Red Hat Enterprise Linux 6

Administration du gestionnaire de volumes logiques

Guide de l'administrateur LVM

Édition 1

Last Updated: 2017-10-13

Red Hat Enterprise Linux 6 Administration du gestionnaire de volumes logiques

Guide de l'administrateur LVM
Édition 1

Landmann
rlandmann@redhat.com

Notice légale

Copyright © 2011 Red Hat, Inc. and others.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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 clusterisé.

Table des matières

INTRODUCTION **5**
1. À PROPOS DE CE GUIDE 5
2. PUBLIC VISÉ 5
3. VERSIONS DU LOGICIEL 5
4. DOCUMENTATION ANNEXE 5
5. VOS COMMENTAIRES SONT BIENVENUS ! 6

CHAPITRE 1. LE GESTIONNAIRE DE VOLUMES LOGIQUES LVM **7**
1.1. NOUVELLES FONCTIONNALITÉS ET FONCTIONNALITÉS MODIFIÉES 7
1.1.1. Nouvelles fonctionnalités et fonctionnalités modifiées de Red Hat Enterprise Linux 6.0 7
1.1.2. Nouvelles fonctionnalités et fonctionnalités modifiées de Red Hat Enterprise Linux 6.1 8
1.2. VOLUMES LOGIQUES 9
1.3. APERÇU DE L'ARCHITECTURE LVM 10
1.4. LE GESTIONNAIRE DE VOLUMES LOGIQUES LVM EN CLUSTER (CLVM) 11
1.5. APERÇU DU DOCUMENT 12

CHAPITRE 2. COMPOSANTS LVM **14**
2.1. LES VOLUMES PHYSIQUES 14
2.1.1. Structure d'un volume physique LVM 14
2.1.2. Plusieurs partitions sur un disque 15
2.2. LES GROUPES DE VOLUMES 15
2.3. LES VOLUMES LOGIQUES LVM 16
2.3.1. Les volumes linéaires 16
2.3.2. Les volumes logiques en mode stripe 18
2.3.3. Les volumes logiques en miroir 19
2.3.4. Les volumes d'instantanés 20

CHAPITRE 3. APERÇU DE L'ADMINISTRATION LVM **22**
3.1. CRÉATION DE VOLUMES LVM DANS UN CLUSTER 22
3.2. APERÇU DE LA CRÉATION D'UN VOLUME LOGIQUE 23
3.3. AUGMENTATION DE LA TAILLE D'UN SYSTÈME DE FICHIERS SUR UN VOLUME LOGIQUE 23
3.4. SAUVEGARDE D'UN VOLUME LOGIQUE 24
3.5. JOURNALISATION 24

CHAPITRE 4. ADMINISTRATION LVM AVEC LES COMMANDES CLI **25**
4.1. UTILISATION DES COMMANDES CLI 25
4.2. ADMINISTRATION DE VOLUMES PHYSIQUES 26
4.2.1. Création de volumes physiques 26
4.2.1.1. Paramétrage du type de partition 26
4.2.1.2. Initialisation des volumes physiques 27
4.2.1.3. Recherche de périphériques blocs 27
4.2.2. Affichage des volumes physiques 28
4.2.3. Empêcher l'allocation sur un volume physique 29
4.2.4. Redimensionnement de volumes physiques 29
4.2.5. Suppression de volumes physiques 29
4.3. ADMINISTRATION D'UN GROUPE DE VOLUMES 29
4.3.1. Création de groupes de volumes 29
4.3.2. Création de groupes de volumes dans un cluster 30
4.3.3. Ajout de volumes physiques à un groupe de volumes 31
4.3.4. Affichage des groupes de volumes 31
4.3.5. Analyse des disques pour les groupes de volumes afin de construire le fichier de cache 32
4.3.6. Suppression de volumes physiques à partir d'un groupe de volumes 33

4.3.7. Changement des paramètres du groupe de volumes 33
4.3.8. Activation et désactivation des groupes de volumes 33
4.3.9. Suppression de groupes de volumes 34
4.3.10. Partager un groupe de volumes 34
4.3.11. Combinaison de groupes de volumes 34
4.3.12. Sauvegarde des métadonnées d'un groupe de volumes 35
4.3.13. Renommer un groupe de volumes 35
4.3.14. Déplacer un groupe de volumes sur un autre système 35
4.3.15. Recréation du répertoire d'un groupe de volumes 36
4.4. ADMINISTRATION DE VOLUMES LOGIQUES 36
4.4.1. Création de volumes logiques linéaires 36
4.4.2. Création de volumes en mode stripe 38
4.4.3. Création de volumes en miroir 38
4.4.3.1. Politique en cas d'échec des volumes logiques en miroir 41
4.4.3.2. Diviser une image redondante d'un volume logique en miroir 42
4.4.3.3. Réparer un périphérique logique en miroir 42
4.4.3.4. Changement de la configuration du volume en miroir 43
4.4.4. Création d'instantanés de volumes 43
4.4.5. Fusion d'instantanés de volumes 44
4.4.6. Numéros de périphérique persistants 45
4.4.7. Redimensionnement des volumes logiques 45
4.4.8. Changement des paramètres d'un groupe de volumes logiques 46
4.4.9. Renommer les volumes logiques 46
4.4.10. Suppression de volumes logiques 46
4.4.11. Affichage de volumes logiques 46
4.4.12. Augmentez la taille des volumes logiques 47
4.4.12.1. Augmenter la taille d'un volume en mode stripe 48
4.4.12.2. Étendre un volume logique avec la politique d'allocation cling 49
4.4.13. Réduire la taille des volumes logiques 51
4.5. CONTRÔLER L'ANALYSE DES PÉRIPHÉRIQUES LVM AVEC LES FILTRES 51
4.6. DÉPLACEMENT DES DONNÉES EN LIGNE 52
4.7. ACTIVATION DES VOLUMES LOGIQUES SUR LES NOEUDS INDIVIDUELS D'UN CLUSTER 53
4.8. RAPPORT PERSONNALISÉ POUR LVM 53
4.8.1. Contrôle du format 53
4.8.2. Sélection d'objets 55
La commande pvs 56
La commande vgs 58
La commande lvs 59
4.8.3. Trier des rapports LVM 62
4.8.4. Spécification des unités 63

CHAPITRE 5. EXEMPLES DE CONFIGURATION LVM 65
5.1. CRÉATION D'UN VOLUME LOGIQUE LVM SUR TROIS DISQUES 65
5.1.1. Création de volumes physiques 65
5.1.2. Création d'un groupe de volumes 65
5.1.3. Création du volume logique 65
5.1.4. Création du système de fichiers 66
5.2. CRÉATION D'UN VOLUME LOGIQUE EN MODE STRIPE 66
5.2.1. Création de volumes physiques 66
5.2.2. Création d'un groupe de volumes 67
5.2.3. Création du volume logique 67
5.2.4. Création du système de fichiers 67
5.3. PARTAGER UN GROUPE DE VOLUMES 68

5.3.1. Déterminer l'espace libre 68
5.3.2. Déplacer les données 68
5.3.3. Diviser le groupe de volumes 69
5.3.4. Création d'un nouveau volume logique 69
5.3.5. Créer un nouveau système de fichiers et monter le nouveau volume logique 69
5.3.6. Activation et montage du volume logique d'origine 70
5.4. SUPPRESSION D'UN DISQUE DU VOLUME LOGIQUE 70
5.4.1. Déplacer les extensions vers des volumes physiques existants 70
5.4.2. Déplacer les extensions vers un nouveau disque 71
5.4.2.1. Création du nouveau volume physique 71
5.4.2.2. Ajout du nouveau volume physique au groupe de volumes 71
5.4.2.3. Déplacer les données 71
5.4.2.4. Suppression de l'ancien volume physique du groupe de volumes. 72
5.5. CRÉATION D'UN VOLUME LOGIQUE LVM MIROIR DANS UN CLUSTER 72

CHAPITRE 6. RÉSOLUTION DE PROBLÈMES LVM 76
6.1. DIAGNOSTIQUES DE RÉSOLUTION DE PROBLÈMES 76
6.2. AFFICHAGE D'INFORMATIONS À PROPOS DES PÉRIPHÉRIQUES AYANT ÉCHOUÉ. 76
6.3. RÉCUPÉRATION SUITE À UN ÉCHEC MIROIR LVM 77
6.4. RECUPÉRATION DES MÉTADONNÉES DU VOLUME PHYSIQUE 80
6.5. REMPLACEMENT D'UN VOLUME PHYSIQUE MANQUANT 82
6.6. SUPPRIMER LES VOLUMES PHYSIQUES PERDUS D'UN GROUPE DE VOLUMES 82
6.7. EXTENSIONS LIBRES INSUFFISANTES POUR UN VOLUME LOGIQUE 83

CHAPITRE 7. ADMINISTRATION LVM AVEC L'INTERFACE UTILISATEUR GRAPHIQUE LVM 84

ANNEXE A. LE MAPPEUR DE PÉRIPHÉRIQUESUR DE PÉRIPHÉRIQUES 85
A.1. TABLES DES MAPPAGES DE PÉRIPHÉRIQUES 85
A.1.1. La cible de mappage linéaire 86
A.1.2. La cible de mappage striped 87
A.1.3. La cible de mappage miroir 88
A.1.4. Les cibles de mappage snapshot et snapshot-origin 90
A.1.5. La cible de mappage « error » 92
A.1.6. La cible de mappage « zero » 93
A.1.7. La cible de mappage « multipath » 93
A.1.8. La cible de mappage « crypt » 95
A.2. LA COMMANDE DMSETUP 96
A.2.1. La commande dmsetup info 96
A.2.2. La commande dmsetup ls 98
A.2.3. La commande dmsetup status 99
A.2.4. La commande dmsetup deps 99
A.3. PRISE EN CHARGE DU MAPPEUR DE PÉRIPHÉRIQUES POUR LE GESTIONNAIRE DE PÉRIPHÉRIQUES UDEV 100
A.3.1. Intégration udev avec le mappeur de périphériques 100
A.3.2. Commandes et interfaces qui supportent udev 102

ANNEXE B. LES FICHIERS DE CONFIGURATION LVM 105
B.1. LES FICHIERS DE CONFIGURATION LVM 105
B.2. ÉCHANTILLON DU FICHIER LVM.CONF 105

ANNEXE C. LES BALISES DES OBJETS LVM 118
C.1. AJOUT ET SUPPRESSION DES BALISES D'OBJETS 118
C.2. LES BALISES HÔTES 118
C.3. CONTRÔLE D'ACTIVATION AVEC LES BALISES 119

ANNEXE D. MÉTADONNÉES DES GROUPES DE VOLUMES LVM **120**
D.1. L'ÉTIQUETTE DU VOLUME PHYSIQUE 120
D.2. CONTENU DES MÉTADONNÉES 120
D.3. ÉCHANTILLON DE MÉTADONNÉES 121

ANNEXE E. HISTORIQUE DES RÉVISIONS **124**

INDEX **125**

INTRODUCTION

1. À PROPOS DE CE GUIDE

Ce guide décrit le gestionnaire de volumes logiques (LVM, de l'anglais « Logical Volume Manager ») et comprend des informations sur l'exécution de LVM dans un environnement clusterisé.

2. PUBLIC VISÉ

Cet ouvrage est destiné aux administrateurs système gérant des systèmes exécutant des systèmes d'exploitation Linux. Vous devez être familier avec Red Hat Enterprise Linux 6 et l'administration du systèmes de fichiers GFS2.

3. VERSIONS DU LOGICIEL

Tableau 1. Versions du logiciel

Logiciel	Description
RHEL 6	Se réfère à RHEL 6 et les versions plus récentes
GFS2	Se réfère à GFS2 pour RHEL 6 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* — Fournit des informations à propos de l'installation de Red Hat Enterprise Linux 6.
- *Guide déploiement* — Fournit des informations à propos du déploiement, de la configuration et de l'administration de Red Hat Enterprise Linux 6.
- *Guide de l'administration du stockage* — Fournit des informations sur comment gérer de manière efficace les périphériques de stockage et les systèmes de fichiers sur Red Hat Enterprise Linux 6.

Pour davantage d'informations à propos les composants additionnels Haute disponibilité et Stockage résilient Red Hat Enterprise Linux 6, reportez-vous aux ressources suivantes :

- *Aperçu du composant additionnel Haute Disponibilité* — Fournit un aperçu de haut niveau du composant additionnel Haute Disponibilité.
- *Administration de Clusters* — Fournit des informations sur l'installation, la configuration et la gestion du composant additionnel Haute Disponibilité de Red Hat.

- *Système de fichiers GFS2 : 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), qui est inclut dans le composant additionnel Stockage résilient.
- *DM Multipath* — Fournit des informations à propos de l'utilisation de la fonctionnalité « Device-Mapper Multipath » de Red Hat Enterprise Linux 6.
- *Administration de l'Équilibreur de charges* — Fournit des informations sur la configuration de systèmes et services de haute performance avec le composant additionnel Équilibreur de charges, un ensemble de composants logiciels fournissant des serveurs virtuels Linux (LVS, de l'anglais Linux Virtual Server) avec un équilibreur de charges IP sur un ensemble de serveurs réels.
- *Notes de mise à jour* — Fournit des informations sur la version actuelle des produits Red Hat.

La documentation du composant additionnel Haute disponibilité 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. VOS COMMENTAIRES SONT BIENVENUS !

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/ dans la rubrique **Red Hat Enterprise Linux 6** et le composant **doc-Logical_Volume_Manager**. Lors de la soumission d'un rapport de bogue, n'oubliez pas de mentionner l'identifiant du manuel : **`Logical_Volume_Manager_Administration(EN)-6 (2011-05-19-15:20)`**.

Si vous avez des suggestions pour améliorer la documentation, essayez d'être aussi précis que possible lors des descriptions. Si vous avez trouvé une erreur, veuillez inclure le numéro de la section ainsi que des portions du texte qui l'entoure afin que nous puissions la retrouver plus facilement.

CHAPITRE 1. LE GESTIONNAIRE DE VOLUMES LOGIQUES LVM

Ce chapitre propose un sommaire des nouvelles fonctionnalités du gestionnaire de volumes logiques LVM dans la version initiale de Red Hat Enterprise Linux 6 et les versions suivantes. Ce chapitre fournit ainsi un aperçu de haut niveau des composants du gestionnaire de volumes logiques (LVM).

1.1. NOUVELLES FONCTIONNALITÉS ET FONCTIONNALITÉS MODIFIÉES

Cette section répertorie les nouvelles fonctionnalités et les fonctionnalités modifiées du gestionnaire de volumes logiques LVM incluses dans la version initiale de Red Hat Enterprise Linux 6 et les versions suivantes.

1.1.1. Nouvelles fonctionnalités et fonctionnalités modifiées de Red Hat Enterprise Linux 6.0

Red Hat Enterprise Linux 6.0 inclut la documentation et les mises à jour et modifications des fonctionnalités suivantes :

- Vous pouvez maintenant définir de quelle manière un volume logique miroir se comporte au cas où il se produirait une panne de périphérique avec les paramètres **mirror_image_fault_policy** et **mirror_log_fault_policy** dans la section **activation** du fichier **lvm.conf**. Lorsque ce paramètre est réglé sur **remove**, le système tente de supprimer le périphérique défectueux et de s'exécuter sans lui. Lorsque ce paramètre est réglé sur **allocate**, le système tente de supprimer le périphérique défectueux et essaied'allouer de l'espace sur un nouveau périphérique afin de remplacer celui qui est défectueux ; cette politique agit de la même manière que la politique **remove** s'il n'y a pas de périphérique et d'espace approprié qui puisse être alloué en remplacement. Pour obtenir plus d'informations sur les politiques en cas d'échec de miroirs LVM, voir la Section 4.4.3.1, « Politique en cas d'échec des volumes logiques en miroir ».

- Dans Red Hat Enterprise Linux 6, la pile d'E/S Linux a été améliorée afin de traiter les informations sur les limites d'E/S offertes par les fournisseurs. Ceci permet d'utiliser des outils de gestion du stockage, y compris LVM, afin d'optimiser le placement et l'accès aux données. Même s'il n'est pas recommandé de le faire, cette prise en charge peut être désactivée en modifiant les valeurs par défaut de **data_alignment_detection** et de **data_alignment_offset_detection** dans le fichier **lvm.conf**.

  Pour obtenir des informations sur l'alignement des données dans LVM et sur la modification des valeurs par défaut de **data_alignment_detection** et **data_alignment_offset_detection**, voir la documentation en ligne sur le fichier **/etc/lvm/lvm.conf**, qui se trouve dans l'Annexe B, *Les fichiers de configuration LVM*. Pour des informations générales sur la pile d'E/S et sur les limites des E/S dans Red Hat Enterprise Linux 6, voir le *Guide d'administration du stockage*.

- Dans Red Hat Enterprise Linux 6, le mappeur de périphériques (de l'anglais, « Device Mapper » fournit un support direct pour l'intégration **udev**. Ceci synchronise le mappeur de périphériques avec tous les processus **udev** liés aux périphériques du mappeur de périphériques.

- Dans Red Hat Enterprise Linux, vous pouvez utiliser la commande **lvconvert --repair** pour réparer un miroir après un échec du disque. Ceci permet de ramener le miroir dans un état consistant. Pour obtenir des informations sur la commande **lvconvert --repair**, voir la

Section 4.4.3.3, « Réparer un périphérique logique en miroir ».

- À partir de Red Hat Enterprise Linux 6, vous pouvez utiliser l'option **--merge** de la commande **lvconvert** pour fusionner un instantané dans son volume d'origine. Pour obtenir plus d'informations sur la fusion d'instantanés, voir la Section 4.4.5, « Fusion d'instantanés de volumes ».

- À partir de Red Hat Enterprise Linux 6, vous pouvez utiliser l'argument **--splitmirrors** de la commande **lvconvert** pour séparer une image redondante d'un volume logique miroir afin de former un nouveau volume logique. Pour obtenir plus d'informations sur l'utilisation de cette option, voir la Section 4.4.3.2, « Diviser une image redondante d'un volume logique en miroir ».

- Vous pouvez maintenant créer un journal miroir pour un périphérique logique miroir qui est lui-même mis en miroir en utilisant l'argument **--mirrorlog mirrored** de la commande **lvcreate** lors de la création d'un périphérique logique miroir. Pour obtenir plus d'informations sur l'utilisation de cette option, voir la Section 4.4.3, « Création de volumes en miroir ».

1.1.2. Nouvelles fonctionnalités et fonctionnalités modifiées de Red Hat Enterprise Linux 6.1

Red Hat Enterprise Linux 6.1 inclut la documentation et les mises à jour et modifications des fonctionnalités suivantes :

- La version 6.1 de Red Hat Enterprise Linux prend en charge la création d'instantanés de volumes logiques de volumes logiques miroirs, Vous pouvez créer l'instantané d'un volume miroir aussi simplement que vour pourriez créer l'instantané d'un volume logique linéaire ou en mode stripe. Pour obtenir plus d'informations sur la création d'instantanés de volumes, voir Section 4.4.4, « Création d'instantanés de volumes ».

- Lors de l'extension d'un volume LVM, vous pouvez maintenant utiliser l'option **--alloc cling** de la commande **lvextend** pour spécifier la politique d'allocation **cling**. Cette politique choisira l'espace sur les mêmes volumes physiques que le dernier segment du volume logique existant. S'il n'y a pas suffisament d'espace sur les volumes physiques et qu'une liste des balises est définie dans le fichier **lvm.conf**, LVM vérifiera si des balises sont attachées aux volumes physiques et cherchera à faire correspondre ces balises de volumes physiques avec les extensions existantes et les nouvelles extensions.

