dev/testvg/testlv
Do you really want to remove active logical volume "testlv"? [y/n]: y
  Logical volume "testlv" successfully removed
```

Vous pourriez explicitement désactiver le volume logique avant de le supprimer avec la commande **lvchange -an**, dans quel cas vous ne verriez pas l'invite vérifiant si vous voulez supprimer un volume logique actif.

4.4.7. Affichage de volumes logiques

Il y a trois commandes que vous pouvez utiliser pour afficher les propriétés des volumes logiques LVM : **lvs**, **lvdisplay** et **lvscan**.

The **lvs** command provides logical volume information in a configurable form, displaying one line per

logical volume. The **lvs** command provides a great deal of format control, and is useful for scripting. For information on using the **lvs** command to customize your output, see Section 4.9, « Rapport personnalisé pour LVM ».

La commande **lvdisplay** affiche les propriétés des volumes logiques (la taille, la structure, le mappage) dans un format fixe.

La commande suivante illustre les attributs de **lvol2** dans **vg00**. Si des instantanés de volumes logiques ont été créés pour ce volume logique d'origine, la commande affiche une liste de tous les instantanés de volumes logiques et de leur statut (actif ou inactif).

```
lvdisplay -v /dev/vg00/lvol2
```

La commande **lvscan** recherche tous les volumes logiques du système et les affiche, comme dans l'exemple suivant.

```
# lvscan
 ACTIVE                   '/dev/vg0/gfslv' [1.46 GB] inherit
```

4.4.8. Augmentez la taille des volumes logiques

Pour augmenter la taille d'un volume logique, utilisez la commande **lvextend**.

Après avoir étendu le volume logique, vous devrez augmenter la taille du système de fichiers associé en conséquence.

Lorsque vous étendez le volume logique, vous pouvez indiquer de quelle quantité vous voulez l'étendre ou la taille qu'il devrait avoir après que vous l'ayez étendu.

La commande suivante étend le volume logique **/dev/myvg/homevol** de 12 gigaoctets.

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

La commande suivante ajoute un autre giga-octet au volume logique **/dev/myvg/homevol**.

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

Comme avec la commande **lvcreate**, vous pouvez utiliser l'argument **-l** de la commande **lvextend** afin de spécifier le nombre d'extensions pour l'augmentation de la taille du volume logique. Vous pouvez également utiliser cet argument pour spécifier un pourcentage du groupe de volumes ou un pourcentage de l'espace libre restant dans le groupe de volumes. La commande suivante étend le volume logique appelé **testlv** afin de remplir tout l'espace non alloué du groupe de volumes **myvg**.

```
[root@tng3-1 ~]# lvextend -l +100%FREE /dev/myvg/testlv
  Extending logical volume testlv to 68.59 GB
  Logical volume testlv successfully resized
```

Après avoir étendu le volume logique, il est nécessaire d'augmenter la taille du système de fichiers en conséquence.

Par défaut, la plupart des outils de redimensionnent des systèmes de fichiers augmentent la taille d'un système de fichiers afin qu'elle corresponde à celle de son volume logique sous-jacent. Vous n'avez pas besoin de spécifier une taille identique pour chacune de ces commandes.

4.4.9. Augmenter la taille d'un volume en mode stripe

Afin d'augmenter la taille d'un volume logique en mode stripe, il doit y avoir suffisamment d'espace libre sur les volumes physiques sous-jacents qui composent le groupe de volumes afin de supporter le mode stripe. Si vous avez par exemple un stripe à deux sens qui utilise un groupe de volumes entier, l'ajout d'un seul volume physique au groupe de volumes ne vous permettra pas d'étendre le stripe. À la place, vous devez ajouter au moins deux volumes physiques au groupe de volumes.

Considérons par exemple un groupe de volumes **vg** qui se compose deux volumes physiques sous-jacents, comme l'illustre la commande **vgs** suivante.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   0   0 wz--n- 271.31G 271.31G
```

Vous pouvez créer un stripe utilisant tout l'espace du groupe de volumes.

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
  Using default stripesize 64.00 KB
  Rounding up size to full physical extent 271.31 GB
  Logical volume "stripe1" created
# lvs -a -o +devices
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe1 vg   -wi-a- 271.31G
/dev/sda1(0),/dev/sdb1(0)
```

Notez que le groupe de volumes n'a maintenant plus d'espace libre.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   1   0 wz--n- 271.31G    0
```

La commande suivante ajoute un autre volume physique au groupe de volumes, qui aura donc 135Go d'espace supplémentaire.

```
# vgextend vg /dev/sdc1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     3   1   0 wz--n- 406.97G 135.66G
```

À ce stade, vous ne pouvez pas étendre le volume logique en mode stripe à la taille totale du groupe de volumes car deux périphériques sous-jacents sont requis afin de "striper" les données.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
```

```
  Insufficient suitable allocatable extents for logical volume stripe1:
34480
more required
```

Pour étendre le volume logique en mode stripe, ajoutez d'abord un autre volume physique. Dans cet exemple, étant donné que nous avons ajouté de deux volumes physiques dans le groupe de volumes, nous pouvons étendre le volume logique 5A à la taille entière du groupe de volumes.

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 542.00 GB
  Logical volume stripe1 successfully resized
```

Même si vous n'avez pas suffisamment de périphériques physiques sous-jacents afin d'étendre le volume logique en mode stripe, vous pouvez tout de même l'étendre mais l'extension ne sera pas "stripée", ce qui peut résulter en des performances inégales. Lors de l'ajout d'espace au volume logique, l'opération par défaut consiste à utiliser les mêmes paramètres "striping" que ceux utilisés dans le dernier segment du volume logique existant, mais vous pouvez surcharger ces paramètres. L'exemple suivant étend le volume logique en mode stripe existant afin d'utiliser l'espace libre restant après que la commande initiale **lvextend** ait échoué.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1:
34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

4.4.10. Réduire la taille des volumes logiques

Pour réduire la taille d'un volume logique, vous devez d'abord le démonter. Vous pouvez utiliser la commande **lvreduce** pour réduire sa taille. Après avoir réduit la taille du volume, remontez le système de fichiers.

AVERTISSEMENT

Il est important de réduire la taille du système de fichiers ou de tout autre élément résidant au sein du volume avant de réduire le volume lui-même, autrement vous pourriez perdre vos données.

La réduction de la taille d'un volume logique permet de libérer de l'espace au sein du groupe de volumes afin qu'il soit réalloué à d'autres volumes logiques.

L'exemple suivant réduit la taille du volume logique **lvol1** de 3 extensions logiques dans le groupe de volumes **vg00**.

```
lvreduce -l -3 vg00/lvol1
```

4.5. CRÉATION D'INSTANTANÉS DE VOLUMES

Utilisez l'argument **-s** de la commande **lvcreate** pour créer un instantané de volume. Un instantané du volume est en mode écriture.

NOTE

Les instantanés LVM ne sont pas supportées entre les noeuds d'un même groupement.

Since LVM snapshots are not cluster-aware, they require exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see Section 4.8, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

La commande suivante crée un instantané de volume logique d'une taille de 100 méga-octets appelé **/dev/vg00/snap**. Cela crée un instantané du volume logique d'origine appelé **/dev/vg00/lvol1**. Si le volume logique d'origine contient un système de fichiers, vous pouvez monter l'instantané du volume logique sur un répertoire arbitraire afin d'accéder au contenu du système de fichiers pour démarrer une sauvegarde pendant que le système de fichiers d'origine continue à être mis à jour.

```
lvcreate --size 100M --snapshot --name snap /dev/vg00/lvol1
```

Après avoir créé un instantané de volume logique, vous pouvez spécifier un volume d'origine avec la commande **lvdisplay** afin de générer une sortie qui inclue une liste de tous les instantanés de volumes logiques et leur statut (actif ou inactif).

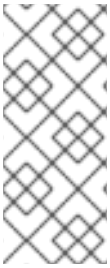
L'exemple suivant affiche le statut du volume logique **/dev/new_vg/lvol0**, pour lequel un instantané de volume **/dev/new_vg/newvgsnap** a été créé.

```
# lvdisplay /dev/new_vg/lvol0
  --- Logical volume ---
  LV Name                /dev/new_vg/lvol0
  VG Name                new_vg
  LV UUID                LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  LV Write Access        read/write
  LV snapshot status     source of
                         /dev/new_vg/newvgsnap1 [active]
  LV Status              available
  # open                 0
  LV Size                52.00 MB
  Current LE             13
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:2
```

Par défaut, la commande **lvs** affiche le volume d'origine et le pourcentage actuel utilisé par chaque instantané de volume. L'exemple suivant illustre la sortie par défaut de la commande **lvs** pour un système qui comprend le volume logique **/dev/new_vg/lvol0**, pour lequel un instantané de volume

/dev/new_vg/newvgsnap a été créé.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```



NOTE

Étant donné que la taille de l'instantané augmente lorsque le volume d'origine change, il est important de régulièrement contrôler le pourcentage de l'instantané de volume avec la commande **lvs** afin de s'assurer qu'il ne soit pas plein. Un instantané qui est rempli à 100% est complètement perdu, car l'écriture sur les parties inchangées du volume d'origine ne pourra pas réussir sans que l'instantané soit corrompu.

4.6. CONTRÔLER L'ANALYSE DES PÉRIPHÉRIQUES LVM AVEC LES FILTRES

Au démarrage, la commande **vgscan** est exécutée afin d'analyser les étiquettes LVM des périphériques blocs du système, pour déterminer les volumes physiques, lire les métadonnées et construire une liste de groupes de volumes. Les noms des volumes physiques sont stockés dans le fichier de cache de chaque noeud du système, **/etc/lvm/.cache**. Les commandes subséquentes peuvent lire ce fichier afin de ne pas les analyser à nouveau.

Vous pouvez contrôler les périphériques analysés par LVM en définissant des filtres dans le fichier de configuration **lvm.conf**. Les filtres se composent d'une série d'expressions régulières simples, appliquée sur les noms de périphériques au sein du répertoire **/dev** afin de déterminer s'il faut accepter ou refuser le périphérique bloc trouvé.

Les exemples suivants illustrent l'utilisation de filtres afin de contrôler les périphériques recherchés par LVM. Notez que certains de ces exemples ne représentent pas forcément une bonne utilisation, étant donné que les expressions régulières peuvent correspondre au nom complet d'un chemin d'accès. Par exemple, **a/loop/** est identique à **a/.*loop.*/** et correspondrait à **/dev/solooperation/lvol1**.

Le filtre suivant ajoute tous les périphériques découverts, ce qui correspond au comportement par défaut étant donné qu'il n'y a pas de filtre configuré dans le fichier de configuration :

```
filter = [ "a/.*/" ]
```

Le filtre suivant supprime le périphérique CD-ROM afin d'éviter un délais d'attente lorsque le lecteur ne contient pas de média :

```
filter = [ "r|/dev/cdrom|" ]
```

Le filtre suivant ajoute tous les "loop" et supprime tous les autres périphériques blocs :

```
filter = [ "a/loop.*/", "r/.*/" ]
```

Le filtre suivant ajoute tous les "loop" et "IDE" et supprime tous les autres périphériques blocs :

```
filter =[ "a|loop.*|", "a|/dev/hd.*|", "r|.*|" ]
```

Le filtre suivant ajoute juste la partition 8 du premier lecteur IDE et supprime tous les autres périphériques blocs :

```
filter = [ "a|^/dev/hda8$|", "r/.*/" ]
```

For more information on the **lvm.conf** file, see Annexe B, *Les fichiers de configuration LVM*and the **lvm.conf**(5) man page.

4.7. DÉPLACEMENT DES DONNÉES EN LIGNE

Vous pouvez déplacer les données pendant que le système est en cours d'utilisation avec la commande **pvmove**.

La commande **pvmove** divise les données devant être déplacées dans des sections et crée un miroir temporaire afin de déplacer chaque section. Pour davantage d'informations à propos de la commande **pvmove**, reportez-vous à la page de manuel **pvmove**(8).

Because the **pvmove** command uses mirroring, it is not cluster-aware and needs exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see Section 4.8, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

La commande suivante déplace tout l'espace alloué, du volume physique **/dev/sdc1** vers les autres volumes physiques libres du groupe de volumes :

```
pvmove /dev/sdc1
```

La commande suivante déplace juste les extensions du volume logique **MyLV**.

```
pvmove -n MyLV /dev/sdc1
```

Étant donné que la commande **pvmove** peut prendre du temps, vous pourriez vouloir l'exécuter en arrière plan afin de ne pas afficher sa progression au premier plan. La commande suivante déplace, en arrière plan, toutes les extensions allouées au volume physique **/dev/sdc1** vers **/dev/sdf1**.

```
pvmove -b /dev/sdc1 /dev/sdf1
```

La commande suivante affiche la progression du déplacement sous forme de pourcentage, à cinq secondes d'intervalle.

```
pvmove -i5 /dev/sdd1
```

4.8. ACTIVATION DES VOLUMES LOGIQUES SUR LES NOEUDS INDIVIDUELS D'UN CLUSTER

Si vous avez LVM d'installé dans un environnement en cluster, vous pourriez parfois avoir besoin d'activer les volumes logiques exclusivement sur un noeud. Par exemple, la commande **pvmove** n'est pas dédiée à un cluster et requiert un accès exclusif au volume. Les instantanés LVM requiert également un accès exclusif au volume.

Pour activer des volumes logiques exclusivement sur un noeud, utilisez la commande **lvchange -aey**. Alternativement, vous pouvez utiliser la commande **lvchange -aly** pour activer des volumes logiques seulement sur le noeud local mais pas de manière exclusive. Vous pouvez par la suite les

activer sur des noeuds supplémentaires de manière concurrente.

You can also activate logical volumes on individual nodes by using LVM tags, which are described in Annexe C, *Les balises des objets LVM* You can also specify activation of nodes in the configuration file, which is described in Annexe B, *Les fichiers de configuration LVM*.

4.9. RAPPORT PERSONNALISÉ POUR LVM

Vous pouvez produire des rapports concis et personnalisés d'objets LVM avec les commandes **pvs**, **lvs** et **vgs**. Les rapports que ces commandes génèrent inclus une ligne de sortie pour chaque objet. Chaque ligne contient une liste ordonnée de champs de propriétés associés à chaque objet. Il existe cinq manières permettant de sélectionner les objets devant être rapportés : par volume physique, par groupe de volumes, par volume logique, par segment de volume physique et par segment de volume logique.

Les sections suivantes fournissent :

- Un résumé des arguments des commandes que vous pouvez utiliser pour contrôler le format du rapport généré.
- Une liste des champs que vous pouvez sélectionner pour chaque objet LVM.
- Un résumé des arguments des commandes que vous pouvez utiliser pour trier le rapport généré.
- Des instructions pour la spécification des unités de la sortie du rapport.

