

Red Hat Enterprise Linux 5

邏輯卷冊管理員的管理指南

LVM Administrator's Guide



Red Hat Enterprise Linux 5 邏輯卷冊管理員的管理指南

LVM Administrator's Guide

版 1

Copyright © 2009 Red Hat Inc..

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution—Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at <http://creativecommons.org/licenses/by-sa/3.0/>. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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, MetaMatrix, 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.

All other trademarks are the property of their respective owners.

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701

本指南詳細描述了 LVM 邏輯卷冊管理員 (logical volume manager) , 包括在叢集環境下執行 LVM 的相關資訊。本文件之內容所針對的是 LVM2 發行版。

簡介	vii
1. 有關本指南	vii
2. 讀者	vii
3. 軟體版本	vii
4. 相關文件	vii
5. 意見	viii
6. 文件常規	viii
6.1. 排字上的常規	viii
6.2. 引述常規	ix
6.3. 註解和警告	x
1. LVM 邏輯卷測管理員	1
1.1. 邏輯卷冊	1
1.2. LVM 架構概要	1
1.3. 叢集邏輯卷測管理員 (Clustered Logical Volume Manager, CLVM)	2
1.4. 文件總覽	4
2. LVM 元件	7
2.1. 實體卷冊	7
2.1.1. LVM Physical Volume Layout	7
2.1.2. 磁碟上的多重分割區	8
2.2. 卷冊群組	8
2.3. LVM 邏輯卷冊	8
2.3.1. 線性邏輯卷冊	9
2.3.2. 等量邏輯卷冊	10
2.3.3. 鏡像邏輯卷冊	11
2.3.4. 快照卷冊	12
3. LVM 管理總覽	15
3.1. 在叢集中建立 LVM 卷冊	15
3.2. 邏輯卷冊建立總覽	16
3.3. 在邏輯卷冊上遞增檔案系統	16
3.4. 邏輯卷冊備份	17
3.5. 記錄	17
4. 透過 CLI 指令來進行 LVM 管理	19
4.1. 使用 CLI 指令	19
4.2. 實體卷冊管理	20
4.2.1. 建立實體卷冊	20
4.2.2. 顯示實體卷冊	22
4.2.3. 避免配置於實體卷冊上	22
4.2.4. 重設實體卷冊的大小	23
4.2.5. 移除實體卷冊	23
4.3. 卷冊群組管理	23
4.3.1. 建立卷冊群組	23
4.3.2. 在叢集中建立卷冊群組	24
4.3.3. 新增實體卷冊至卷冊群組中	24
4.3.4. 顯示卷冊群組	25
4.3.5. 掃描磁碟來找尋卷冊群組以便建立快取檔案	25
4.3.6. 由卷冊群組中移除實體卷冊	26
4.3.7. 更改卷冊群組的參數	26
4.3.8. 啟用和停用卷冊群組	27
4.3.9. 移除卷冊群組	27
4.3.10. 分割卷冊群組	27
4.3.11. 結合卷冊群組	27
4.3.12. 備份卷冊群組的 Metadata	28

4.3.13. 為卷冊群組重新命名	28
4.3.14. 將卷冊群組移至另一部系統	28
4.3.15. 重新建立一個卷冊群組目錄	29
4.4. 邏輯卷冊管理	29
4.4.1. 建立邏輯卷冊	29
4.4.2. 一致的裝置號碼	33
4.4.3. 重設邏輯卷冊大小	34
4.4.4. 更改邏輯卷冊群組的參數	34
4.4.5. 重新為邏輯卷冊命名	34
4.4.6. 移除邏輯卷冊	34
4.4.7. 顯示邏輯卷冊	35
4.4.8. 遞增邏輯卷冊	35
4.4.9. 延伸等量的卷冊	36
4.4.10. 縮減邏輯卷冊	37
4.5. 建立快照卷冊 (Snapshot Volumes)	38
4.6. 透過過濾器來控制 LVM 裝置掃描 (LVM Device Scans)	39
4.7. 線上資料重置 (Online Data Relocation)	39
4.8. 在叢集中啟用各別節點上的邏輯卷冊	40
4.9. LVM 的自訂化回報	40
4.9.1. 格式控制	41
4.9.2. 物件選擇	42
4.9.3. 排序 LVM 報告	47
4.9.4. 指定單位	48
5. LVM 配置範例	51
5.1. 在三個磁碟上建立一個 LVM 邏輯卷冊	51
5.1.1. 建立實體卷冊 (Physical Volumes)	51
5.1.2. 建立卷冊群組	51
5.1.3. 建立邏輯卷冊	51
5.1.4. 建立檔案系統	51
5.2. 建立一個 Striped 邏輯卷冊	52
5.2.1. 建立實體卷冊 (Physical Volumes)	52
5.2.2. 建立卷冊群組	52
5.2.3. 建立邏輯卷冊	53
5.2.4. 建立檔案系統	53
5.3. 分割卷冊群組	53
5.3.1. 判斷可用空間	54
5.3.2. 移動資料	54
5.3.3. 分割卷冊群組	54
5.3.4. 建立新的邏輯卷冊	55
5.3.5. 製作檔案系統和掛載新的邏輯卷冊	55
5.3.6. 啟用和掛載原本的邏輯卷冊	55
5.4. 由邏輯卷冊中移除磁碟	55
5.4.1. 將扇區移至現有的實體卷冊中	55
5.4.2. 將扇區移至新磁碟上	56
5.5. 在叢集中建立鏡像 LVM 邏輯卷冊	57
6. LVM 疑難排解	61
6.1. 疑難排解診斷結果	61
6.2. 顯示錯誤裝置的相關資訊	61
6.3. 由 LVM 鏡像錯誤中復原	62
6.4. 復原實體卷冊的 Metadata	64
6.5. 替換一個遺失的實體卷冊	66
6.6. 將遺失的實體卷冊由一個卷冊群組中移除掉	66
6.7. 邏輯卷冊的可用扇區不足	66

7. 利用 LVM GUI 來進行 LVM 管理	69
A. 裝置映射 (Device Mapper) 設備	71
A.1. 裝置表格映射	71
A.1.1. Linear 映射目標	72
A.1.2. 等量映射目標	72
A.1.3. 鏡像映射目標	73
A.1.4. 快照和 snapshot-origin 映射目標	75
A.1.5. error 映射目標	77
A.1.6. zero 映射目標	77
A.1.7. multipath 映射目標	77
A.1.8. crypt 映射目標	79
A.2. dmsetup 指令	80
A.2.1. dmsetup info 指令	80
A.2.2. dmsetup ls 指令	81
A.2.3. dmsetup status 指令	82
A.2.4. dmsetup deps 指令	82
B. LVM 配置檔案	85
B.1. LVM 配置檔案	85
B.2. 範例 lvm.conf 檔案	85
C. LVM 物件標籤 (Object Tags)	93
C.1. 新增和移除物件標籤	93
C.2. 主機標籤 (Host Tags)	93
C.3. 利用標籤來控制啓用	93
D. LVM 卷冊群組 Metadata	95
D.1. 實體卷冊標籤 (Physical Volume Label)	95
D.2. Metadata 內容	95
D.3. Metadata 範例	96
E. 修訂歷史	99
索引	101

簡介

1. 有關本指南

本指南描述了邏輯卷冊管理員 (Logical Volume Manager, LVM) , 包括有關於在叢集環境下執行 LVM 的相關資訊。本文件之內容所針對的是 LVM2 發行版。

2. 讀者

本指南是為了管理執行 Linux 作業系統的電腦的系統管理員所撰寫的。使用者必須熟悉 Red Hat Enterprise Linux 5 以及 GFS 或 GFS2 檔案系統管理。

3. 軟體版本

表格 1. 軟體版本

軟體	描述
RHEL5	代表 RHEL5 或更新版本
GFS	代表 RHEL5 或更新版本的 GFS

4. 相關文件

如欲取得更多有關於如何使用 Red Hat Enterprise Linux 的相關資訊, 請參閱下列資源:

- Red Hat Enterprise Linux 安裝指南 — 提供了有關於安裝 Red Hat Enterprise Linux 5 的相關資訊。
- Red Hat Enterprise Linux 建置指南 — 提供了有關於建置、配置和管理 Red Hat Enterprise Linux 5 上的相關資訊。

如欲取得更多有關於 Red Hat Enterprise Linux 5 的 Red Hat Cluster Suite 的相關資訊, 請參閱下列資源:

- Red Hat Cluster Suite 總覽 — 提供了高層級的 Red Hat Cluster Suite 總覽。
- 配置和管理 Red Hat 叢集 — 提供了有關於安裝、配置和管理 Red Hat Cluster 元件上的相關資訊。
- 全域檔案系統: 配置和管理 — 提供了有關於安裝、配置和管理 Red Hat GFS (Red Hat 全域檔案系統) 上的相關資訊。
- 全域檔案系統 2: 配置和管理 — 提供了有關於安裝、配置和維護 Red Hat GFS2 (Red Hat 全域檔案系統 2) 上的相關資訊。
- 使用 Device-Mapper Multipath — 提供了有關於使用 Red Hat Enterprise Linux 5 的 Device-Mapper Multipath 功能的相關資訊。
- 使用 GNBD 以及全域檔案系統 — 提供了使用 Red Hat GFS 的全域網路區塊裝置 (Global Network Block Device, GNBD) 上的總覽。
- Linux 虛擬伺服器管理 — 提供了有關於使用 Linux Virtual Server (LVS) 配置高效能系統和服務上的相關資訊。
- Red Hat Cluster Suite 發行公告 — 提供了有關於最新 Red Hat 叢集套件的相關資訊。

Red Hat Cluster Suite 文件和其它 Red Hat 文件在 Red Hat Enterprise Linux Documentation CD 和線上 (<http://www.redhat.com/docs/>) 皆含有 HTML、PDF、和 RPM 版本。

5. 意見

若您發現了錯字，或是您對於本指南有任何改善的意見，我們很樂意傾聽您的意見。請在 Bugzilla (<http://bugzilla.redhat.com/bugzilla/>) 中針對於 rh-cs 這個元件提交一份報告。

Be sure to mention the manual's identifier:

```
Bugzilla component: Documentation-cluster
Book identifier: Logical_Volume_Manager_Administration(EN)-5 (2009-08-18T15:20)
```

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

若您有任何改善文件的建議，請提供我們具體的資訊。若您發現了錯誤，請填入部份號碼和附近文字，如此一來我們便可輕易找到該錯誤。

6. 文件常規

本指南使用了幾種常規，以強調特定文字與詞組，並著重於特定的資訊。

在 PDF 和書面版本中，本指南使用了來自於 [Liberation Fonts](#)¹ 字體組的 typefaces。倘若 Liberation Fonts 字體已安裝在您系統上的話，該字體也會被使用於 HTML 版本中。若是沒有的話，其它相等的 typefaces 便會被顯示。請注意：就預設值，Red Hat Enterprise Linux 5（與更新版本）已包含了 Liberation Fonts 字體。

6.1. 排字上的常規

有四種被用來強調特定文字與詞組的排字常規。這些常規以及它們所適用於的情況如下。

固定粗體字型 (Mono-spaced Bold)

用來強調系統輸入，包括 shell 指令、檔案名稱與路徑。同時也會被使用來強調 key caps 與按鍵組合。例如：

若要查看位於您目前工作目錄中的 my_next_bestselling_novel 檔案的話，請在 shell 提示中輸入 `cat my_next_bestselling_novel` 指令並按下 Enter 來執行該指令。

以上包含了一個檔案名稱、shell 指令，以及 key cap，並且全部以固定粗體字型來顯示。

按鍵組合可透過 keycaps 藉由連字符號連接組合鍵來辨別。例如：

請按下 Enter 來執行指令。

請按下 Ctrl+Alt+F2 來切換至第一個虛擬終端機。按下 Ctrl+Alt+F1 來返回您的 X-Windows session。

第一段落強調了應輸入的特定 keycap。第二段落強調了兩組含有三個 keycaps 的組合鍵，各組組合鍵都是同時按下的。

¹ <https://fedorahosted.org/liberation-fonts/>

若討論到原始碼的話，段落中所提及的 `class` 名稱、`method`、`functions`、`variable` 名稱與回傳值，都將會如上一般地以固定粗體字型顯示。例如：

和檔案相關的 `class`，其中包含了 `filesystem`（檔案系統）、`file`（檔案）以及 `dir`（目錄）。各個 `class` 都有著與它關聯的權限組。

相稱粗體字型（Proportional Bold）

這代表在系統上所會看見的文字或詞組，這包含了應用程式名稱；對話方塊文字；被標記的按鈕；核取方塊與 `radio button` 標籤；選單標題以及子選單標題。例如：

由主選單選取系統 → 偏好設定 → 滑鼠來啟動滑鼠偏好設定。請在按鈕分頁中點選左手操作滑鼠核取方塊並按下關閉來將主要滑鼠按鍵由左邊切換至右邊（這可讓滑鼠適合以左手使用）。

若要將特殊字元插入一個 `gedit` 檔案中的話，請由主選單中選擇應用程式 → 附屬應用程式 → 字元對應表。接下來，請由字元對應表的選單中選擇搜尋 → 尋找…，並在搜尋欄位中輸入字元的名稱，然後按下下一步。您所選擇的字元將會顯示於字元表中。在被選定的字元上點兩下滑鼠便可將它放置在準備複製的文字欄位中，接下來請按下旁邊的複製按鈕。現在，請切換回您的文件並由 `gedit` 的選單上選擇編輯 → 貼上。

以上文字包含了應用程式名稱；系統全域的選單名稱與項目；應用程式特屬的選單名稱；以及在 GUI 介面中所看到的按鈕與文字全部皆以相稱粗體字型來顯示，並且可透過內文來辨別。

Mono-spaced Bold Italic（固定粗體斜體字型）或是 Proportional Bold Italic（相稱粗體斜體字型）

無論是 Mono-spaced Bold 或是 Proportional Bold，額外的斜體字型便表示可替換或是變數文字。斜體字型代表您不會照字面輸入的文字，或是會依照情況而改變的文字。比方說：

若要透過使用 `ssh` 來連至一部遠端機器，請在 `shell` 提示中輸入 `ssh 用戶名稱@網域.名稱`。若遠端機器為 `example.com` 而您在該機器上的用戶名為 `john` 的話，請輸入 `ssh john@example.com`。

`mount -o remount 檔案系統` 指令會將 `named` 檔案系統重新掛載。比方說，若要重新掛載 `/home` 檔案系統的話，該指令就會是 `mount -o remount /home`。

使用 `rpm -q 套件` 指令來查看目前已安裝套件的版本。系統將會回傳此結果：套件發行版本。

請注意以上以粗體斜體字型所表現出的文字 — 用戶名稱、網域.名稱、檔案系統、套件以及發行版本。所有文字皆為佔位符號，這可代表您提供一項指令時所輸入的文字或是系統所顯示的文字。

除了用來顯示一項作業標題這樣的標準用法，斜體字也可代表第一次使用的重要新詞彙。比方說：

`Publican` 是個 `DocBook` 發佈系統。

6.2. 引述常規

終端機輸出與原始碼資料會被設為在附近的文字間不會被看見。

傳送至終端機的輸出設置為 `mono-spaced roman`，並且以此方式顯示：

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads     images  notes  scripts svgs
```

原始碼排列亦設置為 `mono-spaced roman` 不過會如下加上語法強調：

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object          ref    = iniCtx.lookup("EchoBean");
        EchoHome        home   = (EchoHome) ref;
        Echo             echo   = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

6.3. 註解和警告

最後，我們將使用三種視覺上的形式，來強調可能會被遺漏掉的資訊。



註解

註解代表某些作業上的提示、捷徑或是其它完成方式。忽略註解並不會帶來太大的負面影響，不過您可能會忽略掉某些能夠較輕鬆完成工作的方式。



重點

重點方塊會將容易遺漏掉的項目詳細列出：只對應於當下 session 的配置變更，或是某些要套用更新前必須將之重新啓用的服務。倘若您忽略掉重點方塊，雖然不會造成資料遺失，不過卻會造成工作上的不便與其它影響。



警告

任何警告都不該被忽略掉。忽略警告則很有可能會造成資料上的遺失。

LVM 邏輯卷測管理員

此章節提供了高層級的邏輯卷冊管理員（LVM）元件的概要。

1.1. 邏輯卷冊

卷冊管理員會針對於實體儲存裝置建立一個虛擬的儲存裝置，這能讓您建立邏輯儲存卷冊。比起直接使用實體儲存裝置，這提供了更強大的靈活性。當利用邏輯卷冊時，您不會受到實體磁碟大小的限制。此外，硬體儲存配置對軟體來說是隱藏的，如此一來它可在不停止應用程式或卸載檔案系統的情況下，重設大小或進行移動。這可降低作業上的成本。

和直接使用實體儲存相較之下，邏輯卷冊提供了下列優點：

- 可變通的容量

當使用邏輯卷冊時，檔案系統可延伸至多重磁碟上，因為您可將磁碟和分割區聚合為一個單獨的邏輯卷冊。

- 可重設大小的儲存池（Resizeable storage pools）

您可在不格式化與重新分割基本磁碟裝置的情況下透過基本的軟體指令來延伸或遞減邏輯卷冊的大小。

- 線上資料重置（Online data relocation）

若要建置更新、更快，或更有彈性的儲存子系統，您可在您的系統啓用時移動資料。資料可在磁碟使用中的時候被重新整理。比方說，您可在移除一個熱插拔磁碟之前先將它清空。

- 方便的裝置命名

邏輯儲存卷冊可管理於用戶定義群組中，並且您可視您的喜好來進行命名。

- 多磁碟記錄塊串操作（Disk striping）

您可建立一個將資料分散在兩個或更多磁碟上的邏輯卷冊。這會顯著地增加總處理能力。

- 卷冊鏡像

邏輯卷冊提供了一個便利的方式來為您的資料配置一個鏡像。

- 卷冊快照（Volume Snapshots）

當使用邏輯卷冊時，您能夠透過產生裝置快照來進行含有一致性的備份，或在不影響真實資料的情況下測試進行變更後的效果。

這些 LVM 中的功能實做描述於此文件剩下的部份中。

1.2. LVM 架構概要

RHEL 4 發行版的 Linux 作業系統上原本的 LVM1 邏輯卷冊管理員已被 LVM2 取代，它含有個比 LVM1 更加通用的 kernel 架構。LVM2 針對於 LVM1 提供了下列改善：

- 可變通的容量
- 更有效率的 metadata 儲存

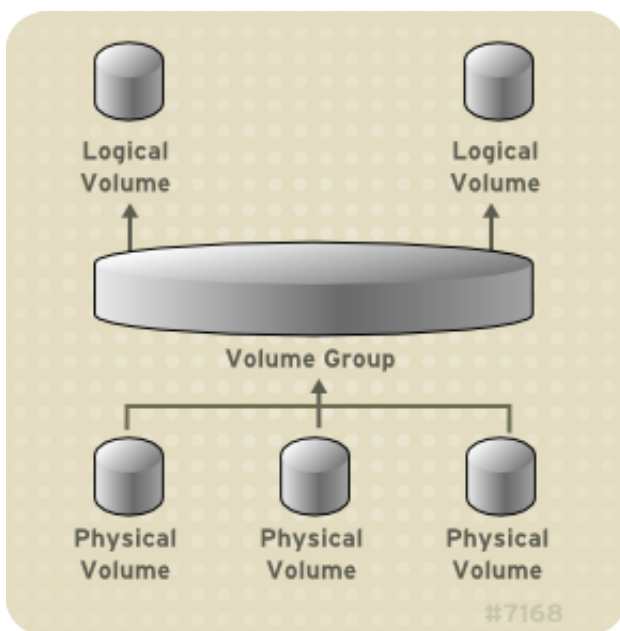
- 較佳的復原格式
- 新的 ASCII metadata 格式
- metadata 的基元變更
- metadata 的重複副本

LVM2 含有 LVM1 的向後相容性，除了快照與叢集支援。您可透過使用 `vgconvert` 指令來將卷冊群組由 LVM1 格式轉換為 LVM2 格式。如欲取得更多有關於轉換 LVM metadata 格式的相關資訊，請參閱 `vgconvert(8) man page`。

一個 LVM 邏輯卷冊的基本實體儲存裝置是個像是分割區或是完整磁碟的區塊裝置。此裝置被初始化為了一個 LVM 實體卷冊 (PV)。

若要建立一個 LVM 邏輯卷冊，實體卷冊會被併入一個卷冊群組 (volume group, VG) 中。這會建立一個磁碟的空間池，從而，LVM 邏輯卷冊 (LVs) 便能被分配出。這項程序和磁碟被劃分為分割區的方式類似。邏輯卷冊是由檔案系統和應用程式（例如資料庫）所使用的。

圖形 1.1, “LVM Logical Volume Components” shows the components of a simple LVM logical volume:



圖形 1.1. LVM Logical Volume Components

For detailed information on the components of an LVM logical volume, see [章 2, LVM 元件](#).

1.3. 叢集邏輯卷測管理員 (Clustered Logical Volume Manager, CLVM)

叢集邏輯卷冊管理員 (Clustered Logical Volume Manager, CLVM) 為一組 LVM 的叢集延伸。這些延伸允許叢集中的元件透過使用 LVM（比方說在 SAN 上）來管理共享的儲存裝置。

您是否應使用 CLVM 取決於您的系統需求：

- 若您系統只有一個節點需要存取您配置來作為邏輯卷冊的儲存裝置，那麼您可使用 LVM 並且不使用 CLVM 的延伸，以該節點建立的邏輯卷冊便都會是節點的本地邏輯卷冊。

- 若您正在使用一部加入叢集的系統來進行容錯，並且存取儲存裝置的單獨節點一次只有一個會被啟用。您應該使用 High Availability Logical Volume Management agents (HA-LVM)。如欲取得更多有關於 HA-LVM 的相關資訊，請參閱 [Configuring and Managing a Red Hat Cluster](#) (配置和管理 Red Hat Cluster 指南)。
- 若您的叢集有超過一個節點需要存取您的儲存裝置並在啓用的節點之間進行共享的話，那麼您就必須使用 CLVM。CLVM 允許用戶透過在配置邏輯卷冊時將實體儲存裝置鎖定以便配置共享儲存裝置上的邏輯卷冊，並使用叢集鎖定服務來管理共享儲存裝置。

若要使用 CLVM 的話，Red Hat Cluster Suite 軟體以及 `clvmd` daemon 都必須執行。`clvmd` daemon 為 LVM 的關鍵叢集延伸。`clvmd` daemon 會在各個叢集電腦中執行並在叢集中分配 LVM metadata 的更新，並提供給各個叢集電腦相同的邏輯卷冊視點。欲取得更多有關於安裝和管理 Red Hat Cluster Suite 上的相關資訊，請參閱配置與管理 Red Hat 叢集。

若要確保 `clvmd` 會在 boot time 時啓動，您可針對於 `clvmd` 服務執行一項 `chkconfig ... on` 指令，如下：