  Pour obtenir des informations sur l'extension de volumes miroirs LVM avec l'option **--alloc cling** de la commande **lvextend**, voir Section 4.4.12.2, « Étendre un volume logique avec la politique d'allocation **cling** ».

- Vous pouvez maintenant spécifier de multiples arguments **--addtag** et **--deltag** avec une commande **pvchange**, **vgchange**, ou **lvchange** unique. Pour obtenir des informations sur l'ajout et la suppression de balises d'objets, voir Section C.1, « Ajout et suppression des balises d'objets ».

- La liste de caractères autorisés dans les balises d'objets LVM a été étendue, et les balises peuvent contenir les caractères "/", "=", "!", ":", "#", et "&". Pour obtenir plus d'informations sur les balises d'objets LVM, voir Annexe C, *Les balises des objets LVM*.

- Vous pouvez maintenant combiner RAID0 (striping) et RAID1 (mirroring) dans un volume logique unique. La création d'un volume logique tout en spécifiant le nombre de miroirs (**--mirrors X**) et le nombre de stripes (**--stripes Y**) résulte en un périphérique miroir dont les périphériques constituants sont en mode stripe. Pour obtenir des informations sur la création de volumes logiques miroirs, voir Section 4.4.3, « Création de volumes en miroir ».

- À partir de la version 6.1 de Red Hat Enterprise Linux, si vous devez créer une copie de sauvegarde des données consistante sur un volume logique clusterisé, vous pouvez activer le volume de manière exclusive puis créer l'instantané. Pour obtenir des informations sur l'activation de volumes logiques sur un seul noeud, voir Section 4.7, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

1.2. VOLUMES LOGIQUES

La gestion de volumes crée une couche abstraite sur un stockage physique, vous permettant ainsi de créer des volumes de stockage logiques. Ceci offre une flexibilité bien plus importante que lors de l'utilisation de stockage physique directement. 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 résistants et 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 « 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.

1.3. APERÇU DE L'ARCHITECTURE LVM

Dans la version Red Hat Enterprise Linux 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, « Composants d'un volume logique LVM » illustre les composants d'un volume logique LVM de base :

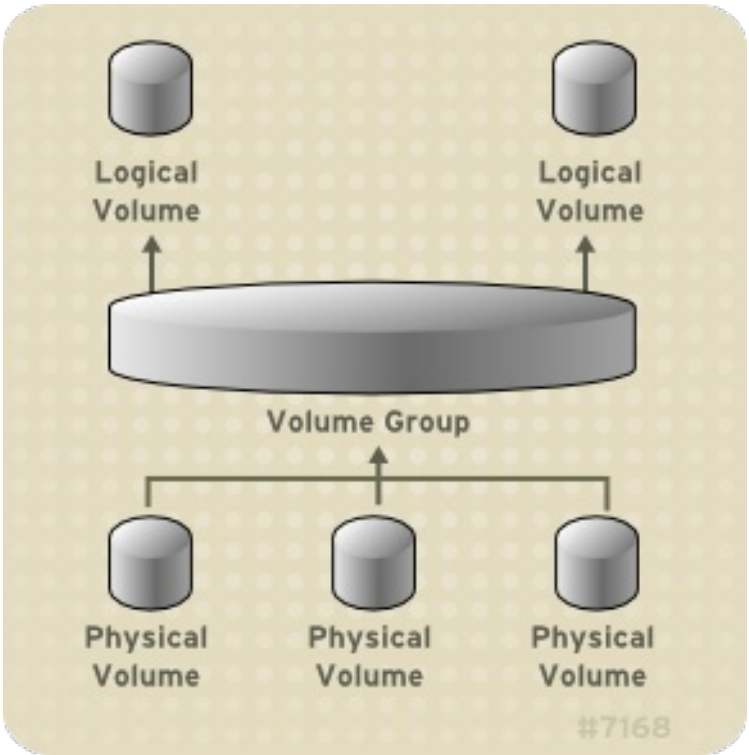


Figure 1.1. Composants d'un volume logique LVM

Pour obtenir des informations détaillées à propos des composants d'un volume logique LVM, reportez-vous au Chapitre 2, *Composants LVM*.

1.4. LE GESTIONNAIRE DE VOLUMES LOGIQUES LVM EN CLUSTER (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 le stockage partagé (par exemple sur un SAN) en utilisant LVM. CLVM fait partie du logiciel additionnel Stockage résilient.

L'utilisation de CLVM dépend des spécifications requises par votre système :

- Si un seul noeud de votre système requiert accès au stockage que vous êtes en train de configurer en tant que volumes logiques, vous pouvez alors utiliser LVM sans les extensions CLVM et les volumes logiques créés avec ce noeud sont tous locaux à celui-ci.
- Si vous utilisez un système en cluster pour basculements alors qu'un seul noeud accédant au stockage est actif à tout moment, vous devriez utiliser les agents de gestion de volume logique de haute disponibilité de l'anglais, « High Availability Logical Volume Management », ou HA-LVM). Pour plus d'informations sur HA-LVM, voir *Configurer et gérer un cluster Red Hat*.
- Si plus d'un noeud de votre cluster requiert accès à votre stockage, qui sera ensuite partagé à travers les différents noeuds actifs, alors vous devrez utiliser CLVM. CLVM permet à un utilisateur de configurer des volumes logiques sur un stockage partagé en verrouillant l'accès au stockage physique pendant qu'un volume est en cours de configuration. CLVM utilise les services de verrouillage clusterisés pour gérer le stockage partagé.

Pour pouvoir utiliser CLVM, les logiciels additionnels Haute disponibilité et Stockage résilient, y compris le démon **clvmd**, doivent tous 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 plus d'informations sur l'installation et l'administration du logiciel additionnel Haute disponiblité, voir *Configurer et gérer un cluster Red Hat*.

Pour vous assurer que **clvmd** est lancé au démarrage, vous pouvez exécuter une commande **chkconfig ... on** sur le service **clvmd** comme suit :

```
# chkconfig clvmd on
```

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

```
# service clvmd start
```

La création de volumes logiques LVM dans un environnement en cluster est identique à la création de volumes logiques sur un seul noeud. Il n'y a pas de différence avec les commandes LVM elles-mêmes, ou dans l'interface utilisateur graphique LVM, comme décrit dans la Chapitre 4, *Administration LVM avec les commandes CLI* et la Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM*. Pour pouvoir activer les volumes LVM que vous êtes en train de créer dans un cluster, l'infrastructure de celui-ci doit être en cours d'exécution et le cluster doit comporter le quorum.

Par défaut, les volumes logiques créés avec CLVM sur un stockage partagé sont visibles à tous les systèmes ayant accès au stockage partagé. Il est possible de créer des groupes de volumes dans lesquels tous les périphériques partagés ne sont visibles qu'à un seul noeud du cluster. Il est aussi

possible d'un groupe de volumes de groupe de volumes local à groupe de volumes clusterisé. Pour plus d'informations, voir la Section 4.3.2, « Création de groupes de volumes dans un cluster » et la Section 4.3.7, « Changement des paramètres du groupe de volumes ».

AVERTISSEMENT

Lorsque vous créez des groupes de volumes avec CLVM sur un stockage partagé, vous devez vous assurer que tous les noeuds dans le cluster ont accès aux volumes physiques constituant le groupe de volumes. Les configurations de cluster asymétriques, avec lesquelles certains noeuds ont accès au stockage et d'autres non, ne sont pas prises en charge.

Figure 1.2, « Aperçu de CLVM » illustre un aperçu de CLVM dans un cluster.

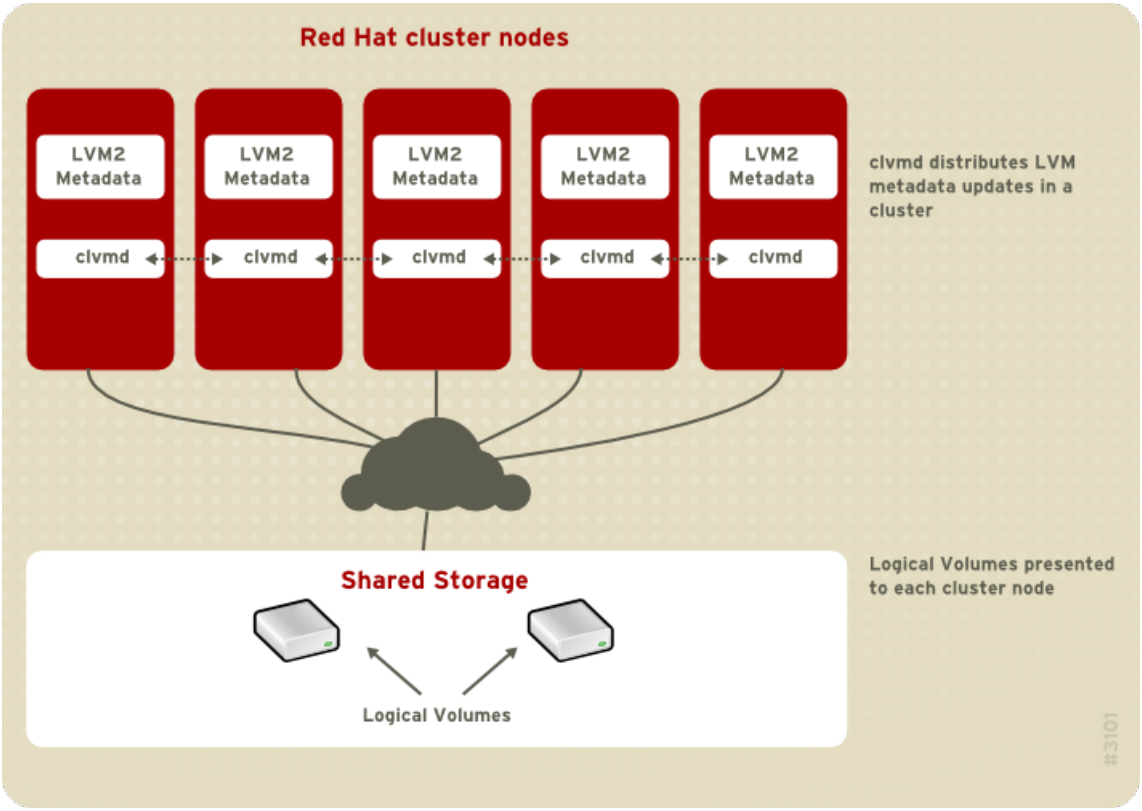


Figure 1.2. Aperçu de CLVM

NOTE

CLVM requiert des modifications dans le fichier **`lvm.conf`** pour un verrouillage au niveau du cluster. Des informations sur la configuration du fichier **`lvm.conf`** pour qu'il prenne en charge le verrouillage clusterisé sont disponibles dans le fichier **`lvm.conf`**lui-même. Pour davantage d'informations à propos de la configuration du fichier **`lvm.conf`**, reportez-vous à l'Annexe B, *Les fichiers de configuration LVM*.

1.5. APERÇU DU DOCUMENT

Le reste de ce document inclut les chapitres suivants :

- Chapitre 2, *Composants LVM* décrit les composants permettant de personnaliser un volume logique LVM.
- Chapitre 3, *Aperçu de l'administration LVM* fournit un aperçu des étapes de base permettant de configurer les volumes logiques LVM, que ce soit en utilisant les commandes CLI (de l'anglais, « Command Line Interface ») LVM ou avec l'interface utilisateur graphique LVM (GUI).
- Chapitre 4, *Administration LVM avec les commandes CLI* résume les tâches d'administration individuelles que vous pouvez effectuer avec les commandes CLI LVM afin de créer et maintenir les volumes logiques.
- Chapitre 5, *Exemples de configuration LVM* fournit plusieurs exemples de configuration LVM.
- Chapitre 6, *Résolution de problèmes LVM* fournit des instructions afin de résoudre différents problèmes LVM.
- Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM* résume les opérations de l'interface utilisateur graphique LVM.
- Annexe A, *Le mappeur de périphériquesur de périphériques* décrit le mappeur de périphériques utilisé par LVM pour mettre en correspondance les volumes logiques et physiques.
- Annexe B, *Les fichiers de configuration LVM* décrit les fichiers de configuration LVM.
- Annexe C, *Les balises des objets LVM* décrit les balises d'objets LVM et les balises hôtes.
- Annexe D, *Métadonnées des groupes de volumes LVM* décrit les métadonnées des groupes de volumes LVM et inclut la copie d'un échantillon de métadonnées d'un groupe de volumes LVM.

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.

Pour obtenir des informations détaillées à propos des métadonnées LVM et du changement de leurs paramètres, reportez-vous à l'Annexe D, *Métadonnées des groupes de volumes LVM*.

2.1.1. Structure d'un volume physique LVM

La Figure 2.1, « Structure d'un volume physique » illustre la structure d'un volume physique LVM. L'étiquette LVM se trouve sur le deuxième secteur, elle est suivie par la zone de métadonnées et par l'espace utilisable sur le périphérique.

NOTE

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

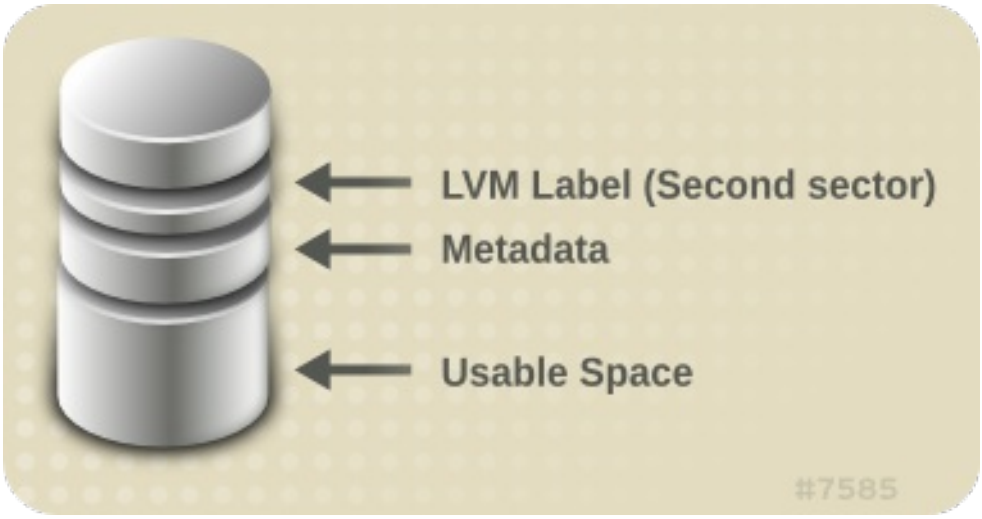


Figure 2.1. Structure d'un volume physique

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.

Lors de la création d'un volume linéaire, un ensemble d'extensions physiques est assigné à une zone du volume logique. Par exemple, dans la Figure 2.2, « Mappage des extensions », les extensions logiques 1 à 99 pourraient correspondre à un premier volume physique et les extensions logiques 100 à 198 pourraient correspondre à un deuxième volume physique. Du point de vue de l'application, il y a un seul périphérique avec une taille de 198 extensions.

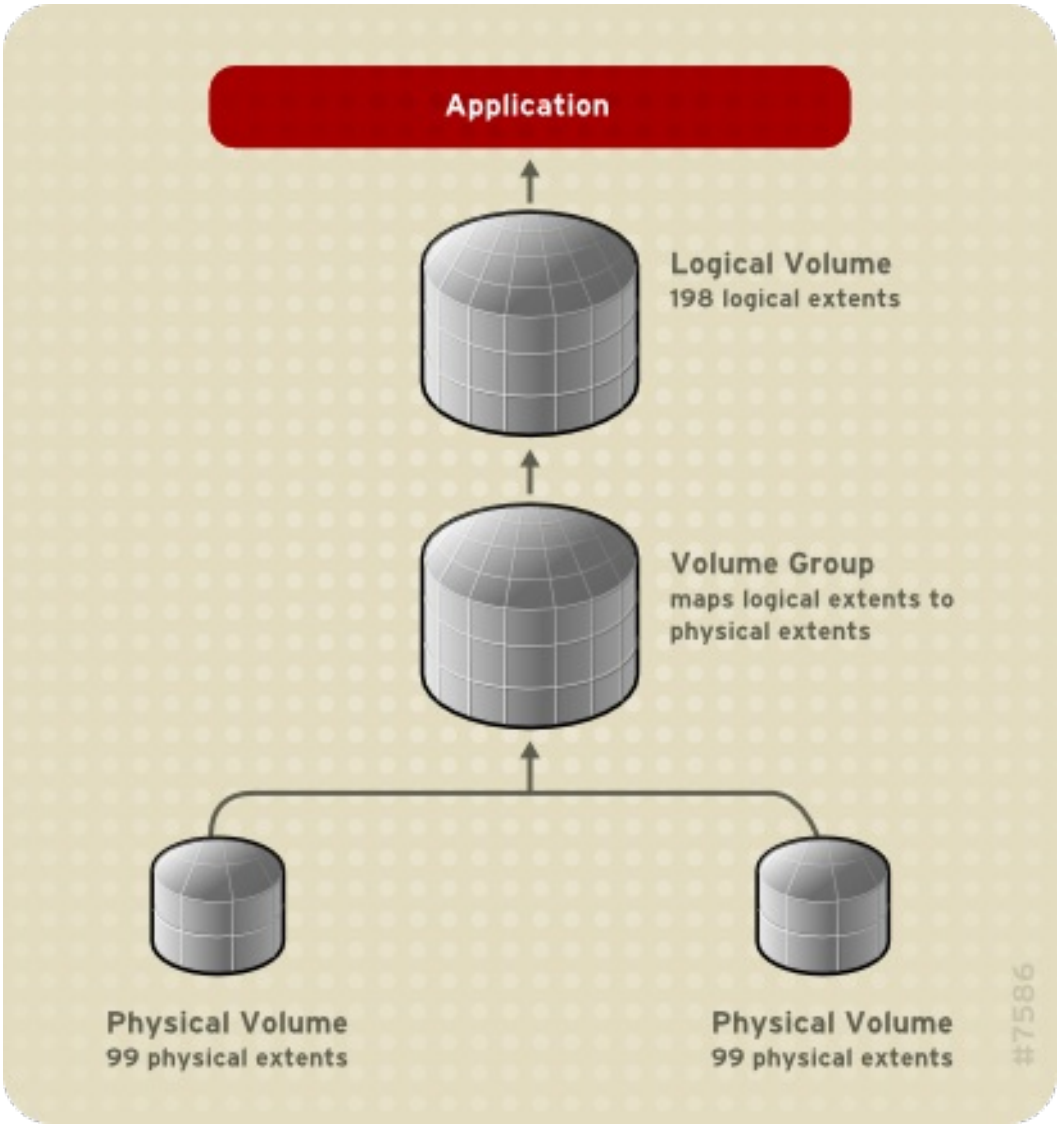


Figure 2.2. Mappage des extensions

Les volumes physiques qui composent le volume logique ne sont pas forcément de la même taille. La Figure 2.3, « Volume linéaire avec des volumes physiques différents » illustre le groupe de volumes **VG1** avec une taille d'extension physique de 4Mo. Ce groupe de volumes inclut 2 volumes physiques appelés **PV1** et **PV2**. Les volumes physiques sont divisés en unités de 4Mo, étant donné qu'il s'agit de la taille d'extension. Dans cet exemple, **PV1** a une taille de 100 extensions (400Mo) et **PV2** a une taille de 200 extensions (800Mo). Vous pouvez créer un volume linéaire avec une taille comprise entre 1 et 300 extensions (de 4Mo à 1200Mo). Dans cet exemple, le volume linéaire appelé **LV1** a une taille de 300 extensions.