4.9.1. Contrôle du format

Que vous utilisez **pvs**, **lvs** ou **vgs**, ces commandes déterminent l'ensemble de champs à afficher par défaut et l'ordre de tri. Vous pouvez contrôler la sortie de ces commandes avec les arguments suivants :

- Vous pouvez modifier les champs à afficher en utilisant l'argument **-o**. La sortie suivante illustre par exemple l'affichage par défaut de la commande **pvs** (qui affiche des informations à propos des volumes physiques).

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

  La commande suivante affiche uniquement le nom et la taille du volume physique.

```
# pvs -o pv_name,pv_size
  PV         PSize
  /dev/sdb1  17.14G
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
```

- Vous pouvez ajouter un champ à la sortie avec le signe plus (**+**), qui est utilisé en association avec l'argument **-o**.

  L'exemple suivant affiche l'UUID du volume physique en plus des champs par défaut.

```
# pvs -o +pv_uuid
  PV         VG     Fmt  Attr PSize  PFree  PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-
M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-
01S9-X08M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-UqkCS
```

- L'ajout de l'argument **-v** à une commande permet d'inclure des champs supplémentaires. La commande **pvs -v** affichera par exemple les champs **DevSize** et **PV UUID** en plus des champs par défaut.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-
ughJ-D9eB-M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-
kuEn-IdwM-01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G  17.14G yvfvZK-Cf31-
j75k-dECm-0RZ3-0dGW-tUqkCS
```

- L'argument **--noheadings** supprime la ligne d'en-têtes. Ceci peut être utile pour écrire des scripts.

  L'exemple suivant utilise l'argument **--noheadings** en association avec l'argument **pv_name** afin de générer une liste de tous les volumes physiques.

```
# pvs --noheadings -o pv_name
  /dev/sdb1
  /dev/sdc1
  /dev/sdd1
```

- L'argument **--separator *separator*** utilise l'élément *separator* afin séparer chaque champ. Ceci peut être utile dans un script si vous exécutez une commande **grep** sur la sortie.

  L'exemple suivant sépare les champs par défaut de la sortie générée par la commande **pvs** par un signe égal (=).

```
# pvs --separator =
  PV=VG=Fmt=Attr=PSize=PFree
  /dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
  /dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
  /dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

  Pour maintenir les champs alignés lors de l'utilisation de l'argument **separator**, utilisez-le en association avec l'argument **--aligned**.

```
# pvs --separator = --aligned
  PV         =VG    =Fmt =Attr=PSize =PFree
  /dev/sdb1 =new_vg=lvm2=a-  =17.14G=17.14G
```

```
  /dev/sdc1 =new_vg=lvm2=a-   =17.14G=17.09G
  /dev/sdd1 =new_vg=lvm2=a-   =17.14G=17.14G
```

You can use the **-P** argument of the **lvs** or **vgs** command to display information about a failed volume that would otherwise not appear in the output. For information on the output this argument yields, see Section 6.2, « Affichage d'informations à propos des périphériques ayant échoué. » .

Pour obtenir une liste complète des arguments d'affichage, reportez-vous aux pages de manuel **pvs**(8), **vgs**(8) et **lvs**(8).

Les champs des groupes de volumes peuvent être une combinaison avec des champs de volumes physiques (et de segment de volumes physiques) ou des champs de volumes logiques (et de segment de volumes logiques) mais les champs de volumes physiques et logiques ne peuvent pas être mélangés. Par exemple, la commande suivante affiche une ligne de sortie pour chaque volume physhique.

```
# vgs -o +pv_name
  VG     #PV #LV #SN Attr   VSize  VFree  PV
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdc1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdd1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdb1
```

4.9.2. Sélection d'objets

Cette section fournit une série de tableaux qui listent les informations que vous pouvez afficher à propos des objets LVM avec les commandes **pvs**, **vgs** et **lvs**.

Pour des raisons pratiques, un préfixe de nom de champ peut être supprimé s'il correspond à la valeur par défaut de la commande. Par exemple, avec la commande **pvs**, **name** signifie **pv_name** alors qu'avec la commande **vgs**, **name** signifie **vg_name**.

L'exécution de la commande suivante est l'équivalent de l'exécution de **pvs -o pv_free**.

```
# pvs -o free
  PFree
  17.14G
  17.09G
  17.14G
```

La commande pvs

Tableau 4.1, « Champs pour l'affichage de la commnade pvs » lists the display arguments of the **pvs** command, along with the field name as it appears in the header display and a description of the field.

Tableau 4.1. Champs pour l'affichage de la commnade pvs

Argument	En-tête	Description
dev_size	DevSize	La taille du périphérique sous-jacent sur lequel le volume physique a été créé

Argument	En-tête	Description
pe_start	1st PE	L'offset au début de la première extension physique dans le périphérique sous-jacent
pv_attr	Attr	Le statut du volume physique : (a)llocatable ou e(x)ported
pv_fmt	Fmt	Le format des métadonnées du volume physique (**lvm2** ou **lvm1**)
pv_free	PFree	L'espace libre restant sur le volume physique
pv_name	PV	Le nom du volume physique
pv_pe_alloc_count	Alloc	Le nombre d'extensions physiques utilisées
pv_pe_count	PE	Le nombre d'extensions physiques
pvseg_size	SSize	La taille du segment du volume physique
pvseg_start	Start	L'extension physique de départ du segment de volume physique
pv_size	PSize	La taille du volume physique
pv_tags	PV Tags	Les balises LVM associées au volume physique
pv_used	Used	La quantité d'espace utilisée sur le volume physique
pv_uuid	PV UUID	L'UUID du volume physique

La commande **pvs** affiche par défaut les champs suivants : **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**. L'affichage est trié par **pv_name**.

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G
```

Utilisez l'argument **-v** avec la commande **pvs** pour ajouter les champs suivants à l'affichage par défaut : **dev_size**, **pv_uuid**.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-
M7iv-6XqA-dqGeXY
```

```
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-
01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-tUqkCS
```

Vous pouvez utiliser l'argument **--segments** de la commande **pvs** pour afficher des informations à propos de chaque segment de volume physique. Un segment est un groupe d'extensions. Cet affichage peut être utile afin de voir si votre volume logique est fragmenté.

La commande **pvs --segments** affiche par défaut les champs suivants : **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**, **pvseg_start**, **pvseg_size**. L'affichage est trié par **pv_name** et **pvseg_size** au sein du volume physique.

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  Start SSize
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M     0  1172
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1172    16
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1188     1
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G     0    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    26    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    50    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    76    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   100    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   126    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   150    22
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   172  4217
  /dev/sdb1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdc1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdd1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sde1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdf1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdg1  vg         lvm2 a-   17.14G 17.14G     0  4389
```

Vous pouvez utiliser la commande **pvs -a** pour afficher les périphériques détectés par LVM qui n'ont pas été initialisés comme des volumes physiques LVM.

```
# pvs -a
  PV                            VG     Fmt  Attr PSize  PFree
  /dev/VolGroup00/LogVol01                   --       0      0
  /dev/new_vg/lvol0                          --       0      0
  /dev/ram                                   --       0      0
  /dev/ram0                                  --       0      0
  /dev/ram2                                  --       0      0
  /dev/ram3                                  --       0      0
  /dev/ram4                                  --       0      0
  /dev/ram5                                  --       0      0
  /dev/ram6                                  --       0      0
  /dev/root                                  --       0      0
  /dev/sda                                   --       0      0
  /dev/sdb                                   --       0      0
  /dev/sdb1                     new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc                                   --       0      0
  /dev/sdc1                     new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd                                   --       0      0
  /dev/sdd1                     new_vg lvm2 a-   17.14G 17.14G
```

La commande vgs

Tableau 4.2, « Champs pour l'affichage de la commnade vgs » lists the display arguments of the **vgs** command, along with the field name as it appears in the header display and a description of the field.

Tableau 4.2. Champs pour l'affichage de la commnade vgs

Argument	En-tête	Description
lv_count	#LV	Le nombre de volumes logiques que le groupe de volumes peut contenir
max_lv	MaxLV	Le nombre maximum de volumes logiques autorisés dans le groupe de volumes (0 si illimité)
max_pv	MaxPV	Le nombre maximum de volumes physiques autorisés dans le groupe de volumes (0 si illimité)
pv_count	#PV	Le nombre de volumes physiques qui définit le groupe de volumes
snap_count	#SN	Le nombre d'instantanés que le groupe de volumes contient
vg_attr	Attr	Le statut du groupe de volumes : (w)riteable, (r)eadonly, resi(z)eable, e(x)ported, (p)artial et (c)lustered.
vg_extent_count	#Ext	Le nombre d'extensions physiques dans le groupe de volumes
vg_extent_size	Ext	La taille des extensions physiques dans le groupe de volumes
vg_fmt	Fmt	Le format des métadonnées du groupe de volumes (**lvm2** or **lvm1**)
vg_free	VFree	La taille de l'espace libre restant dans le groupe de volumes
vg_free_count	Free	Le nombre d'extensions physiques libres dans le groupe de volumes
vg_name	VG	Le nom du groupe de volumes
vg_seqno	Seq	Le nombre représentant la révision du groupe de volumes
vg_size	VSize	La taille du groupe de volumes
vg_sysid	SYS ID	L'ID Système LVM1
vg_tags	VG Tags	Les balises LVM associées au groupe de volumes
vg_uuid	VG UUID	L'UUID du groupe de volumes

La commande **vgs** affiche par défaut les champs suivants : **vg_name**, **pv_count**, **lv_count**, **snap_count**, **vg_attr**, **vg_size**, **vg_free**. L'affichage est trié par **vg_name**.

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  new_vg   3   1   1 wz--n- 51.42G 51.36G
```

Utilisez l'argument **-v** de la commande **vgs** afin d'ajouter les champs suivants à l'affichage par défaut : **vg_extent_size**, **vg_uuid**.

```
# vgs -v
    Finding all volume groups
    Finding volume group "new_vg"
  VG     Attr   Ext   #PV #LV #SN VSize  VFree  VG UUID
  new_vg wz--n- 4.00M   3   1   1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-
nlwO-wwqd-fD5D32
```

La commande lvs

Tableau 4.3, « Champs pour l'affichage de la commnade lvs » lists the display arguments of the **lvs** command, along with the field name as it appears in the header display and a description of the field.

Tableau 4.3. Champs pour l'affichage de la commnade lvs

Argument	En-tête	Description
chunksize **chunk_size**	Chunk	L'unité de taille dans un instantané de volume
copy_percent	Copy%	Le pourcentage de synchronisation d'un volume logique en miroir, également utilisé lorsque les extensions physiques sont déplacées avec la commande **pv_move**
devices	Devices	Les périphériques sous-jacents qui composent le volume logique : les volumes physiques, les volumes logiques et les extensions physiques et logiques de départ

Argument	En-tête	Description
lv_attr	Attr	Le statut du volume logique. Voici les bits des attributs du volume logique : Bit 1 : Type de volume : (m)irrored, (M)irrored without initial sync, (o)rigin, (p)vmove, (s)napshot, invalid (S)napshot, (v)irtual Bit2 : Permissions : (w)riteable, (r)ead-only Bit 3 : Politique d'allocation : (c)ontiguous, (n)ormal, (a)nywhere, (i)nherited. La lettre est en majuscule si le volume est verrouillé contre les modifications d'allocation, par exemple lors de l'exécution de la commande **pvmove**. Bit 4 : fixed (m)inor Bit 5 État : (a)ctive, (s)uspended, (I)nvalid snapshot, invalid (S)uspended snapshot, mapped (d)evice present without tables, mapped device present with (i)nactive table Bit 6 : device (o)pen
lv_kernel_major	KMaj	Le numéro de périphérique majeur du volume logique (-1 si inactif)
lv_kernel_minor	KMIN	Le numéro de périphérique mineur du volume logique (-1 si inactif)
lv_major	Maj	Le numéro de périphérique majeur persistant du volume logique (-1 s'il n'est pas spécifié)
lv_minor	Min	Le numéro de périphérique mineur persistant du volume logique (-1 s'il n'est pas spécifié)
lv_name	LV	Le nom du volume logique
lv_size	LSize	La taille du volume logique
lv_tags	LV Tags	Les balises LVM associées au volume logique
lv_uuid	LV UUID	L'UUID du volume logique
mirror_log	Log	Le périphérique sur lequel réside le fichier journal du miroir

Argument	En-tête	Description
modules	Modules	La cible device-mapper du noyau correspondant nécessaire pour utiliser le volume logique
move_pv	Move	Le volume physique source d'un volume logique temporaire créé avec la commande **pvmove**
origin	Origin	Le périphérique d'origine de l'instantané de volume
regionsize **region_size**	Region	L'unité de taille du volume logique en miroir
seg_count	#Seg	Le nombre de segments dans le volume logique
seg_size	SSize	La taille des segments dans le volume logique
seg_start	Start	L'offset du segment dans le volume logique
seg_tags	Seg Tags	Les balises LVM associées aux segments du volume logique
segtype	Type	Le type de segment d'un volume logique (par exemple : en miroir, en mode stripe, linéaire)
snap_percent	Snap%	Pourcentage d'un instantané de volume en cours d'utilisation
stripes	#Str	Le nombre de stripes ou miroirs dans un volume logique
stripesize **stripe_size**	Stripe	La taille d'un stripe dans volume logique en mode stripe

La commande **lvs** affiche par défaut les champs suivants : **lv_name**, **vg_name**, **lv_attr**, **lv_size**, **origin**, **snap_percent**, **move_pv**, **mirror_log**, **copy_percent**. L'affichage par défaut est trié par **vg_name** et **lv_name** au sein du groupe de volumes.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```

Utilisez l'argument **-v** de la commande **lvs** pour ajouter les champs suivants à l'affichage par défaut : **seg_count**, **lv_major**, **lv_minor**, **lv_kernel_major**, **lv_kernel_minor**, **lv_uuid**.

```
# lvs -v
    Finding all logical volumes
  LV         VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%
Move Copy%  Log LV UUID
  lvol0      new_vg    1 owi-a- 52.00M  -1  -1 253  3
LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20
1ye1OU-1cIu-o79k-20h2-ZGF0-qCJm-CfbsIx
```

Vous pouvez utiliser l'argument **--segments** de la commande **lvs** pour que les colonnes par défaut mettent en évidence les informations à propos des segments. Lorsque vous utilisez l'argument **segments**, le préfixe **seg** est optionnel. La commande **lvs --segments** affiche par défaut les champs suivants : **lv_name**, **vg_name**, **lv_attr**, **stripes**, **segtype**, **seg_size**. L'affichage par défaut est trié par **vg_name**, **lv_name** au sein du groupe de volumes et par **seg_start** au sein du volume logique. Si les volumes logiques étaient fragmentés, la sortie de cette commande afficherait ce qui suit.