```
# chkconfig clvmd on
```

若 `clvmd` daemon 沒有啓動的話，您可針對於 `clvmd` 服務執行一項 `service ... start` 指令，如下：

```
# service clvmd start
```

Creating LVM logical volumes in a cluster environment is identical to creating LVM logical volumes on a single node. There is no difference in the LVM commands themselves, or in the LVM graphical user interface, as described in [章 4, 透過 CLI 指令來進行 LVM 管理](#) and [章 7, 利用 LVM GUI 來進行 LVM 管理](#). In order to enable the LVM volumes you are creating in a cluster, the cluster infrastructure must be running and the cluster must be quorate.

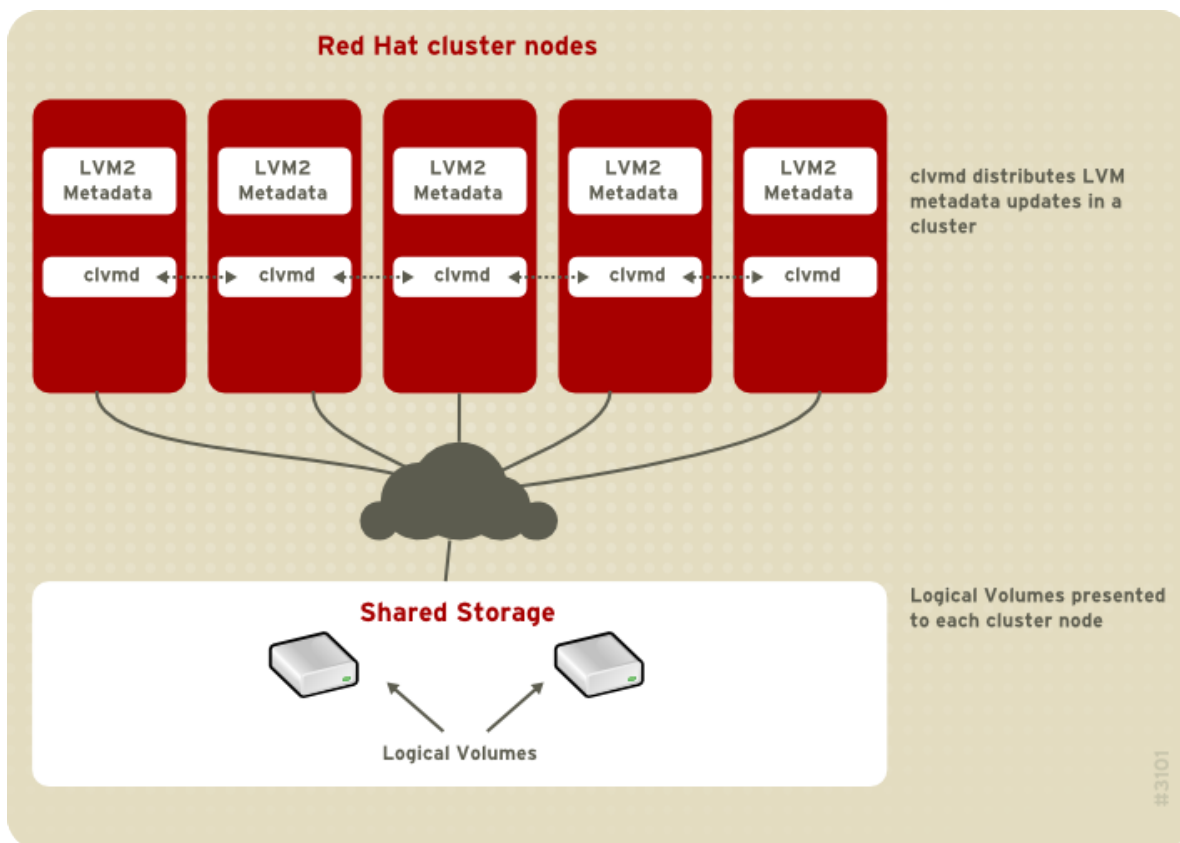
By default, logical volumes created with CLVM on shared storage are visible to all systems that have access to the shared storage. It is possible to create volume groups in which all of the storage devices are visible to only one node in the cluster. It is also possible to change the status of a volume group from a local volume group to a clustered volume group. For information, see [節 4.3.2, “在叢集中建立卷冊群組”](#) and [節 4.3.7, “更改卷冊群組的參數”](#).



警告

當您要在共享儲存裝置上以 CLVM 建立卷冊群組時，您必須確認叢集中所有節點皆可存取組成了卷冊群組的實體卷冊。某些節點可存取儲存裝置，有些則無法存取儲存裝置的這種非對稱式叢集配置不受到支援。

圖形 1.2, “CLVM 概要” shows a CLVM overview in a Red Hat cluster.



圖形 1.2. CLVM 概要



注意

若要使用用於 Red Hat Cluster Suite 中的共享儲存裝置，您必須要執行叢集邏輯卷冊管理員 daemon (cluster logical volume manager daemon, `clvmd`) 或是 High Availability Logical Volume Management (HA-LVM)。若基於作業因素您無法使用 `clvmd` daemon 或是 HA-LVM，或是因為您沒有正確的權限，您絕不可在共享的磁碟上使用單 instance 的 LVM，因為這可能會造成資料損毀。若您有任何疑問，請和您的 Red Hat 服務人員取得聯繫。



注意

CLVM requires changes to the `lvm.conf` file for cluster-wide locking. Information on configuring the `lvm.conf` file to support clustered locking is provided within the `lvm.conf` file itself. For information about the `lvm.conf` file, see [附錄 B, LVM 配置檔案](#).

1.4. 文件總覽

本文件剩下的章節包含了下列內容：

- [章 2, LVM 元件](#) describes the components that make up an LVM logical volume.

- [章 3, LVM 管理總覽](#) provides an overview of the basic steps you perform to configure LVM logical volumes, whether you are using the LVM Command Line Interface (CLI) commands or the LVM Graphical User Interface (GUI).
- [章 4, 透過 CLI 指令來進行 LVM 管理](#) summarizes the individual administrative tasks you can perform with the LVM CLI commands to create and maintain logical volumes.
- [章 5, LVM 配置範例](#) provides a variety of LVM configuration examples.
- [章 6, LVM 疑難排解](#) provides instructions for troubleshooting a variety of LVM issues.
- [章 7, 利用 LVM GUI 來進行 LVM 管理](#) summarizes the operating of the LVM GUI.
- [附錄 A, 裝置映射 \(Device Mapper\) 設備](#) describes the Device Mapper that LVM uses to map logical and physical volumes.
- [附錄 B, LVM 配置檔案](#) describes the LVM configuration files.
- [附錄 C, LVM 物件標籤 \(Object Tags\)](#) describes LVM object tags and host tags.
- [附錄 D, LVM 卷冊群組 Metadata](#) describes LVM volume group metadata, and includes a sample copy of metadata for an LVM volume group.

LVM 元件

此章節描述了 LVM 邏輯卷冊的元件。

2.1. 實體卷冊

一個 LVM 邏輯卷冊的基本實體儲存裝置就是一些像是分割區或是整個磁碟的區塊裝置。若要使用一個 LVM 邏輯卷冊的裝置，該裝置必須被初始化為實體卷冊 (PV)。請在實體卷冊將標籤放置在靠近裝置的起始時將區塊裝置初始化。

就預設值，LVM 標籤會被放置在第二個 512 位元組的磁區中。您可透過將標籤放置在前 4 個磁區中的任何一個磁區上來將此預設值覆寫。在必要的情況下，這能讓 LVM 卷冊和這些磁區的其它用戶並存。

一個 LVM 標籤會提供實體卷冊的正確標示和裝置順序，這是因為系統啓動時，裝置的順序能夠是任意的。LVM 標籤能夠在系統重新啓動的情況下以及叢集的環境中保留。

LVM 標籤會將裝置視為是一個 LVM 實體卷冊。它包含著實體卷冊的亂數唯一識別元 (random unique identifier, UUID)。它同時將區塊裝置的大小以位元組來儲存了起來，並且它會記錄 LVM metadata 被儲存在裝置上的哪裡。

LVM metadata 包含了您系統上的 LVM 卷冊群組的配置詳情。就預設值，metadata 會有個副本被保留在卷冊群組中每個實體卷冊中的所有 metadata 區域裡。LVM metadata 非常小並且會被儲存為 ASCII。

目前，LVM 允許您在各個實體卷冊上儲存 0、1 或 2 個相同的 metadata 副本。一旦您配置了實體卷冊上的 metadata 副本數量之後，您之後便無法修改該數量。第一個副本會被儲存在裝置的起始，就在標籤之後不遠的位置上。若有第二個副本的話，它便會被放置在裝置的最後位置上。若您不小心將您磁碟一開始的區域覆寫掉的話，位於裝置最後的第二個 metadata 副本能讓您將 metadata 復原。

For detailed information about the LVM metadata and changing the metadata parameters, see [附錄 D, LVM 卷冊群組 Metadata](#).

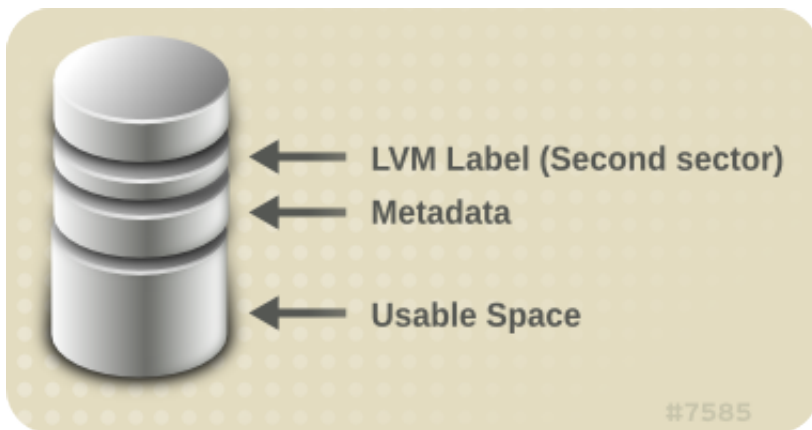
2.1.1. LVM Physical Volume Layout

圖形 2.1, “實體卷冊配置” shows the layout of an LVM physical volume. The LVM label is on the second sector, followed by the metadata area, followed by the usable space on the device.



注意

在 Linux kernel (和本文件) 中，磁區的大小將會被視為是 512 個位元組。



圖形 2.1. 實體卷冊配置

2.1.2. 磁碟上的多重分割區

LVM 能讓您透過磁碟分割區來建立實體卷冊。一般我們建議您建立一個能夠完全涵蓋一個被標記為 LVM 實體卷冊的磁碟的單獨分割區，理由如下：

- 管理上的方便性

在系統中若各個真實的磁碟只出現一次的話，這將能簡化系統中的硬體追蹤（特別是當磁碟發生錯誤時）。此外，單獨磁碟上的多重實體卷冊可能會造成 kernel 在啟動時發出有關於不明分割區類型的相關警告。

- 等量磁碟效能 (Striping performance)

LVM 無法得知兩個實體卷冊是否位於相同的實體磁碟上。若您在兩個實體卷冊位於相同實體磁碟上時建立了一個等量的邏輯卷冊的話，等量磁碟可能會位於相同磁碟的不同分割區上。這將會造成效能上的降低。

雖然我們不建議，不過您可能會遇到需要將磁碟分割為各別 LVM 實體卷冊的情況。比方說，在一部有幾個磁碟的系統上，當您要將一個現有的系統遷移至 LVM 卷冊時，您可能需要將資料在分割區上進行移動。此外，若您擁有一個非常大的磁碟並且基於管理的原因而希望擁有超過一個卷冊群組的話，那麼您便需要將磁碟進行分割。若您沒有一個含有超過一個分割區的磁碟，並且這些分割區都位於相同卷冊群組中的話，當您在建立等量卷冊時，您應小心注意指定哪個分割區需要包含在邏輯卷冊中。

2.2. 卷冊群組

實體卷冊會被合併為卷冊群組 (VG)。這建立了一個磁碟空間的 pool，並且邏輯卷冊可從而被進行分配。

在一個卷冊群組中，可被用來進行分配的磁碟空間會被劃分為固定大小的單位稱為扇區。扇區為能夠被分配的最小空間單元。在實體卷冊中，扇區被稱為實體扇區 (physical extent)。

邏輯卷冊會被分配至與實體扇區相同大小的邏輯扇區中。因此卷冊群組中所有邏輯卷冊的扇區大小都會是相同的。卷冊群組會將邏輯扇區映射至實體扇區。

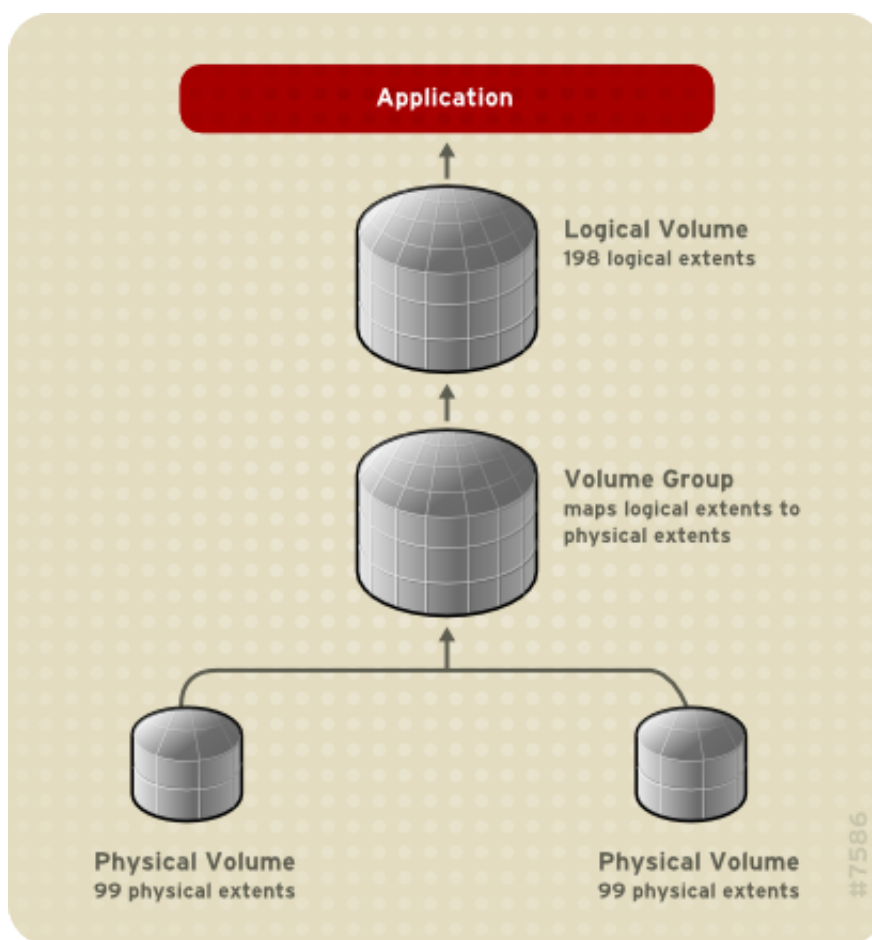
2.3. LVM 邏輯卷冊

在 LVM 中，邏輯卷冊會被劃分為邏輯卷冊。LVM 邏輯卷冊的類型有三種：linear（線性）卷冊、striped（等量）卷冊，以及 mirrored（鏡像）卷冊。這些邏輯卷冊描述於下列部份中。

2.3.1. 線性邏輯卷冊

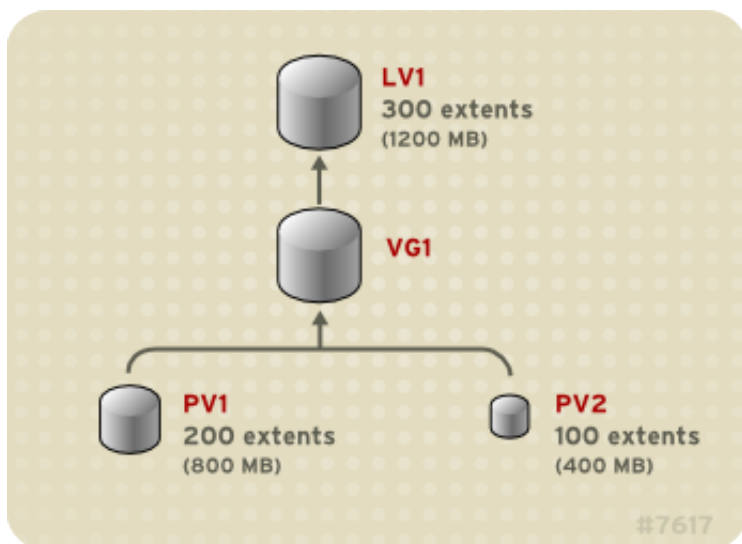
線性卷冊會將多個實體卷冊聚集為單一個邏輯卷冊。比方說，若您有兩個 60GB 的磁碟，您便可建立一個 120GB 的邏輯卷冊。實體儲存裝置會被序連在一起。

Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, as shown in 圖形 2.2, “扇區映射 (Extent Mapping)” logical extents 1 to 99 could map to one physical volume and logical extents 100 to 198 could map to a second physical volume. From the point of view of the application, there is one device that is 198 extents in size.



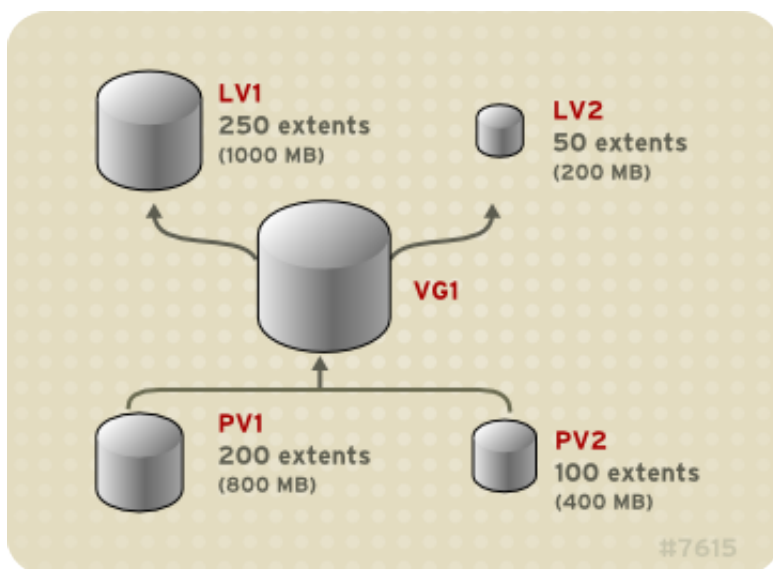
圖形 2.2. 扇區映射 (Extent Mapping)

The physical volumes that make up a logical volume do not have to be the same size. 圖形 2.3, “含有不平實的實體卷冊的線性卷冊” shows volume group VG1 with a physical extent size of 4MB. This volume group includes 2 physical volumes named PV1 and PV2. The physical volumes are divided into 4MB units, since that is the extent size. In this example, PV1 is 100 extents in size (400MB) and PV2 is 200 extents in size (800MB). You can create a linear volume any size between 1 and 300 extents (4MB to 1200MB). In this example, the linear volume named LV1 is 300 extents in size.



圖形 2.3. 含有不平均的實體卷冊的線性卷冊

You can configure more than one linear logical volume of whatever size you desire from the pool of physical extents. 圖形 2.4, “多重邏輯卷冊” shows the same volume group as in 圖形 2.3, “含有不平均的實體卷冊的線性卷冊”, but in this case two logical volumes have been carved out of the volume group: LV1, which is 250 extents in size (1000MB) and LV2 which is 50 extents in size (200MB).



圖形 2.4. 多重邏輯卷冊

2.3.2. 等量邏輯卷冊

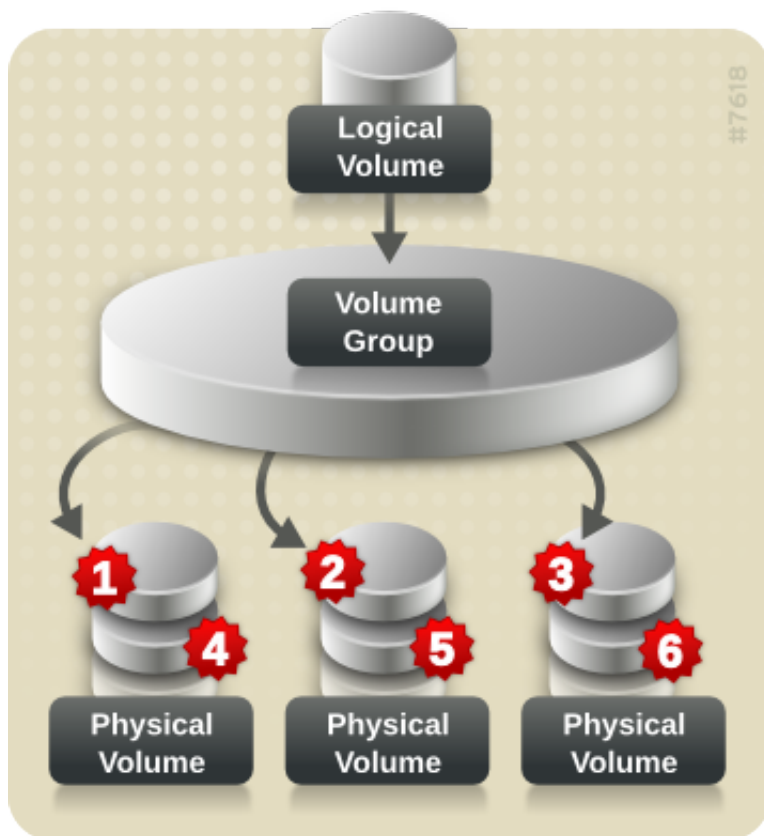
當您將資料寫至一個 LVM 邏輯卷冊時，檔案系統會將資料配置在基本實體卷冊之間。您可藉由建立一個等量的邏輯卷冊來控制資料寫至實體卷冊的方式。對於較大量的讀取和寫入循序來講，這可有效改善資料 I/O 的效率。

建立等量磁碟可藉由 round-round 的方式將資料寫至數量已預先決定的實體卷冊來增強效能。當建立等量磁碟時，I/O 可以平行的方式來進行。在某些情況下，這可能會使等量磁碟中的各個實體卷冊都達到近線性 (near-linear) 的效能。

下列描述了資料如何被等量分配在三個實體卷冊之間。在此圖形中：

- 第一個等量磁碟的資料已被寫至 PV1
- 第二個等量磁碟的資料已被寫至 PV2
- 第三個等量磁碟的資料已被寫至 PV3
- 第四個等量磁碟的資料已被寫至 PV1

在等量邏輯卷冊中，等量磁碟的大小不可超過扇區的大小。



圖形 2.5. 將資料等量分配在三個 PV 之間

Striped logical volumes can be extended by concatenating another set of devices onto the end of the first set. In order to extend a striped logical volume, however, there must be enough free space on the underlying physical volumes that make up the volume group to support the stripe. For example, if you have a two-way stripe that uses up an entire volume group, adding a single physical volume to the volume group will not enable you to extend the stripe. Instead, you must add at least two physical volumes to the volume group. For more information on extending a striped volume, see [節 4.4.9, “延伸等量的卷冊”](#).

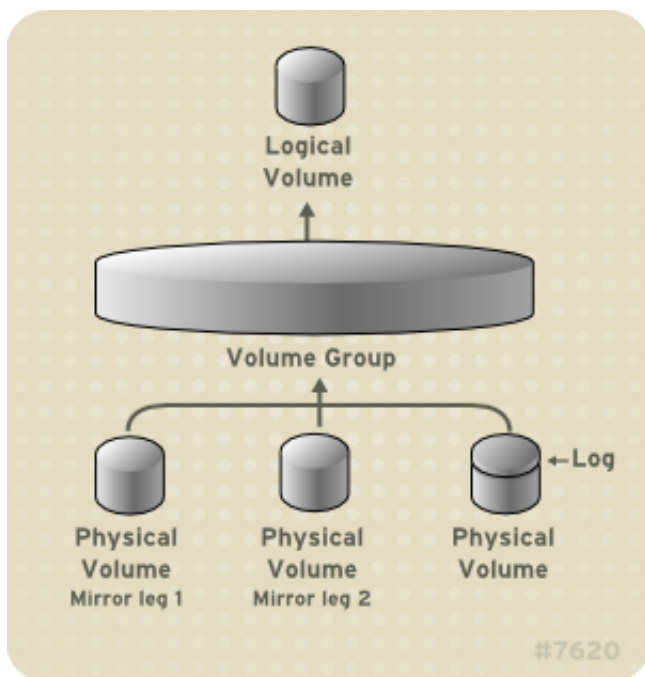
2.3.3. 鏡像邏輯卷冊

鏡像會將資料的相同副本保留在不同的裝置上。當資料被寫至一個裝置上時，它也會同時被寫至第二個裝置上，並成為該資料的鏡像。這提供了裝置發生錯誤時的保障。當鏡像的其中一個 leg 發生錯誤時，邏輯卷冊會成為一個線性卷冊並且依然能被存取。

LVM 支援鏡像卷冊。當您建立了鏡像邏輯卷冊時，LVM 會確保寫至基本實體卷冊的資料會被 mirror 至另一個實體卷冊上。透過 LVM，您可建立多重鏡像的鏡像邏輯卷冊。

LVM 鏡像會將被複製的裝置劃分為大小各為 512KB 的不同區域。LVM 會保留一個小型的日誌檔，它使用了該日誌檔來記錄哪個區域已和哪些鏡像同步化。這個日誌檔可被保留在磁碟上，如此一來它便可保留一致性並不受系統重新啓動影響，或是它亦可被保留在記憶體中。

圖形 2.6, “Mirrored Logical Volume” shows a mirrored logical volume with one mirror. In this configuration, the log is maintained on disk.



圖形 2.6. Mirrored Logical Volume



注意

從 RHEL 5.3 發行版開始，鏡像邏輯卷冊在叢集環境中將會受到支援。

For information on creating and modifying mirrors, see 節 4.4.1.3, “建立鏡像卷冊”.

2.3.4. 快照卷冊

LVM 快照提供了一項在特定一瞬間建立某個裝置的虛擬映像檔並且不干擾服務的功能。當針對於原始裝置進行變更並建立了快照之後，快照功能會建立一份遭到變更後的部份資料的副本，如此一來它便可重建該裝置的狀態。



注意

LVM snapshot 在叢集環境中的節點之間並不受到支援。您無法在叢集卷冊群組中建立 snapshot 卷冊。

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



注意

檔案系統的快照副本屬於虛擬副本，而不是實際的檔案系統媒介備份。快照無法取代備份程序。

快照的大小可支配被設為儲存原始卷冊之變更的空間大小。比方說，若您製作了一個快照並將原始快照完整覆蓋過去的話，快照至少必須和原始卷冊一樣大，如此一來才能夠儲存變更。您需要根據預期的變更程度來設定快照的大小。比方說，一個大部分時間為唯讀的卷冊（例如 `/usr`）的短暫 snapshot 所需要的空間，會比將被大量寫入的卷冊（例如 `/home`）的長續 snapshot 要來得多。

若快照滿出的話，該快照便會無效，因為它將已無法再追蹤原始卷冊上的變更。您應該定時監控快照的大小。快照大小能夠完全地被重設，不過，若您有足夠的儲存容量您便能增加快照卷冊的大小來避免它被 drop 掉。相反的，若您發現快照卷冊的大小比您所需的還要大，您可減少該卷冊的大小來釋出其它邏輯卷冊所需要的空間。

當您建立了快照檔案系統時，原始裝置的完整讀取和寫入存取權限還是可被保留。若快照上有一小區塊受到變更的話，該區塊會被標記並且永遠不會被由原始卷冊中複製出去。

快照功能有幾個用處：

- 快照功能基本上最常在您需要在邏輯卷冊上進行備份而不影響到持續進行資料更新的即時系統的情況下使用。
- 您可在一個快照檔案系統上執行 `fsck` 這項指令來檢查檔案系統的整合性並判斷原始的檔案系統是否需要進行檔案系統修復。
- 因為快照為 `read/write`，因此您可透過製作一個 snapshot 並針對於該 snapshot 執行測試的方式來使用應用程式針對於生產資料進行測試，並且完全不動到真實的資料。
- 您可建立卷冊來與 Xen 虛擬機器監控程式一起使用。您可使用快照功能來建立一個磁碟映像檔，對它進行一個 snapshot 並修改特定 `domU instance` 的快照。接著您便可建立另一個快照並修改另一個 `domU instance`。因為唯一被使用到的儲存容量只有在原始裝置或快照上遭到變更的區塊，因此卷冊絕大部分都會被共享。

LVM 管理總覽

This chapter provides an overview of the administrative procedures you use to configure LVM logical volumes. This chapter is intended to provide a general understanding of the steps involved. For specific step-by-step examples of common LVM configuration procedures, see [章 5, LVM 配置範例](#).

For descriptions of the CLI commands you can use to perform LVM administration, see [章 4, 透過 CLI 指令來進行 LVM 管理](#). Alternately, you can use the LVM GUI, which is described in [章 7, 利用 LVM GUI 來進行 LVM 管理](#).

3.1. 在叢集中建立 LVM 卷冊

To create logical volumes in a cluster environment, you use the Clustered Logical Volume Manager (CLVM), which is a set of clustering extensions to LVM. These extensions allow a cluster of computers to manage shared storage (for example, on a SAN) using LVM. In order to use CLVM, the Red Hat Cluster Suite software, including the `clvmd` daemon, must be started at boot time, as described in [節 1.3, “叢集邏輯卷測管理員 \(Clustered Logical Volume Manager, CLVM\)”](#).

在叢集環境中建立 LVM 邏輯卷冊和在單節點上建立 LVM 邏輯卷冊相似。LVM 指令本身以及 LVM GUI 介面並沒有不同。若要啓用您在叢集中所建立的 LVM 卷冊，該叢集設備必須要是執行中並且為 `quorate`。

CLVM requires changes to the `lvm.conf` file for cluster-wide locking. Information on configuring the `lvm.conf` file to support clustered locking is provided within the `lvm.conf` file itself. For information about the `lvm.conf` file, see [附錄 B, LVM 配置檔案](#).

By default, logical volumes created with CLVM on shared storage are visible to all systems that have access to the shared storage. It is possible to create volume groups in which all of the storage devices are visible to only one node in the cluster. It is also possible to change the status of a volume group from a local volume group to a clustered volume group. For information, see [節 4.3.2, “在叢集中建立卷冊群組”](#) and [節 4.3.7, “更改卷冊群組的參數”](#)



警告

當您在共享儲存裝置上透過使用 CLVM 來建立卷冊群組時，您必須確認叢集中所有的節點皆可存取組成該卷冊群組的實體卷冊。非對稱式的叢集配置（也就是有些節點可存取儲存裝置，有些節點則無法存取儲存裝置）不受到支援。



注意

若要使用用於 Red Hat Cluster Suite 中的共享儲存裝置，您必須要執行叢集邏輯卷冊管理員 daemon (cluster logical volume manager daemon, clvmd) 或是 High Availability Logical Volume Management (HA-LVM)。若基於作業因素您無法使用 clvmd daemon 或是 HA-LVM，或是因為您沒有正確的權限，您絕不可在共享的磁碟上使用單 instance 的 LVM，因為這可能會造成資料損毀。若您有任何疑問，請和您的 Red Hat 服務人員取得聯繫。

欲取得有關於如何安裝 Red Hat Cluster Suite 與設定叢集設備的相關資訊，請查看 [配置和管理 Red Hat Cluster](#)。

For an example of creating a mirrored logical volume in a cluster, see [節 5.5, “在叢集中建立鏡像 LVM 邏輯卷冊”](#)。

3.2. 邏輯卷冊建立總覽

下列為建立 LVM 邏輯卷冊所需步驟的概要。

1. 初始化您將會使用來作為實體卷冊的 LVM 卷冊分割區（這會將它們標記下來）。
2. 建立卷冊群組。
3. 建立邏輯卷冊

建立了邏輯卷冊之後，您便可建立並掛載檔案系統。此文件中的範例使用了 GFS 檔案系統。

1. 藉由 `gfs_mkfs` 指令來在邏輯卷冊上建立一個 GFS 檔案系統。
2. 透過 `mkdir` 指令來建立新的掛載點。在叢集的系統中，請在叢集中所有的節點上建立掛載點。
3. 掛載檔案系統。您可能會希望為系統中的各個節點附加一行行列至 `fstab` 檔案中。

另外，您可使用 LVM GUI 來建立與掛載 GFS 檔案系統。

LVM 的建立基於各別的機器，因為 LVM 設定資訊的儲存位置位於實體卷冊上而不是卷冊所被建立於的機器上。使用該儲存裝置的伺服器含有本地副本，不過能藉由實體卷冊上所含有的內容重新建立。若 LVM 版本相容的話，您可將實體卷冊連至一個不同的伺服器上。

3.3. 在邏輯卷冊上遞增檔案系統

若要在邏輯卷冊上遞增檔案系統的話，請執行下列步驟：

1. 建立一個新的實體卷冊
2. 將包含著邏輯卷冊以及您所希望遞增的檔案系統的卷冊群組延伸來包含新的實體卷冊。
3. 將邏輯卷冊延伸來包含新的實體卷冊。
4. 遞增檔案系統。

若您的卷冊群組中含有足夠的未使用空間，您可使用這個空間來延伸邏輯卷冊而不用進行步驟 1 和 2。

3.4. 邏輯卷冊備份

Metadata 的備份和 archive 會在每個卷冊群組和邏輯卷冊配置遭到變更時自動被建立，除非在 `lvm.conf` 檔案中停用。就預設值，metadata 的備份儲存於 `/etc/lvm/backup` 檔案中，並且 metadata archive 則儲存在 `/etc/lvm/archive` 檔案中。Metadata archive 儲存在 `/etc/lvm/archive` 中的保留時間長短和 archive 檔案的數量取決於 `lvm.conf` 檔案中的參數。每日的系統備份都應包含著 `/etc/lvm` 目錄的內容於備份中。

請注意，metadata 備份不會備份包含在邏輯卷冊中的用戶與系統資料。

You can manually back up the metadata to the `/etc/lvm/backup` file with the `vgcfgbackup` command. You can restore metadata with the `vgcfgrestore` command. The `vgcfgbackup` and `vgcfgrestore` commands are described in [節 4.3.12](#), “[備份卷冊群組的 Metadata](#)”.

3.5. 記錄

所有輸出的訊息都會通過一個記錄模組並含有針對於下列的各別記錄層級選項：

- 標準輸出/錯誤
- syslog
- 日誌檔案
- 外部記錄功能

The logging levels are set in the `/etc/lvm/lvm.conf` file, which is described in [附錄 B](#), [LVM 配置檔案](#).

透過 CLI 指令來進行 LVM 管理

本章節包含了透過 LVM 指令列介面 (CLI) 來建立與維護邏輯卷冊的各別管理作業的概述。



注意

If you are creating or modifying an LVM volume for a clustered environment, you must ensure that you are running the clvmd daemon. For information, see [節 3.1](#), “在叢集中建立 LVM 卷冊”。

4.1. 使用 CLI 指令

所有 LVM CLI 指令都有幾個通用的功能。

當在指令列引數中需要使用到大小時，單位可被明確地指定。若您不指定單位的話，那麼預設值便會被使用，一般會是 KB 或是 MB。LVM CLI 指令不接受分數。

當在指令列引數中指定單位時，LVM 並不區分大小寫；比方說，指定 M 或 m 都是相等的，在此情況下 2 進位 (1024 的倍數) 會被使用。不過，當在指令中指定了 `--units` 引數時，小寫表示單位為 1024 的倍數，而大寫則表示單位為 1000 的倍數。

當指令以卷冊群組或是邏輯卷冊名稱來作為引數時，完整路徑名稱將會是任選的。一個稱為 `vg0` 的卷冊群組中的 `lvo10` 邏輯卷冊可被指定為 `vg0/lvo10`。當需要一系列必要的卷冊群組，不過卻被保留為空白時，一系列含有所有卷冊群組的清單將會被用來代替。當需要一系列邏輯卷冊時，不過被提供的卻是卷冊群組時，該卷冊群組中的所有邏輯卷冊便會被帶入。比方說，`lvdisplay vg0` 指令將會顯示 `vg0` 卷冊群組中所有的邏輯卷冊。

所有的 LVM 指令都接受 `-v` 引數，您可多重輸入該引數來增加輸出的詳細度。比方說，下列範例顯示了 `lvcreate` 指令的預設輸出。