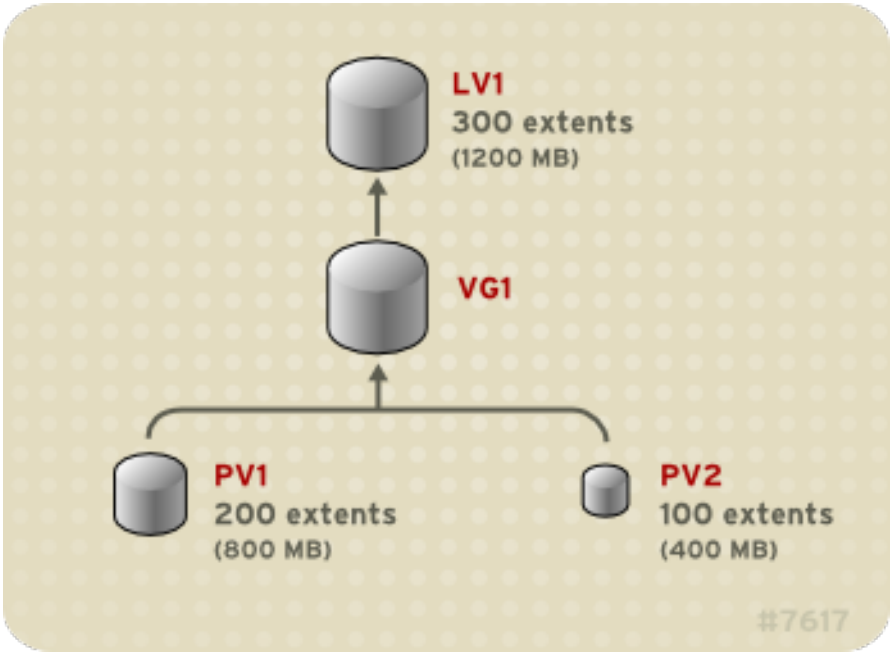


Figure 2.3. Volume linéaire avec des volumes physiques différents

Vous pouvez configurer plus d'un volume logique linéaire, de la taille désirée, à partir du pool d'extensions physiques. Figure 2.4, « Plusieurs volumes logiques » illustre le même groupe de volumes que dans la Figure 2.3, « Volume linéaire avec des volumes physiques différents » mais dans ce cas, deux volumes logiques sont issus du groupe de volumes : **LV1**, avec un taille de 250 extensions (1000Mo) et **LV2**, avec une taille de 50 extensions (200Mo).

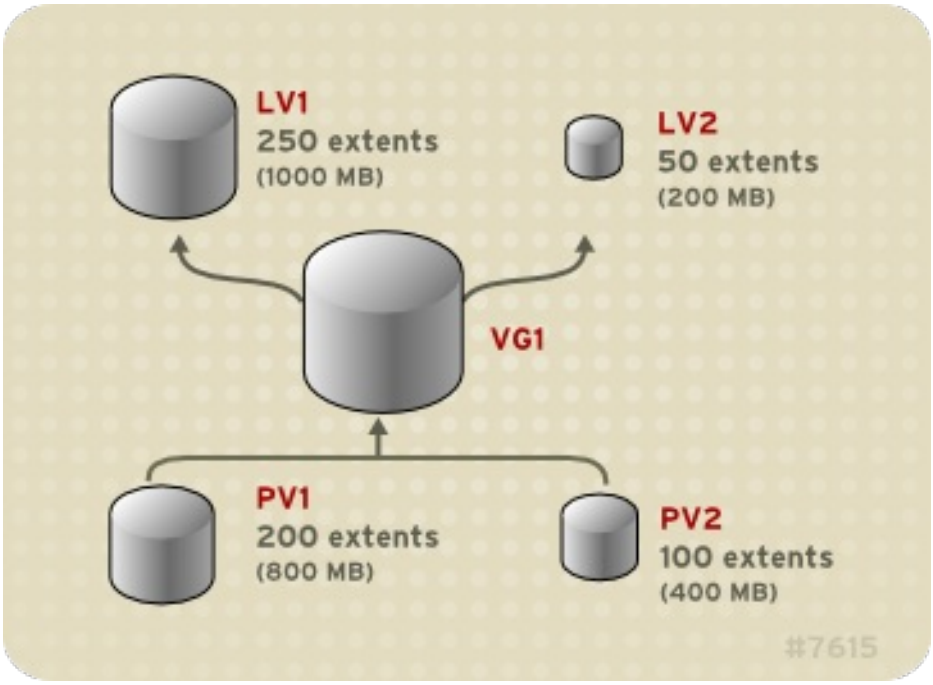


Figure 2.4. Plusieurs volumes logiques

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-robin (tourniquet). 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.

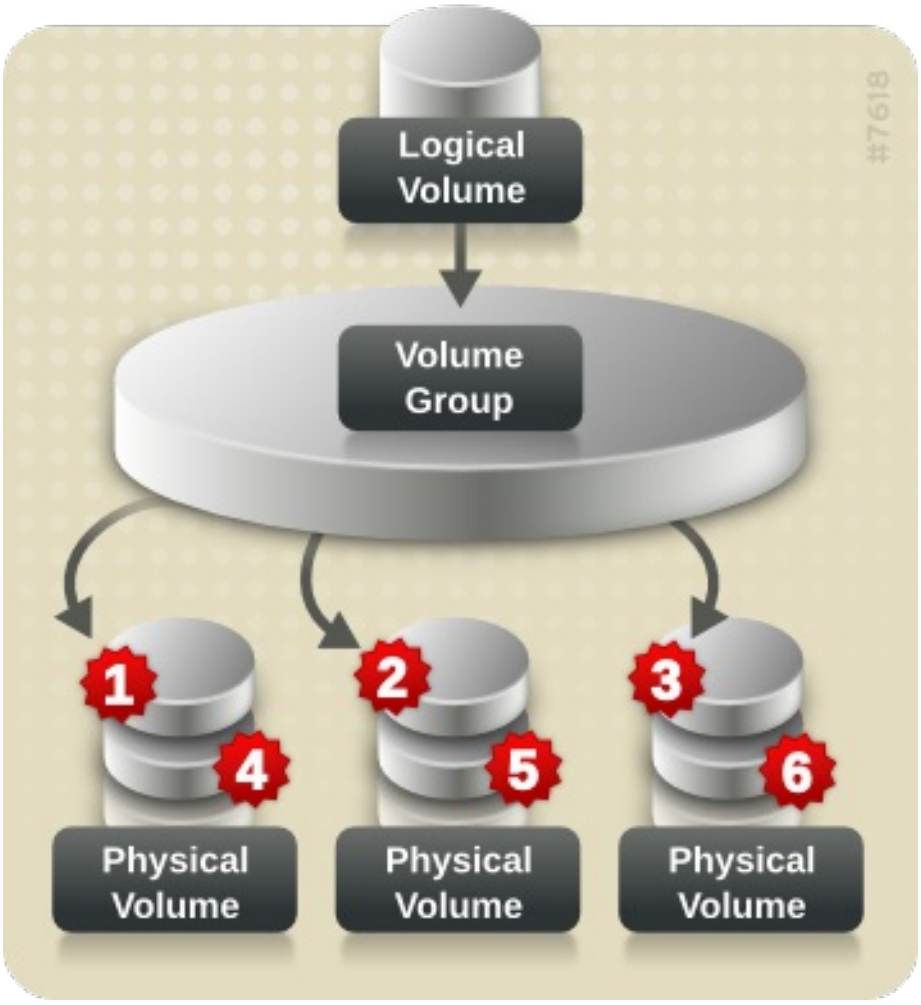


Figure 2.5. Striping des données à travers 3 PV

Les volumes logiques en mode stripe peuvent être étendus en concaténant un autre groupe de périphériques à la fin du premier groupe. Cependant, pour étendre un volume logique en mode stripe, il doit y avoir suffisamment d'espace libre sur les volumes physiques sous-jacents composant le groupe de volumes afin que le stripe soit supporté. Si vous avez par exemple un stripe à double 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. Pour davantage d'informations à propos de l'extension d'un volume en mode stripe, reportez-vous à la Section 4.4.12.1, « 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.

La Figure 2.6, « Volume logique en miroir » illustre un volume logique en miroir avec un seul miroir. Dans cette configuration, le fichier journal est maintenu sur le disque.

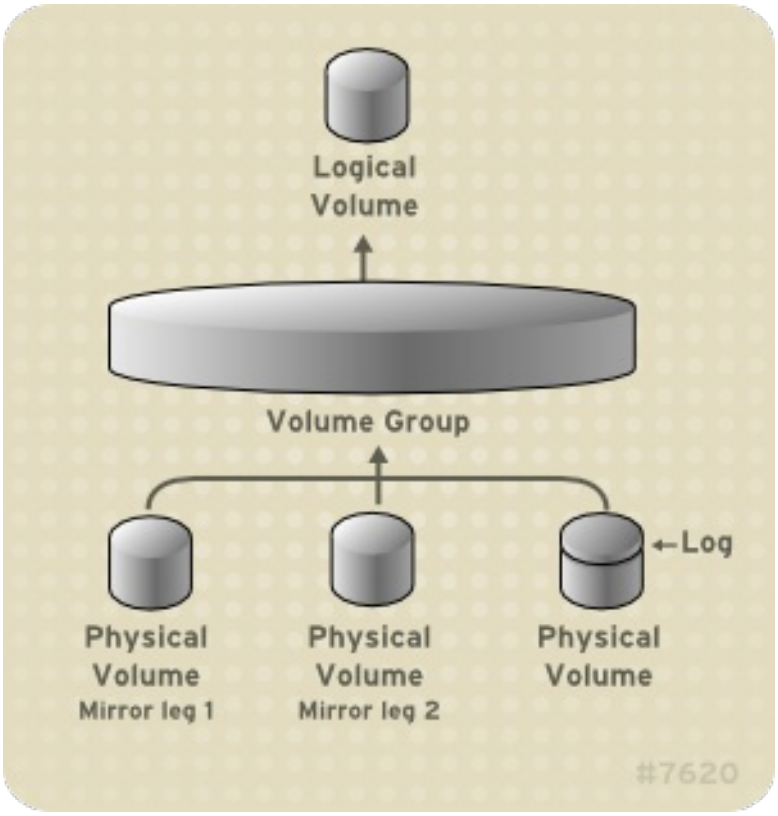


Figure 2.6. Volume logique en miroir

Pour davantage d'informations à propos de la création et de la modification de miroirs, reportez-vous à la Section 4.4.3, « 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.

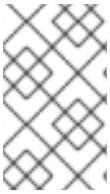
NOTE

Les instantanés LVM ne sont pas supportés à travers les noeuds d'un cluster. Vous ne pouvez pas créer de volume instantané dans un groupe de volumes en clusters.

NOTE

Les instantanés LVM ne sont pas supportés par les volumes logiques LVM en miroir.

Étant donné que seules les zones de données modifiées sont copiées suite à la création d'un instantané, la fonctionnalité relative aux instantanés requiert une quantité de stockage minimale. Par exemple, pour un périphérique d'origine rarement modifié, 3 à 5 % de la capacité d'origine est suffisante pour maintenir l'instantané.



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.

La taille de l'instantané définit la quantité d'espace assignée pour le stockage des modifications au volume d'origine. Par exemple, si vous avec effectué un instantané, puis complètement remplacé l'origine, l'instantané devra être au moins aussi grand que le volume d'oringine pour conserver les modifications. Un instantané doit être dimensionné en fonction du niveau des modifications auquel vous vous attendez. Par exemple, un instantané éphémère d'un volume en lecture principale , tel que **/usr**, nécessitera moins d'espace qu'un instantané à long terme d'un volume comportant un plus grand nombre d'écritures, comme **/home**.

Si un instantané dépasse sa capacité de stockage, ce dernier devient invalide car il ne peut plus suivre les changements sur le volume 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'instantanés afin d'éviter que ce dernier soit ignoré. À l'inverse, si vous trouvez que le volume d'instantanés 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 LVM pour une utilisation avec la virtualisation Red Hat. Les instantanés LVM peuvent être utilisés pour créer des instantanés d'images d'invités virtuels. Ces instantanés peuvent fournir une manière pratique de modifier des invités existants ou de créer de nouveaux invités avec un stockage supplémentaire minimal. Pour obtenir plus d'informations sur la création d'instantanés LVM d'invités virtualisés, voir le *Guide de virtualisation Red Hat Enterprise Linux*.

Pour davantage d'informations à propos de la création de volumes instantanés, reportez-vous à la Section 4.4.4, « Création d'instantanés de volumes ».

À partir de Red Hat Enterprise Linux 6, il est possible d'utiliser l'option **--merge** de la commande **lvconvert** afin de fusionner un instantané dans son volume d'origine. L'une des raisons d'utilisation de cette fonctionnalité est le fait de pouvoir effectuer une restauration du système si vous perdez des données ou fichiers, ou si vous devez restaurer votre système sur un état ultérieur. Après avoir fusionné le volume instantané, le volume logique en résultant contiendra le nom, le numéro mineur, et l'UUID du volume d'origine, et l'instantané de la fusion est supprimé.

CHAPITRE 3. APERÇU DE L'ADMINISTRATION LVM

Ce chapitre fournit un aperçu des procédures administratives utilisées pour configurer les volumes logiques LVM. Ce chapitre a pour objectif d'offrir une compréhension générale des étapes impliquées. Pour obtenir des exemples, étape par étape, des procédures communes de configuration de LVM, reportez-vous au Chapitre 5, *Exemples de configuration LVM*.

Pour obtenir une description des commandes CLI utilisées pour l'administration de LVM, reportez-vous au Chapitre 4, *Administration LVM avec les commandes CLI*. Alternativement, vous pouvez utiliser l'interface utilisateur graphique LVM décrite dans le Chapitre 7, *Administration LVM avec l'interface utilisateur graphique LVM*.

3.1. CRÉATION DE VOLUMES LVM DANS UN CLUSTER

Pour créer des volumes logiques dans un environnement clusterisé, vous pouvez utiliser CLVM (Clustered Logical Volume Manager), CLVM est un ensemble d'extensions de cluster pour LVM. Ces extensions permettent à un cluster d'ordinateurs de gérer un stockage partagé (par exemple sur un réseau SAN) à l'aide de LVM. Pour utiliser CLVM, les logiciels additionnels Haute disponibilité et Stockage résilient, y comprisle démon **clvmd**, doivent être lancés au démarrage, comme décrit dans la Section 1.4, « Le gestionnaire de volumes logiques LVM en cluster (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 requiert des modifications dans le fichier **lvm.conf** pour un verrouillage au niveau du cluster. Des informations sur la configuration du fichier **lvm.conf** pour qu'il prenne en charge le verrouillage clusterisé sont disponibles dans le fichier **lvm.conf**lui-même. Pour davantage d'informations à propos de la configuration du fichier **lvm.conf**, reportez-vous à l'Annexe B, *Les fichiers de configuration LVM*.

Par défaut, les volumes logiques créés avec CLVM sur un stockage partagé sont visibles par tous les systèmes ayant accès à ce stockage partagé. Il est possible de créer des groupes de volumes dans lesquels tous les périphériques de stockage ne sont visibles qu'à un seul noeud du cluster. Il est aussi possible de changer le status d'un groupe de volumes d'un groupe de volumes local en groupe de volumes clusterisé. Pour obtenir plus d'informations, voir la Section 4.3.2, « Création de groupes de volumes dans un cluster » et la Section 4.3.7, « Changement des paramètres du groupe de volumes ».

AVERTISSEMENT

Lorsque vous créez des groupes de volumes avec CLVM sur un stockage partagé, vous devez vous assurer que tous les noeuds dans le cluster ont accès aux volumes physiques constituant le groupe de volumes. Les configurations de cluster asymétriques, avec lesquelles certains noeuds ont accès au stockage et d'autres non, ne sont pas prises en charge.

Pour davantage d'informations sur la façon d'installer le composant additionnel Haute Disponibilité et de configurer une infrastructure de cluster, reportez-vous à l'*Administration de clusters*.

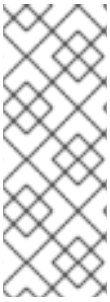
Pour voir un exemple de création d'un volume logique miroir dans un cluster, reportez-vous à la Section 5.5, « Création d'un volume logique LVM miroir dans un cluster ».

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 GFS2.



NOTE

Même si un système de fichiers GFS2 peut être implémenté sur un système autonome ou en faisant partie d'une configuration en clusters, dans la version de Red Hat Enterprise 6, Red Hat ne prend pas en charge l'utilisation de GFS2 en tant que système de fichiers à noeud unique. Red Hat continuera à prendre en charge les systèmes de fichiers GFS2 pour monter des extraits de systèmes de fichiers clusterisés (par exemple pour conserver des copies de sauvegarde).

1. Créez un système de fichiers GFS2 sur le volume logique avec la commande **`mkfs.gfs2`**.
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 GFS2 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 des 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 à l'aide des paramètres définis dans le fichier **lvm.conf**. Une sauvegarde système quotidienne 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.

Vous pouvez manuellement sauvegarder les métadonnées dans le fichier **/etc/lvm/backup** avec la commande **vgcfgbackup**. Vous pouvez restaurer les métadonnées avec la commande **vgcfgrestore**. Les commandes **vgcfgbackup** et **vgcfgrestore** sont décrites dans la 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

Les niveaux de journalisation sont définis dans le fichier **/etc/lvm/lvm.conf**, décrit dans l'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

Si vous êtes en train de créer ou de modifier un volume LVM pour un environnement clusterisé, vous devez vous assurer que vous exécutez bien le démon **clvmd**. Pour obtenir plus d'informations, voir 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 requise mais est laissée vide, une liste de tous les groupes de volumes sera substituée. Si, à la place d'une liste de volumes logiques requis, 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.

Tous les objets LVM sont référencés en interne par un UUID, qui est assigné lorsque vous créez l'objet. Cela peut être utile dans une situation où vous retirez un volume physique appelé **/dev/sdf** d'un groupe de volumes qui, lorsque vous le remettez à nouveau, change de nom pour s'appeler **/dev/sdk**. LVM trouvera encore le volume physique car il l'identifie par son UUID et non pas son nom de périphérique. Pour davantage d'informations à propos de la spécification de l'UUID d'un volume physique lors de sa création, reportez-vous à la 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**.