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    1 linear 512.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear  88.00M
```

Utilisez l'argument **-v** de la commande **lvs --segments** pour ajouter les champs suivants à l'affichage par défaut : **seg_start**, **stripesize**, **chunksize**.

```
# lvs -v --segments
    Finding all logical volumes
  LV         VG     Attr   Start SSize  #Str Type   Stripe Chunk
  lvol0      new_vg owi-a-    0  52.00M    1 linear     0     0
  newvgsnap1 new_vg swi-a-    0   8.00M    1 linear     0  8.00K
```

L'exemple suivant illustre la sortie par défaut de la commande **lvs** sur un système avec un volume logique configuré, suivi par la sortie par défaut de la commande **lvs** avec l'argument **segments** spécifié.

```
# lvs
  LV    VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 new_vg -wi-a- 52.00M
# lvs --segments
  LV    VG     Attr   #Str Type   SSize
  lvol0 new_vg -wi-a-    1 linear 52.00M
```

4.9.3. Trier des rapports LVM

En règle générale, la sortie des commandes **lvs**, **vgs** et **pvs** doit être générée et stockée en interne afin que les colonnes puissent être triées et alignées correctement. Vous pouvez spécifier l'argument **--unbuffered** pour afficher des sorties non triées aussitôt qu'elles sont générées.

Pour spécifier une liste de colonnes ordonnées alternative à trier, utilisez l'argument **-O** de l'une de ces commandes. Il n'est pas nécessaire d'inclure ces champs au sein de la sortie elle-même.

L'exemple suivant illustre la sortie de la commande **pvs** qui affiche le nom, la taille et l'espace libre du volume physique.

```
# pvs -o pv_name,pv_size,pv_free
  PV         PSize  PFree
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
```

L'exemple suivant illustre la même sortie, triée avec le champ pour l'espace libre.

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
  PV         PSize  PFree
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
```

Comme illustré dans l'exemple suivant, vous n'avez pas besoin d'afficher le champ utilisé pour le tri.

```
# pvs -o pv_name,pv_size -O pv_free
  PV         PSize
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
  /dev/sdb1  17.14G
```

Pour afficher un tri inversé, précédez un champ que vous avez spécifié après l'argument **-O** par le caractère **-**.

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
  PV         PSize  PFree
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
```

4.9.4. Spécification des unités

Afin de spécifier une unité pour l'affichage des rapports LVM, utilisez l'argument **--units** de la commande de rapport. Vous pouvez spécifier : (b)ytes, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (e)xabytes, (p)etabytes et (h)uman-readable. L'affichage par défaut est (h)uman-readable. Vous pouvez surcharger cette valeur par défaut en définissant le paramètre **units** dans la section **global** du fichier de configuration **lvm.conf**

L'exemple suivant affiche la sortie de la commande **pvs** en méga-octets plutôt qu'en giga-octets.

```
# pvs --units m
  PV         VG     Fmt  Attr PSize     PFree
```

```
  /dev/sda1         lvm2 --   17555.40M 17555.40M
  /dev/sdb1  new_vg lvm2 a-   17552.00M 17552.00M
  /dev/sdc1  new_vg lvm2 a-   17552.00M 17500.00M
  /dev/sdd1  new_vg lvm2 a-   17552.00M 17552.00M
```

Par défaut, les unités sont affichées en puissances de 2 (multiples de 1024). Vous pouvez afficher des unités en multiples de 1000 si vous les spécifiez en lettres majuscules (B, K, M, G, T, H).

La commande suivante affiche la sortie comme un multiple de 1024, le comportement par défaut.

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

La commande suivante affiche la sortie comme un multiple de 1000.

```
#  pvs --units G
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   18.40G 18.40G
  /dev/sdc1  new_vg lvm2 a-   18.40G 18.35G
  /dev/sdd1  new_vg lvm2 a-   18.40G 18.40G
```

Vous pouvez également spécifier des (s)ectors (définis comme 512 octets) ou des unités personnalisées.

L'exemple suivant affiche la sortie de la commande **pvs** comme un nombre de secteurs ("sectors").

```
# pvs --units s
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sdb1  new_vg lvm2 a-   35946496S 35946496S
  /dev/sdc1  new_vg lvm2 a-   35946496S 35840000S
  /dev/sdd1  new_vg lvm2 a-   35946496S 35946496S
```

L'exemple suivant affiche la sortie de la commande **pvs** avec des unités de 4 méga-octets.

```
# pvs --units 4m
  PV         VG     Fmt  Attr PSize    PFree
  /dev/sdb1  new_vg lvm2 a-   4388.00U 4388.00U
  /dev/sdc1  new_vg lvm2 a-   4388.00U 4375.00U
  /dev/sdd1  new_vg lvm2 a-   4388.00U 4388.00U
```

CHAPITRE 5. EXEMPLES DE CONFIGURATION LVM

Ce chapitre fournit des exemples de configuration LVM de base.

5.1. CRÉATION D'UN VOLUME LOGIQUE LVM SUR TROIS DISQUES

Cet exemple crée un volume logique LVM appelé **new_logical_volume** qui se compose des disques **/dev/sda1**, **/dev/sdb1** et **/dev/sdc1**.

5.1.1. Création de volumes physiques

Pour utiliser des disques dans un groupe de volumes, vous devez les étiqueter comme des volumes physiques.

AVERTISSEMENT

Cette commande supprime toutes les données sur **/dev/sda1**, **/dev/sdb1** et **/dev/sdc1**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

5.1.2. Création d'un groupe de volumes

La commande suivante permet de créer le groupe de volumes **new_vol_group**.

```
[root@tng3-1 ~]# vgcreate new_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "new_vol_group" successfully created
```

Vous pouvez utiliser la commande **vgs** pour afficher les attributs du nouveau groupe de volumes.

```
[root@tng3-1 ~]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  new_vol_group   3   0   0 wz--n- 51.45G 51.45G
```

5.1.3. Création du volume logique

La commande suivante permet de créer le volume logique **new_logical_volume** à partir du groupe de volumes **new_vol_group**. Un volume logique qui utilise 2Go du groupe de volumes sera créé.

```
[root@tng3-1 ~]# lvcreate -L2G -n new_logical_volume new_vol_group
  Logical volume "new_logical_volume" created
```

5.1.4. Création du système de fichiers

La commande suivante permet de créer un système de fichiers GFS sur le volume logique.

```
[root@tng3-1 ~]# gfs_mkfs -plock_nolock -j 1
/dev/new_vol_group/new_logical_volume
This will destroy any data on /dev/new_vol_group/new_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/new_vol_group/new_logical_volume
Blocksize:                 4096
Filesystem Size:           491460
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Les commandes suivantes montent le volume logique et affichent l'utilisation d'espace disque du système de fichiers.

```
[root@tng3-1 ~]# mount /dev/new_vol_group/new_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/new_vol_group/new_logical_volume
                       1965840        20   1965820   1% /mnt
```

5.2. CRÉATION D'UN VOLUME LOGIQUE EN MODE STRIPE

Cet exemple permet de créer un volume logique LVM en mode stripe appelé **striped_logical_volume** qui distribue les données à travers les disques **/dev/sda1**, **/dev/sdb1** et **/dev/sdc1**.

5.2.1. Création de volumes physiques

Étiquetez le disque que vous utiliserez dans les groupes de volumes en tant que volumes physiques LVM.

AVERTISSEMENT

Cette commande supprime toutes les données sur **/dev/sda1**, **/dev/sdb1** et **/dev/sdc1**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
```

```
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

5.2.2. Création d'un groupe de volumes

La commande suivante crée un groupe de volumes **striped_vol_group**.

```
[root@tng3-1 ~]# vgcreate striped_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "striped_vol_group" successfully created
```

Vous pouvez utiliser la commande **vgs** pour afficher les attributs du nouveau groupe de volumes.

```
[root@tng3-1 ~]# vgs
  VG                #PV #LV #SN Attr   VSize  VFree
  striped_vol_group   3   0   0 wz--n- 51.45G 51.45G
```

5.2.3. Création du volume logique

La commande suivante permet de créer le volume logique en mode stripe **striped_logical_volume** à partir du groupe de volumes **striped_vol_group**. Cet exemple crée un volume logique de 2 giga-octets, avec trois bandes de 4 kilo-octets chacune.

```
[root@tng3-1 ~]# lvcreate -i3 -I4 -L2G -nstriped_logical_volume
striped_vol_group
  Rounding size (512 extents) up to stripe boundary size (513 extents)
  Logical volume "striped_logical_volume" created
```

5.2.4. Création du système de fichiers

La commande suivante permet de créer un système de fichiers GFS sur le volume logique.

```
[root@tng3-1 ~]#  gfs_mkfs -plock_nolock -j 1
/dev/striped_vol_group/striped_logical_volume
This will destroy any data on
/dev/striped_vol_group/striped_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/striped_vol_group/striped_logical_volume
Blocksize:                 4096
Filesystem Size:           492484
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Les commandes suivantes montent le volume logique et affichent l'utilisation d'espace disque du système de fichiers.

```
[root@tng3-1 ~]# mount /dev/striped_vol_group/striped_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      13902624   1656776  11528232  13% /
/dev/hda1               101086     10787     85080  12% /boot
tmpfs                   127880         0    127880   0% /dev/shm
/dev/striped_vol_group/striped_logical_volume
                       1969936        20   1969916   1% /mnt
```

5.3. PARTAGER UN GROUPE DE VOLUMES

Dans cet exemple, un groupe de volumes est composé de trois volumes physiques. S'il y a suffisamment d'espace disque non utilisé sur les volumes logiques, un nouveau groupe de volumes peut être créé sans ajouter de nouveaux disques.

Dans la configuration initiale, le volume logique **mylv** est créé à partir du groupe de volumes **myvol**, qui se compose de trois volumes physiques, **/dev/sda1**, **/dev/sdb1** et **/dev/sdc1**.

Après avoir complété cette procédure, le groupe de volumes **myvg** se composera de **/dev/sda1** et **/dev/sdb1**. Un deuxième groupe de volumes, **yourvg**, se composera de **/dev/sdc1**.

5.3.1. Déterminer l'espace libre

Vous pouvez utiliser la commande **pvscan** pour déterminer la quantité d'espace libre disponible dans le groupe de volumes.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1  VG myvg   lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

5.3.2. Déplacer les données

Vous pouvez déplacer toutes les extensions physiques utilisées dans **/dev/sdc1** vers **/dev/sdb1** avec la commande **pvmove**. La commande **pvmove** peut prendre du temps à s'exécuter.

```
[root@tng3-1 ~]# pvmove /dev/sdc1 /dev/sdb1
  /dev/sdc1: Moved: 14.7%
  /dev/sdc1: Moved: 30.3%
  /dev/sdc1: Moved: 45.7%
  /dev/sdc1: Moved: 61.0%
  /dev/sdc1: Moved: 76.6%
  /dev/sdc1: Moved: 92.2%
  /dev/sdc1: Moved: 100.0%
```

Après avoir déplacé les données, vous remarquerez que tout l'espace sur **/dev/sdc1** est libre.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 10.80 GB free]
  PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 17.15 GB free]
```

```
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

5.3.3. Diviser le groupe de volumes

Pour créer le nouveau groupe de volumes **yourvg**, utilisez la commande **vgsplit** pour diviser le groupe de volume **myvg**.

Avant de pouvoir diviser le groupes de volumes, le volume logique doit être inactif. Si le système de fichiers est monté, vous devez démonter le système de fichiers avant de désactiver le volume logique.

Vous pouvez désactiver les volumes logiques avec la commande **lvchange** ou la commande **vgchange**. La commande suivante désactive le volume logique **mylv** et divise ensuite le groupe de volumes **yourvg** à partir de **myvg**, en déplaçant le volume physique **/dev/sdc1** dans le nouveau groupe de volumes **yourvg**.

```
[root@tng3-1 ~]# lvchange -a n /dev/myvg/mylv
[root@tng3-1 ~]# vgsplit myvg yourvg /dev/sdc1
  Volume group "yourvg" successfully split from "myvg"
```

Vous pouvez utiliser la commande **vgs** pour afficher les attributs des deux groupes de volumes.

```
[root@tng3-1 ~]# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  myvg     2   1   0 wz--n- 34.30G 10.80G
  yourvg   1   0   0 wz--n- 17.15G 17.15G
```

5.3.4. Création d'un nouveau volume logique

Après avoir créé le nouveau groupe de volumes, vous pouvez créer le nouveau volume logique **yourlv**.

```
[root@tng3-1 ~]# lvcreate -L5G -n yourlv yourvg
  Logical volume "yourlv" created
```

5.3.5. Créer un nouveau système de fichiers et monter le nouveau volume logique

Vous pouvez créer un système de fichiers sur le nouveau volume logique et le monter.

```
[root@tng3-1 ~]#  gfs_mkfs -plock_nolock -j 1 /dev/yourvg/yourlv
This will destroy any data on /dev/yourvg/yourlv.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/yourvg/yourlv
Blocksize:                 4096
Filesystem Size:           1277816
Journals:                  1
Resource Groups:           20
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
```

```
All Done

[root@tng3-1 ~]# mount /dev/yourvg/yourlv /mnt
```

5.3.6. Activation et montage du volume logique d'origine

Étant donné que vous avez désactivé le volume logique **mylv**, vous devez le réactiver avant de pouvoir le monter.

```
root@tng3-1 ~]# lvchange -a y mylv

[root@tng3-1 ~]# mount /dev/myvg/mylv /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/yourvg/yourlv    24507776        32  24507744   1% /mnt
/dev/myvg/mylv        24507776        32  24507744   1% /mnt
```

5.4. SUPPRESSION D'UN DISQUE DU VOLUME LOGIQUE

Cette exemple vous illustre comment supprimer le disque d'un volume logique existant, soit en remplaçant le disque, soit l'utilisant en tant que volume différent. Afin de supprimer un disque, vous devez d'abord déplacer les extensions du volume physique LVM vers un disque ou un groupe de disques différent.

5.4.1. Déplacer les extensions vers des volumes physiques existants

Dans cet exemple, le volume logique est distribué à travers quatre volumes physiques dans le groupe de volumes **myvg**.

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Nous voulons déplacer les extensions de **/dev/sdb1** afin de le supprimer du groupe de volumes.

S'il y a suffisamment d'extensions libres sur les autres volumes physiques du groupe de volumes, vous pouvez exécuter la commande **pvmove** sans option sur le périphérique que vous voulez supprimer afin que les extensions soient distribuées sur les autres périphériques.

```
[root@tng3-1 ~]# pvmove /dev/sdb1
  /dev/sdb1: Moved: 2.0%
 ...
  /dev/sdb1: Moved: 79.2%
 ...
  /dev/sdb1: Moved: 100.0%
```

Après que la commande **pvmove** ait été exécutée, les extensions sont distribuées comme ci-dessous :

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
```

```
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Utilisez la commande **vgreduce** pour supprimer le volume physique **/dev/sdb1** du groupe de volumes.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
[root@tng3-1 ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G
  /dev/sdb1       lvm2 --   17.15G 17.15G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G
```

Le disque peut maintenant être supprimé physiquement ou alloué à d'autres utilisateurs.