```
# lvcreate -L 50MB new_vg
Rounding up size to full physical extent 52.00 MB
Logical volume "lvo10" created
```

下列指令顯示了 `lvcreate` 指令以及 `-v` 引數的輸出。

```
# 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 lvo10
Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
Found volume group "new_vg"
Creating new_vg-lvo10
Loading new_vg-lvo10 table
Resuming new_vg-lvo10 (253:2)
Clearing start of logical volume "lvo10"
Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
Logical volume "lvo10" created
```

您也能使用 `-vv`、`-vvv` 或是 `-vvvv` 引數來顯示更加詳細的指令執行資訊。目前，`-vvvv` 引數提供了大量的資訊。下列範例僅顯示了 `lvcreate` 指令以及 `-vvvv` 引數的輸出的前幾個行列。

```
# 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
...
```

您可透過使用指令的 `--help` 引數來顯示任何 LVM CLI 指令的協助畫面。

```
commandname --help
```

若要顯示某項指令的 man page，請執行 `man` 指令：

```
man commandname
```

`man lvm` 指令提供了有關於 LVM 的一般線上資訊。

All LVM objects are referenced internally by a UUID, which is assigned when you create the object. This can be useful in a situation where you remove a physical volume called `/dev/sdf` which is part of a volume group and, when you plug it back in, you find that it is now `/dev/sdk`. LVM will still find the physical volume because it identifies the physical volume by its UUID and not its device name. For information on specifying the UUID of a physical volume when creating a physical volume, see see [節 6.4, “復原實體卷冊的 Metadata”](#) .

4.2. 實體卷冊管理

此部份描述了進行各種實體卷冊管理的指令。

4.2.1. 建立實體卷冊

下列小部分描述了幾項使用來建立實體卷冊的指令。

4.2.1.1. 設定分割區類型

若您要使用整個磁碟裝置來作為您的實體卷冊，該磁碟上必須沒有分割表。對於 DOS 磁碟分割區來講，分割區 id 應被透過使用 `fdisk` 或是 `cdisk` 指令來設為 `0x8e`。當使用整個磁碟裝置時，只有分割表必須被清除掉，如此一來便能有效地清除該磁碟上的所有資料。您可藉由使用下列指令來將第一個磁區化零以便移除現有的分割表：

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

4.2.1.2. 初始化實體卷冊

您可使用 `pvccreate` 指令來初始化一個將被用來作為實體卷冊的區塊裝置。初始化和格式化檔案系統是相似的。

下列指令會初始化 `/dev/sdd1`、`/dev/sde1` 以及 `/dev/sdf1` 來作為 LVM 實體卷冊。

```
pvccreate /dev/sdd1 /dev/sde1 /dev/sdf1
```

若要初始化分割區，而不是整個磁碟：請在分割區上執行 `pvccreate` 指令。下列範例會將 `/dev/hdb1` 分割區初始化為一個 LVM 實體卷冊，並作為事後 LVM 邏輯卷冊的一部分來使用。

```
pvccreate /dev/hdb1
```

4.2.1.3. 掃描區塊裝置

您可透過 `lvmdiskscan` 指令來掃描可用來作為實體卷冊的區塊裝置，如下列範例所示。