La commande **pvs** fournit des informations sur les volumes physiques dans une fenêtre configurable, en affichant une ligne par volume physique. La commande **pvs** fournit de nombreuses options de contrôle de format et elle est utile pour les scripts. Pour davantage d'informations à propos de l'utilisation de la commande **pvs** afin de personnaliser votre sortie, reportez-vous à la Section 4.8, « 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]
```

Vous pouvez définir un filtre dans le fichier **lvm.conf** afin que cette commande évite l'analyse de certains volumes physiques. Pour davantage d'informations sur l'utilisation des filtres afin de contrôler les périphériques à analyser, reportez-vous à la Section 4.5, « 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.

Si le volume physique que vous voulez supprimer fait partie d'une groupe de volumes, vous devez le supprimer du groupe de volumes avec la commande **vgreduce**, comme illustré dans la 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

Pour créer un groupe de volumes à partir d'un ou plusieurs volumes physiques, utilisez la commande **vgcreate**. La commande **vgcreate** permet de créer un nouveau groupe de volumes par nom et de lui ajouter au moins un volume physique.

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 taille de l'extension par défaut ne correspond pas à vos besoins, vous pouvez la spécifier avec la commande **vgcreate** et l'option **-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 **normale**. 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**.

Pour obtenir des informations sur l'utilisation de la politique **cling** en conjonction avec les balises LVM afin de spécifier quels volumes physiques supplémentaires utiliser lors de l'extension d'un volume LVM, voir Section 4.4.12.2, « Étendre un volume logique avec la politique d'allocation **cling** ».

En général, les politiques d'allocation autres que **normal** sont uniquement requises 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 dans un cluster

Vous pouvez créer des groupes de volumes dans un environnement clusterisé avec la commande **vgcreate**, tout comme lorsque vous les créez sur un noeud unique.

Par défaut , les groupes de volumes créés avec CLVM sur un stockage partagé sont visibles à tous les ordinateurs qui ont accès au stockage partagé. Il est toutefois possible de créer des groupes de volumes qui sont locaux et visibles à un seul noeud dans le cluster en utilisant **-c n** avec la commande **vgcreate**.

La commande suivante, lorsqu'exécutée dans un environnement clusterisé, crée un groupe de volumes qui est local au noeud depuis lequel 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
```

Vous pouvez modifier un groupe de volumes existant pour qu'il soit local ou clusterisé avec l'option **-c** de la commande **vgchange**,comme décrit dans la Section 4.3.7, « Changement des paramètres du groupe de volumes ».

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

Pour davantage d'informations à propos de la commande **vgs**, reportez-vous à la Section 4.3.4, « Affichage des groupes de volumes » Section 4.8, « Rapport personnalisé pour LVM », et à la page man **vgs**.

4.3.3. Ajout de volumes physiques à un groupe de volumes

Pour ajouter des volumes physiques supplémentaires à un groupe de volumes existant, utilisez la commande **vgextend**. La commande **vgextend** augmente la capacité d'un groupe de volumes en ajoutant un ou plusieurs volumes physiques libres.

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**.

La commande **vgscan**, qui scanne tous les disques pour trouver les groupes de volumes et reconstruit le fichier cache LVM, affiche aussi les groupes de volumes. Pour davantage d'informations à propos de la commande **vgscan**, reportez-vous à la Section 4.3.5, « Analyse des disques pour les groupes de volumes afin de construire le fichier de cache ».

La commande **vgs** fournit des informations sur les groupes de volumes sous une forme configurable, en affichant une ligne par groupe de volumes. La commande **vgs** fournit de nombreuses options de contrôle du format et est utile pour les scripts. Pour davantage d'informations sur l'utilisation de la commande **vgs** afin de personnaliser votre sortie, reportez-vous à la Section 4.8, « Rapport personnalisé pour LVM ».

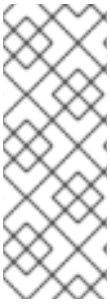
La commande **vgdisplay** affiche les propriétés des groupes de volumes (la taille, les extensions, le nombre de volumes physiques, etc.) dans un format fixe. L'exemple suivant illustre la sortie d'une commande **vgdisplay** pour le groupe de volumes **new_vg**. Si vous ne spécifiez pas un groupe de volumes, tous les groupes de volumes existants seront 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 la commande **vgscan** automatiquement lors du démarrage du système ainsi qu'à d'autres moments lors des opérations LVM, par exemple lorsque vous exécutez une commande **vgcreate**, ou lorsque LVM détecte une incohérence.



NOTE

Vous pourriez devoir exécuter la commande **vgscan** manuellement lorsque vous changez la configuration de votre matériel et lorsque vous ajoutez ou supprimez un périphérique d'un noeud, rendant ainsi visibles des périphériques qui n'étaient pas présents au démarrage du 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.

Vous pouvez définir un filtre dans le fichier **lvm.conf** afin de restreindre l'analyse à des périphériques particuliers. Pour obtenir des informations à propos de l'utilisation des filtres afin de contrôler les fichiers devant être analysés, reportez à la Section 4.5, « 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

Pour supprimer les volumes physiques inutilisés d'un groupe de volumes, utilisez la commande **vgreduce**. La commande **vgreduce** diminue la capacité d'un groupe de volumes en supprimant un ou plusieurs volumes physiques vides. Cela permet de libérer ces volumes physiques afin qu'ils soient utilisés dans d'autres groupes de volumes ou supprimés du système.

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

La commande **vgchange** est utilisée pour activer et désactiver les groupes de volumes, comme décrit dans Section 4.3.8, « Activation et désactivation des groupes de volumes ». Vous pouvez aussi utiliser cette commande pour changer certains paramètres de groupes de volumes pour un groupe de volumes existant.

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.

Vous pouvez désactiver individuellement des volumes logiques avec la commande **lvchange**, comme le décrit la Section 4.4.8, « Changement des paramètres d'un groupe de volumes logiques ». Pour obtenir des informations à propos de l'activation de volumes logiques sur des noeuds individuels au sein d'un cluster, reportez-vous à la Section 4.7, « 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. Partager 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

Pour combiner deux groupes de volumes en un, utilisez la commande **vgmerge**. Vous pouvez fusionner un volume "source" inactif avec un volume de "destination" actif ou inactif si les tailles des extensions physiques du volume sont les mêmes et si les résumés des volumes logiques et physiques des deux groupes de volumes tiennent dans les limites des groupes de volumes de destination.

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 verbeuses.

```
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.

Pour obtenir un exemple à propos de l'utilisation de la commande **vgcfgrestore** pour récupérer les métadonnées d'un volume physique, reportez-vous à la 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** à la commande **vgscan**.

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 linéaires

Pour créer un volume logique, utilisez la commande **lvcreate**. Si vous ne spécifiez pas de nom pour le volume logique, le nom par défaut **lvol#** est utilisé lorsque *#* correspond au numéro interne du volume logique.

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
```

Vous pouvez utiliser l'argument **-l** de la commande **lvcreate** pour créer un volume logique qui utilise le groupe de volumes entier. Une autre manière de créer un volume logique qui utilise le groupe de volumes entier est d'utiliser la commande **vgdisplay** afin de déterminer la taille "Total PE" et d'utiliser les résultats en tant qu'entrée de la commande **lvcreate**.

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
```

Les volumes physiques sous-jacents utilisés pour créer un volume logique peuvent être importants si le volume physique doit être supprimé, vous devez ainsi considérer cette possibilité lorsque vous créez un volume logique. Pour davantage d'informations à propos de la suppression d'un volume physique d'un groupe de volumes, reportez-vous à la 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 dans un 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 un volume logique. L'exemple suivant crée un volume logique linéaire à partir des extensions 0 à 24 du volume physique **/dev/sda1** et à partir des extensions 50 à 124 du volume physique**/dev/sdb1** dans le groupe de volumes **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-24 /dev/sdb1:50-124
```

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-
```

La politique par défaut quant à la façon dont les extensions d'un volume logique sont allouées est **inherit**. Celle-ci s'applique également au groupe de volumes. Ces politiques peuvent être changées en utilisant la commande **lvchange**. Pour davantage d'informations à propos des politiques d'allocation, reportez-vous à la Section 4.3.1, « Création de groupes de volumes ».

4.4.2. Création de volumes en mode stripe

Pour les lectures et écritures séquentielles volumineuses, la création d'un volume logique en mode stripe peut améliorer l'efficacité des E/S de données. Pour davantage d'informations à propos des volumes en mode stripe, reportez-vous à la 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 se trouvant sur deux volumes physiques. Il s'appelle **stripelv** et fait partie du groupe de volumes **testvg**. Le stripe utilisera les secteurs 0 à 49 de **/dev/sda1** et les secteurs 50 à 99 de **/dev/sdb1**.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-49 /dev/sdb1:50-99
  Using default stripesize 64.00 KB
  Logical volume "stripelv" created
```

4.4.3. Création de volumes en miroir

NOTE

Créer un volume logique LVM en miroir au sein d'un cluster requiert les mêmes commandes et procédures que la création d'un volume logique LVM en miroir sur un noeud unique. Cependant, afin de créer un volume LVM en miroir dans un cluster, le cluster et l'infrastructure du miroir du cluster doivent être en cours d'exécution, le cluster doit avoir atteint le quorum, et le type de verrouillage dans le fichier **lvm.conf** doit être configuré correctement afin de permettre le verrouillage du cluster. Pour un exemple de création d'un volume en miroir au sein d'un cluster, reportez-vous à la Section 5.5, « Création d'un volume logique LVM miroir dans un cluster ».

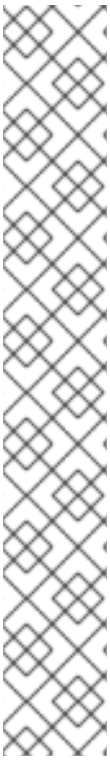
Tenter d'exécuter de multiples commandes de création et de conversions de miroirs LVM en succession rapide depuis de multiples noeuds dans un cluster peut provoquer une file d'attente de ces commandes. Cela peut faire expirer certaines des opérations requises, et les faire échouer. Pour éviter ce problème, il est recommandé d'exécuter les commandes de création de miroirs en cluster depuis un seul noeud du cluster.

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 mirrorlv vg0
```

Un miroir LVM divise le périphérique en train d'être copié en régions faisant une taille de 512 Ko par défaut. Vous pouvez utiliser l'argument **-R** de la commande **lvcreate** pour spécifier la taille des régions en Mo. Vous pouvez aussi changer la taille des régions par défaut en modifiant le paramètre **mirror_region_size** dans le fichier **lvm.conf**.



NOTE

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

En règle générale pour spécifier la taille des régions pour des miroirs faisant plus de 1,5 To, vous pouvez prendre la taille de votre miroir en téraoctets et arrondir ce nombre à la puissance de 2 suivante, en utilisant ce nombre en tant qu'argument **-R** de la commande **lvcreate**. Par exemple, si la taille de votre miroir est de 1,5 To, vous pourriez spécifier **-R 2**. Si la taille de votre miroir est de 3 To, vous pourriez spécifier **-R 4**. Si votre miroir fait 5 To, vous pourriez spécifier **-R 8**

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

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

LVM maintient un petit fichier journal afin de garder une trace des régions synchronisées avec le ou les miroirs. Par défaut, ce fichier journal est stocké sur le disque, il est donc persistant à travers les redémarrages et assure ainsi que le miroir ne doit pas être resynchronisé à chaque fois qu'une machine redémarre ou se bloque. Au lieu de cela, vous pouvez spécifier que ce fichier journal soit stocké en mémoire avec l'argument **--mirrorlog core** ; ceci élimine le besoin de posséder un périphérique de fichiers journaux supplémentaire, mais le miroir entier devra être resynchronisé à chaque redémarrage.

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

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

Le fichier journal du miroir est créé sur un périphérique séparé à partir des périphériques sur lesquels n'importe quelle branche du miroir peut être créée. Il est cependant possible de créer un fichier journal sur le même périphérique que celui des branches du miroir à l'aide de l'argument **--alloc anywhere** de la commande **vgcreate**. Ceci peut dégrader la performance, mais vous permet aussi de créer un miroir, même si vous ne possédez que deux périphériques sous-jacents.

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

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

NOTE

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

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

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

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

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 et les branches du miroir, ainsi que les extensions des périphériques à utiliser. 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 et un seul fichier journal qui n'est pas en miroir. Le volume a une taille de 500 méga-octets, il s'appelle **mirrorlv** et est issu du groupe de volumes **vg0**. La première branche du miroir se trouve sur le périphérique **/dev/sda1**, la seconde sur **/dev/sdb1** et le fichier journal du miroir se trouve sur **/dev/sdc1**.

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

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

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499
/dev/sdc1:0
```

NOTE

À partir de la version 6.1 de Red Hat Enterprise Linux, vous pouvez combiner RAID0 (striping) et RAID1 (mirroring) dans un seul volume logique. La création d'un volume logique tout en spécifiant le nombre de miroirs (**--mirrors X**) et le nombre de stripes (**--stripes Y**) résulte en un périphérique miroir dont les périphériques constituants sont en mode stripe.

4.4.3.1. Politique en cas d'échec des volumes logiques en miroir

Au cas où un échec de périphérique surviendrait, vous pouvez définir de quelle manière un volume logique en miroir se comporte avec les paramètres **mirror_image_fault_policy** et **mirror_log_fault_policy** dans la section **activation** du fichier **lvm.conf**. Lorsque ces paramètre sont définis sur **remove**, le système tente de supprimer le périphérique défectueux et de s'exécuter sans celui-ci. Lorsque ce paramètre est définit sur **allocate**, le système tente de supprimer le périphérique défectueux et essaie d'allouer de l'espace sur un nouveau périphérique afin de remplacer le périphérique défectueux ; cette politique fonctionne comme la politique **remove** s'il n'y a pas de périphérique ni d'espace convenable à allouer pour le remplacement.

Par défaut, le paramètre **mirror_log_fault_policy** est défini sur **allocate**. L'utilisation de cette politique pour le fichier journal est rapide et offre la possibilité de rester capable de se rappeler de l'état de la synchronisation à travers les pannes et les redémarrages. Si vous définissez cette politique sur

remove, lorsqu'un périphérique de fichier journal échoue, le miroir se mettra à utiliser un journal se trouvant dans la mémoire et le miroir ne se rappellera pas de sont statut de synchronisation à travers les pannes et les redémarrages. Le miroir devra donc être entièrement resynchronisé.

Par défaut, le paramètre **mirror_image_fault_policy** est défini sur **remove**. Avec cette politique, si une image de miroir échoue, le miroir se convertira en périphérique sans miroir s'il ne reste qu'une seule copie correcte. Définir cette politique sur **allocate** pour un périphérique en miroir requiert que le miroir resynchronise les périphériques; ce processus est lent, mais préservera les caractéristiques de miroir du périphérique.

NOTE

Lorsqu'un miroir LVM souffre d'un échec de périphérique, une restauration en deux étapes s'effectue. La première étape implique la suppression des périphériques en échec. Ceci peut faire que le miroir soit réduit à l'état de périphérique linéaire. La deuxième étape, si le paramètre **mirror_log_fault_policy** est défini sur **allocate**, consiste à tenter de remplacer l'un des périphériques en échec. Remarquez cependant qu'il n'y a pas de garanties que la seconde étape choisira des périphériques qui étaient utilisés auparavant par le miroir qui ne faisait pas partie de l'échec si d'autres sont disponibles.

Pour plus d'informations sur la restauration manuelle d'échec de miroir LVM, voir Section 6.3, « Récupération suite à un échec miroir LVM ».

4.4.3.2. Diviser une image redondante d'un volume logique en miroir

VOus pouvez diviser une image redondante d'un volume logique en miroir afin de former un nouveau volume logique. Pour diviser une image, vous devez utiliser l'argument **--splitmirrors** de la commande **lvconvert** en spécifiant le nombre d'images redondantes à diviser. Vous devrez utiliser l'argument **--name** de la commande pour spécifier un nom pour le volume logique divisé.

La commande suivante divise un nouveau volume logique nommé **copy** à partir du volume logique **vg/lv**. Le nouveau volume logique contient deux branches du miroir, Dans cet exemple, LVM sélectionne quels sont les périphériques à diviser.

```
lvconvert --splitmirrors 2 --name copy vg/lv
```

Vous pouvez spécifier quels périphériques diviser. La commande suivante divise un nouveau volume logique nommé **copy** depuis le volume logique en miroir **vg/lv**. Le nouveau volume logique contient deux branches de miroir composées des périphériques **/dev/sdc1** et **/dev/sde1**.

```
lvconvert --splitmirrors 2 --name copy vg/lv /dev/sd[ce]1
```

4.4.3.3. Réparer un périphérique logique en miroir

Vous pouvez utiliser la commande **lvconvert --repair** pour réparer un miroir après un échec du disque. Ceci remet le miroir dans un état consistant. La commande **lvconvert --repair** est une commande interactive qui vous demande d'indiquer si vous souhaitez que le système tente de remplacer le(s) périphérique(s) en panne.

- Pour ignorer les demandes et remplacer tout les périphériques en échec, spécifiez l'option **-y** dans la ligne de commande.

- Pour ignorer les demandes et ne remplacer aucun des périphériques en échec, spécifiez l'option **-f** dans la ligne de commande.
- Pour ignorer les demandes mais continuer à indiquer les différentes politiques de remplacement pour l'image du miroir et le fichier journal du miroir, vous pouvez spécifier l'argument **--use-policies** pour utiliser les politiques de remplacement de périphérique spécifiées par les paramètres **mirror_log_fault_policy** et **mirror_device_fault_policy** dans le fichier **lvm.conf**.

4.4.3.4. 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 miroir 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.

Si vous perdez une branche d'un miroir, LVM convertit le volume en un volume linéaire afin que vous ayez encore accès au volume, sans la redondance du miroir. Après avoir remplacé la branche, vous pouvez utiliser la commande **lvconvert** pour restaurer le miroir. Cette procédure est expliquée dans la 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.

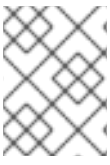
```
lvconvert -m0 vg00/lvol1
```

4.4.4. 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 pris en charge par les noeuds dans un cluster. Vous ne pouvez pas créer d'instantané de volume dans un groupe de volumes clusterisé. Cependant, à partir de la version 6.1 de Red Hat Enterprise Linux, si vous devez créer une copie de sauvegarde de données consistante sur un volume logique clusterisé, vous pouvez activer le volume de manière exclusive, puis créer l'instantané. Pour obtenir des informations sur l'activation des volumes logiques exclusivement sur un noeud, voir Section 4.7, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».



NOTE

À partir de la version 6.1 de Red Hat Enterprise Linux, les instantanés LVM sont pris en charge par les volumes logiques en miroir.

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
```

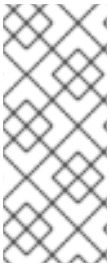
Après avoir créé un instantané de volume logique, vous pouvez spécifier un volume d'origine avec la commande **lvdisplay** afin de générer une sortie qui inclue une liste de tous les instantanés de volumes logiques et leur statut (actif ou inactif).

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 de l'instantané du volume utilisé pour 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.

4.4.5. Fusion d'instantanés de volumes

À partir de la version 6 de Red Hat Enterprise Linux, vous pouvez utiliser l'option **--merge** de la

commande **lvconvert** pour fusionner un instantané dans son volume d'origine. Si le volume d'origine et le volume de l'instantané ne sont pas ouverts, la fusion démarrera immédiatement. Sinon, la fusion démarrera lorsque l'instantané ou l'origine sont activés et lorsque les deux sont fermés. Fusionner un instantané dans un volume d'origine qui ne peut pas être fermé, un système de fichier root par exemple, est reporté jusqu'à la prochaine activation du volume d'origine. Lorsque la fusion démarre, le volume logique en résultant comportera le nom, le numéro mineur et l'UUID du volume d'origine. Pendant la fusion, les opérations de lecture et d'écriture sur le volume d'origine apparaissent telles qu'elles le devraient à l'instantané en cours de fusion. Lorsque la fusion se termine, l'instantané fusionné est supprimé.

La commande suivante fusionne le volume instantané **vg00/lvol1_snap** dans son volume d'origine.