5.4.2. Déplacer les extensions vers un nouveau disque

Dans cet exemple, le volume logique est distribué à travers trois volumes physiques dans le groupe de volumes **myvg** comme ci-dessous :

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

Nous voulons déplacer les extensions de **/dev/sdb1>** vers un nouveau périphérique, **/dev/sdd1**.

5.4.2.1. Création du nouveau volume physique

Créez un nouveau volume physique à partir de **/dev/sdd1**.

```
[root@tng3-1 ~]# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

5.4.2.2. Ajout du nouveau volume physique au groupe de volumes

Ajoutez **/dev/sdd1** au groupe de volumes existant **myvg**.

```
[root@tng3-1 ~]# vgextend myvg /dev/sdd1
  Volume group "myvg" successfully extended
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 17.15G     0
```

5.4.2.3. Déplacer les données

Utilisez la commande **pvmove** pour déplacer les données de **/dev/sdb1** vers **/dev/sdd1**.

```
[root@tng3-1 ~]# pvmove /dev/sdb1 /dev/sdd1
  /dev/sdb1: Moved: 10.0%
...
  /dev/sdb1: Moved: 79.7%
...
  /dev/sdb1: Moved: 100.0%

[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

5.4.2.4. Suppression de l'ancien volume physique du groupe de volumes.

Après avoir déplacé les données de **/dev/sdb1**, vous pouvez le supprimer du groupe de volumes.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
```

Vous pouvez maintenant réaffecter le disque à un autre groupe de volumes ou supprimer le disque du système.

CHAPITRE 6. RÉSOLUTION DE PROBLÈMES LVM

Ce chapitre fournit des instructions permettant de résoudre une variété de problèmes LVM.

6.1. DIAGNOSTIQUES DE RÉSOLUTION DE PROBLÈMES

Si une commande ne fonctionne pas comme prévu, vous pouvez établir des diagnostics de l'une des manières suivantes :

- Utilisez l'argument **-v**, **-vv**, **-vvv** ou **-vvvv** avec n'importe quelle commande afin d'augmenter le niveau de verbosité de sa sortie.
- If the problem is related to the logical volume activation, set 'activation = 1' in the 'log' section of the configuration file and run the command with the **-vvvv** argument. After you have finished examining this output be sure to reset this parameter to 0, to avoid possible problems with the machine locking during low memory situations.
- Exécutez la commande **lvmdump** afin d'obtenir des informations à des fins de diagnostic. Pour davantage d'informations, reportez-vous à la page de manuel **lvmdump**(8).
- Exécutez la commande **lvs -v**, **pvs -a** ou **dmsetup info -c** pour davantage d'informations système.
- Examinez la dernière sauvegarde de métadonnées dans **/etc/lvm/backup** et les versions archivées dans **/etc/lvm/archive**.
- Vérifiez les informations de configuration en exécutant la commande **lvm dumpconfig**.
- Vérifiez le fichier **.cache** dans **/etc/lvm** pour savoir quels sont les périphériques qui disposent de volumes physiques.

6.2. AFFICHAGE D'INFORMATIONS À PROPOS DES PÉRIPHÉRIQUES AYANT ÉCHOUÉ.

Vous pouvez utiliser l'argument **-P** des commandes **lvs** et **vgs** afin d'afficher des informations à propos d'un périphérique ayant échoué qui, autrement, n'apparaîtraient pas dans la sortie. Par exemple, si l'un des périphériques qui compose le groupe de volumes **vg** a échoué, la commande **vgs** pourrait afficher la sortie suivante.

```
[root@link-07 tmp]# vgs -o +devices
  Volume group "vg" not found
```

Si vous spécifiez l'argument **-P** avec la commande **vgs**, le groupe de volumes sera toujours inutilisable mais vous pourrez voir davantage d'informations à propos du périphérique ayant échoué.

```
[root@link-07 tmp]# vgs -P -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  VG   #PV #LV #SN Attr   VSize VFree Devices
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(0)
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(5120),/dev/sda1(0)
```

Dans cet exemple, le périphérique ayant échoué a provoqué l'échec d'un volume logique en mode stripe et d'un volume logique linéaire dans le groupe de volumes. La commande **lvs** sans l'argument **-P** affiche la sortie suivante.

```
[root@link-07 tmp]# lvs -a -o +devices
  Volume group "vg" not found
```

En utilisant l'argument **-P** vous affichez les volumes logiques qui ont échoué.

```
[root@link-07 tmp]# lvs -P -a -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  LV     VG   Attr   LSize  Origin Snap%  Move Log Copy%  Devices
  linear vg   -wi-a- 20.00G                                unknown
device(0)
  stripe vg   -wi-a- 20.00G                                unknown
device(5120),/dev/sda1(0)
```

Les exemples suivants affichent la sortie des commandes **pvs** et **lvs** avec l'argument **-P**, suite à l'échec d'une branche d'un volume logique en miroir.

```
root@link-08 ~]#  vgs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  VG    #PV #LV #SN Attr   VSize VFree Devices
  corey   4   4   0 rz-pnc 1.58T 1.34T
my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T unknown device(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdb1(0)
```

```
[root@link-08 ~]# lvs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  LV                VG    Attr   LSize   Origin Snap%  Move Log
Copy%  Devices
  my_mirror         corey mwi-a- 120.00G
my_mirror_mlog   1.95 my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  [my_mirror_mimage_0] corey iwi-ao 120.00G
unknown device(0)
  [my_mirror_mimage_1] corey iwi-ao 120.00G
/dev/sdb1(0)
  [my_mirror_mlog]     corey lwi-ao   4.00M
/dev/sdd1(0)
```

6.3. RÉCUPÉRATION SUITE À UN ÉCHEC MIROIR LVM

Cette section fournit un exemple de récupération dans une situation où la branche d'un volume LVM en miroir échoue car le périphérique sous-jacent d'un volume physique tombe en panne. Lorsque la branche d'un miroir échoue, LVM convertit le volume en un volume linéaire, qui continue à opérer de la même manière mais sans la redondance en miroir. À ce moment, vous pouvez ajouter un nouveau périphérique disque au système afin de l'utiliser en tant que périphérique physique de remplacement et reconstruire le miroir.

La commande suivante crée les volumes physiques qui seront utilisés pour le miroir.

```
[root@link-08 ~]# pvcreate /dev/sd[abcdefgh][12]
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sda2" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdb2" successfully created
  Physical volume "/dev/sdc1" successfully created
  Physical volume "/dev/sdc2" successfully created
  Physical volume "/dev/sdd1" successfully created
  Physical volume "/dev/sdd2" successfully created
  Physical volume "/dev/sde1" successfully created
  Physical volume "/dev/sde2" successfully created
  Physical volume "/dev/sdf1" successfully created
  Physical volume "/dev/sdf2" successfully created
  Physical volume "/dev/sdg1" successfully created
  Physical volume "/dev/sdg2" successfully created
  Physical volume "/dev/sdh1" successfully created
  Physical volume "/dev/sdh2" successfully created
```

Les commandes suivantes créent le groupe de volumes **vg** et le volume en miroir **groupfs**.

```
[root@link-08 ~]# vgcreate vg /dev/sd[abcdefgh][12]
  Volume group "vg" successfully created
[root@link-08 ~]# lvcreate -L 750M -n groupfs -m 1 vg /dev/sda1 /dev/sdb1
/dev/sdc1
  Rounding up size to full physical extent 752.00 MB
  Logical volume "groupfs" created
```

Vous pouvez utiliser la commande **lvs** pour vérifier la disposition du volume en miroir et des périphériques sous-jacents pour la branche du miroir et le fichier journal du miroir. Notez que dans le premier exemple, le miroir n'est pas encore complètement synchronisé ; vous devriez attendre jusqu'à ce que le champ **Copy%** affiche 100.00 avant de continuer.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy% Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
21.28 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)

[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy%  Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
100.00 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M     i
/dev/sdc1(0)
```

–

Dans cet exemple, la branche primaire du miroir **/dev/sda1** échoue. Toute activité d'écriture sur le volume en miroir permet à LVM de détecter le miroir échouant. Lorsque cela se produit, LVM convertit le miroir en un volume linéaire unique. Dans ce cas, pour déclencher la conversion, nous exécutons une commande **dd**.

```
[root@link-08 ~]# dd if=/dev/zero of=/dev/vg/groupfs count=10
10+0 records in
10+0 records out
```

Vous pouvez utiliser la commande **lvs** pour vérifier que le périphérique soit un périphérique linéaire. À cause du disque ayant échoué, des erreurs d'E/S se produisent.

```
[root@link-08 ~]# lvs -a -o +devices
  /dev/sda1: read failed after 0 of 2048 at 0: Input/output error
  /dev/sda2: read failed after 0 of 2048 at 0: Input/output error
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  groupfs vg   -wi-a- 752.00M                             /dev/sdb1(0)
```

À ce moment, vous devriez encore être capable d'utiliser le volume logique, mais il n'y aura pas de redondance en miroir.

To rebuild the mirrored volume, you replace the broken drive and recreate the physical volume. If you use the same disk rather than replacing it with a new one, you will see "inconsistent" warnings when you run the **pvcreate** command.

```
[root@link-08 ~]# pvcreate /dev/sda[12]
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sda2" successfully created

[root@link-08 ~]# pvscan
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sda1           lvm2 [603.94 GB]
  PV /dev/sda2           lvm2 [603.94 GB]
  Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

Ensuite, vous étendez le groupe de volumes d'origine avec le nouveau volume physique.

```
[root@link-08 ~]# vgextend vg /dev/sda[12]
  Volume group "vg" successfully extended

[root@link-08 ~]# pvscan
```

```
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sda1   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  PV /dev/sda2   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  Total: 16 [2.11 TB] / in use: 16 [2.11 TB] / in no VG: 0 [0   ]
```

Reconvertissez le volume linéaire à son état en miroir d'origine.

```
[root@link-08 ~]# lvconvert -m 1 /dev/vg/groupfs /dev/sda1 /dev/sdb1
/dev/sdc1
  Logical volume mirror converted.
```

Vous pouvez utiliser la commande **lvs** pour vérifier si le miroir a été restauré.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy% Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
68.62 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)
```

6.4. RECUPÉRATION DES MÉTADONNÉES DU VOLUME PHYSIQUE

Si la zone de métadonnées du groupe de volumes d'un volume physique est accidentellement surchargée ou détruite, vous obtiendrez un message d'erreur indiquant que la zone de métadonnées est incorrecte ou que le système était incapable de trouver un volume physique avec un UUID particulier. Vous pourriez restaurer les données du volume physique en écrivant une nouvelle zone de métadonnées sur ce volume et en spécifiant le même UUID que pour les métadonnées qui ont été perdues.

AVERTISSEMENT

Vous ne devriez pas essayer cette procédure avec un volume logique LVM fonctionnant. Vous perdrez vos données si vous spécifiez un UUID incorrect.

L'exemple suivant illustre le type de sortie que vous verriez si la zone de métadonnées est manquante ou corrompue.

```
[root@link-07 backup]# lvs -a -o +devices
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  ...
```

Vous pourriez trouver l'UUID du volume physique qui a été surchargé en regardant dans le répertoire **/etc/lvm/archive**. Recherchez dans le fichier ***VolumeGroupName_xxxx*.vg** les dernières métadonnées LVM archivées valides pour le groupe de volumes.

Alternativement, en désactivant le volume et en utilisant l'argument **partial** (**-P**) vous pourrez trouver l'UUID du volume physique corrompu manquant.

```
[root@link-07 backup]# vgchange -an --partial
  Partial mode. Incomplete volume groups will be activated read-only.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  ...
```

Utilisez les arguments **--uuid** et **--restorefile** de la commande **pvcreate** pour restaurer le volume physique. L'exemple suivant étiquette le périphérique **/dev/sdh1** en tant que volume physique avec l'UUID indiqué ci-dessus, **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**. Cette commande restaure l'étiquette du volume physique avec les informations de métadonnées contenues dans **VG_00050.vg**, les métadonnées archivées les plus récentes pour le groupe de volumes . L'argument **restorefile** permet à la commande **pvcreate** de rendre le nouveau volume physique compatible avec l'ancien sur le groupe de volumes, en s'assurant que les nouvelles métadonnées ne soient pas placées là où l'ancien volume physique stockait ses données (cela pourrait par exemple se produire si la commande d'origine **pvcreate** avait utilisé les arguments en ligne de commande qui permettent de contrôler l'emplacement des métadonnées ou si le volume physique avait été créé avec une version logicielle différente qui utilise d'autres paramètres par défaut). La commande **pvcreate** surcharge uniquement les zones de métadonnées LVM et n'affecte pas les zones de données existantes.

```
[root@link-07 backup]# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-
NYM5Sk" --restorefile /etc/lvm/archive/VG_00050.vg /dev/sdh1
  Physical volume "/dev/sdh1" successfully created
```

You can then use the **vgcfgrestore** command to restore the volume group's metadata.

```
[root@link-07 backup]# vgcfgrestore VG
  Restored volume group VG
```

Vous pouvez maintenant afficher les volumes logiques.

```
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi--- 300.00G                                /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi--- 300.00G                                /dev/sdh1
(34728),/dev/sdb1(0)
```

Les commandes suivantes activent les volumes et affichent les volumes actifs.

```
[root@link-07 backup]# lvchange -ay /dev/VG/stripe
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(34728),/dev/sdb1(0)
```

Si les métadonnées LVM sur le disque prennent au moins autant d'espace que celles qui les surchagaient, cette commande peut récupérer le volume physique. Si celles qui surchagaient les métadonnées dépassent la zone de métadonnées, les données sur le volume peuvent être affectées. Vous pourriez utiliser la commande **fsck** pour récupérer ces données.

6.5. REMPLACEMENT D'UN VOLUME PHYSIQUE MANQUANT

If a physical volume fails or otherwise needs to be replaced, you can label a new physical volume to replace the one that has been lost in the existing volume group by following the same procedure as you would for recovering physical volume metadata, described in Section 6.4, « Recupération des métadonnées du volume physique ». You can use the **--partial** and **--verbose** arguments of the **vgdisplay** command to display the UUIDs and sizes of any physical volumes that are no longer present. If you wish to substitute another physical volume of the same size, you can use the **pvcreate** command with the **--restorefile** and **--uuid** arguments to initialize a new device with the same UUID as the missing physical volume. You can then use the **vgcfgrestore** command to restore the volume group's metadata.