```
# 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/Vo1Group00/LogVo101 [    512.00 MB]
/dev/ram2          [      16.00 MB]
/dev/new_vg/lvo10  [      52.00 MB]
/dev/ram3          [      16.00 MB]
/dev/pk1_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
```

4.2.2. 顯示實體卷冊

您可使用三項指令來顯示 LVM 實體卷冊的內容：pvs、pvdisplay 以及 pvscan。

The pvs command provides physical volume information in a configurable form, displaying one line per physical volume. The pvs command provides a great deal of format control, and is useful for scripting. For information on using the pvs command to customize your output, see 節 4.9, “LVM 的自訂化回報”。

pvdisplay 指令會為各個實體卷冊提供詳細的多行輸出。它會以一個固定的格式來顯示實體的內容（大小、扇區、卷冊群組等等）。

下列範例顯示了針對於單獨實體卷冊所執行的 pvdisplay 指令的輸出。

```
# 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-X08M-mcpsVe
```

pvscan 指令會掃描系統中所有受支援的 LVM 區塊裝置來作為實體卷冊。

下列指令顯示了所有發現的實體裝置：

```
# pvscan
PV /dev/sdb2  VG vg0  1vm2 [964.00 MB / 0 free]
PV /dev/sdc1  VG vg0  1vm2 [964.00 MB / 428.00 MB free]
PV /dev/sdc2  VG vg0  1vm2 [964.84 MB]
Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

You can define a filter in the lvm.conf so that this command will avoid scanning specific physical volumes. For information on using filters to control which devices are scanned, see 節 4.6, “透過過濾器來控制 LVM 裝置掃描 (LVM Device Scans)”。

4.2.3. 避免配置於實體卷冊上

您可透過使用 pvchange 指令來避免將實體扇區分配在一個或更多實體卷冊的可用空間上。若有磁碟錯誤或是若您要將實體卷冊移除的話，這將會是必要的。

下列指令會使實體扇區無法分配在 /dev/sdk1 上。

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

您也可使用 `pvchange` 指令的 `-xy` 引數來允許分配。

4.2.4. 重設實體卷冊的大小

若您基於任何理由需要更改一個區塊裝置的大小的話，請使用 `pvresize` 指令來將 LVM 更改為新的大小。您可在 LVM 使用實體卷冊時執行這項指令。

4.2.5. 移除實體卷冊

若 LVM 已不再需要使用某個裝置，您可透過 `pvremove` 指令來將 LVM 標籤移除掉。執行 `pvremove` 指令可將一個空的實體卷冊上的 LVM metadata 零化掉。

If the physical volume you want to remove is currently part of a volume group, you must remove it from the volume group with the `vgreduce` command, as described in [節 4.3.6](#), “由卷冊群組中移除實體卷冊”。

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

4.3. 卷冊群組管理

此部份描述了執行各種卷冊群組管理的指令。

4.3.1. 建立卷冊群組

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

下列指令會建立一個稱為 `vg1` 的卷冊群組，它將會包含 `/dev/sdd1` 和 `/dev/sde1` 這兩個實體卷冊。

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

當實體卷冊被使用來建立卷冊群組時，就預設值它的磁碟空間會被平分為幾個 4MB 扇區。這個扇區為邏輯卷冊大小可被遞增或遞減的最小值。大量扇區將不會對於 I/O 邏輯卷冊的效能造成任何影響。

若是預設扇區大小不適合的話，您可透過使用 `vgcreate` 指令的 `-s` 選項來指定扇區的大小。您亦可透過使用 `vgcreate` 指令的 `-p` 和 `-l` 引數來限制卷冊群組能擁有的實體或邏輯卷冊數量。

就預設值，卷冊群組會根據一般常規（例如不將平行磁帶放置在相同實體卷冊上）來分配實體扇區。這是 `normal` 分配政策。您可使用 `vgcreate` 指令的 `--alloc` 引數來指定 `contiguous`、`anywhere` 或 `cling` 這些分配政策。

`contiguous` 政策會需要新的扇區和現有的扇區相鄰。若有足夠的可用扇區來滿足分配請求，不過 `normal` 分配政策卻不使用它們的話，`anywhere` 分配政策則會使用它們，儘管將兩個磁帶放置在相同的實體卷冊上會降低效能。`cling` 政策會將新的扇區放置在與現有扇區（位於邏輯卷冊的相同磁帶中）相同的實體卷冊上。這些政策可透過使用 `vgchange` 指令來進行變更。

一般來講，normal 以外的分配政策只有在特殊的情況下（當您需要指定不尋常，或是非標準扇區分配時）才需要被使用到。

LVM 卷冊群組和基本的邏輯卷冊包含在位於 /dev 目錄中的裝置特殊檔案目錄樹中，格式如下：

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

比方說，若您建立了兩個卷冊群組 myvg1 和 myvg2，各個都含有三個稱為 lv01、lv02 和 lv03 的邏輯卷冊，這將會建立六個裝置特殊檔案：

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

LVM 的裝置大小在 64 位元 CPU 上最大值為 8 百萬兆位元組（Exabytes）。

4.3.2. 在叢集中建立卷冊群組

在叢集環境中，您可透過 vgcreate 指令來建立卷冊群組，就如同您在單獨的節點上建立它們一樣。

就預設值，透過 CLVM 在共享儲存裝置上所建立的卷冊群組能讓所有可存取該共享儲存裝置的電腦看見。不過您也可透過使用 vgcreate 指令的 -c n 選項來建立一個本機、只有叢集中的一個節點可看見的卷冊群組。

當在叢集環境中執行下列指令時，它會在指令被執行的節點上建立一個本機卷冊群組。這項指令會建立一個名為 vg1 並且包含著實體卷冊 /dev/sdd1 和 /dev/sde1 的本機卷冊。

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

You can change whether an existing volume group is local or clustered with the -c option of the vgchange command, which is described in [節 4.3.7](#)，[“更改卷冊群組的參數”](#)。

您可透過 vgs 指令來檢查一個現有的卷冊群組是否是個叢集卷冊群組，若卷冊已在叢集中，這項指令便會顯示 c 屬性。下列指令顯示了卷冊群組 VolGroup00 和 testvg1 的屬性。在此範例中，我們可藉由 Attr 標頭下的 c 屬性來得知，VolGroup00 尚未被納入叢集中，而 testvg1 則已納入叢集中。

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

For more information on the vgs command, see [節 4.3.4](#)，[“顯示卷冊群組”](#) [節 4.9](#)，[“LVM 的自訂化回報”](#)，and the vgs man page.

4.3.3. 新增實體卷冊至卷冊群組中

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

下列指令會將 `/dev/sdf1` 實體卷冊新增至卷冊群組 `vg1` 中。

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

4.3.4. 顯示卷冊群組

您可使用兩項指令來顯示 LVM 卷冊群組的內容：`vgs` 和 `vgdisplay`。

The `vgscan` command, which scans all the disks for volume groups and rebuilds the LVM cache file, also displays the volume groups. For information on the `vgscan` command, see [節 4.3.5, “掃描磁碟來找尋卷冊群組以便建立快取檔案”](#)。

The `vgs` command provides volume group information in a configurable form, displaying one line per volume group. The `vgs` command provides a great deal of format control, and is useful for scripting. For information on using the `vgs` command to customize your output, see [節 4.9, “LVM 的自訂化回報”](#)。

`vgdisplay` 指令會以固定的格式來顯示卷冊群組內容（例如大小、扇區、實體卷冊數量等等）。下列範例顯示了 `vgdisplay` 指令針對於 `new_vg` 卷冊群組的輸出。若您不指定卷冊群組的話，所有現有的卷冊群組就會被顯示出。

```
# 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-ZKk0-0pM0-0118-nlw0-wwqd-fD5D32
```

4.3.5. 掃描磁碟來找尋卷冊群組以便建立快取檔案

`vgscan` 指令會掃描系統中所有受支援的磁碟裝置並尋找 LVM 實體卷冊和卷冊群組。這會將 LVM 快取建立在 `/etc/lvm/.cache` 檔案中，該檔案含有一列現有的 LVM 裝置清單。

LVM 會在系統啟動時以及 LVM 作業時的其它時候（例如當您執行一項 `vgcreate` 指令或是當 LVM 偵測到不一致的情況時）自動地執行 `vgscan` 指令。當您更改您的硬體設定時，您可能需要手動式地執行

`vgscan` 指令，這會使得系統能看見系統開機時所不存在的新裝置。這可能會是必要的，例如當您新增磁碟至一部 SAN 上的系統或是當您熱插拔一個被標記為實體卷冊的新磁碟時。

You can define a filter in the `lvm.conf` file to restrict the scan to avoid specific devices. For information on using filters to control which devices are scanned, see [節 4.6, “透過過濾器來控制 LVM 裝置掃描 \(LVM Device Scans\)”](#).

下列範例顯示了一項 `vgscan` 指令的輸出。

```
# 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. 由卷冊群組中移除實體卷冊

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

在由一個卷冊群組中移除實體卷冊之前，您可藉由使用 `pvdisplay` 指令來確認實體卷冊並未被任何邏輯卷冊使用中。

```
# 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-M0kn-76iP-iftz-0VSen7
```

若實體卷冊依然使用中，您便必須透過使用 `pvmove` 指令來將資料遷移至另一個實體卷冊上。然後使用 `vgreduce` 指令來移除掉實體卷冊：

下列指令會將實體卷冊 `/dev/hda1` 由 `my_volume_group` 卷冊群組中移除。

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

4.3.7. 更改卷冊群組的參數

The `vgchange` command is used to deactivate and activate volume groups, as described in [節 4.3.8, “啟用和停用卷冊群組”](#). You can also use this command to change several volume group parameters for an existing volume group.

下列指令會將卷冊群組 `vg00` 的最大邏輯卷冊數量更改為 128。

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

欲取得 `vgchange` 指令所能更改的卷冊群組參數的相關資訊，請參閱 `vgchange(8) man page`。

4.3.8. 啓用和停用卷冊群組

當您建立卷冊群組時，就預設值它會是被啓用的。這代表在該群組中的邏輯卷冊可被存取並且可能會有變更。

在幾種情況下您將需要停用卷冊群組，並使 `kernel` 無法偵測到該卷冊群組。若要停用或啓用卷冊群組，請使用 `vgchange` 指令的 `-a` (`--available`) 引數。

下列範例將會停用卷冊群組 `my_volume_group`。

```
vgchange -a n my_volume_group
```

若叢集鎖定被啓用的話，請附加「e」來將卷冊群組啓用或停用於一個專屬的節點上，或是附加「1」來啓用或停用本機節點上的卷冊群組。只有單獨主機快照的邏輯卷冊總是會特別地被啓用，因為它們一次只能使用於一個節點上。

You can deactivate individual logical volumes with the `lvchange` command, as described in [節 4.4.4, “更改邏輯卷冊群組的參數”](#), For information on activating logical volumes on individual nodes in a cluster, see [節 4.8, “在叢集中啓用各別節點上的邏輯卷冊”](#).

4.3.9. 移除卷冊群組

若要移除不包含邏輯卷冊的卷冊群組，請使用 `vgremove` 指令。

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

4.3.10. 分割卷冊群組

若要分割卷冊群組的實體卷冊並建立新的卷冊群組，請使用 `vgsplit` 指令。

邏輯卷冊無法在邏輯群組之間分割。所有現有的邏輯卷冊都必須整個位於實體卷冊上並形成舊的或新的卷冊群組。不過若有必要的話，您可使用 `pvmove` 指令來強制分割。

下列範例由原本的 `bigvg` 卷冊群組分割出新的卷冊群組 `smallvg`。

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

4.3.11. 結合卷冊群組

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

下列指令會將未啓用的卷冊群組 `my_vg` 合併入啓用或未啓用的卷冊群組 `databases` 中，並提供詳細的 runtime 資訊。

```
vgmerge -v databases my_vg
```

4.3.12. 備份卷冊群組的 Metadata

Metadata 備份和 `archive` 會在每個卷冊群組和邏輯卷冊配置遭到變更時被自動地建立，除非您在 `lvm.conf` 檔案中將它停用。就預設值，metadata 備份儲存在 `/etc/lvm/backup` 檔案中，並且 metadata `archive` 則儲存在 `/etc/lvm/archives` 檔案中。您可透過 `vgcfgbackup` 指令來手動式地將 metadata 備份至 `/etc/lvm/backup` 檔案中。

`vgcfrestore` 指令會將卷冊群組的 metadata 由 `archive` 復原至卷冊群組中所有的實體卷冊中。

For an example of using the `vgcfrestore` command to recover physical volume metadata, see [節 6.4, “復原實體卷冊的 Metadata”](#).

4.3.13. 為卷冊群組重新命名

使用 `vgrename` 指令來為現有的卷冊群組重新命名。

下列兩項指令皆可將現有的卷冊群組重新由 `vg02` 命名為 `my_volume_group`

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

```
vgrename vg02 my_volume_group
```

4.3.14. 將卷冊群組移至另一部系統

您可將整個 LVM 卷冊群組移至另一部系統上。我們建議您使用 `vgexport` 和 `vgimport` 指令來這麼作。

`vgexport` 指令會使得系統無法存取一個未啓用的卷冊群組，這能讓您將它的實體卷冊拆卸。`vgimport` 指令會使得卷冊群組在由 `vgexport` 指令使其停用後能再次地被系統存取。

若要將卷冊群組由一部系統移至另一部系統，請執行下列步驟：

1. 請確認沒有用戶正在存取卷冊群組中啓用中卷冊上的檔案，然後再將邏輯卷冊卸載。
2. 請使用 `vgchange` 指令的 `-a n` 引數來將卷冊群組標記為停用，這可避免在卷冊群組上再有任何活動進行。

3. 請使用 `vgexport` 指令來匯出卷冊群組。這可避免它被您要從之移除的系統存取它。

當您匯出了卷冊群組後，當您執行 `pvsan` 指令時，實體卷冊便會顯示在一個已匯出的卷冊群組中。

```
[root@tng3-1]# pvsan
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]
...
```

當系統關閉後，您便可將構成卷冊群組的磁碟拔除然後將它們連接至新的系統上。

4. 當磁碟被插入新的系統上時，請使用 `vgimport` 指令來匯入卷冊群組，並使新的系統能夠存取它。
5. 請使用 `vgchange` 指令的 `-a y` 引數來啓用卷冊群組。
6. 掛載檔案系統來使它能被使用。

4.3.15. 重新建立一個卷冊群組目錄

若要重新建立卷冊群組目錄以及邏輯卷冊特殊檔案，請使用 `vgmknodes` 指令。這項指令會檢查啓用邏輯卷冊所需的 `/dev` 目錄中的 LVM2 特殊檔案。它會建立任何遺失的特殊檔案，並移除未被使用的特殊檔案。

您可藉由使用 `vgscan` 指令時指定 `mknodes` 引數至指令來將 `vgmknodes` 指令合併入 `vgscan` 指令。

4.4. 邏輯卷冊管理

此部份描述了執行可進行各種邏輯卷冊管理的各項指令。

4.4.1. 建立邏輯卷冊

若要建立邏輯卷冊，請使用 `lvcreate` 指令。您可建立線性卷冊 (linear volumes)、等量卷冊 (striped volumes) 以及鏡像卷冊 (mirrored volumes)，如下列小部份中所示。

若您不為邏輯卷冊指定一組名稱的話，預設的 `lv01#` 就會被使用，並且 `#` 為邏輯卷冊的內部號碼。

下列部份提供了您可透過 LVM 建立的三種邏輯卷冊的邏輯卷冊建立範例。

4.4.1.1. 建立線性卷冊

當您建立邏輯卷冊時，邏輯卷冊會由一個使用實體卷冊上可用扇區所組成的卷冊群組來形成。一般來講，邏輯卷冊會使用掉基本實體卷冊上的所有可用空間。修改邏輯卷冊可空出和重新分配實體卷冊中的空間。

下列指令將會在 `vg1` 卷冊群組中建立大小為 10GB 的邏輯卷冊。

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

下列指令將會在 `testvg` 卷冊群組中建立一個 1500MB、名為 `testlv` 的線性卷冊，並建立 `/dev/testvg/testlv` 這個區塊裝置。

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

下列指令會由 `vg0` 卷冊群組中的可用扇區建立一個 50GB、名為 `gfs1v` 的邏輯卷冊。

```
lvcreate -L 50G -n gfs1v vg0
```

您可使用 `lvcreate` 指令的 `-l` 引數來指定扇區中的邏輯卷冊大小。您也能使用此引數來指定使用於邏輯卷冊的卷冊群組比例。下列指令將會建立一個名為 `mylv` 的邏輯卷冊，它使用了卷冊群組 `testvol` 中 60% 的總空間。

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

您也能使用 `lvcreate` 指令的 `-l` 引數來將卷冊群組中的剩下的可用空間比例作為邏輯卷冊的大小。下列指令將會建立一個名為 `yourlv` 的邏輯卷冊，它將會使用卷冊群組 `testvol` 中所有未分配的空間。

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

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

下列指令將會建立一個名為 `mylv` 的邏輯卷冊，該邏輯卷冊會將名為 `testvg` 的卷冊群組填滿。

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

The underlying physical volumes used to create a logical volume can be important if the physical volume needs to be removed, so you may need to consider this possibility when you create the logical volume. For information on removing a physical volume from a volume group, see [節 4.3.6](#)，[“由卷冊群組中移除實體卷冊”](#)。

若要建立一個由卷冊群組中的實體卷冊所分配的邏輯卷冊，請在 `lvcreate` 指令列後方指定實體卷冊或卷冊。下列指令將會在 `testvg` 卷冊群組中建立一個由實體卷冊 `/dev/sdgl` 分配並稱為 `testlv` 的邏輯卷冊。

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

您可指定實體卷冊的哪些扇區可用來作為邏輯卷冊。下列範例透過了卷冊群組 `testvg` 中的實體卷冊 `/dev/sda1` 中的扇區 0 至 25，以及實體卷冊 `/dev/sdb1` 中的扇區 50 至 125 建立了一個線性邏輯卷冊。

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

下列範例由 `/dev/sda1` 實體卷冊的扇區 0 至 25 建立了一個線性邏輯卷冊，然後在扇區 100 繼續編排邏輯卷冊。

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

The default policy for how the extents of a logical volume are allocated is `inherit`, which applies the same policy as for the volume group. These policies can be changed using the `lvchange` command. For information on allocation policies, see [節 4.3.1, “建立卷冊群組”](#)。

4.4.1.2. 建立等量卷冊

For large sequential reads and writes, creating a striped logical volume can improve the efficiency of the data I/O. For general information about striped volumes, see [節 2.3.2, “等量邏輯卷冊”](#)。

當您建立等量邏輯卷冊時，您可透過 `lvcreate` 指令的 `-i` 引數來指定磁條的數量。這可決定邏輯卷冊需要被分散至多少實體卷冊上。磁條的數量不可大於卷冊群組中的實體卷冊數量（除非使用了 `--alloc anywhere` 引數）。

若構成等量邏輯卷冊的基本實體裝置的大小不同的話，等量卷冊的最大大小便會取決於最小的基本裝置。比方說，在一個 `two-legged` 的磁條中，最大大小將會等於較小裝置大小的兩倍。在一個 `three-legged` 的磁條中，最大大小則等於最小裝置大小的三倍。

下列指令將會在兩個實體卷冊上建立等量的邏輯卷冊，並且磁帶為 64KB。邏輯卷冊大小為 50GB、名為 `gfslv`，並由卷冊群組 `vg0` 中所分割出。

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

就與 `linear` 卷冊相同，您可指定您將會使用於磁條的實體卷冊的扇區。下列指令將會在兩個實體卷冊上建立一個大小為 100 扇區的等量卷冊，名為 `stripelv` 並位於卷冊群組 `testvg` 中。該磁條會使用 `/dev/sda1` 的磁區 0-50 以及 `/dev/sdb1` 的磁區 50-100。

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

4.4.1.3. 建立鏡像卷冊

當您要建立鏡像卷冊時，您需要以 `lvcreate` 指令的 `-m` 引數來指定複製的資料數量。指定 `-m1` 的話會建立一個鏡像，並產生出兩份檔案系統：一個 `linear` 邏輯卷冊和一個副本。相同地，指定了 `-m2` 的話會建立兩個鏡像，並產生出三份檔案系統。

下列指令會建立一個單鏡像的鏡像的鏡像邏輯卷冊。卷冊大小為 50GB，名為 `mirrorlv`，並由卷冊群組 `vg0` 中所分割出：

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

LVM 鏡像會將要被複製的裝置分為不同的區域，就預設值大小會是 512KB。您可使用 `lvcreate` 指令的 `-R` 引數來以 MB 為單位指定區域大小。您亦可藉由編輯 `lvm.conf` 檔案中的 `mirror_region_size` 設定來更改預設的區域大小。

請注意

基於叢集架構中的限制，大於 1.5TB 的叢集鏡像無法以 512KB 的預設區域大小來建立。需要較大鏡像的使用者應將預設大小增加。若沒有增加區域大小的話將會造成 LVM 的建立程序停擺，並且可能會使其它 LVM 指令也停止運作。

就一般指導綱要來說，當要為大於 1.5TB 的鏡像指定區域大小時，您可將您的鏡像大小以 TB 來進行二進位的四捨五入，並使用這組數字來作為 `lvcreate` 指令的 `-R` 引數。比方說，若您的鏡像大小為 1.5TB，您可指定 `-R 2`。若您的鏡像大小為 3TB，您可指定 `-R 4`。當鏡像大小為 5TB 時，您則能夠指定 `-R 8`。

下列指令將會建立區域大小為 2MB 的鏡像邏輯卷冊：

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

LVM 含有一個小型的日誌，它會使用該日誌來記錄哪個區域和鏡像已同步化。就預設值，這個日誌會被存放在磁碟上，使其在系統重新啟動後依然可保留設定。您亦可透過 `--corelog` 引數來指定將此日誌存放在記憶體中；這可省略使用額外的日誌裝置。不過若要如此，每當系統重新啟動時，整個鏡像就必須被重新同步化。

下列指令將會由 `bigvg` 卷冊群組建立鏡像邏輯卷冊。該邏輯卷冊名為 `ondiskmirvol` 並且含有單鏡像。卷冊大小為 12MB，並且會將 `mirror log` 存放在記憶體中。