```
lvconvert --merge vg00/lvol1_snap"
```

Vous pouvez spécifier de multiples instantanés sur la ligne de commande, ou vous pouvez utiliser des balises d'objets LVM pour spécifier que de multiples instantanés doivent être fusionnés dans leurs volumes d'origine respectifs. Dans l'exemple suivant, les volumes logiques **vg00/lvol1**, **vg00/lvol2**, et **vg00/lvol3** possèdent tous la balise **@some_tag**. La commande suivante fusionne les instantanés de volumes logiques pour les trois volumes en série : **vg00/lvol1**, **vg00/lvol2**, puis **vg00/lvol3**. Si l'option **--background** avait été utilisée, toutes les fusions des instantanés de volumes logiques auraient démarrées en parallèle.

```
lvconvert --merge @some_tag"
```

Pour obtenir des informations sur le balisage des objets LVM, voir Annexe C, *Les balises des objets LVM*. Pour plus d'informations sur la commande **lvconvert --merge**, reportez-vous à la page de manuel **lvconvert**(8).

4.4.6. 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
```

Utilisez un numéro de miroir élevé pour vous assurez qu'il n'est pas déjà été alloué dynamiquement à un autre périphérique.

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.

4.4.7. Redimensionnement des volumes logiques

Pour réduire 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 une taille identique ou supérieure à celle du système de fichiers.

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

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

4.4.8. 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).

Vous pouvez utiliser la commande **lvchange** pour activer et désactiver les volumes logiques. Pour activer et désactiver tous les volumes logiques d'un groupe de volumes, utilisez la commande **vgchange**, comme le décrit la 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 seule (read-only).

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

4.4.9. 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
```

Pour davantage d'informations à propos de l'activation de volumes logiques sur les noeuds individuels d'un cluster, reportez-vous à la Section 4.7, « Activation des volumes logiques sur les noeuds individuels d'un cluster ».

4.4.10. Suppression de volumes logiques

Pour supprimer un volume logique inactif, utilisez la commande **lvremove**. Si le volume logique est actuellement monté, démontez-le avant de le supprimer. De plus, dans un environnement en clusters, vous devez désactiver un volume logique avant qu'il puisse être supprimé.

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.

4.4.11. 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**.

La commande **lvs** fournit des informations sur les volumes logiques sous un format configurable, en affichant une ligne par volume logique. La commande **lvs** fournit de nombreuses options de contrôle de format et est utile pour les scripts. Pour davantage d'informations à propos de l'utilisation de la commande **lvs** afin de personnaliser votre sortie, reportez-vous à la Section 4.8, « 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
```

4.4.12. Augmentez la taille des volumes logiques

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

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** jusqu'à 12 giga-octets.

```
# 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.

4.4.12.1. 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 un autre volume physique, puis étendez le volume logique. Dans cet exemple, étant donné que nous avons ajouté deux volumes physiques dans le groupe de volumes, nous pouvons étendre le volume logique jusqu'à la totalité de la taille 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 remplacer ces paramètres. L'exemple suivant étend le volume logique en mode stripe existant afin d'utiliser l'espace libre restant après l'échec de la commande initiale **lvextend**.

```
# 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
```

4.4.12.2. Étendre un volume logique avec la politique d'allocation `cling`

Lors de l'extension d'un volume LVM, vous pouvez utiliser l'option **--alloc cling** de la commande **lvextend** afin de spécifier la politique d'allocation **cling**. Cette politique choisira l'espace sur les mêmes volumes physiques que le dernier segment du volume logique existant. Si l'espace sur les volumes physiques est insuffisant et qu'une liste de balises est définie dans le fichier **lvm.conf**, LVM vérifiera s'il y a des balises attachées aux volumes physiques et cherchera à faire correspondre ces balises de volumes physiques avec les extensions existantes et les nouvelles extensions.

Par exemple, su vous possédez des volumes logiques qui sont en miroir entre deux sites dans un groupe de volumes unique, vous pouvez baliser les volumes physiques en fonction de leur emplacement en marquant les volumes physiques avec les balises @site1 et @site2, puis spécifiez la ligne suivante dans le fichier **lvm.conf** :

```
cling_tag_list = [ "@site1", "@site2" ]
```

Pour davantage d'informations sur le balisage de volumes physiques, voir Annexe C, *Les balises des objets LVM*.

Dans l'exemple suivant, le fichier **lvm.conf** a été modifié afin de contenir la ligne suivante :

```
cling_tag_list = [ "@A", "@B" ]
```

Toujours dans cet exemple, un groupe de volumes **taft**a été créé, il est composé des volumes physiques **/dev/sdb1**, **/dev/sdc1**, **/dev/sdd1**, **/dev/sde1**, **/dev/sdf1**, **/dev/sdg1**, et **/dev/sdh1**. Ces volumes physiques ont été marqués avec les balises **A**, **B**, et **C**. L'exemple n'utilise pas la balise **C**, mais il montrera que utilise les balises pour sélectionner quels volumes physiques utiliser pour les branches du miroir.

```
[root@taft-03 ~]# pvs -a -o +pv_tags /dev/sd[bcdefgh]1
  PV         VG   Fmt  Attr PSize   PFree   PV Tags
  /dev/sdb1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdc1  taft lvm2 a-   135.66g 135.66g B
  /dev/sdd1  taft lvm2 a-   135.66g 135.66g B
  /dev/sde1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdf1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdg1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdh1  taft lvm2 a-   135.66g 135.66g A
```

La commande suivante permet de créer un volume miroir de 100G pour le groupe de volumes **taft**.

```
[root@taft-03 ~]# lvcreate -m 1 -n mirror --nosync -L 100G taft
```

La commmande suivante affiche les périphériques utilisés pour les branches et le journal dumiroir. miroir

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                VG        Attr   LSize   Log         Copy%  Devices
  mirror            taft      Mwi-a- 100.00g mirror_mlog 100.00
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      iwi-ao 100.00g
/dev/sdb1(0)
  [mirror_mimage_1] taft      iwi-ao 100.00g
/dev/sdc1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

La commande suivante agrandit la taille du volume miroir à l'aide de la politique d'allocation **cling** pour indiquer que les branches du miroir devraient être étendues à l'aide des volumes physiques avec la même balise.

```
[root@taft-03 ~]# lvextend --alloc cling -L +100G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 200.00 GiB
  Logical volume mirror successfully resized
```

La commande d'affichage suivante montre que les branches du miroir ont été étendues à l'aide des volumes physiques avec la même balise que la branche. Remarquez que les volumes physiques avec la balise **C** ont été ignorés.ignorés.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                VG        Attr   LSize   Log         Copy%  Devices
  mirror            taft      Mwi-a- 200.00g mirror_mlog  50.16
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdb1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdg1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdc1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdd1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

4.4.13. 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. 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 du fichier **lvm.conf** se composent d'une série d'expressions régulières simples appliquée sur les noms des 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 la meilleure 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/.*/" ]
```

Pour davantage d'informations à propos du fichier **lvm.conf**, reportez-vous à la Annexe B, *Les fichiers de configuration LVM* et à la page de manuel **lvm.conf**(5).

4.6. 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).

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 uniquement 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
```

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

Si LVM est installé dans un environnement en cluster, vous pourriez parfois avoir besoin d'activer les volumes logiques exclusivement sur un noeud.

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.

Vous pouvez également activer des volumes logiques sur des noeuds individuels en utilisant les balises LVM, qui sont décrites dans la Annexe C, *Les balises des objets LVM*. Vous pouvez aussi spécifier l'activation de noeuds dans le fichier de configuration, qui est décrit dans la Annexe B, *Les fichiers de configuration LVM*.

4.8. 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.

4.8.1. 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`**. Par exemple, la sortie suivante illustre 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.

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
```

Vous pouvez utiliser l'argument **-P** des commandes **lvs** ou **vgs** afin d'afficher les informations d'un volume ayant échoué qui n'auraient normalement pas dû apparaître dans la sortie. Pour davantage d'informations à propos de la sortie que cet argument génère, reportez-vous à la 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
```

4.8.2. 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 » liste les arguments d'affichage de la commande **pvs**, ainsi que le nom de champ tel qu'il apparaît dans l'en-tête d'affichage et une description du champ.

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
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 » liste les arguments d'affichage de la commande **vgs**, ainsi que le nom de champ tel qu'il apparaît dans l'en-tête d'affichage et une description du champ.

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** ou **lvm1**)
vg_free	VFree	La taille de l'espace libre restant dans le groupe de volumes

Argument	En-tête	Description
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 » liste les arguments d'affichage de la commande **lvs**, ainsi que le nom de champ tel qu'il apparaît dans l'en-tête d'affichage et une description du champ.

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

Argument	En-tête	Description
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
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 Bit 2: 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: State: (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

Argument	En-tête	Description
mirror_log	Journal	Le périphérique sur lequel réside le fichier journal du miroir
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
```

4.8.3. 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
```

4.8.4. 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 GFS2 sur le volume logique.

```
[root@tng3-1 ~]# mkfs.gfs2 -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 permet de créer le groupe de volumes **volgroup01**.

```
[root@tng3-1 ~]# vgcreate volgroup01 /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "volgroup01" 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
  volgroup01         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 **volgroup01**. 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 volgroup01
  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 GFS2 sur le volume logique.

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/volgroup01/striped_logical_volume
This will destroy any data on /dev/volgroup01/striped_logical_volume.

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

Device:                    /dev/volgroup01/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/volgroup01/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/volgroup01/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 ~]#  mkfs.gfs2 -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.

5.5. CRÉATION D'UN VOLUME LOGIQUE LVM MIROIR DANS UN CLUSTER

La création d'un volume logique LVM miroir dans un cluster requiert les mêmes commandes et procédures que la création d'un volume logique LVM miroir sur un noeud unique. Cependant, afin de créer un volume LVM miroir dans un cluster, le cluster et l'infrastructure miroir du cluster doivent être en cours d'exécution, le cluster doit comporter le quorum et le type de verrouillage dans le fichier **lvm.conf** doit être correctement configuré pour permettre le verrouillage du cluster soit directement, soit par le biais de la commande **lvmconf** comme décrit dans la Section 3.1, « Création de volumes LVM dans un cluster ».

La procédure suivante crée un volume LVM miroir dans un cluster. D'abord, la procédure vérifie si les services du cluster sont bien installés et en cours d'exécution, puis elle crée le volume miroir.

1. Pour créer un volume logique miroir partagé par tous les noeuds d'un cluster, le type de verrouillage doit être correctement défini dans le fichier **lvm.conf** dans chaque noeud du cluster. Par défaut, le type de verrouillage est défini sur « local ». Pour modifier ceci, exécutez la commande suivante dans chaque noeud du cluster afin de permettre le verrouillage clusterisé.

   ```
   # /sbin/lvmconf --enable-cluster
   ```

2. Pour créer un volume logique clusterisé, l'infrastructure du cluster doit être en cours d'exécution sur chaque noeud du cluster. L'exemple suivant vérifie que le démon **clvmd** est en cours d'exécution sur le noeud d'où il provient :

   ```
   [root@doc-07 ~]# ps auxw | grep clvmd
   root     17642  0.0  0.1 32164 1072 ?        Ssl  Apr06   0:00
   clvmd -T20 -t 90
   ```

   La commande suivante affiche une vue locale du statut du cluster :

```
[root@example-01 ~]# cman_tool services
fence domain
member count  3
victim count  0
victim now    0
master nodeid 2
wait state    none
members       1 2 3

dlm lockspaces
name          clvmd
id            0x4104eefa
flags         0x00000000
change        member 3 joined 1 remove 0 failed 0 seq 1,1
members       1 2 3
```

3. Assurez-vous que le paquetage **cmirror** est bien installé.

4. Lancez le service **cmirrord**.

```
[root@hexample-01 ~]# service cmirrord start
Starting cmirrord:                                         [  OK
]
```

5. Créez le miroir. La première étape est la création des volumes physiques. Les commandes suivantes créent trois volumes physiques. Deux des volumes physiques seront utilisés comme branches du miroir (de l'anglais, « mirror legs »), et le troisième volume physique contiendra le journal miroir.

```
[root@doc-07 ~]# pvcreate /dev/xvdb1
  Physical volume "/dev/xvdb1" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdb2
  Physical volume "/dev/xvdb2" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdc1
  Physical volume "/dev/xvdc1" successfully created
```

6. Créez le groupe de volumes. Cet exemple crée le groupe de volumes **vg001**, qui est composé des trois volumes physiques créés dans l'étape précédente.

```
[root@doc-07 ~]# vgcreate vg001 /dev/xvdb1 /dev/xvdb2 /dev/xvdc1
  Clustered volume group "vg001" successfully created
```

   Remarquez que la sortie de la commande **vgcreate** indique que le groupe de volumes est clusterisé. Vous pouvez vérifier si un groupe de volumes est clusterisé avec la commande **vgs**, qui affiche les attributs d'un groupe de volumes. Si un groupe de volumes est clusterisé, l'attribut « c » sera affiché.

```
[root@doc-07 ~]# vgs vg001
  VG       #PV #LV #SN Attr   VSize  VFree
  vg001      3   0   0 wz--nc 68.97G 68.97G
```

7. Créez le volume logique miroir. Cet exemple crée le volume logique **mirrorlv** à partir du groupe de volumes **vg001**. Ce volume possède une seule branche de miroir (de l'anglais,

« mirror leg »). L'exemple spécifie les extensions du volume physique qui seront utilisées pour le volume logique.

```
[root@doc-07 ~]# lvcreate -l 1000 -m1 vg001 -n mirrorlv
/dev/xvdb1:1-1000 /dev/xvdb2:1-1000 /dev/xvdc1:0
  Logical volume "mirrorlv" created
```

Vous pouvez utiliser la commande **lvs** pour afficher le progrès de la création du miroir. L'exemple suivant affiche que le miroir est synchronisé à 47%, puis synchronisé à 91%, le miroir est créé lorsque la synchronisation est à 100%.

```
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
47.00
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
91.00
[root@doc-07 ~]#  lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
100.00
```

La création du miroir est notée dans le journal système :

```
May 10 14:52:52 doc-07 [19402]: Monitoring mirror device vg001-
mirrorlv for events
May 10 14:55:00 doc-07 lvm[19402]: vg001-mirrorlv is now in-sync
```

8. Vous pouvez utiliser la commande **lvs** avec les options **-o +devices** pour afficher la configuration du miroir, y compris pour afficher quels périphériques remplissent le rôle de branches du miroir (de l'anglais, « mirror legs »). On peut observer dans cet exemple que le volume logique est composé de deux images linéaires et d'un journal.

```
[root@doc-07 ~]# lvs -a -o +devices
  LV                 VG         Attr   LSize  Origin Snap%  Move
Log           Copy%  Convert Devices
  mirrorlv           vg001      mwi-a-  3.91G
mirrorlv_mlog 100.00
mirrorlv_mimage_0(0),mirrorlv_mimage_1(0)
  [mirrorlv_mimage_0] vg001      iwi-ao  3.91G
/dev/xvdb1(1)
  [mirrorlv_mimage_1] vg001      iwi-ao  3.91G
/dev/xvdb2(1)
  [mirrorlv_mlog]     vg001      lwi-ao  4.00M
/dev/xvdc1(0)
```

Vous pouvez utiliser l'option **seg_pe_ranges** de la commande **lvs** pour afficher la structure des données. Cette option peut être utilisée pour vérifier que votre structure est correctement redondante. La sortie de cette commande afficher les échelles PE sous le même format que les

saisies des commande **lvcreate** et **lvresize**.

```
[root@doc-07 ~]# lvs -a -o +seg_pe_ranges --segments
  PE Ranges
  mirrorlv_mimage_0:0-999 mirrorlv_mimage_1:0-999
  /dev/xvdb1:1-1000
  /dev/xvdb2:1-1000
  /dev/xvdc1:0-0
```

NOTE

Pour plus d'informations sur la restauration d'échec d'une des branches d'un volume miroir LVM, voir Section 6.3, « Récupération suite à un échec miroir LVM ».

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.
- Si un problème est lié à l'activation du volume logique, définissez 'activation = 1' dans la section 'log' du fichier de configuration et exécutez la commande avec l'argument **-vvvv**. Après avoir examiné cette sortie assurez-vous de réinitialiser le paramètre à 0 afin d'éviter d'éventuels problèmes avec le verrouillage de la machine en cas de mémoire insuffisante.
- 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. Cet argument permet certaines opérations même si les métadonnées internes ne sont pas totalement consistantes. 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 et le paramètre **mirror_log_fault_policy** est défini sur **remove**, nécessitant ainsi que vous reconstruisiez le miroir manuellement. Pour obtenir des informations sur l'ajustement du paramètre **mirror_log_fault_policy**, reportez-vous à la Section 6.3, « Récupération suite à un échec miroir LVM ».

Lorsque la branche d'un miroir échoue, LVM convertit le volume miroir 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.

Pour reconstruire le volume en miroir, vous remplacez le périphérique cassé et recréez le volume physique. Si vous utilisez le même disque plutôt que de le remplacer par un nouveau, vous verrez des avertissements "inconsistent" lorsque vous exécuterez la commande **pvcreate**. Vous pouvez éviter l'apparition de ces avertissements en exécutant la commande **vgreduce --removemissing**.

```
[root@link-08 ~]# pvcreate /dev/sdi[12]
  Physical volume "/dev/sdi1" successfully created
  Physical volume "/dev/sdi2" 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/sdi1           lvm2 [603.94 GB]
  PV /dev/sdi2           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/sdi[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/sdi1   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  PV /dev/sdi2   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/sdi1 /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/sdi1(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** n'écrase que 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
```

Vous pouvez ensuite utiliser la commande **vgcfgrestore** pour restaurer les métadonnées d'un groupe de volumes.

```
[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

Si un volume physique échoue ou a besoin d'être remplacé, vous pouvez en étiqueter un nouveau pour remplacer celui qui a été perdu dans le groupe de volumes existant en suivant la même procédure que celle utilisée pour la récupération des métadonnées d'un volume physique, décrite dans la Section 6.4, « Recupération des métadonnées du volume physique ». Vous pouvez utiliser les arguement **--partial** et **--verbose** de la commande **vgdisplay** pour afficher les UUID et les tailles des volumes physiques qui ne sont plus présents. Si vous souhaitez substituer un autre volume physique de même taille, vous pouvez utiliser la commande **pvcreate** avec les arguments **--restorefile** et **--uuid** pour initialiser un nouveau périphérique avec le même UUID que le volume physique manquant. Vous pouvez ensuite utiliser la commande **vgcfgrestore** pour restaurer les métadonnées du groupe de volumes.

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

Vous pourriez obtenir un message d'erreur "Insufficient free extents" lors de la création d'un volume logique bien que vous pensiez avoir suffisamment d'extensions en vous basant sur la sortie des commandes **vgdisplay** et **vgs**. Ceci est dû au fait que ces commandes arrondissent à 2 décimales près afin de fournir une sortie plus lisible. Pour spécifier une taille exacte, utilisez le nombre d'extensions physiques libres plutôt qu'un multiple d'octets déterminé par la taille du volume logique.