6.6. SUPPRIMER LES VOLUMES PHYSIQUES PERDUS D'UN GROUPE DE VOLUMES

Si vous perdez un volume physique, vous pouvez activer les volumes physiques restants dans le groupe de volumes avec l'argument **--partial** de la commande **vgchange**. Vous pouvez supprimer tous les volumes logiques utilisant ce volume physique à partir du groupe de volumes avec l'argument **--removemissing** de la commande **vgreduce**.

Il est recommandé d'exécuter la commande **vgreduce** avec l'argument **--test** pour vérifier ce que vous supprimez.

Comme la plupart des opérations LVM, la commande **vgreduce** est réservible si vous utilisez immédiatement la commande **vgcfgrestore** afin de restaurer les métadonnées du groupe de volumes à leur état précédent. Par exemple, si vous avez utilisé l'argument **--removemissing** de la

commande **vgreduce** sans l'argument **--test** et que vous vous apercevez d'avoir supprimé les volumes logiques que vous vouliez garder, vous pouvez toujours remplacer le volume physique et utiliser une autre commande **vgcfgrestore** pour restaurer le groupe de volumes à son état précédent.

6.7. EXTENSIONS LIBRES INSUFFISANTES POUR UN VOLUME LOGIQUE

You may get the error message "Insufficient free extents" when creating a logical volume when you think you have enough extents based on the output of the **vgdisplay** or **vgs** commands. This is because these commands round figures to 2 decimal places to provide human-readable output. To specify exact size, use free physical extent count instead of some multiple of bytes to determine the size of the logical volume.

La commande **vgdisplay** inclut par défaut cette ligne de sortie qui indique les extensions physiques libres.

```
# vgdisplay
  --- Volume group ---
  ...
  Free  PE / Size       8780 / 34.30 GB
```

Alternativement, vous pouvez utiliser les arguments **vg_free_count** et **vg_extent_count** de la commande **vgs** pour afficher les extensions libres et le nombre total d'extensions.

```
[root@tng3-1 ~]# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree  Free #Ext
  testvg   2   0   0 wz--n- 34.30G 34.30G 8780 8780
```

Avec 8780 extensions physiques libres, vous pouvez exécuter la commande suivante en utilisant l'argument l, en lettre minuscule, pour utiliser les extensions plutôt que les octets :

```
# lvcreate -l8780 -n testlv testvg
```

Cela utilise toutes les extensions libres du groupe de volumes.

```
# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree Free #Ext
  testvg   2   1   0 wz--n- 34.30G    0     0 8780
```

Alternately, you can extend the logical volume to use a percentage of the remaining free space in the volume group by using the **-l** argument of the **lvcreate** command. For information, see Section 4.4.1.1, « Création de volumes linéaires » .

CHAPITRE 7. ADMINISTRATION LVM AVEC L'INTERFACE UTILISATEUR GRAPHIQUE LVM

En plus de l'interface en lignes de commande (CLI de l'anglais Command Line Interface), LVM fournit une interface utilisateur graphique (GUI) que vous pouvez utiliser pour configurer les volumes logiques LVM. Vous pouvez obtenir cet utilitaire en saisissant **`system-config-lvm`**. Le chapitre LVM du *Guide de déploiement de Red Hat Enterprise Linux* fournit des instructions, étape par étape, afin de configurer un volume logique LVM en utilisant cet utilitaire.

De plus, l'interface utilisateur graphique LVM est disponible dans le cadre de l'interface de gestion Conga. Pour davantage d'informations à propos de l'utilisation de l'interface utilisateur graphique LVM avec Conga, reportez-vous à l'aide en ligne de Conga.

ANNEXE A. DEVICE MAPPER (MAPPEUR DE PÉRIPHÉRIQUES)

Le Device Mapper est un pilote noyau qui offre un framework générique pour la gestion des volumes. Il permet de créer génériquement des périphériques mappés, qui peuvent être utilisés en tant que volumes logiques. Il ne connaît pas, de manière spécifique, le format des métadonnées et groupes de volumes.

Le Device Mapper fournit la base pour un nombre de technologies de haut niveau. En plus de LVM, le device-mapper multipath et la commande **dmraid** utilisent le mappeur de périphériques. L'interface d'application du mappeur de périphériques est l'appel système **ioctl**. L'interface d'utilisateur est la commande **dmsetup**.

LVM logical volumes are activated using the Device Mapper. Each logical volume is translated into a mapped device. Each segment translates into a line in the mapping table that describes the device. The Device Mapper supports a variety of mapping targets, including linear mapping, striped mapping, and error mapping. So, for example, two disks may be concatenated into one logical volume with a pair of linear mappings, one for each disk. When LVM2 creates a volume, it creates an underlying device-mapper device that can be queried with the **dmsetup** command. For information about the format of devices in a mapping table, see Section A.1, « Mappages de tables de périphériques » . For information about using the **dmsetup** command to query a device, see Section A.2, « La commande dmsetup » .

A.1. MAPPAGES DE TABLES DE PÉRIPHÉRIQUES

Un périphérique mappé est déterminé par une table qui spécifie comment mapper chaque partie des secteurs logiques, en utilisant un mappage de tables de périphériques pris en charge. La table périphérique mappé, est construite à partir d'une liste de lignes sous la forme suivante :

```
start length mapping [mapping_parameters...]
```

Sur la première ligne d'une table de Device Mapper, le paramètre ***start*** doit être égal à zéro. Les paramètres ***start*** + ***length*** d'une ligne, doivent être égaux au paramètre ***start*** de la lige suivante. Les paramètres de mappage spécifiés sur une ligne de table de mappage dépendent du type de ***mapping*** spécifié sur la ligne.

Les tailles, dans le mappeur de périphérique, sont toujours spécifiées dans les secteurs (512 octets).

Quand un périphérique est spécifié en tant que paramètre dans mappeur de périphérique, il peut être connu sous le nom du périphérique dans le système de fichiers (par exemple, **/dev/hda**) ou par un nombre majeur ou mineur sous les formats suivants ***major:minor***. Le format major:minor est le format que l'on préfère, car il évite les recherches de noms de chemins d'accès.

Ce qui suit montre un échantillon de table de mappage pour un périphérique. Dans cette table, il y a quatre cibles linéaires :

```
0 35258368 linear 8:48 65920
35258368 35258368 linear 8:32 65920
70516736 17694720 linear 8:16 17694976
88211456 17694720 linear 8:16 256
```

Les 2 premiers paramètres de chaque ligne correspondent au bloc de départ et à la longueur du segment. Le prochain mot-clé est la cible de mappage, qui, dans tous les cas de cet exemple, est **linear** (linéaire). Le reste de la ligne consiste en paramètres pour une cible **linéaire**.

Les sous-sections suivantes décrivent le format des mappages suivants :

- linéaire
- par bandes
- miroir
- instantané et origine-instantané
- erreur
- zéro
- multivoies
- cryptage

A.1.1. La cible de mappage linéaire

Une cible de mappage linéaire mappe un ensemble de blocs contunus vers un autre périphérique bloc. Le format d'une cible linéaire est comme suit :

```
start length linear device offset
```

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

device

périphérique en bloc, référencé par le nom d'un périphérique dans le système de fichiers ou par les nombres mineurs ou majeurs sous les formats ***major*:*minor***

offset

démarrer le déport du mappage sur le périphérique

L'exemple suivant montre une cible linéaire avec un bloc de départ dans le périphérique virtuel de 0, une longueur de segment de 1638400, une paire de nombre majeur:mineur de 8:2, et un déport de départ de 41146992 pour le périphérique.

```
0 16384000 linear 8:2 41156992
```

L'exemple suivant montre une cible linéaire avec un paramètre de périphérique spécifié en tant que périphérique **/dev/hda**.

```
0 20971520 linear /dev/hda 384
```

A.1.2. La cible de mappage par bandes

La cible de mappage par bandes prend en charge le dénudage pour tous les périphériques physiques. Elle prend pour arguments le nombre de bandes et la taille de la tranche de la bande , suivi par une liste de paires de nom et secteur. Le format de la cible par bandes est le suivante :

```
start length striped #stripes chunk_size device1 offset1 ... deviceN
offsetN
```

Il y a un ensemble de paramètres de ***device*** (périphérique) et de ***offset*** (déport) pour chaque bande.

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

#stripes

nombre de bandes pour chaque périphérique virtuel

chunk_size

le nombre de secteurs écrit pour chaque bande avant de passer à la prochaine, doit être à la puissance 2 au moins en taille par rapport à la page du noyau.

device

périphérique en bloc, référencé par le nom du périphérique dans le système de fichiers ou par les nombres mineurs ou majeurs sous le format ***major*:*minor***.

offset

démarrer le déport du mappage sur le périphérique

L'exemple suivant montre une cible par bandes comprenant trois bandes et une taille de tranche de 128 :

```
0 73728 striped 3 128 8:9 384 8:8 384 8:7 9789824
```

0

Blocs de départ en périphérique virtuel

73728

longueur de ce segment

par bandes 3 128

bandes sur trois périphériques avec des tailles de tranches de 128 blocs

8:9

Nombres majeurs:mineurs du premier périphérique

384

démarrage du déport de mappage sur le premier périphérique

8:8

Nombres majeurs:mineurs du second périphérique

384

démarrage du déport de mappage sur le second périphérique

8:7

Nombres majeurs:mineurs sur le troisième périphérique

9789824

démarrage du déport du mappage sur le troisième périphérique

L'exemple suivant montre une cible pour 2 bandes comprenant 256 Kio, avec des paramètres de périphérique spécifiés par les noms de périphérique dans le système de fichiers plutôt que par les nombres majeurs ou mineurs.

```
0 65536 striped 2 512 /dev/hda 0 /dev/hdb 0
```

A.1.3. La cible de mappage en miroir

La cible de mappage en miroir prend en charge le mappage du périphérique logique en miroir. Le format de la cible en miroir est le suivant :

```
start length mirror log_type #logargs logarg1 ... logargN #devs device1
offset1 ... deviceN offsetN
```

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

log_type

Les types de log possibles et leurs arguments sont les suivants :

core

Le miroir est local et le log de miroir est gardé dans la mémoire centrale. Ce type de log contient 1 - 3 arguments :

regionsize **[[no]sync] [block_on_error]**

disk

Le miroir est local et le log du miroir est gradé sur le disque. Ce type de log contient 2 - 4 arguments :

logdevice regionsize **[[no]sync] [block_on_error]**

clustered_core

Le miroir est clusterisé et le log de miroir est conservé dans la mémoire centrale. Ce type de log contient 2 - 4 arguments :

regionsize UUID **[[no]sync] [block_on_error]**

clustered_disk

Le miroir est clusterisé et le log du miroir est conservé sur le disque. Ce type de log contient 3 - 5 arguments :

logdevice regionsize UUID **[[no]sync] [block_on_error]**

LVM maintient un petit journal qui est utilisé pour garder la trace des régions qui sont synchrônisées avec le ou les miroirs. L'argument *regionsize* précise la taille de ces régions.

Dans un environnement clusterisé, l'argument *UUID* est un identifiant unique, associé au périphérique du log du miroir, de façon à ce que l'état du log puisse être maintenu à travers le cluster.

The optional **[no]sync** argument can be used to specify the mirror as "in-sync" or "out-of-sync". The **block_on_error** argument is used to tell the mirror to respond to errors rather than ignoring them.

#log_args

nombre d'arguments de logs qui seront spécifiés dans le mappage

logargs

Les arguments de log du miroir, le nombre d'arguments de log fourni est spécifié par le paramètre ***#log-args*** et les arguments de log recevables sont déterminés par le paramètre ***log_type***.

#devs

le nombre de branches du miroir; un périphérique et un déport (offset) sont spécifiés pour chaque branche.

device

périphérique en bloc pour chaque branche du miroir, référencé par le nom du périphérique dans le système de fichiers ou par les nombres mineurs ou majeurs sous le format ***major:minor***. Un périphérique en bloc et un déport (offset) sont précisés pour chaque branche de miroir, comme indiqué par le paramètre ***#devs***.

offset

démarrage du mappage sur le périphérique. Un périphérique en bloc est un déport (offset) sont spécifiés pour chaque branche de miroir, comme indiqué par le paramètre ***#devs***.

L'exemple suivant montre une cible de mappage en miroir pour un miroir clusterisé, accompagné d'un log de miroir attaché au disque.

```
0 52428800 mirror clustered_disk 4 253:2 1024 UUID block_on_error 3 253:3
0 253:4 0 253:5 0
```

0

Blocs de départ en périphérique virtuel

52428800

longueur de ce segment

mirror clustered_disk

cible en miroir avec un type de log spécifiant que le miroir est clusterisé et que le log en miroir est gardé sur le disque.

4

4 arguments de log en miroir vont suivre

253:2

nombres majeur:mineur de périphériques de logs

1024

taille de région que le log en miroir utilise pour garder la trace de ce qui est synchrônisé

UUID

UUID du périphérique du log en miroir pour conserver l'information du log dans tout le cluster

block_on_error

le miroir doit répondre aux erreurs

3

nombres de branches du miroir

253:3 0 253:4 0 253:5 0

les nombres majeur:mineur et les déports (offset) des périphériques constituent chaque branche du miroir

A.1.4. Les cibles de mappage de l'instantané et de l'instantané d'origine

Quand vous créez le premier instantané LVM d'un volume, on utilise quatre mappeurs de périphérique :

1. Un périphérique comprenant un mappage **linéaire** contenant la table de mappage d'origine du volume de la source.
2. Un périphérique avec un mappage **linéaire**utilisé comme périphérique COW (de l'anglais copy-on-write ou copie-sur-écriture) pour le volume source, pour chaque écriture, les données d'origine sont sauvegardées dans le périphérique COW de chaque instantané pour garder son contenu visible inchangé (jusqu'à ce que le périphérique COW soit rempli).
3. Un périphérique contenant un mappage **instantané** combinant #1 et #2, qui est le volume d'instantanés visibles
4. The "original" volume (which uses the device number used by the original source volume), whose table is replaced by a "snapshot-origin" mapping from device #1.

On utilise un modèle de noms fixes pour créer ces périphériques. Par exemple, vous pourrez utiliser les commandes suivantes pour créer un volume LVM intitulé **base** et un volume d'instantané intitulé **snap** bas. sur ce volume.

```
# lvcreate -L 1G -n base volumeGroup
# lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

Cela produit quatre périphériques, que vous pouvez apercevoir par les commandes suivantes :

```
# dmsetup table|grep volumeGroup
volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-snap-cow: 0 204800 linear 8:19 2097536
volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-*
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-
real
brw-------  1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-
cow
brw-------  1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
brw-------  1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

Le format de la cible **instantané-d'origine** est le suivant :

```
start length snapshot-origin origin
```

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

origin

volume de base d'instantané

L'**instantané-d'origine** aura normalement un ou plusieurs instantanés basés dessus. Les lectures seront directement mappées sur le périphérique de sauvegarde. Pour chaque écriture, les données d'origine seront sauvegardées dans le périphérique COW de chaque instantané pour garder leur contenu visible inchangé, jusqu'à ce que le périphérique COW soit rempli.