```
# lvcreate -L 12MB -ml --corelog -n ondiskmirvol bigvg
Logical volume "ondiskmirvol" created
```

`mirror log` 會被建立在一個與任何 `mirror leg` 都會被建立於的裝置區隔開的裝置上。不過您亦可透過使用 `vgcreate` 指令的 `--alloc anywhere` 引數來在和其中一個 `mirror leg` 的裝置相同的裝置上建立一個 `mirror log`。這可能會使效能降低，不過它卻能讓您建立鏡像，儘管您只有兩個基本裝置。

下列指令會建立一個單鏡像的鏡像邏輯卷冊，它的 `mirror log` 位於與其中一個 `mirror leg` 相同的裝置上。在此範例中，`vg0` 這個卷冊群組只包含著兩個裝置。這項指令會在 `vg0` 卷冊群組中建立大小為 500MB、名為 `mirrorlv` 的鏡像卷冊。

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

當鏡像被建立後，鏡像區域會被同步化。對於大型的鏡像元件來說，同步程序可能會花上許多時間。當您要建立一個無須再生的新鏡像時，您可指定 `nosync` 引數來顯示來自於第一個裝置的初始同步化是非必要的。

您可指定使用來作為 `mirror log` 和日誌的裝置為何，以及該使用裝置的哪個扇區。若要強制將日誌存放在某個特定磁碟上，請在磁碟上確切指定一個要存放的扇區。LVM 並不一定會遵照指令列中所列出的裝置順序。若有任何實體卷冊被列出的話，那將會是唯一被進行分配的空間。任何包含在清單中並且已被分配的實體扇區都會被忽略掉。

下列指令會建立一個含有單獨鏡像的鏡像邏輯卷冊。卷冊大小為 500MB，名為 `mirrorlv`，並且是由卷冊群組 `vg0` 中所切割出來的。鏡像的第一個 `leg` 位於 `/dev/sda1` 裝置上，鏡像的第二個 `leg` 位於 `/dev/sdb1` 裝置上，`mirror logs` 則位於 `/dev/sdc1` 上。

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

下列指令會建立一個含有單獨鏡像的鏡像邏輯卷冊。卷冊大小為 500MB，名為 `mirrorlv`，並且是由卷冊群組 `vg0` 中所切割出來的。鏡像的第一個 `leg` 位於 `/dev/sda1` 裝置的扇區 0 至 499 上，鏡像的第二個 `leg` 位於 `/dev/sdb1` 裝置的扇區 0 至 499 上，並且 `mirror log` 則是由 `/dev/sdc1` 裝置的扇區 0 上起始。這些為大小 1MB 的扇區。若有任何指定的扇區早已被分配的話，它們將會被忽略掉。

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



注意

As of the RHEL 5.3 release, mirrored logical volumes are supported in a cluster. Creating a mirrored LVM logical volume in a cluster requires the same commands and procedures as creating a mirrored LVM logical volume on a single node. However, in order to create a mirrored LVM volume in a cluster the cluster and cluster mirror infrastructure must be running, the cluster must be quorate, and the locking type in the `lvm.conf` file must be set correctly to enable cluster locking. For an example of creating a mirrored volume in a cluster, see [節 5.5, “在叢集中建立鏡像 LVM 邏輯卷冊”](#).

4.4.1.4. 更改鏡像卷冊配置

您可透過 `lvconvert` 指令來將一個邏輯卷冊由鏡像卷冊轉換為線性卷冊，或由線性卷冊轉換為鏡像卷冊。您亦可使用這項指令來重新配置現有邏輯卷冊的其它鏡像參數，例如 `corelog`。

當您將邏輯卷冊轉換為鏡像卷冊時，您基本上就是在為一個現有的卷冊建立 `mirror leg`。這代表您的卷冊群組必須包含著 `mirror leg` 以及 `mirror log` 的裝置與空間。

If you lose a leg of a mirror, LVM converts the volume to a linear volume so that you still have access to the volume, without the mirror redundancy. After you replace the leg, you can use the `lvconvert` command to restore the mirror. This procedure is provided in [節 6.3, “由 LVM 鏡像錯誤中復原”](#)。

下列指令會將線性邏輯卷冊 `vg00/lvol1` 轉換為鏡像邏輯卷冊。

```
lvconvert -ml vg00/lvol1
```

下列指令會將鏡像邏輯卷冊 `vg00/lvol1` 轉換為線性邏輯卷冊，並移除 `mirror leg`。

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

4.4.2. 一致的裝置號碼

重大與非重大裝置號碼會於模組載入時被動態式地分配。有些應用程式可因為區塊裝置總是透過相同的裝置（重大與非重大）號碼來啓用而得到最佳的效果。您可透過 `lvcreate` 和 `lvchange` 指令並使用下列引數來這麼指定：

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

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

若您希望使用 NFS 來匯出檔案系統的話，在 `exports` 檔案中指定 `fsid` 參數可能能夠避免在 LVM 中設置一致性的裝置號碼的必要性。

4.4.3. 重設邏輯卷冊大小

若要更改邏輯卷冊的大小，請使用 `lvreduce` 指令。若邏輯卷冊包含著一個檔案系統，請確認先減少檔案系統（或使用 LVM GUI）。如此一來，邏輯卷冊的大小就總是至少能與檔案系統所預期的大小相等。

下列指令會使 `vg00` 卷冊群組中的邏輯卷冊 `lv011` 的大小減少 3 個邏輯扇區。

```
lvreduce -l -3 vg00/lv011
```

4.4.4. 更改邏輯卷冊群組的參數

若要更改邏輯卷冊的參數，請使用 `lvchange` 指令。欲取得一列包含著您可更改的參數之清單，請查看 `lvchange(8) man page`。

You can use the `lvchange` command to activate and deactivate logical volumes. To activate and deactivate all the logical volumes in a volume group at the same time, use the `vgchange` command, as described in [節 4.3.7](#)，[“更改卷冊群組的參數”](#)。

下列指令會將卷冊群組 `vg00` 中的 `lv011` 卷冊的權限更改為唯讀。

```
lvchange -pr vg00/lv011
```

4.4.5. 重新為邏輯卷冊命名

若要為現有的邏輯卷冊重新命名，請使用 `lvrename` 指令。

下列兩項指令皆可將卷冊群組 `vg02` 中的邏輯卷冊 `lvold` 重新命名為 `lvnew`。

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

```
lvrename vg02 lvold lvnew
```

For more information on activating logical volumes on individual nodes in a cluster, see [節 4.8](#)，[“在叢集中啟用各別節點上的邏輯卷冊”](#)。

4.4.6. 移除邏輯卷冊

若要移除一個非啓用中的邏輯卷冊，請使用 `lvremove` 這項指令。若是邏輯卷冊目前已掛載，請先將該邏輯卷冊卸載才移除它。此外，在一個叢集環境中，您必須先將邏輯卷冊停用才可將之移除。

下列指令會將邏輯卷冊 `/dev/testvg/testlv` 由卷冊群組 `testvg` 中移除。請注意，在此情況下，邏輯卷冊並未被停用。

```
[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
```

您能夠透過 `lvchange -an` 指令來在移除某個邏輯卷冊前先將它停用，在此情況下，您將不會看見一個提示要求您驗證是否要將一個啓用中的邏輯卷冊移除掉。

4.4.7. 顯示邏輯卷冊

您可使用三項指令來顯示 LVM 邏輯卷冊的內容：`lvs`、`lvdisplay` 以及 `lvscan`。

The `lvs` command provides logical volume information in a configurable form, displaying one line per logical volume. The `lvs` command provides a great deal of format control, and is useful for scripting. For information on using the `lvs` command to customize your output, see 節 4.9, “LVM 的自訂化回報”。

`lvdisplay` 指令可利用固定的格式來顯示邏輯卷冊的內容（像是大小、格式和映射）。

下列指令顯示了 `vg00` 中的 `lvo12` 的屬性。若是已為此原始邏輯卷冊建立了快照邏輯卷冊的話，這項指令便會顯示一列含有所有快照邏輯卷冊以及其狀態（啓用中或停用中）的清單。

```
lvdisplay -v /dev/vg00/lvo12
```

`lvscan` 指令會掃秒系統中所有的邏輯卷冊並將它們列出，如下列範例。

```
# lvscan
ACTIVE                               '/dev/vg0/gfs1v' [1.46 GB] inherit
```

4.4.8. 遞增邏輯卷冊

若要增加邏輯卷冊的大小，請使用 `lvextend` 指令。

延伸了邏輯卷冊之後，您將會需要增加關聯檔案系統的大小來進行匹配。

當您延伸邏輯卷冊時，您可指定要將卷冊延伸多少或是當您將它延伸後它應該要多大。

下列指令會將邏輯卷冊 `/dev/myvg/homevol` 延伸為 12 GB。

```
# 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
```

下列指令會增加額外的 1GB 至邏輯卷冊 /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
```

就和 lvcreate 指令相同，您可使用 lvextend 指令的 -l 引數來指定扇區的數量並遞增邏輯卷冊的大小。您亦可使用此引數來指定卷冊群組的百分比，或是卷冊群組剩下可用空間的百分比。下列指令將會延伸一個名為 testlv 的邏輯卷冊，以填滿卷冊群組 myvg 中所有未分配的空間。

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

在您延伸了邏輯卷冊後，您需要增加檔案系統大小來進行匹配。

就預設值來講，大部分用來重設檔案系統大小的工具都會將檔案系統的大小遞增為基本邏輯卷冊的大小，如此一來您便無須擔心得為這兩項指令指定相同的大小。

4.4.9. 延伸等量的卷冊

若要增加等量邏輯卷冊的大小，用來製作支援 stripe 的卷冊群組的基本實體卷冊上就必須要有足夠的空間。比方說，若您有個雙向的 stripe 而它使用了所有的卷冊群組，新增一個單獨的實體卷冊至卷冊群組將無法讓您延伸該 stripe。您必須要新增至少兩個實體卷冊至卷冊群組中。

比方說，請參考一個包含著兩個基本實體卷冊的卷冊群組 vg，如下列透過 vgs 指令所示。

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

您可透過使用卷冊群組中的所有空間來建立磁條。

```
# 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)
```

請注意，卷冊群組現在已無可用空間。

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

下列指令會增加額外的實體卷冊至卷冊群組，如此一來它便會含有 135G 的額外空間。

```
# 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
```

在此情況下，您無法將等量邏輯卷冊延伸為整個卷冊群組的大小，因為需要兩個基本的裝置才可 stripe 資料。

```
# 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
```

若要延伸等量的邏輯卷冊，請新增另一個實體卷冊，然後再延伸邏輯卷冊。在此範例中，在新增了兩個實體卷冊至卷冊群組之後，我們便可將邏輯卷冊延伸為卷冊群組的完整大小。

```
# 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
```

在您沒有足夠的基本實體裝置來延伸等量邏輯卷冊的情況下，若延伸沒有被 stripe 無所謂的話，您依然還是能夠延伸該卷冊，而這將有可能造成效能上的不一致。當增加邏輯卷冊的空間時，預設的作業就是使用和現有邏輯卷冊的最後一個區段相同的 striping 參數，不過您可將這些參數置換掉。下列範例在初始的 lvextend 指令失敗後延伸了現有的等量邏輯卷冊來使用剩下的可用空間。

```
# 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 -il -l+100%FREE vg/stripe1
```

4.4.10. 縮減邏輯卷冊

若要縮減邏輯卷冊的大小，首先請將檔案系統卸載。接著您便可使用 lvreduce 指令來縮減卷冊。將卷冊縮減過後，請將檔案系統重新掛載。



警告

有一點相當重要，那就是您在縮減卷冊本身時，您一定要先將檔案系統大小或是保存在卷冊中的一切都先縮減，否則您將面臨資料遺失的危機。

縮減邏輯卷冊可空出一些卷冊群組空間來分配給該卷冊群組中的其它邏輯卷冊。

下列範例縮減了位於 `vg00` 卷冊群組中的邏輯卷冊 `lvo11` 的 3 個邏輯扇區。

```
lvreduce -l -3 vg00/lvo11
```

4.5. 建立快照卷冊 (Snapshot Volumes)

使用 `lvcreate` 指令的 `-s` 引數來建立快照卷冊。快照卷冊可被寫入。



注意

LVM 快照在叢集環境中的節點之間並不受支援。您無法在一個叢集卷冊群組中建立快照卷冊。

下列指令建立了一個大小為 100 MB 名為 `/dev/vg00/snap` 的快照邏輯卷冊。它建立了名為 `/dev/vg00/lvo11` 的原始邏輯卷冊的快照。若原始的邏輯卷冊包含了一個檔案系統，您可將快照邏輯卷冊掛載在一個任意的目錄上以便在原始檔案系統進行更新的同時存取檔案系統的內容來進行備份。

```
lvcreate --size 100M --snapshot --name snap /dev/vg00/lvo11
```

在您建立了快照邏輯卷冊後，利用 `lvdisplay` 指令來指定原始的卷冊可產生出包含著一個含有所有快照邏輯卷冊與其狀態（啟用或停用）之清單的輸出。

下列範例顯示了邏輯卷冊 `/dev/new_vg/lvo10` 的狀態，並且有個快照卷冊 `/dev/new_vg/newvgsnap` 被建立了。

```
# lvdisplay /dev/new_vg/lvo10
--- Logical volume ---
LV Name                /dev/new_vg/lvo10
VG Name                new_vg
LV UUID                LBylTz-sr23-0jsI-LT03-nHLC-y8XW-EhC178
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
```

就預設值，`lvs` 指令會顯示初始卷冊以及各個快照卷冊目前被使用到的比例。下列範例顯示了在一部包含著邏輯卷冊 `/dev/new_vg/lvo10` 的系統上輸入 `lvs` 指令的預設輸出。在此情況下，有個快照卷冊 `/dev/new_vg/newvgsnap` 被建立了。

```
# lvs
LV          VG      Attr   LSize  Origin Snap%  Move Log Copy%
lvo10       new_vg  owi-a- 52.00M
newvgsnap1  new_vg  swi-a- 8.00M  lvo10   0.20
```

**注意**

因為快照的大小會隨著原始卷冊的改變而增加，所以請記得時常透過 `lvs` 指令來監控快照卷冊的比例以確保它不會滿出。一個用滿 100% 的快照基本上一定會發生問題，因為若要寫至未修改的初始卷冊部份中一定得將該快照損毀才能成功。

4.6. 透過過濾器來控制 LVM 裝置掃描 (LVM Device Scans)

在開機時，`vgscan` 指令會被執行來掃描系統上的區塊裝置以搜尋 LVM 標籤，判斷出它們哪一個才是實體卷冊，並讀取 `metadata` 來建立一系列卷冊群組清單。實體卷冊的名稱儲存在系統中各個節點的 `cache` 檔案中 (`/etc/lvm/.cache`)。後續的指令可讀取該檔案來避免重新掃描 (`rescanning`)。

您可藉由在 `lvm.conf` 配置檔案中設定過濾器來控制 LVM 該掃描哪些裝置。`lvm.conf` 檔案中的過濾器包含著一系列會被套用至 `/dev` 目錄中的裝置名稱的基本正規表示式，以決定是否要接受或拒絕發現的各個區塊裝置。

下列範例顯示了如何使用此過濾器來控制 LVM 掃描哪些裝置。請注意，以下有些範例並不全然代表最佳作法，因為正規表示式被自由地拿來和完整的路徑名稱作比較。比方說，`a/loop/` 相當於 `a/*.loop.*/` 並且將會符合 `/dev/loopoperation/lvol1`。

下列過濾器新增了所有被發現的裝置，這是預設的特性，因為在配置檔案中未配置過濾器：

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

下列過濾器一除了 `cdrom` 裝置以避免在光碟機中沒有光碟時所造成的延緩：

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

下列過濾器新增了所有的 `loop` 並移除了所有其它的區塊裝置：

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

下列過濾器新增了所有 `loop` 和 `IDE` 並移除了所有其它的區塊裝置：

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

下列過濾器只在第一個 `IDE drive` 上新增了分割區 8 並移除了所有其它的區塊裝置：

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

For more information on the `lvm.conf` file, see [附錄 B, LVM 配置檔案](#) and the `lvm.conf(5)` man page.

4.7. 線上資料重置 (Online Data Relocation)

您可透過使用 `pvmove` 指令來在系統使用中的時候移動資料。

`pvmove` 這項指令會將要移至不同部份中的資料分開並建立一個暫時性的鏡像來移動各個部份。欲取得更多有關於 `pvmove` 指令作業上的相關資訊，請查看 `pvmove(8) man page`。

Because the `pvmove` command uses mirroring, it is not cluster-aware and needs exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see [節 4.8, “在叢集中啓用各別節點上的邏輯卷冊”](#)。

下列指令會將所有經過分配的空間由實體卷冊 `/dev/sdc1` 上移至卷冊群組中其它可使用的實體卷冊上：

```
pvmove /dev/sdc1
```

下列指令只會移動邏輯卷冊 `MyLV` 的扇區。

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

因為 `pvmove` 指令的執行可能會花上一段時間，我們建議您在背景環境（background）中執行這項指令來避免完成度更新顯示在前景環境（foreground）中。下列指令會將所有分配至實體卷冊 `/dev/sdc1` 的扇區移至背景環境中的 `/dev/sdf1`。

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

下列指令會以每五秒間隔和百分比的方式來回報移動的完成度。

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

4.8. 在叢集中啓用各別節點上的邏輯卷冊

若您在一個叢集環境下安裝了 LVM 的話，您可能有時需要特別地在某個節點上啓用邏輯卷冊。比方說，`pvmove` 這項指令無法偵測到叢集並且需要特別的卷冊存取權限。

若要特別地在某個節點上啓用邏輯卷冊，請使用 `lvchange -aey` 這項指令。另外，您亦可使用 `lvchange -aly` 指令來只在本機節點上啓用邏輯卷冊，而非特別地啓用。您之後可同時地在額外的節點上啓用它們。

You can also activate logical volumes on individual nodes by using LVM tags, which are described in [附錄 C, LVM 物件標籤 \(Object Tags\)](#). You can also specify activation of nodes in the configuration file, which is described in [附錄 B, LVM 配置檔案](#).