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
```

Alternativement, vous pouvez étendre le volume logique pour utiliser un pourcentage de l'espace libre restant dans le groupe de volumes en spécifiant l'argument **-l** de la commande **lvcreate**. Pour davantage d'informations, reportez-vous à la Section 4.4.1, « Création de volumes logiques 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 d'adminstration du stockage* fournit des instructions, étape par étape, afin de configurer un volume logique LVM à l'aide de cet utilitaire.

ANNEXE A. LE MAPPEUR DE PÉRIPHÉRIQUESUR DE PÉRIPHÉRIQUES

Le mappeur de périphériques est un pilote de noyau qui offre un framework 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 particulièrement le format des métadonnées et des groupes de volumes.

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

Les volumes logiques LVM sont activés en utilisant le mappeur de périphériques. Chaque volume logique est traduit en un périphérique mappé. Chaque segment est représenté par une ligne dans la table de mappages qui décrit le périphérique. Le gestionnaire de périphériques fournit, entre autres, un mappage linéaire, un mappage en mode stripe et un mappage d'erreurs. Deux disques peuvent être concaténés dans un seul volume logique avec une paire de mappages linéaires, un pour chaque disque. Lorsque LVM crée un volume, il crée un périphérique sous-jacent qui est un mappeur de périphériques et pouvant être questionné avec la commande **dmsetup**. Pour obtenir des informations sur le format des périphériques dans une table de mappage, voir la Section A.1, « Tables des mappages de périphériques ». Pour obtenir des informations sur l'utilisation de la commande **dmsetup** pour questionner un périphérique, voir la Section A.2, « La commande dmsetup ».

A.1. TABLES DES MAPPAGES DE PÉRIPHÉRIQUES

Un périphérique mappé est défini par une table qui spécifie comment mapper chaque gamme de secteurs logiques du périphérique à l'aide d'un mappage de table de périphérique supporté. La table d'un périphérique mappé est construite à partir d'une liste de lignes sous la forme suivante :

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

Dans la première ligne d'une table de périphérique mappé, le paramètre ***start*** doit être égal à 0. Les paramètres ***start*** + ***length*** sur une ligne doivent être égaux à ***start*** sur la ligne 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 en secteurs (512 octets).

Lorsqu'un périphérique est spécifié en tant que paramètre dans le mappeur de périphériques, il peut être référencé par le nom du périphérique dans le système de fichiers (par exemple, **/dev/hda**) ou par le numéro majeur ou mineur sous le format ***major***:***minor***. Le format major:minor (majeur:mineur) est préféré car il permet d'éviter les recherches de noms de chemins.

Ci-dessous figure un exemple 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 deux premiers paramètres de chaque ligne correspondent au bloc de démarrage du segment et à la longueur de celui-ci. Le mot-clé qui suit est la cible de mappage, et qui est, dans tous les cas de figure de cet exemple, **linear**. Le reste de la ligne est composé de paramètres pour une cible **linear**.

Les sous-sections qui suivent décrivent le format des mappages ci-dessous :

- linéaire
- striped
- miroir
- snapshot et snapshot-origin
- erreur
- zéro
- multipath
- crypt

A.1.1. La cible de mappage linéaire

Une cible de mappage linéaire mappe un éventail continu de blocs sur un autre périphérique bloc. Le format d'une cible linéaire est comme suit :

```
start length linear device offset
```

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

device

périphérique bloc, répertorié par le nom de périphérique dans le système de fichiers, ou par le numéro majeur ou mineur sous le format ***major*:*minor***

offset

offset de démarrage du mappage du périphérique

L'exemple suivant montre une cible linéaire avec un bloc de démarrage dans le périphérique virtuel de 0, une longueur de segment de 1638400, une paire de numéros major:minor de 8:2, et un offset de démarrage du périphérique de 41146992.

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

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

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

A.1.2. La cible de mappage striped

La cible de mappage striped supporte le striping sur les périphériques physiques. Elle prend le nombre de stripes et la taille du bloc striping en tant qu'arguments, suivis par une liste de paires de nom de périphériques et de secteurs. Le format d'une cible striped est comme suit :

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

Il y a un ensemble de paramètres ***device*** et ***offset*** pour chaque stripe.

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

#stripes

nombre de stripes pour le périphérique virtuel

chunk_size

nombre de secteurs écrits sur chaque stripe avant de passer à la stripe suivante ; doit être à la puissance 2, au moins aussi gros que la taille de la page du noyau.

device

périphérique bloc, peut être référencé par le nom de périphérique dans le système de fichiers ou par le numéro majeur ou mineur sous le format ***major***:***minor***.

offset

offset de démarrage du mappage du périphérique

L'exemple suivant montre une cible striped avec trois stripes et une taille de bloc de 128 :

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

0

bloc de démarrage du périphérique virtuel

73728

longueur de ce segment

striped 3 128

stripe sur trois périphériques avec une taille de bloc de 128

8:9

numéros major:minor du premier périphérique

384

démarrage du décalage du mappage sur le premier périphérique

8:8

numéros major:minor du second périphérique

384

démarrage du décalage du mappage sur le deuxième périphérique

8:7

numéros major:minor du troisième périphérique

9789824

démarrage du décalage du mappage sur le troisième périphérique

L'exemple suivant montre une cible striped pour deux stripes avec des blocs de 256 Ko, avec les paramètres de périphérique spécifiés par les noms de périphériques dans le système de fichiers plutôt que par les numéros majeur et mineur (major et minor).

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

A.1.3. La cible de mappage miroir

La cible de mappage miroir supporte le mappage d'un périphérique logique miroir. Le format d'une cible miroir est comme suit :

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

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

log_type

Les types de journaux possibles et leurs arguments sont comme suit :

core

Le miroir est local et le journal du miroir est conservé dans la mémoire centrale. Ce type de journal prend de 1 à 3 arguments :

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

disk

Le miroir est local et le journal du miroir est conservé sur disque. Ce type de journal prend de 2 à 4 arguments :

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

clustered_core

Le miroir est clusterisé et le journal du miroir est conservé dans la mémoire centrale. Ce type de journal prend de 2 à 4 arguments :

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

clustered_disk

Le miroir est clusterisé et le journal du miroir est conservé sur disque. Ce type de journal prend de 3 à 5 arguments :

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

LVM maintient un petit journal utilisé pour savoir quelles régions sont synchronisées avec le (ou les) miroir(s). L'argument *regionsize* spécifie la taille de ces régions.

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

L'argument optionnel **[no]sync** peut être utilisé pour spécifier le miroir comme étant "in-sync" ou "out-of-sync". L'argument **block_on_error** est utilisé pour dire au miroir de répondre aux erreurs plutôt que de les ignorer.

#log_args

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

logargs

les arguments de journal pour le miroir ; le nombre d'arguments de journal fournit est spécifié par le paramètre ***#log-args*** et les arguments valides du journal sont déterminés par le paramètre ***log_type***.

#devs

le nombre de branches du miroir ; un périphérique et un décalage est spécifié pour chaque branche.

device

périphérique bloc pour chaque branche de miroir, référencé par le nom de périphérique dans le système de fichiers ou par le numéro majeur ou mineur sous le format ***major***:***minor***. Un périphérique bloc et un décalage sont spécifiés pour chaque branche de miroir, comme indiqué par paramètre ***#devs***.

offset

démarrage du décalage du mappage sur le périphérique. Un périphérique bloc et un décalage est spécifié pour chaque branche de miroir, comme indiqué par le paramètre ***#devs***.

L'exemple suivant montre une cible de mappage miroir pour un miroir clusterisé avec un journal miroir sur 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

bloc de démarrage du périphérique virtuel

52428800

longueur de ce segment

mirror clustered_disk

cible de miroir avec un type de journal spécifiant que le miroir est clusterisé et que la branche du miroir est maintenue sur disque.

4

4 arguments du journal miroir suivent

253:2

numéros major:minor du périphérique du journal

1024

taille de la région que le journal miroir utilise pour savoir ce qui est synchronisé

UUID

UUID du périphérique du journal miroir pour maintenir les informations de journal à travers un cluster

`block_on_error`

le miroir devrait répondre aux erreurs

3

nombre de branches en miroir

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

numéros major:minor et offset des périphériques composant chaque branche du miroir

A.1.4. Les cibles de mappage snapshot et snapshot-origin

Lorsque vous créez le premier snapshot (ou instantané) LVM d'un volume, quatre périphériques de mappage de périphériques sont utilisés :

1. Périphérique avec un mappage **`linear`** contenant la table de mappage originale du volume source.

2. Périphérique avec un mappage **`linear`** utilisé comme périphérique de cliché instantané (de l'anglais, « copy-on-write », ou COW) pour le volume source ; pour chaque écriture, les données d'origine sont enregistrées dans le périphérique COW de chaque snapshot pour garder son contenu visible inchangé (jusqu'à ce que le périphérique COW soit rempli).

3. Un périphérique avec un mappage **snapshot** combinant #1 et #2, qui est le volume snapshot visible

4. Le volume "original" (qui utilise le numéro du périphérique utilisé par le volume source original), dont la table est remplacée par un mappage "snapshot-origin" du périphérique #1.

Un schéma de dénomination fixe est utilisé pour créer ces périphériques. Par exemple, vous pourriez utiliser les commandes suivantes pour créer un volume LVM nommé **base** et un volume snapshot nommé **snap** basé sur ce volume.

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

Ceci produit quatre périphériques, que vous pouvez observer à l'aide des 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 **snapshot-origin** est comme suit :

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

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

origin

volume de base du snapshot

snapshot-origin possède normalement un ou plusieurs snapshot qui lui sont basés dessus. Les lectures seront directement mappées sur le périphérique de sauvegarde. Pour chaque écriture, les données d'origine seront enregistrées sur le périphérique COW de chaque snapshot afin que son contenu visible reste inchangé jusqu'à ce que le périphérique COW soit rempli.

Le format de la cible **snapshot** est comme suit :

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

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

origin

volume de base du snapshot

COW-device

Périphérique sur lequel des portions de données modifiées sont stockées

P|N

P (persistant) ou N (not persistant) ; indique si le snapshot survivra au redémarrage. Pour les snapshots temporaires (N), moins de métadonnées doivent être enregistrées sur disque, celles-ci peuvent être conservées en mémoire par le noyau.

chunksize

Taille en secteurs des portions de données modifiées qui seront stockées sur le périphérique COW.

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

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

L'exemple suivant montre une cible **snapshot** avec un périphérique d'origine de 254:11 et un périphérique COW de 254:12. Ce périphérique snapshot est persistant à travers les redémarrage et la taille des portions 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 « error »

Avec une cible de mappage « error », toute opération d'E/S sur le secteur mappé échoue.

Une cible de mappage « error » peut être utilisée pour effectuer des tests. Pour voir de quelle manière se comporte un périphérique en cas d'échec, vous pouvez créer un mappage de périphérique avec un secteur défectueux au milieu du périphérique, ou vous pouvez échanger une branche d'un miroir et la remplacer par une cible erronée.

Une cible erronée peut être utilisée à la place d'un périphérique en éche afin d'éviter les délais et les nouvelles tentatives sur le périphérique. Celle-ci peut servir de cible intermédiaire tandis que vous réarrangez les métadonnées LVM pendant les pannes.

À l'exception des paramètres *start* et *length*, la cible de mappage **error** ne prend aucun paramètre supplémentaire.

L'exemple suivant présente une cible **error**.

```
0 65536 error
```

A.1.6. La cible de mappage « zero »

La cible de mappage **zero** est un périphérique bloc équivalent à **/dev/zero**. Une opération de lecture sur ce mappage retourne des blocs de zéros. Les données écrites sur ce mappage sont abandonnées, mais l'écriture est réussie. À l'exception des paramètres *start* et *length*, la cible de mappage **zero** ne prend pas de paramètres supplémentaires.

L'exemple suivant présente une cible **zero** pour un périphérique 16Tb.

```
0 65536 zero
```

A.1.7. La cible de mappage « multipath »

La cible de mappage « multipath » prend en charge le mappage d'un périphérique multipath. Le format de la cible **multipath** est comme suit :

```
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.

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

#features

Nombre de fonctionnalités multipath, suivi par celles-ci. Si ce paramètre est à zéro, alors il n'y a aucun paramètre ***feature*** et le paramètre de mappage de périphérique qui suit est ***#handlerargs***. Une seule fonctionnalité multipath est prise en charge à l'instant présent, **queue_if_no_path**. Ceci indique que le périphérique multipath est actuellement réglé pour mettre les opérations d'E/S en file d'attente s'il n'y a aucun chemin disponible.

Par exemple, si l'option **no_path_retry** dans le fichier **multipath.conf** a été réglé pour mettre les opérations d'E/S en file d'attente uniquement lorsque tous les chemins on été marqués comme étant en échec, et aprés qu'un certain nombre de tentatives d'emprunt de ces chemins n'ait été réalisé, le mappage apparaitra comme suit jusqu'à ce que toutes les vérification des vérificateurs de chemins aient échoué.

```
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
```

Une fois que les vérificateurs de chemins auront échoué à vérifier le nombre spécifié de vérifications, le mappage apparaitra comme 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

Nombre d'arguments du gestionnaire du matériel, suivis par ceux-ci. Un gestionnaire de matériel spécifie un module qui sera utilisé pour effectuer des actions spécifiques au matériel lorsque vous basculez d'un groupe de chemins à un autre ou lors de la gestion d'erreurs d'E/S. S'il est réglé sur 0, alors le paramètre qui suit sera ***#pathgroups***.

#pathgroups

Nombre de groupes de chemins. Un groupe de chemins est l'ensemble des chemins sur lesquels un périphérique mutlipath équilibre les charges. Il existe un ensemble de paramètres ***pathgroupargs*** pour chaque groupe de chemins.

pathgroup

Le prochain groupe de chemins à essayer.

pathgroupsargs

Chaque groupe de chemins est composé des arguments suivants :

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

Il y a un ensemble d'arguments de chemins pour chaque chemin du groupe de chemins.

pathselector

Spécifie l'algorithme à utiliser pour déterminer quel chemin dans ce groupe de chemins doit être utilisé pour la prochaine opération d'E/S.

#selectorargs

Nombre d'arguments du sélecteur d'arguments qui suit cet argument dans le mappage multipath. Actuellement, la valeur de cet argument est toujours 0.

#paths

Nombre de chemins dans ce groupe de chemins.

#pathargs

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

device

Numéro du périphérique bloc du chemin, référencé par les numéros majeur et mineur sous le format ***major***:***minor***

ioreqs

Nombre de requêtes d'E/S à acheminer vers ce chemin avant de basculer sur le chemin suivant du groupe actuel.

Figure A.1, « Cible de mappage multipath » montre le format d'une cible multipath avec deux groupes de chemins

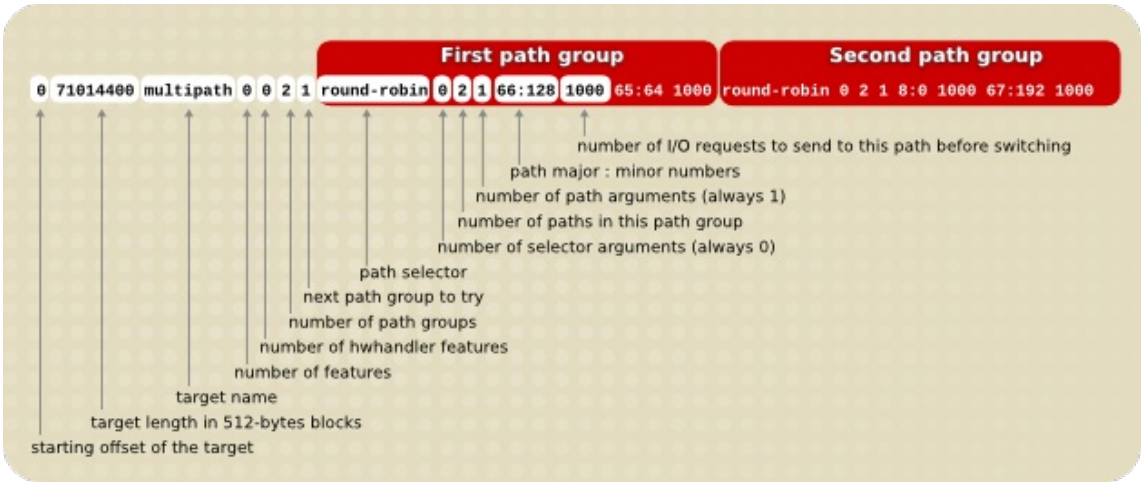


Figure A.1. Cible de mappage multipath

L'exemple suivant montre une définition de cible de basculement pure pour le même périphérique multipath. Dans cette cible, il y a quatre groupes de chemins, avec seulement un chemin ouvert par groupe de chemins, et ce de manière à ce que le périphérique multipath 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 une définition de cible en étalement complet (multibus) pour le même périphérique multipath. Dans cette cible, il n'y a qu'un seul groupe de chemins, qui inclut tous les chemins. Dans cette installation, multipath étale les charges de manière uniformément sur tous les chemins.

```
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 de plus amples informations sur le « multipathing », voir le document *Using Device Mapper Multipath*.

A.1.8. La cible de mappage « crypt »

La cible **crypt** chiffre les données passant à travers le périphérique spécifié. Elle utilise l'API du noyau Cypto.

Le format pour la cible **crypt** est comme suit :

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

start

bloc de démarrage du périphérique virtuel

length

longueur de ce segment

cipher

Un cipher est composé de ***cipher[-chainmode]-ivmode[:iv options]***.

cipher

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

chainmode

Veuillez toujours utiliser **cbc**. Ne pas utiliser **ebc** ; car il n'utilise pas un vecteur initial (IV).

ivmode[:iv options]

IV est un vecteur initial utilisé afin de varier le chiffrage. Le mode IV est **plain** ou **essiv:hash**. Un ***ivmode*** de **-plain** utilise le numéro du secteur (plus un décalage d'IV) comme IV. Un ***ivmode*** de **-essiv** est une amélioration évitant une faiblesse face aux tatouages numériques (ou watermarking)

key

Clé de chiffrement, fournie en hex

IV-offset

Décalage du vecteur initial (IV)

device

périphérique bloc, répertorié par le nom de périphérique dans le système de fichiers, ou par le numéro majeur ou mineur sous le format ***major*:*minor***

offset

offset de démarrage du mappage du périphérique

Ci-dessous figure un exemple de 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, les options **info**, **ls**, **status**, et **deps** de la commande **dmsetup** se révéleront utiles, comme les sous-sections suivantes le décrivent :

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 dmsetup info

La commande **dmsetup info *device*** fournit un résumé des informations sur les périphériques mappeurs de périphériques. Si vous ne spécifiez pas un nom de périphérique, la sortie sera des informations sur tous les périphériques mappeurs de périphériques actuellement configurés. Si vous

spécifiez un périphérique, alors cette commande se limitera aux informations de ce périphérique uniquement.

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

Nom

Nom du périphérique. Un périphérique LVM est exprimé comme le nom du groupe de volumes et le nom du volume logique séparé par un trait d'union. Un trait d'union dans le nom d'origine est traduit en deux traits d'union.

État

Les états possibles des périphériques sont **SUSPENDED** (Suspendu), **ACTIVE** (actif), et **READ-ONLY** (lecture seule). La commande **dmsetup suspend** définit l'état du périphérique sur **SUSPENDED**. Lorsqu'un périphérique est suspendu, toutes les opérations d'E/S de ce périphérique s'arrêtent. La commande **dmsetup resume** restaure l'état du périphérique sur **ACTIVE**.

Lecture anticipée

Nombre de blocs de données que le système lit à l'avance pour tout fichier ouvert sur lequel des opérations de lecture sont en cours. Par défaut, le noyau choisit une valeur correcte automatiquement. Vous pouvez changer cette valeur avec l'option **--readahead** de la commande **dmsetup**.

Tables présentes

Les états possibles pour cette catégorie sont **LIVE** et **INACTIVE**. L'état **INACTIVE** (inactif) indique qu'une table a été chargée et sera échangée lorsque la commande **dmsetup resume** restaurera l'état du périphérique sur **ACTIVE**, moment auquel létat de la table deviendra **LIVE**. Pour plus d'informations, voir la page man **dmsetup**.

Compte d'ouvertures

Le compte d'ouvertures de références indique combien de fois le périphérique a été ouvert. La commande **mount** ouvre un périphérique.

Nombre d'événements

Nombre actuel d'événements reçus. La saisie de la commande **dmsetup wait *n*** permet à l'utilisateur d'attendre l'événement « n », bloquant ainsi l'appel jusqu'à ce que cet événement soit reçu.

Majeur, mineur

Numéro majeur et mineur du périphérique

Nombre de cibles

Nombre de fragments formant un périphérique. Par exemple, un périphérique linéaire couvrant 3 disques devrait avoir 3 cibles. Un périphérique linéaire composé du début et de la fin d'un disque, mais pas du milieu n'aurait que 2 cibles.

UUID

UUID du périphérique.

L'exemple suivant montre une partie de la sortie de 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 répertorier les noms de périphériques des périphériques mappés avec la commande **dmsetup ls**. Vous pouvez répertorier les périphériques qui possède au moins une cible du type spécifié avec la commande **dmsetup ls --target *target_type***. Pour voir les autres options de **dmsetup ls**, reportez-vous à la page man de **dmsetup**.

L'exemple suivant montre la commande pour répertorier les noms de périphérique des périphériques mappés 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 répertorier les noms de périphérique des mappages miroirs 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)
```

Les configurations LVM qui sont empilées sur des périphériques multipath ou sur d'autres périphériques mappeurs de périphériques peuvent être assez complexes à arranger. La commande **dmsetup ls** propose une option **--tree** qui affiche les dépendances entre périphériques sous la forme d'une arborescence, comme illustré dans l'exemple suivant.

```
# dmsetup ls --tree
vgtest-lvmir (253:13)
 ├─vgtest-lvmir_mimage_1 (253:12)
 │  └─mpathep1 (253:8)
 │     └─mpathe (253:5)
 │        ├─ (8:112)
 │        └─ (8:64)
 ├─vgtest-lvmir_mimage_0 (253:11)
 │  └─mpathcp1 (253:3)
 │     └─mpathc (253:2)
 │        ├─ (8:32)
 │        └─ (8:16)
 └─vgtest-lvmir_mlog (253:4)
    └─mpathfp1 (253:10)
       └─mpathf (253:6)
          ├─ (8:128)
          └─ (8:80)
```

A.2.3. La commande dmsetup status

La commande **dmsetup status *device*** fournit des informations sur le statut de chaque cible d'un périphérique spécifié. Si vous ne spécifiez pas de nom de périphérique, la sortie contiendra les informations sur tous les périphériques mappeurs de périphériques actuellement configurés.