Le format de la cible **instantané** est le suivant :

```
start length snapshot origin COW-device P|N chunksize
```

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

origin

volume de base d'instantané

COW-device

Périphérique stockant les morceaux de données modifiées

P|N

P (Persistant) ou N (Non persistant); indique si l'instantané survivra après le redémarrage. Pour les instantanés transitoires (N), on doit sauvegarder moins de metadonnées sur le disque. Ils peuvent être contenus dans la mémoire par le noyau.

chunksize

Taille en sectuers de morceaux de données modifiées, qui seront stockés dans le périphérique COW

L'exemple suivant montre une cible **instantané-d'origine** avec un périphérique d'origine de 254:11.

```
0 2097152 snapshot-origin 254:11
```

L'exemple suivant montre une cible **instantané** avec un périphérique d'origine de 254:11 et un périphérique COW de 254:12. Ce périphérique d'instantané est persistant à travers les démarrages et la taille du morceau des données stockées sur le périphérique COW est de 16 secteurs.

```
0 2097152 snapshot 254:11 254:12 P 16
```

A.1.5. La cible de mappage d'erreurs

Avec une cible de mappage d'erreurs, toute opération E/S vers le secteur mappé échouera.

Une cible de mappage d'erreurs peut être utilisée pour tester. Pour tester comment un périphérique va se comporter en situation d'échec, vous pouvez créer un mappage de périphérique avec un mauvais secteur au milieu d'un périphérique, ou bien, vous pouvez échanger la branche du miroir par une cible d'erreur.

Une cible d'erreurs peut être utilisée à la place d'un périphérique défaillant, comme moyen d'éviter les timeouts et les nouvelles tentatives sur les périphériques. Elle peut servir en tant que cible intermédiaire tandis que vous réarrangez les métadonnées LVM en cas d'échec.

La cible de mappage **erreur** n'accepte pas de paramètres supplémentaires en dehors de *start*(démarrage) et de *length* (longueur).

L'exemple suivant montre une cible **erreur**.

```
0 65536 error
```

A.1.6. Le cible de mappage zéro

La cible de mappage **zero** est un périphérique en bloc équivalent à **/dev/zero**. Toute opération lecture de ce mappage retourne des blocs de zéro. Les données inscrites sur ce mappage sont ignorées, mais les opérations écriture aboutissent à leur fin. La cible de mappage **zero** n'accepte pas

de paramètres supplémentaires en dehors de *start* (démarrage) et de *length* (longueur).

L'exemple suivant montre une cible **zéro** pour un périphérique de 16To.

```
0 65536 zero
```

A.1.7. La cible de mappage multivoies

La cible de mappage multivoies supporte le mappage d'un périphérique multivoies. Le format de la cible **multivoie** est la suivante :

```
start length  multipath  #features [feature1 ... featureN] #handlerargs
[handlerarg1 ... handlerargN] #pathgroups pathgroup pathgroupargs1 ...
pathgroupargsN
```

Il existe un ensemble de paramètres ***pathgroupargs*** pour chaque groupe de chemins d'accès.

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

#features

Le nombre de fonctionnalités en multivoies, suivi par ces fonctionnalités. Si ce paramètre est zéro, alors il n'y a pas de paramètre ***fonctionnalité*** et le prochain paramètre de mappage de périphérique sera ***#handlerargs***. Actuellement, il existe une fonctionnalité multivoies prise en charge, ***#handlerargs***. Elle indique que ce périphérique multivoies est actuellement configuré pour empiler en liste d'attente des opérations E/S s'il n'y a pas de chemin disponible.

Ainsi, si l'option **no_path_retry** du fichier **multipath.conf** a été configuré de façon à s'empiler en file d'attente des opérations E/S jusqu'à ce que tous les chemins d'accès aient été marqués 'failed' (échec) au bout d'un certain nombre de tentatives pour utiliser les chemins d'accès, le mappage apparaîtrait ainsi jusqu'à ce que tous les contrôleurs de chemin d'accès aient 'échoué' le nombre de contrôles spécifiés.

```
0 71014400 multipath 1 queue_if_no_path 0 2 1 round-robin 0 2 1 66:128 \
1000 65:64 1000 round-robin 0 2 1 8:0 1000 67:192 1000
```

Après que tous les contrôleurs aient échoué sur le nombre de contrôles spécifiés, le mappage ressemblerait à ce qui suit :

```
0 71014400 multipath 0 0 2 1 round-robin 0 2 1 66:128 1000 65:64 1000 \
round-robin 0 2 1 8:0 1000 67:192 1000
```

#handlerargs

Le nombre d'arguments de gestionnaires de matériel, suivi par ces arguments. Un gestionnaire de matériel spécifie un module qui sera utilisé pour effectuer des actions spécifiques à des matériels particuliers au moment des changements de chemins d'accès de groupes ou en cours de gestion des erreurs E/S. Si la configuration est 0, alors le prochain paramètre sera ***#pathgroups***.

#pathgroups

Le nombre de groupes de chemins d'accès. Un groupe de chemins d'accès correspond à l'ensemble des chemins d'accès à partir duquel un périphérique multivoies effectuera l'équilibre de ses charges. Il existe un ensemble de paramètres ***pathgroupargs*** pour chaque

pathgroup

Le prochain groupe de chemins d'accès à essayer.

pathgroupsargs

Chaque groupe de chemins d'accès consiste aux arguments suivants :

```
pathselector #selectorargs #paths #pathargs device1 ioreqs1 ... deviceN
ioreqsN
```

Il existe un ensemble d'arguments de chemins d'accès pour chaque chemin dans un groupe de chemins d'accès.

pathselector

Précise l'algorithme qui est utilisé pour déterminer le chemin d'accès dans ce groupe de chemins d'accès, à utiliser pour la prochaine opération E/S.

#selectorargs

Le nombre d'arguments de sélecteurs de chemins d'accès qui suivent cet argument, est le mappage multivoies. Actuellement, la valeur de cet argument est toujours 0.

#paths

Le nombre de chemins d'accès dans ce groupe de chemins d'accès.

#pathargs

Le nombre d'arguments de chemins d'accès spécifiés pour chaque chemin compris dans ce groupe. Actuellement, ce nombre est toujours de 1, l'argument ***ioreqs***.

device

Le numéro de périphérique en bloc du chemin, référencé par les nombres majeurs et mineurs, sous le format ***major:minor***

ioreqs

Le nombre de requêtes E/S à diriger vers ce chemin avant de passer au prochain chemin du groupe en question.

Figure A.1, « Cible de mappage multivoies » shows the format of a multipath target with two path groups.

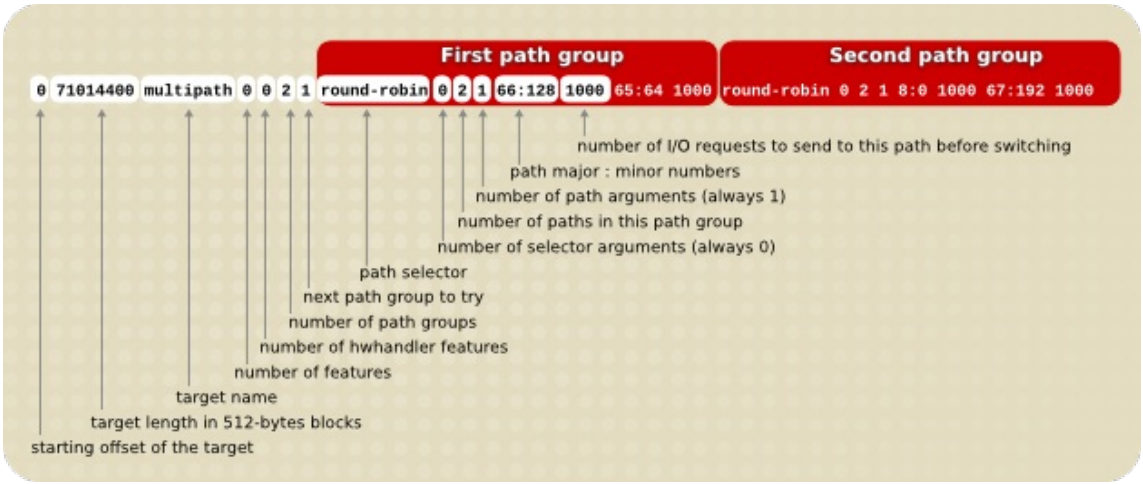


Figure A.1. Cible de mappage multivoies

L'exemple suivant montre une définition de cible en failover pure dans un même périphérique multivoies. Pour cette cible, il existe quatre groupes de chemins, dont un seul chemin ouvert par groupe de chemins, de façon à ce que le périphérique multivoies n'utilise qu'un seul chemin à la fois.

```
0 71014400 multipath 0 0 4 1 round-robin 0 1 1 66:112 1000 \
round-robin 0 1 1 67:176 1000 round-robin 0 1 1 68:240 1000 \
round-robin 0 1 1 65:48 1000
```

L'exemple suivant montre la définition d'une cible multibus complètement déployée, dans un même périphérique multivoies. Pour cette cible, il n'y a qu'un seul groupe de chemins, qui inclut tous les chemins. Dans cette installation, multivoies déploie la charge

```
0 71014400 multipath 0 0 1 1 round-robin 0 4 1 66:112 1000 \
 67:176 1000 68:240 1000 65:48 1000
```

Pour plus d'informations à propos de multivoies, consultez le document *Using Device Mapper Multipath* (Utiliser le mappeur de périphériques multivoies).

A.1.8. La cible de mappage de cryptage

La cible **crypt** cible la codification des données qui passent par le périphérique spécifié. Il utilise l'API Crypto du noyau.

Le format de la cible **crypt** est le suivant :

```
start length crypt cipher key IV-offset device offset
```

start

Blocs de départ en périphérique virtuel

length

longueur de ce segment

cipher

Cipher consiste en ***cipher[-chainmode]-ivmode[:iv options]***.

cipher

Les ciphers disponibles sont listés dans **/proc/crypto** (par exemple, **aes**).

chainmode

Toujours utiliser **cbc**. Ne pas utiliser **ebc**; qui n'utilise pas un Vecteur Initial (IV) (de l'anglais Initial Vector).

ivmode[:iv options]

IV est un vecteur initial qui est utilisé pour faire varier le cryptage. Le mode de IV est soit **plain** ou bien **essiv:hash**. Un ***ivmode*** de **-plain** utilise le nombre de secteurs (plus IV offset) comme IV. Un ***ivmode*** de **-essiv** consistue un progrès par rapport au problème de filigrane numérique (watermark).

key

Clé de cryptage, fournie en hex

IV-offset

Initial Vector (IV) offset

device

périphérique en bloc, référencé par le nom d'un périphérique dans le système de fichiers ou par les nombres mineurs ou majeurs sous les formats ***major:minor***

offset

démarrer le déport du mappage sur le périphérique

L'exemple suivant est un exemple d'une cible **crypt**.

```
0 2097152 crypt aes-plain 0123456789abcdef0123456789abcdef 0 /dev/hda 0
```

A.2. LA COMMANDE DMSETUP

La commande **dmsetup** est une enveloppe de lignes de commande pour la communication avec le mappeur de périphériques. Pour obtenir des informations système à propos des périphériques LVM, vous allez sûrement trouver les options **info**, **ls**, **status**, et **deps** de la commande **dmsetup** utiles, comme expliqué dans les sous-sections suivantes.

Pour obtenir davantage d'informations à propos des fonctionnalités et des options de la commande **dmsetup**, reportez-vous à la page de manuel **dmsetup**(8).

A.2.1. La commande info dmsetup

La commande **dmsetup info *device*** fournit des informations sur les périphériques de Device Mapper. Si vous spécifiez un périphérique, alors cette commande produira des informations pour ce périphérique uniquement.

La commande **dmsetup info** fournit des informations sur les catégories suivantes :

Name

Le nom du périphérique. Un périphérique LVM est exprimé en tant que nom de groupe de volume et le nom du volume logique, séparé par un trait d'union. Un trait d'union qui se trouve dans le nom d'origine, est traduit par deux traits d'union.

State

Les états de périphériques possibles sont **SUSPENDED**, **ACTIVE**, et **READ-ONLY**. La commande **dmsetup suspend** configure l'état du périphérique à **SUSPENDED**. Quand un périphérique est suspendu, toutes les opérations E/S de ce périphérique s'interrompent. La commande **dmsetup resume** restaure un état de périphérique à **ACTIVE**.

Read Ahead

Le nombre de blocs de données qu'un système lit préliminairement à l'ouverture d'un fichier pour lequel les opérations de lecture sont en cours. Par défaut, le noyau choisit une valeur convenable automatiquement. Vous pourrez changer cette valeur par l'option **--readahead** de la commande **dmsetup**.

Tables present

Possible states for this category are **LIVE** and **INACTIVE**. An **INACTIVE** state indicates that a table has been loaded which will be swapped in when a **dmsetup resume** command restores a device state to **ACTIVE**, at which point the table's state becomes **LIVE**. For information, see the **dmsetup** man page.

Open count

La référence ouverte 'count' indique le nombre de fois que le périphérique a été ouvert. Une commande **mount** ouvre un périphérique.

Event number

The current number of events received. Issuing a **dmsetup wait *n*** command allows the user to wait for the n'th event, blocking the call until it is received.

Major, minor

Nombre de périphériques majeur ou mineur

Number of targets

Le nombre de fragments qui constituent un périphérique. Par exemple, un périphérique linéaire portant sur 3 disques, aurait 3 cibles. Un périphérique linéaire composé par le début et la fin d'un disque, mais sans de partie intermédiaire (milieu) aurait 2 cibles.

UUID

UUID du périphérique

L'exemple suivant montre une sortie incomplète pour la commande **dmsetup info**.

```
[root@ask-07 ~]# dmsetup info
Name:              testgfsvg-testgfslv1
State:             ACTIVE
Read Ahead:        256
```

```
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 2
Number of targets: 2
UUID: LVM-K528WUGQgPadNXYcFrrf9LnPlUMswgkCkpgPIgYzSvigM7SfeWCypddNSWtNzc2N
...
Name:              VolGroup00-LogVol00
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        1
Event number:      0
Major, minor:      253, 0
Number of targets: 1
UUID: LVM-tOcS1kqFV9drb0X1Vr8sxeYP0tqcrpdegyqj5lZxe45JMGlmvtqLmbLpBcenh2L3
```

A.2.2. La commande dmsetup ls

Vous pouvez lister les noms des périphériques mappés par la commande **dmsetup ls**. Vous pourrez lister des périphériques qui ont au moins une cible spécifique par la commande **dmsetup ls --target *target_type***. Pour les autres options de la commande **dmsetup ls**, voir les pages de manuel **dmsetup**.