4.9. LVM 的自訂化回報

您可透過使用 `pvs`、`lvs` 和 `vgs` 指令來產生出簡明與可自訂化的 LVM 物件報告。這些指令所產生的報告包含著各個物件的一行輸出。各個行列都包含著一列和物件相關、經過排序的屬性的欄位。有五種方式可選擇欲回報的物件：藉由實體卷冊、卷冊群組、邏輯卷冊、實體卷冊區段，以及邏輯卷冊區段。

下列部份提供了：

- 您可使用來控制產生出的報告格式的指令引數之摘要。

- 您能為 LVM 物件選擇的欄位之清單。
- 您可使用來排序產生出的報告的指令引數之 摘。要
- 指定回報輸出單位的指示。

4.9.1. 格式控制

無論您是使用 `pvs`、`lvs` 或 `vgs` 指令都能看見顯示出的預設欄位和排序順序。您可藉由下列引數來控制這些指令的輸出：

- 您可藉由使用 `-o` 引數來改變預設顯示的欄位。比方說，下列輸出為 `pvs` 指令的預設輸出（它顯示出了有關於實體卷冊的相關資訊）。

```
# 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
```

下列指令只會顯示實體卷冊的名稱和大小。

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

- 您可透過一個加號 (+) 來將一個欄位附加至輸出，這能夠和 `-o` 引數一起組合使用。

下列範例除了會顯示預設的欄位還會顯示實體卷冊的 UUID。

```
# 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 Joq1ch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
/dev/sdd1   new_vg  lvm2 a-   17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-UqkCS
```

- 新增 `-v` 引數至一項指令可增加額外的欄位。比方說，輸入 `pvs -v` 指令的話將會顯示出預設欄位以及 `DevSize` 和 `PV UUID` 這兩個額外的欄位。

```
# 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 Joq1ch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
/dev/sdd1   new_vg  lvm2 a-   17.14G 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

- `--noheadings` 引數會將標題的行列抑制住。這對於編寫 `script` 相當有幫助。

下列範例合併使用了 `--noheadings` 和 `pv_name` 引數來產生了一系列包含著所有實體卷冊的清單。

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

```
/dev/sdc1
/dev/sdd1
```

- `--separator separator` 這個引數使用了 `separator` 來區隔了各個欄位。

下列範例透過了一個等於符號 (=) 來區分了 `pvs` 的預設輸出欄位。

```
# pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lv2=a- =17.14G=17.14G
/dev/sdc1=new_vg=lv2=a- =17.14G=17.09G
/dev/sdd1=new_vg=lv2=a- =17.14G=17.14G
```

若要在使用 `separator` 引數時讓欄位對稱的話，請合併使用 `separator` 引數以及 `--aligned` 引數。

```
# pvs --separator = --aligned
PV      =VG      =Fmt =Attr=PSize =PFree
/dev/sdb1 =new_vg=lv2=a-  =17.14G=17.14G
/dev/sdc1 =new_vg=lv2=a-  =17.14G=17.09G
/dev/sdd1 =new_vg=lv2=a-  =17.14G=17.14G
```

You can use the `-P` argument of the `lvs` or `vgs` command to display information about a failed volume that would otherwise not appear in the output. For information on the output this argument yields, see [節 6.2, “顯示錯誤裝置的相關資訊”](#)。

欲取得顯示引數的完整清單，請查看 `pvs(8)`、`vgs(8)` 以及 `lvs(8)` man page。

卷冊群組欄位能與實體卷冊（和實體卷冊區段）欄位或邏輯卷冊（和邏輯卷冊區段）欄位混合在一起，不過實體卷冊與邏輯卷冊欄位則無法混合在一起。比方說，下列指令將會顯示各個實體卷冊的一行輸出。

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

4.9.2. 物件選擇

此部份提供了一系列的表格，這些表格列出了您可透過使用 `pvs`、`vgs` 和 `lvs` 來顯示出有關於 LVM 物件的相關資訊。

為求方便，欄位名稱字首若符合指令的預設值的話便可被省略掉。比方說 `pvs` 指令，`name` 代表 `pv_name`，不過對 `vgs` 指令來講，`name` 會被解譯為 `vg_name`。

執行下列指令相當於執行 `pvs -o pv_free`。

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

4.9.2.1. pvs 指令

表格 4.1, “pvs 顯示欄位” lists the display arguments of the pvs command, along with the field name as it appears in the header display and a description of the field.

表格 4.1. pvs 顯示欄位

引數	標頭	描述
dev_size	DevSize	實體卷冊所建立於的基本裝置上的大小
pe_start	第一個 PE	設為基本裝置中第一個實體扇區的起始的偏差值
pv_attr	Attr	實體卷冊狀態: (a)lllocatable 或 e(x)ported。
pv_fmt	Fmt	實體卷冊的 metadata 格式 (lvm2 或 lvm1)
pv_free	PFree	實體卷冊上剩下的可用空間
pv_name	PV	實體卷冊名稱
pv_pe_alloc_count	Alloc	已使用的實體扇區數量
pv_pe_count	PE	實體扇區數量
pvseg_size	SSize	實體卷冊的區段大小
pvseg_start	起始	實體卷冊區段的起始實體扇區
pv_size	PSize	實體卷冊的大小
pv_tags	PV Tag	連接至實體卷冊的 LVM 標籤
pv_used	已使用	實體卷冊上目前已使用的空間
pv_uuid	PV UUID	實體卷冊的 UUID

pvs 指令就預設值會顯示下列欄位: pv_name、vg_name、pv_fmt、pv_attr、pv_size、pv_free。它們是藉由 pv_name 來排序的。

```
# 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
```

使用 -v 引數以及 pvs 指令可將下列欄位附加至預設的輸出: 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 Joq1ch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
/dev/sdd1   new_vg  lvm2 a-   17.14G 17.13G 17.14G yvfVZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

您可使用 pvs 指令的 --segments 引數來顯示有關於各個實體卷冊區段的相關資訊。區段相當於扇區的群組。區段視點 (segment view) 可有效幫助您查看您的邏輯卷冊是否已分散掉。

pvs --segments 指令就預設值會顯示下列欄位: pv_name、vg_name、pv_fmt、pv_attr、pv_size、pv_free、pvseg_start、pvseg_size。它們是藉由實體卷冊中的 pv_name 和 pvseg_size 來排序的。

```
# 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

```

您可使用 `pvs -a` 指令來查看 LVM 所偵測到不過尚未被初始化為 LVM 實體卷冊的裝置。

```

# 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

```

4.9.2.2. vgs 指令

表格 4.2, “vgs 顯示欄位” lists the display arguments of the vgs command, along with the field name as it appears in the header display and a description of the field.

表格 4.2. vgs 顯示欄位

引數	標頭	描述
lv_count	#LV	卷冊群組中所包含的邏輯卷冊數量
max_lv	MaxLV	卷冊群組中可允許的最大邏輯卷冊數量 (若無限制的話便是 0)
max_pv	MaxPV	卷冊群組中可允許的最大實體卷冊數量 (若無限制的話便是 0)
pv_count	#PV	定義卷冊群組的實體卷冊數量
snap_count	#SN	卷冊群組所包含的快照數量
vg_attr	Attr	卷冊群組的狀態: 可寫入 (w)、唯讀 (r)、可重設大小 (z)、已匯出 (x)、部分的 (p) 以及已聚成叢集 (c)。

引數	標頭	描述
vg_extent_count	#Ext	卷冊群組中的實體扇區數量
vg_extent_size	Ext	卷冊群組中的實體扇區大小
vg_fmt	Fmt	卷冊群組的 metadata 格式 (lvm2 或 lvm1)
vg_free	VFree	卷冊群組中剩下的可用空間大小
vg_free_count	Free	卷冊群組中可使用的實體扇區數量
vg_name	VG	卷冊群組名稱
vg_seqno	Seq	表示卷冊群組修訂版本的號碼
vg_size	VSize	卷冊群組的大小
vg_sysid	SYS ID	LVM1 System ID
vg_tags	VG Tags	連接至卷冊群組的 LVM 標籤
vg_uuid	VG UUID	卷冊群組的 UUID

vg 指令就預設值會顯示下列欄位: vg_name、pv_count、lv_count、snap_count、vg_attr、vg_size、vg_free。它們是藉由 vg_name 來排序的。

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

使用 vgs 指令和 -v 引數會將下列欄位新增至預設的畫面: 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-ZKk0-0pM0-0118-nlw0-wwqd-fD5D32
```

4.9.2.3. lvs 指令

表格 4.3. “lvs 顯示欄位” lists the display arguments of the lvs command, along with the field name as it appears in the header display and a description of the field.

表格 4.3. lvs 顯示欄位

引數	標頭	描述
chunksize chunk_size	Chunk	快照卷冊中的單位大小
copy_percent	Copy%	鏡像卷冊的同步化百分比; 在實體卷冊被透過 pv_move 指令移動時也會使用到
devices	裝置	構成邏輯卷冊的基本裝置: 實體卷冊、邏輯卷冊, 以及起始實體扇區和邏輯扇區
lv_attr	Attr	邏輯卷冊的狀態。邏輯卷冊的 attribute bit 如下: Bit 1: 卷冊類型 (Volume type): (m)irrored、(M)irrored (無初始同步化)、(o)rigin、(p)vmove、(s)napshot、invalid (S)napshot、(v)irtual

引數	標頭	描述
		Bit2: 權限 (Permissions) : (w)riteable、(r)ead-only Bit 3: 分配政策 (Allocation policy) : (c)ontiguous、(n)ormal、(a)nywhere、(i)nherited。 若卷冊目前無法進行分配上的變更 (比方說在執行 pvmove 指令時), 這就會是大寫的。 Bit 4: fixed (m)inor Bit 5 狀態 (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	邏輯卷冊的實際重大裝置代碼 (若未啓用的話就是 -1)
lv_kernel_minor	KMIN	邏輯卷冊的實際鏡像裝置代碼 (若未啓用的話就是 -1)
lv_major	Maj	邏輯卷冊的持續性重大裝置代碼 (若未指定的話就是 -1)
lv_minor	Min	邏輯卷冊的持續性非重大裝置代碼 (若未指定的話就是 -1)
lv_name	LV	邏輯卷冊名稱
lv_size	LSize	邏輯卷冊大小
lv_tags	LV Tags	連接至邏輯卷冊的 LVM 標籤
lv_uuid	LV UUID	邏輯卷冊的 UUID
mirror_log	Log	鏡像 log 所位於的裝置
modules	模組	使用此邏輯卷冊所需要的相應 kernel 裝置映射 (device-mapper) 目標
move_pv	移動	提供一個透過 pvmove 指令來建立的暫時性邏輯卷冊的實體卷冊
origin	起源	快照卷冊的起源裝置 (origin device)
regionsize region_size	範圍	鏡像邏輯卷冊的單位大小
seg_count	#Seg	邏輯卷冊中的區段數量
seg_size	SSize	邏輯卷冊中的區段大小
seg_start	起始	邏輯卷冊中的區段偏差值
seg_tags	Seg Tags	連接至邏輯卷冊之區段的 LVM 標籤
segtype	類型	邏輯卷冊的區段類型 (例如: mirror、striped、linear)
snap_percent	Snap%	快照目前的使用量百分比
stripes	#Str	邏輯卷冊中的磁帶或鏡像數量
stripesize stripe_size	Stripe	等量邏輯卷冊中的磁條單位大小

lvs 指令就預設值會顯示下列欄位: lv_name、vg_name、lv_attr、lv_size、origin、snap_percent、move_pv、mirror_log、copy_percent。它們就預設值會被藉由卷冊群組中的 vg_name 和 lv_name 來排序。

```
# lvs
LV          VG      Attr   LSize  Origin Snap%  Move Log Copy%
lvol10      new_vg  owi-a- 52.00M
newvgsnap1 new_vg  swi-a- 8.00M lvol10  0.20
```

使用 `lvs` 指令以及 `-v` 引數便能將下列欄位新增至預設的畫面中：`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
lvol10      new_vg  1 owi-a- 52.00M -1 -1 253 3          LBylTz-sr23-0jsI-LT03-
nHLC-y8XW-EhC178
newvgsnap1 new_vg  1 swi-a- 8.00M -1 -1 253 5    lvol10  0.20          1ye10U-1cIu-o79k-20h2-
ZGF0-qCJm-CfbsIx
```

您可使用 `lvs` 指令的 `--segments` 引數來顯示含有強調區段資訊的預設欄位的相關資訊。當您使用 `segments` 引數時，`seg` 前綴字元為非必要的。`lvs --segments` 指令就預設值會顯示下列欄位：`lv_name`、`vg_name`、`lv_attr`、`stripes`、`segtype`、`seg_size`。畫面就預設值會藉由卷冊群組中的 `vg_name`、`lv_name` 以及邏輯卷冊中的 `seg_start` 來排序。若邏輯卷冊被分散的話，這項指令的輸出便會將其顯示出來。

```
# lvs --segments
LV          VG      Attr   #Str Type  SSize
LogVol100 VolGroup00 -wi-ao 1 linear 36.62G
LogVol101 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
```

若使用 `lvs --segments` 指令以及 `-v` 引數便可新增下列欄位至預設的畫面中：`seg_start`、`stripesize`、`chunksize`。

```
# lvs -v --segments
Finding all logical volumes
LV          VG      Attr   Start SSize #Str Type  Stripe Chunk
lvol10      new_vg  owi-a- 0 52.00M 1 linear 0 0
newvgsnap1 new_vg  swi-a- 0 8.00M 1 linear 0 8.00K
```

下列範例顯示了 `lvs` 指令在一部配置了一個邏輯卷冊的系統上的預設輸出，以及 `lvs` 指令和 `segments` 引數的預設輸出。

```
# lvs
LV          VG      Attr   LSize  Origin Snap%  Move Log Copy%
lvol10 new_vg -wi-a- 52.00M
# lvs --segments
LV          VG      Attr   #Str Type  SSize
lvol10 new_vg -wi-a- 1 linear 52.00M
```

4.9.3. 排序 LVM 報告

通常 `lvs`、`vgs` 或 `pvs` 指令的所有輸出都要先被產生並儲存於內部之後才可進行排序並且行列才可正確對稱。您可指定 `--unbuffered` 這個引數來在未排序的輸出一被產生時即刻將它顯示出。

若要指定另一列行列來進行排序的話，請使用任何回報指令的 `-O` 引數。無須在輸出中包含這些欄位。

下列範例顯示了 `pvs` 指令的輸出，它顯示了實體卷冊的名稱、大小以及可用空間。

```
# 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
```

下列範例顯示了相同的輸出，並藉由可用空間的欄位來排序。

```
# 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
```

下列範例顯示了您無須顯示您正在排序的欄位。

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

若要顯示反向的排序，在 `-O` 引數之後，於您所指定的欄位之前前置 `-` 這個字元。

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

4.9.4. 指定單位

若要為 LVM 回報顯示指定單位，請使用回報指令的 `--units` 引數。您可指定 `(b)ytes`、`(k)ilobytes`、`(m)egabytes`、`(g)igabytes`、`(t)erabytes`、`(e)xabytes`、`(p)etabytes`、以及 `(h)uman-readable`。預設的顯示為 `human-readable`。您可透過在 `lvm.conf` 檔案的 `global` 部份中設定 `units` 這個參數來置換預設值。

下列範例指定了 `pvs` 指令的輸出為 MB 而非預設的 GB。

```
# 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
```

就預設值，單位會以 2 的因子（1024 的倍數）來顯示出。您可藉由將單位規格（B、K、M、G、T、H）大寫化來將單位指定為 1000 的倍數。

下列指令會將輸出顯示為 1024 的倍數，這是預設的特性。

```
# 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
```

下列指令會將輸出顯示為 1000 的倍數。

```
# 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
```

您亦可指定區段（s）（定義為 512 位元組）或是自訂單位。

下列範例將 pvs 指令的輸出顯示成了幾個磁區。

```
# 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
```

下列範例會將 pvs 指令的輸出以 4 MB 為單位的方式顯示出。

```
# 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
```

LVM 配置範例

本章節提供了一些基本的 LVM 配置範例。

5.1. 在三個磁碟上建立一個 LVM 邏輯卷冊

此範例建立了一個稱為 `new_logical_volume` 的 LVM 邏輯卷冊，它包含著位於 `/dev/sda1`、`/dev/sdb1` 以及 `/dev/sdc1` 的磁碟。

5.1.1. 建立實體卷冊 (Physical Volumes)

若要使用卷冊群組中的磁碟，您必須將它們標記為 LVM 實體卷冊。



警告

這項指令會將 `/dev/sda1`、`/dev/sdb1` 以及 `/dev/sdc1` 上的所有資料都損毀。

```
[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. 建立卷冊群組

下列指令將會建立卷冊群組 `new_vol_group`。

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

您可使用 `vgs` 指令來顯示新卷冊群組的屬性。

```
[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. 建立邏輯卷冊

下列指令會由 `new_vol_group` 卷冊群組建立邏輯卷冊 `new_logical_volume`。此範例會建立一個使用卷冊群組 2GB 的邏輯卷冊。

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

5.1.4. 建立檔案系統

下列指令可在邏輯卷冊上建立一個 GFS 檔案系統。

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

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

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

Syncing...
All Done
```

下列指令會將邏輯卷冊掛載並回報檔案系統磁碟空間上的使用量。

```
[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. 建立一個 Striped 邏輯卷冊

此範例建立了一個稱為 `striped_logical_volume` 的 LVM 磁條邏輯卷冊 (LVM striped logical volume)，它會將資料分散到 `/dev/sda1`、`/dev/sdb1` 和 `/dev/sdc1` 磁碟上去。

5.2.1. 建立實體卷冊 (Physical Volumes)

標記您將會在卷冊群組中使用來作為 LVM 實體卷冊的磁碟。



警告

這項指令會將 `/dev/sda1`、`/dev/sdb1` 以及 `/dev/sdc1` 上的所有資料都損毀。

```
[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. 建立卷冊群組

下列指令將會建立卷冊群組 `volgroup01`。

```
[root@tng3-1 ~]# vgcreate volgroup01 /dev/sda1 /dev/sdb1 /dev/sdc1
Volume group "volgroup01" successfully created
```

您可使用 `vgs` 指令來顯示新卷冊群組的屬性。

```
[root@tng3-1 ~]# vgs
VG                #PV #LV #SN Attr   VSize VFree
volgroup01        3   0   0 wz--n- 51.45G 51.45G
```

5.2.3. 建立邏輯卷冊

下列指令會由 `volgroup01` 這個卷冊群組建立磁條邏輯卷冊 `striped_logical_volume`。此範例會建立一個大小為 2GB 的邏輯卷冊，它將含有三個磁條並且每個磁條將會是 4 KB。

```
[root@tng3-1 ~]# lvcreate -i3 -l4 -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. 建立檔案系統

下列指令可在邏輯卷冊上建立一個 GFS 檔案系統。

```
[root@tng3-1 ~]# gfs_mkfs -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
```

下列指令會將邏輯卷冊掛載並回報檔案系統磁碟空間上的使用量。

```
[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-LogVol100
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. 分割卷冊群組

在此範例中，有個現有的卷冊群組包含了三個實體卷冊。實體卷冊上若有足夠的未使用空間的話，您便可在不新增磁碟的情況下新建卷冊群組。

在初始設定中，邏輯卷冊 `mylv` 是由 `myvol` 這個卷冊群組所分割的，並且它包含著 `/dev/sda1`、`/dev/sdb1` 以及 `/dev/sdc1` 這三個實體卷冊。

完成了這項程序後，myvg 這個卷冊群組便會包含著 /dev/sda1 和 /dev/sdb1。第二個邏輯群組 yourvg 則會包含著 /dev/sdc1。

5.3.1. 判斷可用空間

您可透過使用 pvscan 指令來判斷出卷冊群組中目前有多少可用空間。

```
[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. 移動資料

您可透過使用 pvmove 指令來將 /dev/sdc1 中所有已使用的實體扇區移至 /dev/sdb1。pvmove 指令的執行可能會花上一段時間。

```
[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%
```

移動了資料之後，您便會看見 /dev/sdc1 上的所有空間都可使用。

```
[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. 分割卷冊群組

若要建立新的卷冊群組 yourvg，請使用 vgsplit 指令來分割卷冊群組 myvg。

在您能夠分割卷冊群組之前，邏輯卷冊必須被停用。若檔案系統已被掛載，您必須在停用邏輯卷冊前先將檔案系統卸載。

您可透過使用 lvchange 指令或是 vgchange 指令來停用邏輯卷冊。下列指令可停用邏輯卷冊 mylv 然後由 myvg 卷冊群組分割 yourvg 卷冊群組，然後將實體卷冊 /dev/sdc1 移至新的卷冊群組 yourvg 中。

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

您可使用 vgs 指令來查看這兩個卷冊群組的屬性。

```
[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. 建立新的邏輯卷冊

在建立了新的卷冊群組之後，您便可新建邏輯卷冊 `yourlv`。

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

5.3.5. 製作檔案系統和掛載新的邏輯卷冊

您可在新的邏輯卷冊上建立檔案系統然後將它掛載。

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

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

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

Syncing...
All Done

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

5.3.6. 啟用和掛載原本的邏輯卷冊

因為您必須停用邏輯卷冊 `mylv`，因此在您掛載它之前您必須再次啓用它。

```
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. 由邏輯卷冊中移除磁碟

此範例顯示了如何將磁碟由現有的邏輯卷冊中移除（為了更換磁碟或是使用該磁碟來作為不同卷冊的一部分）。若要移除磁碟，您首先必須將 LVM 實體卷冊上的扇區移至一個或一組不同的磁碟中。

5.4.1. 將扇區移至現有的實體卷冊中

在此範例中，邏輯卷冊被分配至 `myvg` 卷冊群組中的四個實體卷冊上。

```
[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
```

我們希望將 `/dev/sdb1` 的扇區移除，如此一來我們才能將該磁碟由卷冊群組中移除。

若卷冊群組中的其它實體卷冊上有足夠的可用扇區的話，您可針對於您希望移除的裝置執行 `pvmove` 指令並且不使用其它選項，如此一來扇區便會被分配置其它裝置上。

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

當 `pvmove` 指令執行完成後，扇區的分配會如下所示：

```
[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
```