L'exemple suivant montre la commande pour répertorier le statut des cibles dans tous les 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 des paires (majeures, mineures) de périphériques référencées par la table de mappage pour le périphérique spécifié. Si vous ne spécifiez pas de nom de périphérique, la sortie contiendra des informations sur tous les périphériques mappeurs de périphériques actuellement configurés.

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

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

A.3. PRISE EN CHARGE DU MAPPEUR DE PÉRIPHÉRIQUES POUR LE GESTIONNAIRE DE PÉRIPHÉRIQUES UDEV

Le principal rôle du gestionnaire de périphériques **udev** est de permettre de paramétrer des noeuds de manière dynamique dans le répertoire **/dev**. La création de ces noeuds est réalisée par l'application des règles **udev** dans l'espace utilisateur. Ces règles sont traitées sur les événements **udev** envoyés directement depuis le noyau, et résultent de l'ajout, de la suppression, ou de la modification de périphériques particuliers, ce qui fournit un mécanisme pratique et central pour le support des connexions à chaud.

À part créer des noeuds, le gestionnaire de périphériques **udev** est capable de créer tout lien symbolique avec leur propres noms, offrant aux utilisateurs la liberté de choisir leur structure de répertoire et de dénomination personnalisée dans le répertoire **/dev** si nécessaire

Chaque événement **udev** contient des informations de base sur le périphérique traité, comme son nom, le sous-système auquel il appartient, son type, son numéro majeur et mineur utilisé, et le type de l'événement. Avec ces informations et la possibilité d'accéder à toutes les informations trouvées dans le répertoire **/sys**, qui est aussi accessible avec les règles **udev**, les utilisateurs sont en mesure d'utiliser de simples filtres basés sur ces informations et d'exécuter les règles à condition de se baser sur ces informations.

Le gestionnaire de périphériques **udev** offre aussi une manière centralisée de paramétrer les permissions des noeuds. Un utilisateur peut facilement ajouter un ensemble de règles personnalisées pour définir les permissions pour tout périphérique spécifié par toute information disponible pendant le traitement de l'événement.

Il est aussi possible d'ajouter des hooks de programmes dans les règles **udev** directement. Le gestionnaire de périphériques **udev** peut appeler tous ces programmes afin de fournir un traitement supplémentaire nécessaire à la gestion de l'événement. Le programme peut aussi exporter des variables d'environnement en tant que résultat de ce traitement. Tous les résultats donnés peuvent être utilisés dans les règles en tant que source d'informations supplémentaire.

Tout logiciel utilisant la bibliothèque **udev** est en mesure de recevoir et de traiter des événements **udev** avec toutes les informations disponibles. Ainsi, le traitement n'est pas limité au démon **udev** uniquement.

A.3.1. Intégration udev avec le mappeur de périphériques

Dans RHEL 6, le mappeur de périphériques fournit un support direct pour l'intégration **udev**. Ceci synchronise le mappeur de périphériques avec tous les traitements **udev** concernant les périphériques du mappeur de périphériques, y compris les périphériques LVM. La synchronisation est nécessaire car l'application de la règle dans le démon **udev** est une forme de traitement parallèle avec le programme qui est la source des modifications du périphérique (comme **dmsetup** et LVM). Sans ce support, un

problème commun, résultant d'un changement d'événement, survenait lorsqu'un utilisateur tentait de supprimer un périphérique qui était encore ouvert et traité par les règles **udev** ; ceci était particulièrement commun lorsqu'il y avait de très courts laps de temps entre les changements pour ce périphérique.

La version de RHEL 6 fournit des règles **udev** officiellement prises en charge pour les périphériques de Device Mapper en général ainsi que pour LVM. La Tableau A.1, « Règles udev pour les périphériques du mappeur de périphériques » résume ces règles, qui sont installées dans **/lib/udev/rules.d**.

Tableau A.1. Règles udev pour les périphériques du mappeur de périphériques

Nom du fichier	Description
10-dm.rules	Contient des règles basiques/générales du mappeur de périphériques et crée des symlinks dans **/dev/mapper** avec une cible **/dev/dm-*N***, où N est un numéro assigné dynamiquement à un périphérique par le noyau (**/dev/dm-*N*** est un noeud). REMARQUE : les noeuds **/dev/dm-*N*** ne doivent *jamais* être utilisés dans des scripts pour accéder au périphérique puisque le numéro N est assigné dynamiquement et des changements de la séquence des périphériques how sont activés. Ainsi, les véritables noms dans le répertoire **/dev/mapper** devraient être utilisés. Cette structure est ainsi afin de prendre en charge les prérequis **udev** dictant comment les noeuds/symlinks doivent être créés.
11-dm-lvm.rules	Contient des règles appliquées pour les périphériques LVM et crée les symlinks pour les volumes logiques du groupe de volumes. Les symlinks sont créés dans le répertoire **/dev/*vgname*** avec une cible **/dev/dm-*N***. REMARQUE : Pour rester consistant avec le standard de dénomination de toutes les futures des sous-systèmes du mappeur de périphériques, les règles udev devraient être sous le format **11-dm-*nom_du_sous_système*.rules**. Tous les utilisateurs **libdevmapper** fournissant des règles **udev** devraient aussi suivre ce standard.
13-dm-disk.rules	Contient des règles à appliquer à tous les périphériques du mappeur de périphériques en général et crée des symlinks dans les répertoires **/dev/disk/by-id**, **/dev/disk/by-uuid** et **/dev/disk/by-uuid**.
95-dm-notify.rules	Contient la règle qui notifie du processus d'attente à l'aide de **libdevmapper** (tout comme LVM et **dmsetup**). La notification est effectuée une fois que toutes les règles précédentes sont appliquées, afin de s'assurer que tous les traitements **udev** sont terminés. Le processus notifié est ensuite repris.

Vous pouvez ajouter des règles de permissions personnalisées à l'aide du fichier **12-dm-**

permissions.rules. Ce fichier *n'est pas* installé dans le répertoire **/lib/udev/rules** ; il peut être trouvé dans le répertoire **/usr/share/doc/device-mapper-*version***. Le fichier **12-dm-permissions.rules** est un modèle contenant des indices pour paramétrer les permissions basé sur des règles de correspondance données comme exemple. Le fichier contient des exemples pour certaines situations communes. Vous pouvez modifier ce fichier et le placer manuellement dans le répertoire **/etc/udev/rules.d** où il survivra aux mises à jour. Ainsi, les paramètres subsisteront.

Ces règles paramètrent toutes les variables de base qui peuvent être utilisées par toutes les autres règles pendant le traitement des événements.

Les variables suivantes sont paramétrées dans 10-dm.rules :

- **DM_NAME** : nom du périphérique du mappeur de périphériques
- **DM_UUID** : UUID du périphérique du mappeur de périphériques
- **DM_SUSPENDED** : état suspendu du périphérique du mappeur de périphériques
- **DM_UDEV_RULES_VSN** : version des règles **udev** (principalement pour que toutes les autres règles puissent vérifier que les variables mentionnées au préalable sont directement paramétrées par les règles du mappeur de périphériques)

Les variables suivantes sont paramétrées dans **11-dm-lvm.rules** :

- **DM_LV_NAME** : nom du volume logique
- **DM_VG_NAME** : nom du groupe de volumes
- **DM_LV_LAYER** : nom de la couche LVM

Toutes ces variables peuvent être utilisées dans le fichier **12-dm-permissions.rules** pour définir une permission pour des périphériques spécifiques du mappeur de périphériques, comme documenté dans le fichier **12-dm-permissions.rules**.

A.3.2. Commandes et interfaces qui supportent udev

La Tableau A.2, « Commandes dmsetup pour supporter udev » résume les commandes **dmsetup** qui supportent l'intégration **udev**.

Tableau A.2. Commandes dmsetup pour supporter udev

Commande	Description
dmsetup udevcomplete	Utilisé pour notifer que udev a terminé le traitement des règles et déverrouille le procesus d'attente (appelé depuis les règles **udev** dans **95-dm-notify.rules**)
dmsetup udevcomplete_all	Utilisé à des fins de débogages, pour déverrouiller manuellement tous les processus d'attente
dmsetup udevcookies	Utilisé à des fins de débogages, pour afficher tous les cookies existants (sémaphores à l'échelle du système)

Commande	Description
dmsetup udevcreatecookie	Utilisé pour créer un cookie (sémaphore) manuellement. Ceci est utile pour exécuter plus de processeurs sous une ressource de synchronisation.
dmsetup udevreleasecookie	Utilisé pour attendre tous les processus **udev** qui concernent tous les processus placés sous cet unique cookie de synchronisation.

Les options **dmsetup** qui supportent l'intégration **udev** sont comme suit.

--udevcookie

Doit être défini pour tous les processus dmsetup que vous souhaitez ajouter à une transaction udev. Utilisé en conjonction avec **udevcreatecookie** et **udevreleasecookie**:

```
COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command --udevcookie $COOKIE ....
  dmsetup command --udevcookie $COOKIE ....
  ....
  dmsetup command --udevcookie $COOKIE ....
dmsetup udevreleasecookie --udevcookie $COOKIE
```

Outre l'utilisation de l'option **--udevcookie**, vous pouvez aussi exporter la variable dans un environnement du processus :

```
export DM_UDEV_COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command ...
  dmsetup command ...
  ...
  dmsetup command ...
```

--noudevrules

Désactive les règles udev. Les noeuds/symlinks seront créés par **libdevmapper** (qui est l'ancienne manière). Cette option sert à des fins de débogage, au cas où **udev** ne fonctionne pas correctement.

--noudevsync

Déssactive la synchronisation **udev**. Ceci peut aussi servir au débogage.

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

Les commande LVM supportent les options suivantes, qui supportent elles-mêmes l'intégration **udev** :

- **--noudevrules** : tout comme pour la commande **dmsetup**, désactive les règles **udev**.
- **--noudevsync** : tout comme pour la commande **dmsetup**, désactive la synchronisation **udev**.

Le fichier **lvm.conf** inclut les options suivantes qui elles-mêmes supportent l'intégration **udev** :

- **udev_rules** : active/désactive **udev_rules** pour toutes les commandes LVM2 à l'échelle du système
- **udev_sync** : active/désactive la synchronisation **udev** pour toutes les commandes LVM à l'échelle du système

Pour obtenir davantage d'informations sur les options du fichier **lvm.conf**, voir les commandes de la ligne dans le fichier **lvm.conf**.

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`**.

Pour obtenir des informations à propos du chargement des fichiers de configuration supplémentaires, reportez-vous à la 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

Pour chaque balise hôte, un fichier de configuration supplémentaire est lu s'il existe : **`lvm_hosttag.conf`**. Si ce fichier définit de nouvelles balises, d'autres fichiers de configuration seront alors ajoutés dans la liste des fichiers à lire. Pour obtenir des informations à propos des balises hôtes, reportez-vous à la 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

Ci-dessous figure un exemple du fichier de configuration **lvm.conf**. Votre fichier de configuration peut être légèrement différent de celui-ci.

```
# Ceci est un exemple de fichier de configuration du système LVM2.
# Il contient les paramètres par défaut qui seraient utilisés s'il n'y
avait pas
# de fichier /etc/lvm/lvm.conf.
#
# Reportez-vous à 'man lvm.conf' pour obtenir plus d'informations, y
compris sur la structure du fichier.
#
# Pour inclure ce fichier dans un autre répertoire et remplacer /etc/lvm,
définissez
# la variable d'environnement LVM_SYSTEM_DIR avant d'exécuter les outils.


# Cette section vous permet de configurer quels périphériques blocs
devraient être utilisés
# par le système LVM.
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 md device's stripe-width.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1
```

```
    # Default alignment of the start of a data area in MB.  If set to 0,
    # a value of 64KB will be used.  Set to 1 for 1MiB, 2 for 2MiB, etc.
    # default_data_alignment = 1

    # By default, the start of a PV's data area will be a multiple of
    # the 'minimum_io_size' or 'optimal_io_size' exposed in sysfs.
    # - minimum_io_size - the smallest request the device can perform
    #   w/o incurring a read-modify-write penalty (e.g. MD's chunk size)
    # - optimal_io_size - the device's preferred unit of receiving I/O
    #   (e.g. MD's stripe width)
    # minimum_io_size is used if optimal_io_size is undefined (0).
    # If md_chunk_alignment is enabled, that detects the optimal_io_size.
    # This setting takes precedence over md_chunk_alignment.
    # 1 enables; 0 disables.
    data_alignment_detection = 1

    # Alignment (in KB) of start of data area when creating a new PV.
    # md_chunk_alignment and data_alignment_detection are disabled if set.
    # Set to 0 for the default alignment (see: data_alignment_default)
    # or page size, if larger.
    data_alignment = 0

    # By default, the start of the PV's aligned data area will be shifted
by
    # the 'alignment_offset' exposed in sysfs.  This offset is often 0 but
    # may be non-zero; e.g.: certain 4KB sector drives that compensate for
    # windows partitioning will have an alignment_offset of 3584 bytes
    # (sector 7 is the lowest aligned logical block, the 4KB sectors start
    # at LBA -1, and consequently sector 63 is aligned on a 4KB boundary).
    # But note that pvcreate --dataalignmentoffset will skip this
detection.
    # 1 enables; 0 disables.
    data_alignment_offset_detection = 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

    # During each LVM operation errors received from each device are
counted.
    # If the counter of a particular device exceeds the limit set here, no
    # further I/O is sent to that device for the remainder of the
respective
    # operation. Setting the parameter to 0 disables the counters
altogether.
    disable_after_error_count = 0

    # Allow use of pvcreate --uuid without requiring --restorefile.
    require_restorefile_with_uuid = 1
}
```

```
# This section allows you to configure the way in which LVM selects
# free space for its Logical Volumes.
#allocation {
#    When searching for free space to extend an LV, the "cling"
#    allocation policy will choose space on the same PVs as the last
#    segment of the existing LV.  If there is insufficient space and a
#    list of tags is defined here, it will check whether any of them are
#    attached to the PVs concerned and then seek to match those PV tags
#    between existing extents and new extents.
#    Use the special tag "@*" as a wildcard to match any PV tag.
#
#    Example: LVs are mirrored between two sites within a single VG.
#    PVs are tagged with either @site1 or @site2 to indicate where
#    they are situated.
#
#    cling_tag_list = [ "@site1", "@site2" ]
#    cling_tag_list = [ "@*" ]
#}

# 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 {
```

```
    # 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"

    # Since version 2.02.54, the tools distinguish between powers of
    # 1024 bytes (e.g. KiB, MiB, GiB) and powers of 1000 bytes (e.g.
    # KB, MB, GB).
    # If you have scripts that depend on the old behaviour, set this to 0
    # temporarily until you update them.
    si_unit_consistency = 1

    # 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 "lvm2".
    # format = "lvm2"

    # 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.
    # Type 4 uses read-only locking which forbids any operations that
```

```
might
    # change metadata.
    locking_type = 1

    # Set to 0 to fail when a lock request cannot be satisfied
immediately.
    wait_for_locks = 1

    # 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"

    # Whenever there are competing read-only and read-write access
requests for
    # a volume group's metadata, instead of always granting the read-only
    # requests immediately, delay them to allow the read-write requests to
be
    # serviced.  Without this setting, write access may be stalled by a
high
    # volume of read-only requests.
    # NB. This option only affects locking_type = 1 viz. local file-based
    # locking.
    prioritise_write_locks = 1

    # 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"

    # Treat any internal errors as fatal errors, aborting the process that
```

```
    # encountered the internal error. Please only enable for debugging.
    abort_on_internal_errors = 0

    # If set to 1, no operations that change on-disk metadata will be
permitted.
    # Additionally, read-only commands that encounter metadata in need of
repair
    # will still be allowed to proceed exactly as if the repair had been
    # performed (except for the unchanged vg_seqno).
    # Inappropriate use could mess up your system, so seek advice first!
    metadata_read_only = 0
}

activation {
    # Set to 0 to disable udev synchronisation (if compiled into the
binaries).
    # Processes will not wait for notification from udev.
    # They will continue irrespective of any possible udev processing
    # in the background.  You should only use this if udev is not running
    # or has rules that ignore the devices LVM2 creates.
    # The command line argument --nodevsync takes precedence over this
setting.
    # If set to 1 when udev is not running, and there are LVM2 processes
    # waiting for udev, run 'dmsetup udevcomplete_all' manually to wake
them up.
    udev_sync = 1

    # Set to 0 to disable the udev rules installed by LVM2 (if built with
    # --enable-udev_rules). LVM2 will then manage the /dev nodes and
symlinks
    # for active logical volumes directly itself.
    # N.B. Manual intervention may be required if this setting is changed
    # while any logical volumes are active.
    udev_rules = 1

    # 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. This applies to automatic repairs (when the mirror is
being
    # monitored by dmeventd) and to manual lvconvert --repair when
    # --use-policies is given.
    #
    # "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.
    #
    # "allocate_anywhere" - Not yet implemented. Useful to place the log
device
    #            temporarily on same physical volume as one of the mirror
    #            images. This policy is not recommended for mirror devices
    #            since 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_image_fault_policy = "remove"

    # 'snapshot_autoextend_threshold' and 'snapshot_autoextend_percent'
define
    # how to handle automatic snapshot extension. The former defines when
the
    # snapshot should be extended: when its space usage exceeds this many
    # percent. The latter defines how much extra space should be allocated
for
    # the snapshot, in percent of its current size.
    #
    # For example, if you set snapshot_autoextend_threshold to 70 and
    # snapshot_autoextend_percent to 20, whenever a snapshot exceeds 70%
usage,
    # it will be extended by another 20%. For a 1G snapshot, using up 700M
will
    # trigger a resize to 1.2G. When the usage exceeds 840M, the snapshot
will
    # be extended to 1.44G, and so on.
    #
    # Setting snapshot_autoextend_threshold to 100 disables automatic
    # extensions. The minimum value is 50 (A setting below 50 will be
treated
    # as 50).

    snapshot_autoextend_threshold = 100
    snapshot_autoextend_percent = 20

    # While activating devices, I/O to devices being (re)configured is
    # suspended, and as a precaution against deadlocks, LVM2 needs to pin
    # any memory it is using so it is not paged out.  Groups of pages that
    # are known not to be accessed during activation need not be pinned
    # into memory.  Each string listed in this setting is compared against
    # each line in /proc/self/maps, and the pages corresponding to any
    # lines that match are not pinned.  On some systems locale-archive was
    # found to make up over 80% of the memory used by the process.
    # mlock_filter = [ "locale/locale-archive", "gconv/gconv-
modules.cache" ]

    # Set to 1 to revert to the default behaviour prior to version 2.02.62
    # which used mlockall() to pin the whole process's memory while
activating
    # devices.
    use_mlockall = 0

    # Monitoring is enabled by default when activating logical volumes.
    # Set to 0 to disable monitoring or use the --ignoremonitoring option.
    monitoring = 1

    # When pvmove or lvconvert must wait for the kernel to finish
    # synchronising or merging data, they check and report progress
    # at intervals of this number of seconds.  The default is 15 seconds.
    # If this is set to 0 and there is only one thing to wait for, there
    # are no progress reports, but the process is awoken immediately the
```

```
    # operation is complete.
    polling_interval = 15
}


####################
# 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

    # Default number of copies of metadata to maintain for each VG.
    # If set to a non-zero value, LVM automatically chooses which of
    # the available metadata areas to use to achieve the requested
    # number of copies of the VG metadata.  If you set a value larger
    # than the the total number of metadata areas available then
    # metadata is stored in them all.
    # The default value of 0 ("unmanaged") disables this automatic
    # management and allows you to control which metadata areas
    # are used at the individual PV level using 'pvchange
    # --metadataignore y/n'.

    # vgmetadatacopies = 0

    # 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
    # the snapshot exceeds 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot is filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"

    # Full path of the dmeventd binary.
    #
    # executable = "/sbin/dmeventd"
}
```