L'exemple suivant montre les commandes pour lister les noms des périphériques mappés et actuellement configurés.

```
[root@ask-07 ~]# dmsetup ls
testgfsvg-testgfslv3    (253, 4)
testgfsvg-testgfslv2    (253, 3)
testgfsvg-testgfslv1    (253, 2)
VolGroup00-LogVol01     (253, 1)
VolGroup00-LogVol00     (253, 0)
```

L'exemple suivant montre la commande pour lister les noms de périhériques des mappagesen miroir actuellement configurés.

```
[root@grant-01 ~]# dmsetup ls --target mirror
lock_stress-grant--02.1722      (253, 34)
lock_stress-grant--01.1720      (253, 18)
lock_stress-grant--03.1718      (253, 52)
lock_stress-grant--02.1716      (253, 40)
lock_stress-grant--03.1713      (253, 47)
lock_stress-grant--02.1709      (253, 23)
lock_stress-grant--01.1707      (253, 8)
lock_stress-grant--01.1724      (253, 14)
lock_stress-grant--03.1711      (253, 27)
```

A.2.3. La commande de statut dmsetup

La commande **dmsetup status *device*** fournit des informations pour chaque cible pour un périphérique particulier. Si vous ne spécifiez pas le nom du périphérique, la sortie correspondra à des informations sur tous les périphériques de Device Mapper actuellement configurés. Vous ne pouvez

uniquement lister que les statuts de périphériques qui ont au moins une cible d'un type particulier par la commande **dmsetup status --target *target_type***.

L'exemple suivant montre la commande qui est utilisée pour lister le statut des cibles des périphériques mappés actuellement configurés.

```
[root@ask-07 ~]# dmsetup status
testgfsvg-testgfslv3: 0 312352768 linear
testgfsvg-testgfslv2: 0 312352768 linear
testgfsvg-testgfslv1: 0 312352768 linear
testgfsvg-testgfslv1: 312352768 50331648 linear
VolGroup00-LogVol01: 0 4063232 linear
VolGroup00-LogVol00: 0 151912448 linear
```

A.2.4. La commande dmsetup deps

La commande **dmsetup deps *device*** fournit une liste de paires (major, minor) pour des périphériques référencés par la table de mappage, pour un périphérique particulier. Si vous ne spécifiez pas un nom de périphérique, la sortie portera sur des informations provenant des périphériques de Device Mapper actuellement configurés.

L'exemple suivant montre la commande qui est utilisée pour lister les dépendances de tous les périphériques mappés actuellement configurés.

```
[root@ask-07 ~]# dmsetup deps
testgfsvg-testgfslv3: 1 dependencies    : (8, 16)
testgfsvg-testgfslv2: 1 dependencies    : (8, 16)
testgfsvg-testgfslv1: 1 dependencies    : (8, 16)
VolGroup00-LogVol01: 1 dependencies     : (8, 2)
VolGroup00-LogVol00: 1 dependencies     : (8, 2)
```

L'exemple suivant montre la commande utilisée pour lister les dépendances du périphérique uniquement **lock_stress-grant--02.1722**:

```
[root@grant-01 ~]# dmsetup deps lock_stress-grant--02.1722
3 dependencies  : (253, 33) (253, 32) (253, 31)
```

ANNEXE B. LES FICHIERS DE CONFIGURATION LVM

LVM supporte plusieurs fichiers de configuration. Au démarrage du système, le fichier de configuration **lvm.conf** est chargé à partir du répertoire spécifié par la variable d'environnement **LVM_SYSTEM_DIR** dont la valeur par défaut est **/etc/lvm**.

Le fichier **lvm.conf** peut spécifier des fichiers de configuration supplémentaires à charger. Les paramètres des fichiers chargés en dernier surchargent les paramètres des premiers fichiers. Pour afficher les paramètres en cours d'utilisation après avoir chargé tous les fichiers de configuration, exécutez la commande **lvm dumpconfig**.

For information on loading additional configuration files, see Section C.2, « Les balises hôtes ».

B.1. LES FICHIERS DE CONFIGURATION LVM

Les fichiers suivants sont utilisés pour la configuration LVS :

/etc/lvm/lvm.conf

Fichier de configuration central lu par les outils.

etc/lvm/lvm_*hosttag*.conf

For each host tag, an extra configuration file is read if it exists: **lvm_*hosttag*.conf**. If that file defines new tags, then further configuration files will be appended to the list of tiles to read in. For information on host tags, see Section C.2, « Les balises hôtes ».

En plus des fichiers de configuration LVM, un système exécutant LVS comporte des fichiers qui affectent l'installation du système LVM.

/etc/lvm/.cache

Fichier de cache pour le filtre des noms de périphériques (configurable).

/etc/lvm/backup/

Répertoire pour les sauvegardes automatiques des métadonnées d'un groupe de volumes (configurable).

/etc/lvm/archive/

Répertoire pour l'archivage automatique des métadonnées d'un groupe de volumes (configurable en fonction du chemin d'accès au répertoire et de la taille des archives).

/var/lock/lvm/

Dans les configurations avec un seul hôte, verrouillez les fichiers pour empêcher la corruption des métadonnées due à l'exécution parallèle de plusieurs outils ; dans un cluster, DLM à l'échelle du cluster est utilisé.

B.2. ÉCHANTILLON DU FICHIER LVM.CONF

Voici un exemple de fichier de configuration **lvm.conf**. Ce fichier de configuration est le fichier par défaut de la version de sortie RHEL 5.3. Si votre système exécute un version différente de RHEL5, certains paramètres de configuration pourraient varier.

```
[root@tng3-1 lvm]# cat /etc/lvm/lvm.conf
# This is an example configuration file for the LVM2 system.
# It contains the default settings that would be used if there was no
# /etc/lvm/lvm.conf file.
#
# Refer to 'man lvm.conf' for further information including the file
layout.
#
# To put this file in a different directory and override /etc/lvm set
# the environment variable LVM_SYSTEM_DIR before running the tools.


# This section allows you to configure which block devices should
# be used by the LVM system.
devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # If several entries in the scanned directories correspond to the
    # same block device and the tools need to display a name for device,
    # all the pathnames are matched against each item in the following
    # list of regular expressions in turn and the first match is used.
    preferred_names = [ ]

    # Try to avoid using undescriptive /dev/dm-N names, if present.
    # preferred_names = [ "^/dev/mpath/", "^/dev/mapper/mpath",
"^/dev/[hs]d" ]

    # A filter that tells LVM2 to only use a restricted set of devices.
    # The filter consists of an array of regular expressions.  These
    # expressions can be delimited by a character of your choice, and
    # prefixed with either an 'a' (for accept) or 'r' (for reject).
    # The first expression found to match a device name determines if
    # the device will be accepted or rejected (ignored).  Devices that
    # don't match any patterns are accepted.

    # Be careful if there there are symbolic links or multiple filesystem
    # entries for the same device as each name is checked separately
against
    # the list of patterns.  The effect is that if any name matches any
'a'
    # pattern, the device is accepted; otherwise if any name matches any
'r'
    # pattern it is rejected; otherwise it is accepted.

    # Don't have more than one filter line active at once: only one gets
used.

    # Run vgscan after you change this parameter to ensure that
    # the cache file gets regenerated (see below).
    # If it doesn't do what you expect, check the output of 'vgscan -
```

```
vvvv'.


    # By default we accept every block device:
    filter = [ "a/.*/" ]

    # Exclude the cdrom drive
    # filter = [ "r|/dev/cdrom|" ]

    # When testing I like to work with just loopback devices:
    # filter = [ "a/loop/", "r/.*/" ]

    # Or maybe all loops and ide drives except hdc:
    # filter =[ "a|loop|", "r|/dev/hdc|", "a|/dev/ide|", "r|.*|" ]

    # Use anchors if you want to be really specific
    # filter = [ "a|^/dev/hda8$|", "r/.*/" ]

    # The results of the filtering are cached on disk to avoid
    # rescanning dud devices (which can take a very long time).
    # By default this cache is stored in the /etc/lvm/cache directory
    # in a file called '.cache'.
    # It is safe to delete the contents: the tools regenerate it.
    # (The old setting 'cache' is still respected if neither of
    # these new ones is present.)
    cache_dir = "/etc/lvm/cache"
    cache_file_prefix = ""

    # You can turn off writing this cache file by setting this to 0.
    write_cache_state = 1

    # Advanced settings.

    # List of pairs of additional acceptable block device types found
    # in /proc/devices with maximum (non-zero) number of partitions.
    # types = [ "fd", 16 ]

    # If sysfs is mounted (2.6 kernels) restrict device scanning to
    # the block devices it believes are valid.
    # 1 enables; 0 disables.
    sysfs_scan = 1

    # By default, LVM2 will ignore devices used as components of
    # software RAID (md) devices by looking for md superblocks.
    # 1 enables; 0 disables.
    md_component_detection = 1

    # By default, if a PV is placed directly upon an md device, LVM2
    # will align its data blocks with the the chunk_size exposed in sysfs.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1

    # If, while scanning the system for PVs, LVM2 encounters a device-
mapper
    # device that has its I/O suspended, it waits for it to become
accessible.
```

```
    # Set this to 1 to skip such devices.  This should only be needed
    # in recovery situations.
    ignore_suspended_devices = 0
}

# This section that allows you to configure the nature of the
# information that LVM2 reports.
log {

    # Controls the messages sent to stdout or stderr.
    # There are three levels of verbosity, 3 being the most verbose.
    verbose = 0

    # Should we send log messages through syslog?
    # 1 is yes; 0 is no.
    syslog = 1

    # Should we log error and debug messages to a file?
    # By default there is no log file.
    #file = "/var/log/lvm2.log"

    # Should we overwrite the log file each time the program is run?
    # By default we append.
    overwrite = 0

    # What level of log messages should we send to the log file and/or
syslog?
    # There are 6 syslog-like log levels currently in use - 2 to 7
inclusive.
    # 7 is the most verbose (LOG_DEBUG).
    level = 0

    # Format of output messages
    # Whether or not (1 or 0) to indent messages according to their
severity
    indent = 1

    # Whether or not (1 or 0) to display the command name on each line
output
    command_names = 0

    # A prefix to use before the message text (but after the command name,
    # if selected).  Default is two spaces, so you can see/grep the
severity
    # of each message.
    prefix = "  "

    # To make the messages look similar to the original LVM tools use:
    #   indent = 0
    #   command_names = 1
    #   prefix = " -- "

    # Set this if you want log messages during activation.
    # Don't use this in low memory situations (can deadlock).
    # activation = 0
}
```

```
# Configuration of metadata backups and archiving.  In LVM2 when we
# talk about a 'backup' we mean making a copy of the metadata for the
# *current* system.  The 'archive' contains old metadata configurations.
# Backups are stored in a human readeable text format.
backup {

    # Should we maintain a backup of the current metadata configuration ?
    # Use 1 for Yes; 0 for No.
    # Think very hard before turning this off!
    backup = 1

    # Where shall we keep it ?
    # Remember to back up this directory regularly!
    backup_dir = "/etc/lvm/backup"

    # Should we maintain an archive of old metadata configurations.
    # Use 1 for Yes; 0 for No.
    # On by default.  Think very hard before turning this off.
    archive = 1

    # Where should archived files go ?
    # Remember to back up this directory regularly!
    archive_dir = "/etc/lvm/archive"

    # What is the minimum number of archive files you wish to keep ?
    retain_min = 10

    # What is the minimum time you wish to keep an archive file for ?
    retain_days = 30
}

# Settings for the running LVM2 in shell (readline) mode.
shell {

    # Number of lines of history to store in ~/.lvm_history
    history_size = 100
}


# Miscellaneous global LVM2 settings
global {
    library_dir = "/usr/lib64"

    # The file creation mask for any files and directories created.
    # Interpreted as octal if the first digit is zero.
    umask = 077

    # Allow other users to read the files
    #umask = 022

    # Enabling test mode means that no changes to the on disk metadata
    # will be made.  Equivalent to having the -t option on every
    # command.  Defaults to off.
    test = 0
```

```
    # Default value for --units argument
    units = "h"

    # Whether or not to communicate with the kernel device-mapper.
    # Set to 0 if you want to use the tools to manipulate LVM metadata
    # without activating any logical volumes.
    # If the device-mapper kernel driver is not present in your kernel
    # setting this to 0 should suppress the error messages.
    activation = 1

    # If we can't communicate with device-mapper, should we try running
    # the LVM1 tools?
    # This option only applies to 2.4 kernels and is provided to help you
    # switch between device-mapper kernels and LVM1 kernels.
    # The LVM1 tools need to be installed with .lvm1 suffices
    # e.g. vgscan.lvm1 and they will stop working after you start using
    # the new lvm2 on-disk metadata format.
    # The default value is set when the tools are built.
    # fallback_to_lvm1 = 0

    # The default metadata format that commands should use - "lvm1" or
"lvm2".
    # The command line override is -M1 or -M2.
    # Defaults to "lvm1" if compiled in, else "lvm2".
    # format = "lvm1"

    # Location of proc filesystem
    proc = "/proc"

    # Type of locking to use. Defaults to local file-based locking (1).
    # Turn locking off by setting to 0 (dangerous: risks metadata
corruption
    # if LVM2 commands get run concurrently).
    # Type 2 uses the external shared library locking_library.
    # Type 3 uses built-in clustered locking.
    locking_type = 3

    # If using external locking (type 2) and initialisation fails,
    # with this set to 1 an attempt will be made to use the built-in
    # clustered locking.
    # If you are using a customised locking_library you should set this to
0.
    fallback_to_clustered_locking = 1

    # If an attempt to initialise type 2 or type 3 locking failed, perhaps
    # because cluster components such as clvmd are not running, with this
set
    # to 1 an attempt will be made to use local file-based locking (type
1).
    # If this succeeds, only commands against local volume groups will
proceed.
    # Volume Groups marked as clustered will be ignored.
    fallback_to_local_locking = 1

    # Local non-LV directory that holds file-based locks while commands
are
```

```
    # in progress.  A directory like /tmp that may get wiped on reboot is
OK.
    locking_dir = "/var/lock/lvm"

    # Other entries can go here to allow you to load shared libraries
    # e.g. if support for LVM1 metadata was compiled as a shared library
use
    #   format_libraries = "liblvm2format1.so"
    # Full pathnames can be given.

    # Search this directory first for shared libraries.
    #   library_dir = "/lib"

    # The external locking library to load if locking_type is set to 2.
    #   locking_library = "liblvm2clusterlock.so"
}

activation {
    # How to fill in missing stripes if activating an incomplete volume.
    # Using "error" will make inaccessible parts of the device return
    # I/O errors on access.  You can instead use a device path, in which
    # case, that device will be used to in place of missing stripes.
    # But note that using anything other than "error" with mirrored
    # or snapshotted volumes is likely to result in data corruption.
    missing_stripe_filler = "error"

    # How much stack (in KB) to reserve for use while devices suspended
    reserved_stack = 256

    # How much memory (in KB) to reserve for use while devices suspended
    reserved_memory = 8192

    # Nice value used while devices suspended
    process_priority = -18

    # If volume_list is defined, each LV is only activated if there is a
    # match against the list.
    #   "vgname" and "vgname/lvname" are matched exactly.
    #   "@tag" matches any tag set in the LV or VG.
    #   "@*" matches if any tag defined on the host is also set in the LV
or VG
    #
    # volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]

    # Size (in KB) of each copy operation when mirroring
    mirror_region_size = 512

    # Setting to use when there is no readahead value stored in the
metadata.
    #
    # "none" - Disable readahead.
    # "auto" - Use default value chosen by kernel.
    readahead = "auto"

    # 'mirror_image_fault_policy' and 'mirror_log_fault_policy' define
    # how a device failure affecting a mirror is handled.
```

```
    # A mirror is composed of mirror images (copies) and a log.
    # A disk log ensures that a mirror does not need to be re-synced
    # (all copies made the same) every time a machine reboots or crashes.
    #
    # In the event of a failure, the specified policy will be used to
    # determine what happens:
    #
    # "remove" - Simply remove the faulty device and run without it.  If
    #            the log device fails, the mirror would convert to using
    #            an in-memory log.  This means the mirror will not
    #            remember its sync status across crashes/reboots and
    #            the entire mirror will be re-synced.  If a
    #            mirror image fails, the mirror will convert to a
    #            non-mirrored device if there is only one remaining good
    #            copy.
    #
    # "allocate" - Remove the faulty device and try to allocate space on
    #            a new device to be a replacement for the failed device.
    #            Using this policy for the log is fast and maintains the
    #            ability to remember sync state through crashes/reboots.
    #            Using this policy for a mirror device is slow, as it
    #            requires the mirror to resynchronize the devices, but it
    #            will preserve the mirror characteristic of the device.
    #            This policy acts like "remove" if no suitable device and
    #            space can be allocated for the replacement.
    #            Currently this is not implemented properly and behaves
    #            similarly to:
    #
    # "allocate_anywhere" - Operates like "allocate", but it does not
    #            require that the new space being allocated be on a
    #            device is not part of the mirror.  For a log device
    #            failure, this could mean that the log is allocated on
    #            the same device as a mirror device.  For a mirror
    #            device, this could mean that the mirror device is
    #            allocated on the same device as another mirror device.
    #            This policy would not be wise for mirror devices
    #            because it would break the redundant nature of the
    #            mirror.  This policy acts like "remove" if no suitable
    #            device and space can be allocated for the replacement.

    mirror_log_fault_policy = "allocate"
    mirror_device_fault_policy = "remove"
}


####################
# Advanced section #
####################

# Metadata settings
#
# metadata {
    # Default number of copies of metadata to hold on each PV.  0, 1 or 2.
    # You might want to override it from the command line with 0
    # when running pvcreate on new PVs which are to be added to large VGs.
```

```
    # pvmetadatacopies = 1

    # Approximate default size of on-disk metadata areas in sectors.
    # You should increase this if you have large volume groups or
    # you want to retain a large on-disk history of your metadata changes.

    # pvmetadatasize = 255

    # List of directories holding live copies of text format metadata.
    # These directories must not be on logical volumes!
    # It's possible to use LVM2 with a couple of directories here,
    # preferably on different (non-LV) filesystems, and with no other
    # on-disk metadata (pvmetadatacopies = 0). Or this can be in
    # addition to on-disk metadata areas.
    # The feature was originally added to simplify testing and is not
    # supported under low memory situations - the machine could lock up.
    #
    # Never edit any files in these directories by hand unless you
    # you are absolutely sure you know what you are doing! Use
    # the supplied toolset to make changes (e.g. vgcfgrestore).

    # dirs = [ "/etc/lvm/metadata", "/mnt/disk2/lvm/metadata2" ]
#}

# Event daemon
#
dmeventd {
    # mirror_library is the library used when monitoring a mirror device.
    #
    # "libdevmapper-event-lvm2mirror.so" attempts to recover from
    # failures.  It removes failed devices from a volume group and
    # reconfigures a mirror as necessary. If no mirror library is
    # provided, mirrors are not monitored through dmeventd.

    mirror_library = "libdevmapper-event-lvm2mirror.so"

    # snapshot_library is the library used when monitoring a snapshot
device.
    #
    # "libdevmapper-event-lvm2snapshot.so" monitors the filling of
    # snapshots and emits a warning through syslog, when the use of
    # snapshot exceedes 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot are filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"
}
```