請使用 `vgreduce` 指令來將 `/dev/sdb1` 實體卷冊由卷冊群組中移除。

```
[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   myvg lvm2 --   17.15G 17.15G
/dev/sdc1   myvg lvm2 a-   17.15G 12.15G
/dev/sdd1   myvg lvm2 a-   17.15G  2.15G
```

磁碟現在可被實體移除或分配給其他用戶。

5.4.2. 將扇區移至新磁碟上

在此範例中，邏輯卷冊分配在 `myvg` 卷冊群組中的三個實體卷冊上：

```
[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
```

我們希望將 `/dev/sdb1` 的扇區移至新的 `/dev/sdd1` 裝置上。

5.4.2.1. 新建實體卷冊

由 `/dev/sdd1` 建立新的實體卷冊。

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

5.4.2.2. 新增實體卷冊至卷冊群組中

新增 `/dev/sdd1` 至現有的卷冊群組 `myvg` 中。

```
[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. 移動資料

使用 `pvmove` 指令來將資料由 `/dev/sdb1` 移至 `/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. 將舊的實體卷冊由卷冊群組中移除

當您將資料由 `/dev/sdb1` 中移走後，您便可將它由卷冊群組中移除。

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

您現在已能將磁碟重新分配至另一個卷冊群組或將磁碟由系統中移除。

5.5. 在叢集中建立鏡像 LVM 邏輯卷冊

Creating a mirrored LVM logical volume in a cluster requires the same commands and procedures as creating a mirrored LVM logical volume on a single node. However, in order to create a mirrored LVM volume in a cluster the cluster and cluster mirror infrastructure

must be running, the cluster must be quorate, and the locking type in the `lvm.conf` file must be set correctly to enable cluster locking, either directly or by means of the `lvmconf` command as described in 節 3.1, “在叢集中建立 LVM 卷冊”。

下列程序將會在叢集中建立一個鏡像 LVM 卷冊。首先，程序會查看叢集服務是否已安裝並運作中，然後該程序便會建立鏡像卷冊。

1. 若要建立一個會被叢集中所有節點共享的鏡像邏輯卷冊，叢集中所有節點的 `lvm.conf` 檔案裡的鎖定類型皆必須經過正確設置。就預設值，鎖定類型會被設為 `local`。若要進行修改，請在叢集的各個節點中執行下列指令，以啟用叢集鎖定：

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

2. 若要建立叢集邏輯卷冊，叢集架構必須經過設定並執行於叢集中的所有節點上。以下範例驗證了 `clvmd` daemon 正在節點上運作：

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

下列指令顯示了叢集狀態的本機視點：

```
[root@doc-07 ~]# cman_tool services
Service      Name                                GID LID State   Code
...
DLM Lock Space: "clvmd"              7   3 run    -
[1 2 3]
...
```

3. 請確認 `cmirror` 與 `cmirror-kmod` 套件已被安裝。`cmirror-kmod` 套件需根據運作中的 `kernel` 來進行安裝。比方說，若運作中的 `kernel` 為 `kernel-largesmp` 的話，您便需要安裝 `cmirror-kmod-largesmp` 來相應 `kernel` 版本。
4. 啟用 `cmirror` 服務。

```
[root@doc-07 ~]# service cmirror start
Loading clustered mirror log: [ OK ]
```

5. 建立鏡像。第一個步驟就是建立實體卷冊。下列指令會建立三個實體卷冊。其中兩個實體卷冊將會被用來作為鏡像的 `leg`，並且第三個實體卷冊將會包含鏡像的 `log`。

```
[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. 建立卷冊群組。此範例將會建立一個卷冊群組 `vg001`，它將會包含著先前步驟中所建立的三個實體卷冊。

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

Note that the output of the `vgcreate` command indicates that the volume group is clustered. You can verify that a volume group is clustered with the `vgs` command, which will show the volume group's attributes. If a volume group is clustered, it will show a `c` attribute.

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

7. 建立鏡像邏輯卷冊。此範例將會由卷冊群組 `vg001` 建立邏輯卷冊 `mirrorlv`。此卷冊含有一個鏡像 `leg`。此範例會指定實體卷冊的哪個扇區會被用來作為邏輯卷冊。

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

您可使用 `lvs` 指令來顯示鏡像建立的完成度。下列範例顯示了鏡像已完成了 47% 的同步、接著 91%，然後當鏡像完成時將會顯示 100% 同步化。

```
[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
```

鏡像完成後將會被紀錄在系統日誌檔中：

```
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. 您可使用 `lvs` 指令以及 `-o +devices` 選項來顯示鏡像的配置，包括鏡像 `leg` 是由哪些裝置所組成的。在此範例中，您可發現邏輯卷冊是由兩個 `linear image` 和一個 `log` 所組成的。

```
[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)
```

在 RHEL 5.2 或更新版本上，您可使用 `lvs` 的 `seg_pe_ranges` 選項來顯示資料格式。您可使用此選項來驗證您的格式是否正確。這項指令的輸出會將 PE 範圍以 `lvcreate` 與 `lvresize` 指令所接受的輸入格式來顯示。

```
[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
```

當您建立鏡像卷冊時，您將會建立 `clustered_log` dlm 空間，它將會包含著所有鏡像的 dlm lock。

```
[root@doc-07 log]# cman_tool services
Service      Name                                GID LID State  Code
Fence Domain: "default"              4   2 run   -
[1 2 3]

DLM Lock Space: "clvmd"                12   7 run   -
[1 2 3]

DLM Lock Space: "clustered_log"        14   9 run   -
[1 2 3]

User:         "usrm::manager"          10   4 run   -
[1 2 3]
```



注意

For information on recovering from the failure of one of the legs of an LVM mirrored volume, see 節 6.3, “由 LVM 鏡像錯誤中復原”。

LVM 疑難排解

此章節提供了針對於各種 LVM 問題進行疑難排解的指南。

6.1. 疑難排解診斷結果

若指令不如預期地運作的話，您可透過下列方式來取得診斷結果：

- 使用任何指令的 `-v`、`-vv`、`-vvv` 或 `-vvvv` 引數來取得更加詳細的輸出。
- If the problem is related to the logical volume activation, set 'activation = 1' in the 'log' section of the configuration file and run the command with the `-vvvv` argument. After you have finished examining this output be sure to reset this parameter to 0, to avoid possible problems with the machine locking during low memory situations.
- 請執行 `lvm dump` 指令，它提供了用來作為診斷用的資訊傾印。如欲取得更多相關資訊，請參閱 `lvm dump(8)` man page。
- 您可執行 `lvs -v`、`pvs -a` 或 `dmsetup info -c` 指令來取得額外的系統資訊。
- 在 `/etc/lvm/backup` 檔案中（或於 `/etc/lvm/archive` 檔案中的壓縮版本）檢查 metadata 的最後一個備份。
- 透過執行 `lvm dumpconfig` 指令來檢查目前的配置資訊。
- 檢查 `/etc/lvm` 目錄中的 `.cache` 檔案來取得有關於哪個裝置上含有實體卷冊的相關記錄。

6.2. 顯示錯誤裝置的相關資訊

您可使用 `lvs` 或 `vgs` 指令的 `-P` 引數來顯示原本不會出現在輸出中，有關於錯誤卷冊的相關資訊。這個引數允許了一些作業的進行，儘管內部的 metadata 並不完全地一致。比方說，若構成卷冊群組 `vg` 的其中一個裝置發生錯誤的話，`vgs` 指令可能會顯示下列輸出。

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

若您指定了 `vgs` 指令的 `-P` 引數的話，卷冊群組還是無法使用，不過您可看見更多有關於錯誤裝置的相關資訊。

```
[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)
```

在此範例中，錯誤的裝置會造成卷冊群組中的線性和等量邏輯卷冊發生錯誤。缺少了 `-P` 引數的 `lvs` 指令會顯示下列輸出。

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

使用 `-P` 引數來顯示錯誤的邏輯卷冊。

```
[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)
```

下列範例顯示了當某個鏡像邏輯卷冊的 `leg` 發生錯誤時而使用了 `-P` 引數的 `pvs` 和 `lvs` 指令的輸出。

```
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. 由 LVM 鏡像錯誤中復原

此部份提供了一個從「基於實體卷冊的基本裝置發生了錯誤所造成的 LVM 鏡像卷冊的 `leg` 錯誤」的情況下復原之範例。當一個鏡像 `leg` 發生錯誤時，LVM 便會將鏡像卷冊轉換為線性卷冊，它會和先前相同地繼續進行作業不過缺少鏡像的重複。在此情況下，您可新增一個新的磁碟裝置至系統，使用它來取代實體裝置並重建鏡像。

下列指令建立了將會使用於鏡像的實體卷冊。

```
[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
```

下列指令建立了卷冊群組 `vg` 和鏡像卷冊 `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
```

您可使用 `lvs` 指令來驗證 `mirror leg` 和 `mirror log` 的基本裝置和鏡像卷冊的格式。請注意，在第一個範例中，鏡像還未完整同步化；您應等到 `Copy%` 這個欄位顯示了 100.00 之後才繼續進行。

```
[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                /dev/sdc1(0)
```

在此範例中，`/dev/sda1` 這個鏡像的主要 `leg` 發生了錯誤。任何寫入至鏡像卷冊的動作都會造成 LVM 偵測到錯誤的鏡像。當這情況發生時，LVM 會將鏡像轉換為單獨的線性卷冊。在此情況下，若要進行轉換的話，請執行 `dd` 這項指令

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

您可使用 `lvs` 指令來驗證裝置現在是否是個 `linear` 裝置。因為磁碟發生錯誤，因而產生了 I/O 錯誤。

```
[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)
```

在此情況下，您還是應該能使用邏輯卷冊，不過將不會有鏡像重複。

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

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

[root@link-08 ~]# pvscan
PV /dev/sdb1  VG vg   lvm2 [67.83 GB / 67.10 GB free]
PV /dev/sdb2  VG vg   lvm2 [67.83 GB / 67.83 GB free]
```

```
PV /dev/sdc1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sde1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sde2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sda1   VG vg    1vm2 [603.94 GB]
PV /dev/sda2   VG vg    1vm2 [603.94 GB]
Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

接下來請透過新的實體卷冊來延伸原始的卷冊群組。

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

[root@link-08 ~]# pvscan
PV /dev/sdb1   VG vg    1vm2 [67.83 GB / 67.10 GB free]
PV /dev/sdb2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sde1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sde2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh1   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh2   VG vg    1vm2 [67.83 GB / 67.83 GB free]
PV /dev/sda1   VG vg    1vm2 [603.93 GB / 603.93 GB free]
PV /dev/sda2   VG vg    1vm2 [603.93 GB / 603.93 GB free]
Total: 16 [2.11 TB] / in use: 16 [2.11 TB] / in no VG: 0 [0 ]
```

將 linear 卷冊轉換回它原始的鏡像狀態。

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

您可使用 `lvs` 指令來驗證鏡像是否已被儲存。

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

6.4. 復原實體卷冊的 Metadata

若實體卷冊的卷冊群組 metadata 區域不小心被覆寫或損毀的話，您將會看見一則顯示 metadata 區域不正確，或是系統無法找到含有某個 UUID 的實體卷冊的錯誤訊息。您可藉由在實體卷冊上編寫新的 metadata 區域，指定和遺失的 metadata 相同的 UUID 來恢復實體卷冊的資料。



警告

您不該以一個可運作的 LVM 邏輯卷冊來嘗試這項程序。若您指定了錯誤的 UUID，您將會遺失您的資料。

下列範例顯示了當 metadata 遺失或損毀時您所可能會看見的輸出。

```
[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.
...
```

您可能能夠透過查看 /etc/lvm/archive 目錄來找尋被覆寫的實體卷冊的 UUID。若要找尋特定卷冊群組最後一個已知、有效、壓縮過的 LVM metadata，請查看 VolumeGroupName_xxxx.vg 檔案。

此外，您可能會發現停用卷冊並設置 partial (-P) 引數能讓您找到遺失或損毀的實體卷冊的 UUID。

```
[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'.
...
```

您可使用 pvcreate 指令的 --uuid 和 --restorefile 引數來復原實體卷冊。下列範例已將 /dev/sdh1 裝置標記為實體卷冊和以上所顯示的 UUID (FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk)。這項指令會透過包含在 VG_00050.vg 中的 metadata 資訊 (卷冊群組 最近期的良好壓縮的 metadata) 來將實體卷冊的標籤恢復。restorefile 引數會指示 pvcreate 指令去使新的實體卷冊能和卷冊群組上的舊實體卷冊相容，並確保新的 metadata 不會被放置在舊實體卷冊包含著資料的位置上 (這是有可能會發生的，比方說在原始的 pvcreate 指令有使用控制 metadata 定位的指令列引數的情況下，或是實體卷冊原本是透過使用不同預設值的不同版本軟體來建立的情況下)。pvcreate 指令只會將 LVM metadata 區域覆寫而不會影響到現有的資料區域。

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

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

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

您現在能夠顯示邏輯卷冊了。

```
[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)
```

下列指令會啟用卷冊並顯示啟用的卷冊。

```
[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)
```

若 on-disk 的 LVM metadata 所需容量和將它覆寫的資料相同，這項指令便可將實體卷冊復原。若將 metadata 覆寫的資料超過了 metadata 範圍，那麼卷冊上的資料就可能會被影響到。您可能能夠透過使用 fsck 指令來將資料復原。

6.5. 替換一個遺失的實體卷冊

If a physical volume fails or otherwise needs to be replaced, you can label a new physical volume to replace the one that has been lost in the existing volume group by following the same procedure as you would for recovering physical volume metadata, described in 節 6.4, “復原實體卷冊的 Metadata”. You can use the `--partial` and `--verbose` arguments of the `vgdisplay` command to display the UUIDs and sizes of any physical volumes that are no longer present. If you wish to substitute another physical volume of the same size, you can use the `pvccreate` command with the `--restorefile` and `--uuid` arguments to initialize a new device with the same UUID as the missing physical volume. You can then use the `vgcfgrestore` command to restore the volume group's metadata.

6.6. 將遺失的實體卷冊由一個卷冊群組中移除掉

若您遺失了一個實體卷冊，您可透過 `vgchange` 指令的 `--partial` 引數來啟用卷冊群組中剩下的實體卷冊。您可透過 `vgreduce` 指令的 `--removemissing` 引數來將所有使用了該實體卷冊的邏輯卷冊由卷冊群組中移除。

我們建議您執行 `vgreduce` 指令和 `--test` 引數來驗證您所要銷毀的邏輯卷冊。

就和大部分的 LVM 作業一樣，您可透過即刻地使用 `vgcfgrestore` 指令來將卷冊群組的 metadata 復原為先前的狀態以撤消 `vgreduce` 指令。比方說，若您使用了 `vgreduce` 指令的 `--removemissing` 引數而不使用 `--test` 引數並且發現您移除了您想要保留的邏輯卷冊，您還是能夠將實體卷冊替換掉並使用另一項 `vgcfgrestore` 指令來將卷冊群組復原為它先前的狀態。

6.7. 邏輯卷冊的可用扇區不足

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

`vgdisplay` 指令就預設值會包含下列顯示了可用實體扇區的輸出。

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

此外，您亦可使用 `vgs` 指令的 `vg_free_count` 和 `vg_extent_count` 引數來顯示可用的扇區和扇區的總數。

```
[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
```

有了 8780 的可用扇區，您便可執行下列指令，並使用小寫的 `l` 引數來使用扇區並取代位元組：

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

這使用了卷冊群組中所有的可用扇區。

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

Alternately, you can extend the logical volume to use a percentage of the remaining free space in the volume group by using the `-l` argument of the `lvcreate` command. For information, see [節 4.4.1.1, “建立線性卷冊”](#).

利用 LVM GUI 來進行 LVM 管理

除了指令列介面 (Command Line Interface, CLI)，LVM 還提供了一個圖形化用戶介面 (GUI)，您可使用該圖形化介面來配置 LVM 邏輯卷冊。您可藉由輸入 `system-config-lvm` 來啟動此工具程式。Red Hat Enterprise Linux 建置指南的 LVM 章節中提供了如何逐步地使用此工具來配置 LVM 邏輯卷冊的相關指南。

另外，LVM GUI 也包含在 Conga 管理介面中的一部分。如欲取得如何使用 Conga 的 LVM GUI，請查看 Conga 的線上說明。

附錄 A. 裝置映射 (Device Mapper) 設備

Device Mapper 是一個提供了卷冊管理架構的 kernel 驅動程式。它提供了建立映射裝置的一般方式，並且該映射裝置可被用來作為邏輯卷冊。它並不確切地清楚卷冊群組或 metadata 的格式為何。

Device Mapper 提供了幾個高層級技術上的基礎。除了 LVM 之外，Device-Mapper multipath 和 dmraid 指令亦使用 Device Mapper。Device Mapper 的應用程式介面為 ioctl system call。dmsetup 指令則為用戶介面。

LVM logical volumes are activated using the Device Mapper. Each logical volume is translated into a mapped device. Each segment translates into a line in the mapping table that describes the device. The Device Mapper supports a variety of mapping targets, including linear mapping, striped mapping, and error mapping. So, for example, two disks may be concatenated into one logical volume with a pair of linear mappings, one for each disk. When LVM2 creates a volume, it creates an underlying device-mapper device that can be queried with the dmsetup command. For information about the format of devices in a mapping table, see 節 A.1, “裝置表格映射”. For information about using the dmsetup command to query a device, see 節 A.2, “dmsetup 指令”。

A.1. 裝置表格映射

映射裝置是由一個透過受支援的裝置表格 (Device Table) 映射來指定了如何映射邏輯磁區 (logical sector) 的各個範圍的表格來定義的。映射裝置的表格是由一系列具有下列格式的行列所構成的：

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

在 Device Mapper 表格的第一個行列中，start 這個參數必須等於 0。一個行列上的 start + length 參數必須等於下個行列上的 start。指定於映射表格行列中的映射參數取決於該行列上所指定的 mapping。

Device Mapper 中的大小總是以磁區來指定 (512 位元組)。

當某個裝置被指定為 Device Mapper 中的映射參數時，它可被檔案系統中的裝置名稱 (例如 /dev/hda) 參照，或是被格式為 major:minor 的 major 和 minor 數字參照。我們建議使用 major:minor 這個格式因為它避免了路徑名稱搜尋。

以下顯示了一個裝置的映射表格範例。在此表格中有四個 linear 目標：

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

各個行列的前兩個參數都是區段的起始區塊以及區塊的長度。下個關鍵字為映射目標，在此範例中的所有情況下都是 linear。行列剩下的部份則包含著 linear 目標的參數。

下列部份中描述了以下映射的格式：

- 線性
- 等量
- mirror

- snapshot 和 snapshot-origin
- error
- zero
- multipath
- crypt

A.1.1.1. Linear 映射目標

Linear 映射目標會將一個連續範圍的區塊映射至另一個區塊裝置上。Linear 目標的格式如下：

```
start length linear device offset
```

start

在虛擬裝置中啟用區塊

length

此區段的長度

device

區塊裝置，被檔案系統中的裝置名稱所參照，或是被格式為 major:minor 的 major 和 minor 數字所參照

offset

在裝置上啟用映射的偏差值 (offset)

下列範例顯示了一個起始區塊於虛擬裝置 0 中、區段長度為 1638400、major:minor 數字配對為 8:2，以及裝置起始偏差值為 41146992 的 linear 目標。

```
0 1638400 linear 8:2 41146992
```

下列範例顯示了一個裝置參數指定為 /dev/hda 這個裝置的 linear 目標。

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

A.1.1.2. 等量映射目標

等量映射目標 (striped Mapping Target) 支援實體裝置上的 striping。它會取等量磁碟數量和 striping chunk size 為引數以及一系列裝置名稱與磁區的配對。等量目標的格式如下

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

各個等量磁碟都有一組 device 和 offset 參數。

start

在虛擬裝置中啟用區塊

length

此區段的長度

#stripes

虛擬裝置的等量磁碟數量

chunk_size

切換至下一個等量磁碟之前可寫至各個等量磁碟的磁區數量；必須至少和 kernel page 大小的 2 的 n 次方一樣大

device

區塊裝置，被檔案系統中的裝置名稱所參照，或是被格式為 major:minor 的 major 和 minor 數字所參照。

offset

在裝置上啓用映射的偏差值 (offset)

下列範例顯示了一個等量目標以及三個等量磁碟和大小為 128 的 chunk size:

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

0

在虛擬裝置中啓用區塊

73728

此區段的長度

striped 3 128

跨越了三個裝置的 stripe 以及大小為 128 block 的 chunk size

8:9

第一個裝置的 major:minor 數字

384

第一個裝置上的映射的起始偏差值

8:8

第二個裝置上的 major:minor 數字

384

第二個裝置上的映射的起始偏差值

8:7

第三個裝置上的 major:minor 數字

9789824

第三個裝置上的映射的起始偏差值

下列範例顯示了一個擁有兩個 256 KiB chunk 的等量磁碟的等量目標，並且裝置參數是由檔案系統中的裝置名稱來指定的，而不是以 major 和 minor 數字來指定。