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.

À partir de la version 6.1 de Red Hat Enterprise Linux, les balises LVM sont des chaines allant jusqu'à 1024 caractères (dans les versions précédentes la longueur limite était de 128 caractères). Les balises LVM ne peuvent pas commencer par un trait d'union.

Une balise valide consiste uniquement d'une gamme limitée de caractères. Dans Red Hat Enterprise Linux 6.0, les caractères autorisés sont [A-Za-z0-9_+.-]. À partir de la version 6.1 de Red Hat Enterprise Linux, la liste des caractères autorisés a été étendue et les balises peuvent maintenant contenir les caractères "/", "=", "!", ":", "#", and "&".

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**.

À partir de la version 6.1 de Red Hat Enterprise Linux, vous pouvez spécifier de multiples arguments **--addtag** et **--deltag** dans une commande **pvchange**, **vgchange**, ou **lvchange** unique. Par exemple, la commande suivante supprime les balises **T9** et **T10** et ajoute les balises **T13** et **T14** au groupe de volumes **grant**.

```
vgchange --deltag T9 --deltag T10 --addtag T13 --addtag T14 grant
```

C.2. LES BALISES HÔTES

Dans une configuration en clusters, vous pouvez définir des balises hôtes dans les fichiers de configuration. Si vous paramétrez **hosttags = 1** dans la section **tags**, une balise hôte est automatiquement définie en utilisant le nom d'hôte de la machine. Ceci vous permet d'utiliser un fichier

de configuration commun qui peut être répliqué sur toutes vos machines afin qu'elles comportent toutes une copie identique du fichier, mais le comportement peut être différent entre les machines en fonction du nom d'hôte.

Pour obtenir des informations à propos des fichiers de configuration, reportez-vous à l'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" ] }
```

La correspondance spéciale "@*" réussit seulement si une des balises de métadonnées correspond à une des balises hôtes sur cette 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 DES RÉVISIONS

Version 1-4.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Version 1-4 **2012-07-18** **Anthony Towns**

Rebuild for Publican 3.0

Version 2.0-1 **Thu May 19 2011** **Steven Levine**

Publication initiale de Red Hat Enterprise Linux 6.1

Résout : #694619
Documente la nouvelle politique d'allocation **`cling`** lors de l'extension d'un volume logique.

Résout : #682649
Ajoute un avertissement concernant l'exécution de multiples commandes de création de miroir en succession sur des volumes clusterisés.

Résout : #674100
Ajoute un exemple de la sortie de la commande **`dmsetup ls --tree`**.

Résout : #694607
Documente la prise en charge pour inclure des multiples arguments --addtag et --deltag sur une seule ligne de commande.

Résout : #694604
Documente la prise en charge de la liste étendue des caractères dans les balises.

Résout : #694611
Documente la prise en charge des miroirs en stripes.

Résout : #694616
Documente la prise en charge des instantanés de volumes logiques.

Résout : #694618
Documente la prise en charge des instantanés de volumes clusterisés activés de manière exclusive.

Résout : #682648
Documente que lorsque la branche d'un miroir est ré-allouée, le journal du miroir peut aussi être déplacé.

Résout : #661530
Met à jour l'exemple **`cluster.conf`** de manière à documenter les fonctionnalités actuelles.

Résout : #642400
Ajoute une remarque sur la gestion de journaux des clusters maintenue par le noeud du cluster avec l'ID du cluster le plus bas.

Résout : #663462
Supprime les références obsolètes au moniteur de la machine virtuelle Xen.

Version 1.0-1 **Wed Nov 10 2010** **Steven Levine**

Publication initiale de Red Hat Enterprise Linux 6

INDEX

Symboles

/lib/udev/rules.d directory, Intégration udev avec le mappeur de périphériques

A

activation des groupes de volumes, Activation et désactivation des groupes de volumes
  noeud local uniquement, Activation et désactivation des groupes de volumes
  noeuds individuels, Activation et désactivation des groupes de volumes

activation des volumes logiques
  noeuds individuels, Activation des volumes logiques sur les noeuds individuels d'un cluster

affichage
  groupe de volumes, Affichage des groupes de volumes, La commande vgs
  trier la sortie, Trier des rapports LVM
  volume logique, Affichage de volumes logiques, La commande lvs
  volumes physiques, Affichage des volumes physiques, La commande pvs

affichage de l'aide, Utilisation des commandes CLI
affichage de la page man, Utilisation des commandes CLI
allocation
  empêcher, Empêcher l'allocation sur un volume physique
  politique, Création de groupes de volumes

analyse
  périphériques blocs, Recherche de périphériques blocs

analyse des périphériques, filtres, Contrôler l'analyse des périphériques LVM avec les filtres
aperçu
  fonctionnalités nouvelles et modifiées, Nouvelles fonctionnalités et fonctionnalités modifiées

C

CLVM
  définition, Le gestionnaire de volumes logiques LVM en cluster (CLVM)

clvmd démon, Le gestionnaire de volumes logiques LVM en cluster (CLVM)
commande lvmdiskscan, Recherche de périphériques blocs
commande pvscan, Affichage des volumes physiques
commande vgcreate , Création de groupes de volumes, Création de groupes de volumes dans un cluster
commentaires
  Informations de contact pour ce manuel, Vos commentaires sont bienvenus !

création
de volume logique, exemple, Création d'un volume logique LVM sur trois disques
de volumes LVM dans un cluster, Création de volumes LVM dans un cluster
groupe de volumes, clusterisé, Création de groupes de volumes dans un cluster
groupes de volumes, Création de groupes de volumes
volume logique, Création de volumes logiques linéaires
volume logique en mode stripe, exemple, Création d'un volume logique en mode stripe
volumes physiques, Création de volumes physiques

création de volumes LVM
aperçu, Aperçu de la création d'un volume logique

D

déactivation des groupes de volumes
exclusivement sur un seul noeud, Activation et désactivation des groupes de volumes

déplacement de données en ligne, Déplacement des données en ligne
déplacement de données, en ligne, Déplacement des données en ligne
désactivation des groupes de volumes, Activation et désactivation des groupes de volumes
noeud local uniquement, Activation et désactivation des groupes de volumes

E

environnement en clusters, Le gestionnaire de volumes logiques LVM en cluster (CLVM), Création de volumes LVM dans un cluster
étendue physique
empêcher l'allocation, Empêcher l'allocation sur un volume physique

exemples de configuration, Exemples de configuration LVM
extension
allocation, Création de groupes de volumes
définition, Les groupes de volumes, Création de groupes de volumes

F

Fichier archive, Sauvegarde des métadonnées d'un groupe de volumes
fichier cache
construction, Analyse des disques pour les groupes de volumes afin de construire le fichier de cache

Fichier d'archivage, Sauvegarde d'un volume logique
Fichier sauvegarde, Sauvegarde des métadonnées d'un groupe de volumes
filtres, Contrôler l'analyse des périphériques LVM avec les filtres
filtres d'analyse de périphériques, Contrôler l'analyse des périphériques LVM avec les filtres

fonctionnalités nouvelles et modifiées, Nouvelles fonctionnalités et fonctionnalités modifiées
format de rapport, périphériques LVM, Rapport personnalisé pour LVM

G

groupe de volumes
activation, Activation et désactivation des groupes de volumes
administration générale, Administration d'un groupe de volumes
affichage, Affichage des groupes de volumes, Rapport personnalisé pour LVM, La commande vgs
agrandissement, Ajout de volumes physiques à un groupe de volumes
arguments d'affichage vgs, La commande vgs
combiner, Combinaison de groupes de volumes
création, Création de groupes de volumes
création dans un cluster, Création de groupes de volumes dans un cluster
définition, Les groupes de volumes
déplacements entre systèmes, Déplacer un groupe de volumes sur un autre système
désactivation, Activation et désactivation des groupes de volumes
extension, Ajout de volumes physiques à un groupe de volumes
fusionner, Combinaison de groupes de volumes
modification des paramètres, Changement des paramètres du groupe de volumes
partage
exemple de procédure, Partager un groupe de volumes
réduction, Suppression de volumes physiques à partir d'un groupe de volumes
renommer, Renommer un groupe de volumes
séparation, Partager un groupe de volumes
suppression, Suppression de groupes de volumes

I

initialisation
partitions, Initialisation des volumes physiques
volume physique, Initialisation des volumes physiques

instantané de volume logique
création, Création d'instantanés de volumes

J

journalisation, Journalisation

L

La commande lvchange, Changement des paramètres d'un groupe de volumes logiques
La commande lvconvert, Changement de la configuration du volume en miroir

La commande lvcreate, Création de volumes logiques linéaires
La commande lvdisplay, Affichage de volumes logiques
La commande lvextend, Augmentez la taille des volumes logiques
La commande lvreduce, Redimensionnement des volumes logiques, Réduire la taille des volumes logiques
La commande lvremove, Suppression de volumes logiques
La commande lvrename, Renommer les volumes logiques
La commande lvs, Rapport personnalisé pour LVM, La commande lvs
arguments d'affichage, La commande lvs

La commande lvscan, Affichage de volumes logiques
La commande pvdisplay, Affichage des volumes physiques
La commande pvmove, Déplacement des données en ligne
La commande pvremove, Suppression de volumes physiques
La commande pvresize, Redimensionnement de volumes physiques
La commande pvs, Rapport personnalisé pour LVM
arguments d'affichage, La commande pvs

La commande vgcfbackup, Sauvegarde des métadonnées d'un groupe de volumes
La commande vgcfrestore, Sauvegarde des métadonnées d'un groupe de volumes
La commande vgchange, Changement des paramètres du groupe de volumes
La commande vgdisplay, Affichage des groupes de volumes
La commande vgexport, Déplacer un groupe de volumes sur un autre système
La commande vgextend, Ajout de volumes physiques à un groupe de volumes
La commande vgimport, Déplacer un groupe de volumes sur un autre système
La commande vgmerge, Combinaison de groupes de volumes
La commande vgmknodes, Recréation du répertoire d'un groupe de volumes
La commande vgreduce, Suppression de volumes physiques à partir d'un groupe de volumes
La commande vgrename, Renommer un groupe de volumes
La commande vgs, Rapport personnalisé pour LVM
arguments d'affichage , La commande vgs

La commande vgscan, Analyse des disques pour les groupes de volumes afin de construire le fichier de cache
La commande vgsplit, Partager un groupe de volumes
LVM
administration des volumes logiques, Administration de volumes logiques
administration du volume physique, Administration de volumes physiques
aide, Utilisation des commandes CLI
aperçu de l'architecture, Aperçu de l'architecture LVM
clusterisé, Le gestionnaire de volumes logiques LVM en cluster (CLVM)
composants, Aperçu de l'architecture LVM, Composants LVM
étiquette, Les volumes physiques

format de rapport personnalisé, Rapport personnalisé pour LVM
groupe de volumes, définition, Les groupes de volumes
historique, Aperçu de l'architecture LVM
journalisation, Journalisation
structure de répertoire, Création de groupes de volumes
volume physique, définition, Les volumes physiques

LVM1, Aperçu de l'architecture LVM
LVM2, Aperçu de l'architecture LVM

M

Message "Insufficient Free Extents", Extensions libres insuffisantes pour un volume logique
métadonnées
récupération, Recupération des métadonnées du volume physique
sauvegarde, Sauvegarde d'un volume logique, Sauvegarde des métadonnées d'un groupe de volumes

N

noms des chemins, Utilisation des commandes CLI
noms des chemins de périphériques, Utilisation des commandes CLI
numéros de périphérique persistants, Numéros de périphérique persistants
numéros des périphériques
majeur, Numéros de périphérique persistants
mineur, Numéros de périphérique persistants
persistant, Numéros de périphérique persistants

P

paramètre de configuration mirror_image_fault_policy, Politique en cas d'échec des volumes logiques en miroir
paramètre de configuration mirror_log_fault_policy, Politique en cas d'échec des volumes logiques en miroir
partitions
multiples, Plusieurs partitions sur un disque

périphérique bloc
analyse, Recherche de périphériques blocs

périphériques ayant échoué
Affichage d'informations, Affichage d'informations à propos des périphériques ayant échoué.

procédures d'administration, Aperçu de l'administration LVM

R

redimensionnement
- **volume logique,** Redimensionnement des volumes logiques
- **volume physique,** Redimensionnement de volumes physiques

rennomer
- **groupe de volumes,** Renommer un groupe de volumes

renommer
- **volume logique,** Renommer les volumes logiques

répertoire des fichiers spéciaux de périphériques, Création de groupes de volumes
résolution de problèmes, Résolution de problèmes LVM
rules.d directory, Intégration udev avec le mappeur de périphériques

S

sauvegarde
- **fichier,** Sauvegarde d'un volume logique
- **métadonnées,** Sauvegarde d'un volume logique, Sauvegarde des métadonnées d'un groupe de volumes

sortie verbeuse, Utilisation des commandes CLI
suppression
- **d'un disque du volume logique,** Suppression d'un disque du volume logique
- **volume logique,** Suppression de volumes logiques

supprimer
- **volumes physiques,** Suppression de volumes physiques

système de fichiers
- **extensible sur un volume logique,** Augmentation de la taille d'un système de fichiers sur un volume logique

système de fichiers extensible
- **volume logique,** Augmentation de la taille d'un système de fichiers sur un volume logique

T

taille du périphérique, maximum, Création de groupes de volumes
type de partition, paramètre, Paramétrage du type de partition

U

udev device manager, Prise en charge du mappeur de périphériques pour le gestionnaire de périphériques udev
udev rules, Intégration udev avec le mappeur de périphériques
unités de ligne de commande, Utilisation des commandes CLI
unités, ligne de commande, Utilisation des commandes CLI

V

volume instantané
- **définition,** Les volumes d'instantanés

volume logique
- **accès exclusif,** Activation des volumes logiques sur les noeuds individuels d'un cluster
- **accès local,** Activation des volumes logiques sur les noeuds individuels d'un cluster
- **affichage,** Affichage de volumes logiques, Rapport personnalisé pour LVM, La commande lvs
- **arguments d'affichage lvs,** La commande lvs
- **augmenter,** Augmentez la taille des volumes logiques
- **création,** Création de volumes logiques linéaires
- **définition,** Volumes logiques, Les volumes logiques LVM
- **en miroir,** Création de volumes en miroir
- **exemple de création,** Création d'un volume logique LVM sur trois disques
- **extension,** Augmentez la taille des volumes logiques
- **instantané,** Création d'instantanés de volumes
- **linéaire,** Création de volumes logiques linéaires
- **mode stripe,** Création de volumes en mode stripe
- **modifier les paramètres,** Changement des paramètres d'un groupe de volumes logiques
- **redimensionnement,** Redimensionnement des volumes logiques
- **réduction,** Réduire la taille des volumes logiques
- **renommer,** Renommer les volumes logiques
- **suppression,** Suppression de volumes logiques

volume logique en miroir
- **convertir en linéaire,** Changement de la configuration du volume en miroir
- **création,** Création de volumes en miroir
- **définition,** Les volumes logiques en miroir
- **politique d'échec,** Politique en cas d'échec des volumes logiques en miroir
- **reconfiguration,** Changement de la configuration du volume en miroir

volume logique en mode stripe
- **création,** Création de volumes en mode stripe
- **définition,** Les volumes logiques en mode stripe
- **exemple de création,** Création d'un volume logique en mode stripe

volume logique en stripe
- **augmenter,** Augmenter la taille d'un volume en mode stripe
- **extension,** Augmenter la taille d'un volume en mode stripe

volume logique linéaire
- **convertir en miroir,** Changement de la configuration du volume en miroir
- **création,** Création de volumes logiques linéaires

définition, Les volumes linéaires

volume logique miroir

en cluster, Création d'un volume logique LVM miroir dans un cluster

récupération après incident, Récupération suite à un échec miroir LVM

volume physique

administration, générale, Administration de volumes physiques

affichage, Affichage des volumes physiques, Rapport personnalisé pour LVM, La commande pvs

ajouter à un groupe de volumes, Ajout de volumes physiques à un groupe de volumes

arguments d'affichage pvs, La commande pvs

création, Création de volumes physiques

définition, Les volumes physiques

illustration, Structure d'un volume physique LVM

initialisation, Initialisation des volumes physiques

récupération, Remplacement d'un volume physique manquant

redimensionnement, Redimensionnement de volumes physiques

structure, Structure d'un volume physique LVM

suppression du groupe de volumes, Suppression de volumes physiques à partir d'un groupe de volumes

suppression du volume perdu, Supprimer les volumes physiques perdus d'un groupe de volumes

supprimer, Suppression de volumes physiques

volumes logiques

administration générale, Administration de volumes logiques