ANNEXE C. LES BALISES DES OBJETS LVM

Une balise LVM est une chaîne de caractères qui peut être utilisée pour grouper des objets LVM2 de même type. Les balises peuvent être attachées aux objets tels que les volumes physiques, les groupes de volumes, les volumes logiques. Les balises peuvent être attachées à des hôtes dans une configuration en clusters. Les instantanés ne peuvent pas être étiquetés.

Les balises peuvent être utilisées en lignes de commande à la place des arguments PV, VG ou LV. Les balises devraient être préfixées par @ pour éviter toute ambiguïté. Chaque balise est étendue à travers son remplacement par tout objet possédant la balise concernée, du type attendu par sa position sur la ligne de commande.

Les balises LVM sont des chaînes de caractères utilisant [A-Za-z0-9_+.-] et allant jusqu'à 128 caractères. Elles ne peuvent pas commencer par un trait d'union.

Seuls les objets d'un groupe de volumes peuvent être étiquetés. Les volumes physiques perdent leurs balises s'ils sont supprimés d'un groupe de volumes ; en effet, les balises sont stockées comme faisant partie des métadonnées du groupe de volumes et elles sont retirées lorsqu'un volume physique est supprimé. Les instantanés ne peuvent pas être étiquetés.

Les commandes suivantes listent tous les volumes logiques avec le balise **database**.

```
lvs @database
```

C.1. AJOUT ET SUPPRESSION DES BALISES D'OBJETS

Pour ajouter ou supprimer des balises sur les volumes physiques, utilisez l'option **--addtag** ou **--deltag** de la commande **pvchange**.

Pour ajouter ou supprimer des balises sur les groupes de volumes, utilisez l'option **--addtag** ou **--deltag** des commandes **vgchange** ou **vgcreate**.

Pour ajouter ou supprimer des balises sur les volumes logiques, utilisez l'option **--addtag** ou **--deltag** des commandes **lvchange** ou **lvcreate**.

C.2. LES BALISES HÔTES

In a cluster configuration, you can define host tags in the configuration files. If you set **hosttags = 1** in the **tags** section, a host tag is automatically defined using the machine's hostname. This allow you to use a common configuration file which can be replicated on all your machines so they hold identical copies of the file, but the behavior can differ between machines according to the hostname.

For information on the configuration files, see Annexe B, *Les fichiers de configuration LVM*

Pour chaque balise hôte, un fichier de configuration supplémentaire est lu s'il existe : lvm_*hosttag*.conf. Si ce fichier définit de nouvelles balises, plusieurs fichiers de configuration seront ajoutés à la liste des fichiers à lire.

Par exemple, l'entrée suivante du fichier de configuration définit toujours **tag1** et définit **tag2** si le nom d'hôte est **host1**.

```
tags { tag1 { }  tag2 { host_list = ["host1"] } }
```

C.3. CONTRÔLE D'ACTIVATION AVEC LES BALISES

Vous pouvez spécifier dans le fichier de configuration les volumes logiques devant être activés sur cet hôte. Par exemple, l'entrée suivante agit comme un filtre pour les requêtes d'activation (telles que **vgchange -ay**) et active seulement le volume **vg1/lvol0** et tout autre volume logique ou groupe de volumes avec la balise **database** dans les métadonnées.

```
activation { volume_list = ["vg1/lvol0", "@database" ] }
```

There is a special match "@*" that causes a match only if any metadata tag matches any host tag on that machine.

Considérez par exemple une situation où toutes les machines du cluster ont l'entrée suivante dans le fichier de configuration :

```
tags { hosttags = 1 }
```

Si vous voulez activer **vg1/lvol2**, seulement sur l'hôte **db2**, suivez les instructions suivantes :

1. Exécutez **lvchange --addtag @db2 vg1/lvol2** sur n'importe quel hôte du cluster.
2. Exécutez **lvchange -ay vg1/lvol2**.

Cette solution requiert que vous stockiez les noms d'hôte dans les métadonnées du groupe de volumes.

ANNEXE D. MÉTADONNÉES DES GROUPES DE VOLUMES LVM

Les détails de configuration d'un groupe de volumes sont appelés les métadonnées. Par défaut, une copie identique des métadonnées est maintenue dans toutes les zones de métadonnées de tous les volumes physiques au sein du groupe de volumes. Les métadonnées LVM sont petites et stockées en ASCII.

Si un groupe de volumes contient plusieurs volumes physiques, la possession de plusieurs copies redondantes de métadonnées est inefficace. Vous pouvez créer un volume physique sans copie de métadonnées en utilisant l'option **--metadatacopies 0** de la commande **pvcreate**. Une fois que vous aurez sélectionné le nombre de copies de métadonnées que le volume physique peut contenir, vous ne pourrez plus changer cette valeur. La sélection de 0 copie peut résulter en des mises à jour plus rapides sur les changements de configuration. Notez, cependant, que chaque groupe de volumes doit contenir en permanence au moins un volume physique avec une zone de métadonnées (à moins que vous utilisiez des paramètres de configuration avancés qui vous permettent de stocker les métadonnées du groupe de volumes dans un système de fichiers). Si par la suite vous avez l'intention de partager le groupe de volumes, chaque groupe de volumes doit avoir au moins une copie des métadonnées.

Les métadonnées core sont stockées en ASCII. Une zone de métadonnées correspond à un buffer circulaire. Les nouvelles métadonnées sont ajoutées aux anciennes et ensuite le pointeur au départ est mis à jour.

Vous pouvez spécifier la taille des métadonnées avec l'argument **--metadatasize** de la commande **pvcreate**. La taille par défaut est trop petite pour les groupes de volumes ayant beaucoup de volumes logiques ou physiques.

D.1. L'ÉTIQUETTE DU VOLUME PHYSIQUE

Par défaut, la commande **pvcreate** place l'étiquette du volume physique dans le deuxième secteur de 512 octets. Cette étiquette peut optionnellement être placée dans un des 4 premiers secteurs, étant donné que les outils LVM recherchant une étiquette de volume physique vérifient les 4 premiers secteurs. L'étiquette d'un volume physique commence avec la chaîne de caractères **LABELONE**.

L'étiquette du volume physique contient :

- L'UUID du volume physique
- La taille des périphériques blocs en octets
- Une liste terminant par NULL d'emplacements de zone de données
- Des listes terminant par NULL d'emplacements de zone de métadonnées

Les emplacements de métadonnées sont stockés en tant qu'offset et taille (en octets). Il y a un espace dans l'étiquette pour approximativement 15 emplacements mais les outils LVS en utilisent que 3 : une zone de données unique et jusqu'à deux zones de métadonnées.

D.2. CONTENU DES MÉTADONNÉES

Les métadonnées des groupes de volumes contiennent :

- Des informations à propos de la façon dont elles ont été créées et quand
- Des informations à propos du groupe de volumes :

Les informations du groupe de volumes contiennent :

- Le nom et l'identifiant unique
- Un numéro de version qui est incrémenté à chaque fois que les métadonnées sont mises à jour.
- Des propriétés : Lecture/Écriture ? Redimensionnable ?
- Toute limite d'administration du nombre de volumes physiques et logiques qu'il peut contenir
- La taille d'extension (en unités de secteurs qui sont définis comme 512 octets)
- Une liste non ordonnée de volumes physiques qui forment le groupe de volumes, chaque volume avec :
  - Son UUID, utilisé pour déterminer le périphérique bloc le contenant
  - Toute propriété, par exemple une propriété indiquant si le volume physique est allouable
  - L'offset au départ de la première extension au sein du volume physique (en secteurs)
  - Le nombre d'extensions
- Une liste non ordonnée de volumes logiques. Chaque volume consiste en :
  - Une liste ordonnée de segments de volumes logiques. Pour chaque segment, les métadonnées incluent un mappage appliqué à une liste ordonnée de segments de volumes physiques ou logiques.

D.3. ÉCHANTILLON DE MÉTADONNÉES

Ce qui suit illustre un exemple de métadonnées d'un groupe de volumes LVM pour un groupe de volumes appelé **myvg**.

```
# Generated by LVM2: Tue Jan 30 16:28:15 2007

contents = "Text Format Volume Group"
version = 1

description = "Created *before* executing 'lvextend -L+5G /dev/myvg/mylv
/dev/sdc'"

creation_host = "tng3-1"        # Linux tng3-1 2.6.18-8.el5 #1 SMP Fri Jan
26 14:15:21 EST 2007 i686
creation_time = 1170196095      # Tue Jan 30 16:28:15 2007

myvg {
        id = "0zd3UT-wbYT-lDHq-lMPs-EjoE-0o18-wL28X4"
        seqno = 3
        status = ["RESIZEABLE", "READ", "WRITE"]
        extent_size = 8192              # 4 Megabytes
        max_lv = 0
        max_pv = 0

        physical_volumes {
```

```
                pv0 {
                        id = "ZBW5qW-dXF2-0bGw-ZCad-2RlV-phwu-1c1RFt"
                        device = "/dev/sda"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv1 {
                        id = "ZHEZJW-MR64-D3QM-Rv7V-Hxsa-zU24-wztY19"
                        device = "/dev/sdb"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv2 {
                        id = "wCoG4p-55Ui-9tbp-VTEA-jO6s-RAVx-UREW0G"
                        device = "/dev/sdc"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv3 {
                        id = "hGlUwi-zsBg-39FF-do88-pHxY-8XA2-9WKIiA"
                        device = "/dev/sdd"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }
        }
        logical_volumes {

                mylv {
                        id = "GhUYSF-qVM3-rzQo-a6D2-o0aV-LQet-Ur9OF9"
                        status = ["READ", "WRITE", "VISIBLE"]
                        segment_count = 2

                        segment1 {
                                start_extent = 0
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv0", 0
```

```
                                ]
                        }
                        segment2 {
                                start_extent = 1280
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv1", 0
                                ]
                        }
                }
        }
}
```

ANNEXE E. HISTORIQUE DE RÉVISION

Version 3-6.400	**2013-10-31**	**Rüdiger Landmann**
Rebuild with publican 4.0.0		
Version 3-6	**2012-07-18**	**Anthony Towns**
Rebuild for Publican 3.0		
Version 1.0-0	**Thu Jan 29 2009**	

INDEX

A

B

C

M

O

P