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

A.1.3. 鏡像映射目標

鏡像映射目標 (mirror mapping target) 支援鏡像邏輯卷冊的映射。鏡像目標的格式如下:

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

start

在虛擬裝置中啓用區塊

length

此區段的長度

log_type

各種可能的日誌類型與它們的引數如下:

core

鏡像是本地的，並且鏡像日誌 (mirror log) 存放在核心記憶體 (core memory) 中。此日誌類型接受 1 到 3 個引數:

```
regionsize [[no]sync] [block_on_error]
```

disk

鏡像是本地的，並且鏡像日誌存放在磁碟上。此日誌類型接受 2 到 4 個引數:

```
logdevice regionsize [[no]sync] [block_on_error]
```

clustered_core

鏡像已被叢集連結，並且鏡像日誌存放在核心記憶體中。此日誌類型接受 2 到 4 個引數:

```
regionsize UUID [[no]sync] [block_on_error]
```

clustered_disk

鏡像已被叢集連結，並且鏡像日誌存放在磁碟上。此日誌類型接受 3 到 5 個引數:

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

LVM 會保存一個小型的日誌，它會使用該日誌來追蹤哪些區域已和鏡像同步化。regionsize 這個引數能指定這些區域的大小。

在一個叢集環境中，UUID 這個引數是個和鏡像日誌裝置相聯的唯一識別碼 (unique identifier)，日誌狀態可從而在叢集環境下被保留。

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

#log_args

將會被指定在映射中的日誌引數數量

logargs

鏡像的日誌引數；日誌引數的數量是透過 #log-args 參數來指定的，並且有效的日誌引數則是透過 log_type 參數來判斷出的。

#devs

鏡像中的 leg 數量；各個 leg 都有被指定了一個裝置和偏差值。

device

各個鏡像 leg 的區塊裝置，由檔案系統中的裝置名稱參照，或由格式為 major:minor 的 major 和 minor 數字參照。如 #devs 參數所顯示，各個鏡像 leg 都會被指定一個區塊裝置和偏差值。

offset

裝置上的映射的起始偏差值。如 #devs 參數所顯示，各個鏡像 leg 都會被指定一個區塊裝置和偏差值。

下列範例顯示了一個叢集鏡像的鏡像映射目標，並且該叢集鏡像的鏡像日誌被存放在磁碟上。

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

0

在虛擬裝置中啟用區塊

52428800

此區段的長度

mirror clustered_disk

含有一個指定了鏡像已被叢集連結並且鏡像日誌存放在磁碟上的鏡像目標

4

會有四個鏡像日誌

253:2

日誌裝置的 major:minor 數字

1024

鏡像日誌使用來追蹤已同步化之鏡像的區域大小

UUID

用來在叢集環境下保留日誌資訊的鏡像日誌裝置 UUID

block_on_error

鏡像應針對於錯誤做出回應

3

鏡像中的 leg 數量

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

構成各個鏡像 leg 的裝置的 major:minor 數字和偏差值

A.1.4. 快照和 snapshot-origin 映射目標

當您建立了卷冊的第一個 LVM 快照時，會有四個 Device Mapper 裝置被使用到：

1. 一個含有 linear 映射的裝置，並且該映射包含著來源卷冊的原始映射表格。
2. 一個含有 linear 映射的裝置，它會被用來作為來源卷冊的寫入即複製 (copy-on-write, COW) 的裝置；針對於各個寫入動作，原始資料都會被儲存在各個 snapshot 的 COW 裝置中，如此一來它的可見內容便不會遭到更改（直到 COW 裝置滿出）。
3. 一個含有 snapshot 映射的裝置，它結合了 #1 與 #2，也就是可見的快照卷冊
4. The "original" volume (which uses the device number used by the original source volume), whose table is replaced by a "snapshot-origin" mapping from device #1.

有個使用來建立這些裝置的固定命名方案。比方說，您可能會使用下列指令來建立一個名為 base 的 LVM 卷冊以及一個基於該卷冊、名為 snap 的快照卷冊。

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

這會產生四個裝置，並且您可透過下列指令來檢視這些裝置：

```
# 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
```

snapshot-origin 目標的格式如下：

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

start

在虛擬裝置中啟用區塊

length

此區段的長度

origin

快照的基本卷冊

snapshot-origin 通常會有一個或是更多個基於它的快照。讀取動作會直接被映射至 backing device。針對於各個寫入動作，原始資料都會被儲存在各個快照的 COW 裝置中來保存它的可見內容直到 COW 裝置填滿。

snapshot 目標的格式如下：

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

start

在虛擬裝置中啟用區塊

length

此區段的長度

origin

快照的基本卷冊

COW-device

遭到更改的資料區塊所被儲存至的裝置

P|N

P (Persistent [一致性]) 或是 N (Not persistent [非一致性])；顯示了快照在系統重新啟動後是否還會存在。針對於暫時性的快照 (N)，磁碟上必須要儲存較少 metadata；它們可被 kernel 保存在記憶體中。

chunksize

被存放在 COW 裝置上、遭到更改的資料區塊大小（以磁區為單位）

下列範例顯示了一個原始裝置為 254:11 的 snapshot-origin 目標。

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

下列範例顯示了一個原始裝置為 254:11 並且 COW 裝置為 254:12 的 snapshot 目標。這個快照裝置經過了系統重新啟動後將依然有效，並且儲存在 COW 裝置上的資料的區塊大小為 16 個磁區。

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

A.1.5. error 映射目標

若使用了 error 映射目標的話，任何針對於映射的磁區的 I/O 作業都會失敗。

error 映射目標可用來進行測試。若要測試某個裝置在錯誤的情況下會有什麼特性，您可建立一個裝置映射，並且在該裝置中間含有一個錯誤的磁區，或是您可將鏡像的一個 leg 替換為另一個 error 目標。

一個 error 目標可被用來取代一個發生錯誤的裝置，這是個避免在實際的裝置上逾時和進行重新嘗試的方式。它可被用來作為一個當您在發生錯誤時進行 LVM metadata 的重整時的媒介目標。

error 映射目標除了 start 和 length 這兩個參數之外不接受額外的參數。

下列範例顯示了一個 error 目標。

```
0 65536 error
```

A.1.6. zero 映射目標

zero 映射目標是個相當於 /dev/zero 的區塊裝置。對於此映射所進行的讀取作業會回傳一些零的區塊。寫至此映射的資料將會被丟棄，不過寫入作業會成功。zero 映射目標不接受 start 和 length 參數以外的額外參數。

下列範例顯示了一個 16Tb 裝置的 zero 目標。

```
0 65536 zero
```

A.1.7. multipath 映射目標

multipath 映射目標支援多路徑裝置的映射。multipath 目標的格式如下：

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

各個路徑群組都有一組 pathgroupargs 參數。

start

在虛擬裝置中啟用區塊

length

此區段的長度

#features

multipath 功能的數量以及它們的功能。若此參數為零的話，那麼就不會有 feature 這個參數並且下一個裝置映射參數會是 #handlerargs。目前只有一個受支援的 multipath 功能，queue_if_no_path。這表示此多路徑裝置目前已設為若沒有可用路徑的話便會將 I/O 作業置於佇列中。

比方說，若 multipath.conf 檔案中的 no_path_retry 選項被設為只在所有路徑都在經過幾次嘗試並被標記為失敗之後才將 I/O 作業置入佇列中的話，那麼映射就會以下列格式來顯示出，直到所有路徑檢查程式所被指定的檢查次數都已完成後。

```
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
```

當所有路徑檢查程式所被指定的檢查次數都已完成後，映射就會以下列格式顯示出。

```
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

硬體處理程式引數的數量以及那些引數。硬體處理程式會在切換路徑群組或是處理 I/O 錯誤時指定一個用來執行硬體特屬之動作的模組。若這被設為 0 的話，那麼下個參數便是 #pathgroups。

#pathgroups

路徑群組的數量。路徑群組 (path group) 代表一組路徑，並且多路徑的裝置將會在這些路徑上進行負載平衡。各個路徑群組都有一組 pathgroupargs 參數。

pathgroup

下一個嘗試的路徑群組。

pathgroupsargs

各個路徑群組都包含著下列引數：

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

路徑群組中的各個路徑都都有一組路徑引數。

pathselector

可指定使用中的演算法來判斷下個 I/O 作業該使用此路徑群組中的哪個路徑。

#selectorargs

允許此引數使用於 multipath 映射的路徑選擇器引數數量。目前，這個引數的值總會是 0。

#paths

此路徑群組中的路徑數量。

#pathargs

為此群組中各個路徑所指定的路徑引數數量。目前，這個數字總會是 1，也就是 ioreqs 引數。

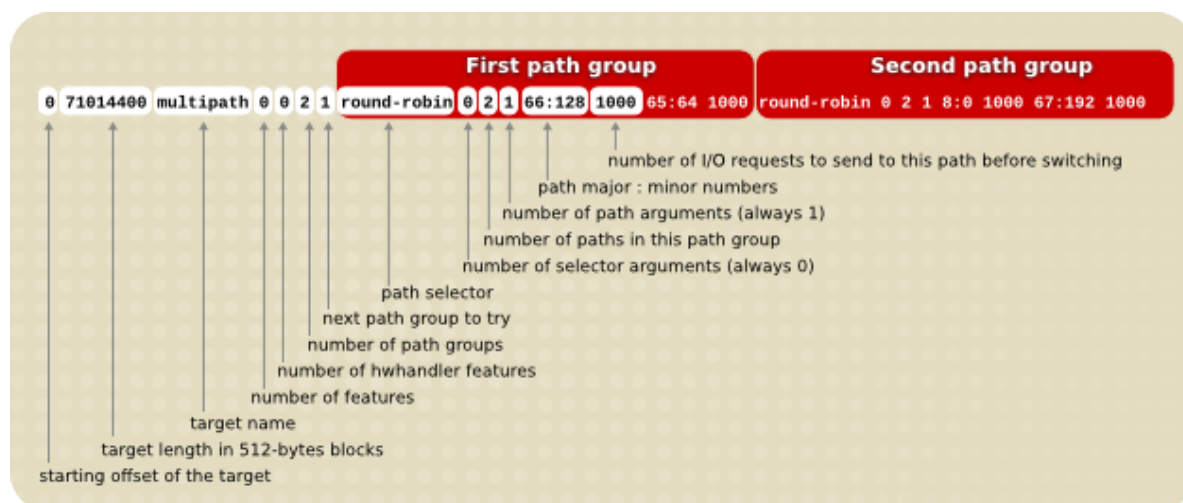
device

路徑的區塊裝置號碼，以格式為 major:minor 的 major 和 minor 數字來參照

ioreqs

要切換至目前群組中的下個路徑前所需要用來 route 至此路徑的 I/O 請求數量。

圖形 A.1, “Multipath 映射目標” shows the format of a multipath target with two path groups.



圖形 A.1. Multipath 映射目標

下列範例顯示了相同 multipath 裝置的純容錯 (failover) 目標定義。在此目標中有四個路徑群組，並且一個路徑群組只有一個開放的路徑，這樣一來 multipath 裝置便只會一次使用一個路徑。

```
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
```

下列範例顯示了相同 multipath 裝置的一個多重匯流排 (multibus) 目標定義。在此目標中只有一個路徑群組，並且它包含了所有的路徑。在此設定中，multipath 會將負載平衡地分配至所有路徑中。

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

如欲取得更多有關於進行 multipath 上的相關資訊，請參閱 Using Device Mapper Multipath 文件。

A.1.8. crypt 映射目標

crypt 目標會將通過指定裝置的資料加密。它使用了 kernel 的 Crypto API。

crypt 目標的格式如下：

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

start

在虛擬裝置中啟用區塊

length

此區段的長度

cipher

Cipher 包含著 cipher[-chainmode]-ivmode[:iv options]。

cipher

可用的 Cipher 列在 /proc/crypto 中 (比方說 aes)。

chainmode

總是使用 cbc。請勿使用 ebc；它並不使用初始向量 (IV)。

ivmode[:iv options]

IV 是個使用來改變加密的初始向量。IV 模式為 plain 或 essiv:hash。-plain 的 ivmode 使用了磁區編號 (加上 IV 偏差值) 來作為 IV。-essiv 的 ivmode 是為了避免 watermark 弱點用的。

key

加密金鑰、以十六進位提供

IV-offset

初始向量 (Initial Vector, IV) 偏差值

device

區塊裝置，被檔案系統中的裝置名稱所參照，或是被格式為 major:minor 的 major 和 minor 數字所參照

offset

在裝置上啟用映射的偏差值 (offset)

下列為 crypt 目標的範例。

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

A.2. dmsetup 指令

dmsetup 指令是一項用來和 Device Mapper 進行通訊的指令列 wrapper。如欲取得有關於 LVM 裝置的一般系統資訊，您可使用描述於下列部份的 dmsetup 指令的 info、ls、status 和 deps 選項。

如欲取得有關於 dmsetup 指令的額外選項和功能上的相關資訊，請查看 dmsetup(8) man page。

A.2.1. dmsetup info 指令

dmsetup info device 這項指令提供了有關於 Device Mapper 裝置的總覽資訊。若您不指定一個裝置名稱的話，輸出便會是有關於目前所有已配置的 Device Mapper 裝置的相關資訊。若您指定了一個裝置，那麼這項指令便只會產生該裝置的相關資訊。

dmsetup info 指令提供了下列種類的資訊：

Name

裝置的名稱。LVM 裝置是以卷冊群組名稱和邏輯卷冊名稱來表示並以連字符號來區隔開。原始名稱中的連字符號會被轉譯為兩個連字符號。

State

可能的裝置狀態有 SUSPENDED、ACTIVE 和 READ-ONLY。dmsetup suspend 指令會將裝置狀態設為 SUSPENDED。當裝置休眠 (suspend) 時，該裝置的所有 I/O 作業都會停下。dmsetup resume 指令則會將裝置狀態恢復為 ACTIVE。

Read Ahead

系統將為讀取作業繼續進行中的任何已開啓的檔案所預讀的資料區塊數量。就預設值，kernel 會自動地選擇一個適當的值。您可透過 `--readahead` option of the `dmsetup` 指令來更改這個值。

Tables present

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

Open count

Open reference count 表示了裝置被開啓了多少次。`mount` 指令可將裝置開啓。

Event number

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

Major, minor

Major 和 minor

Number of targets

構成一個裝置的片段數量。比方說一個跨距了三個磁碟的 `linear` 裝置將會有三個目標。一個由磁碟起始和結尾（少了中間）所構成的 `linear` 裝置將會有兩個目標。

UUID

裝置的 UUID。

下列範例顯示了 `dmsetup info` 指令的部份輸出。

```
[root@ask-07 ~]# dmsetup info
Name:          testgfsvg-testgfs1v1
State:         ACTIVE
Read Ahead:    256
Tables present: LIVE
Open count:    0
Event number:  0
Major, minor:  253, 2
Number of targets: 2
UUID: LVM-K528WUGQgPadNXycFrrf9LnP1UMswgkCkpgPIgYzSvigM7SfeWCypddNSWtNzc2N
...
Name:          VolGroup00-LogVol100
State:         ACTIVE
Read Ahead:    256
Tables present: LIVE
Open count:    1
Event number:  0
Major, minor:  253, 0
Number of targets: 1
UUID: LVM-t0cS1kqFV9drb0X1Vr8sxeYP0tqcrpdegqj51Zxe45JMG1mvtqLmbLpBcenh2L3
```

A.2.2. dmsetup ls 指令

您可透過 `dmsetup ls` 指令來列出映射裝置的裝置名稱。您可透過使用 `dmsetup ls --target target_type` 指令來列出擁有至少一個指定類型的目標的裝置。如欲取得其它 `dmsetup ls` 的選項的相關資訊，請參閱 `dmsetup` man page。

下列範例顯示了用來列出目前已配置的映射裝置之裝置名稱的指令。

```
[root@ask-07 ~]# dmsetup ls
testgfsvg-testgfs1v3    (253, 4)
testgfsvg-testgfs1v2    (253, 3)
testgfsvg-testgfs1v1    (253, 2)
VolGroup00-LogVol101    (253, 1)
VolGroup00-LogVol100    (253, 0)
```

下列範例顯示了用來列出目前已配置的鏡像映射之裝置名稱的指令。

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

A.2.3. dmsetup status 指令

`dmsetup status device` 這項指令提供了指定裝置中的各個目標的狀態資訊。若您不指定裝置名稱的話，輸出便會是有關於所有目前已配置的 Device Mapper 裝置的相關資訊。您只可透過 `dmsetup status --target target_type` 指令來將擁有至少一個指定類型的目標的裝置狀態列出。

下列範例顯示了用來列出目前已配置的映射裝置中的所有目標狀態的指令。

```
[root@ask-07 ~]# dmsetup status
testgfsvg-testgfs1v3: 0 312352768 linear
testgfsvg-testgfs1v2: 0 312352768 linear
testgfsvg-testgfs1v1: 0 312352768 linear
testgfsvg-testgfs1v1: 312352768 50331648 linear
VolGroup00-LogVol101: 0 4063232 linear
VolGroup00-LogVol100: 0 151912448 linear
```

A.2.4. dmsetup deps 指令

`dmsetup deps device` 指令會提供一系列裝置配對 (major、minor)，並且這些裝置配對已被指定裝置的映射表格所參照。若您不指定裝置名稱的話，輸出將會是有關於所有目前已配置的 Device Mapper 裝置的相關資訊。

下列範例顯示了用來列出目前已配置的映射裝置的所有相依的指令。

```
[root@ask-07 ~]# dmsetup deps
testgfsvg-testgfs1v3: 1 dependencies : (8, 16)
testgfsvg-testgfs1v2: 1 dependencies : (8, 16)
testgfsvg-testgfs1v1: 1 dependencies : (8, 16)
VolGroup00-LogVol101: 1 dependencies : (8, 2)
VolGroup00-LogVol100: 1 dependencies : (8, 2)
```

下列範例顯示了只用來列出 `lock_stress-grant--02.1722` 裝置之相依的指令：

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

附錄 B. LVM 配置檔案

LVM 支援多重配置檔案。當系統啟動時，`lvm.conf` 配置檔案會被由 `LVM_SYSTEM_DIR` 這個環境變數所指定的目錄載入，預設值會是 `/etc/lvm`。

`lvm.conf` 檔案可指定所要載入的額外配置檔案。檔案中之後的設定將會覆蓋先前的設定。若要在載入所有配置檔案之後顯示使用中的設定，請執行 `lvm dumpconfig` 指令。

For information on loading additional configuration files, see [節 C.2, “主機標籤 \(Host Tags\)”](#)。

B.1. LVM 配置檔案

下列檔案為 LVM 的相關配置檔案：

`/etc/lvm/lvm.conf`

工具所讀取的中央配置檔案。

`etc/lvm/lvm_hosttag.conf`

For each host tag, an extra configuration file is read if it exists: `lvm_hosttag.conf`. If that file defines new tags, then further configuration files will be appended to the list of files to read in. For information on host tags, see [節 C.2, “主機標籤 \(Host Tags\)”](#)。

除了 LVM 配置檔案以外，執行 LVM 的系統還包含了下列可影響 LVM 系統設定的檔案：

`/etc/lvm/.cache`

裝置名稱過濾快取檔案（可配置）。

`/etc/lvm/backup/`

自動卷冊群組 metadata 備份的目錄（可配置）。

`/etc/lvm/archive/`

自動卷冊群組 metadata 壓縮檔的目錄（可藉由目錄路徑和 `archive history` 深度來進行配置）。

`/var/lock/lvm/`

在單主機的配置中，檔案會被鎖定來預防 `parallel tool` 執行並損壞 metadata；在叢集中，重集全域的 DLM 會被使用。

B.2. 範例 `lvm.conf` 檔案

以下為 `lvm.conf` 配置檔案的範例。此配置檔為 RHEL 5.3 發行版的預設檔案。若您的系統執行的是不同發行版的 RHEL 5 的話，有些預設設定可能會有所不同。

```
[root@tng3-1 lvm]# cat /etc/lvm/lvm.conf
# This is an example configuration file for the LVM2 system.
# It contains the default settings that would be used if there was no
# /etc/lvm/lvm.conf file.
#
# Refer to 'man lvm.conf' for further information including the file layout.
#
# To put this file in a different directory and override /etc/lvm set
# the environment variable LVM_SYSTEM_DIR before running the tools.

# This section allows you to configure which block devices should
# be used by the LVM system.
```

```

devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # If several entries in the scanned directories correspond to the
    # same block device and the tools need to display a name for device,
    # all the pathnames are matched against each item in the following
    # list of regular expressions in turn and the first match is used.
    preferred_names = [ ]

    # Try to avoid using un-descriptive /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 are symbolic links or multiple filesystem
    # entries for the same device as each name is checked separately against
    # the list of patterns. The effect is that if any name matches any 'a'
    # pattern, the device is accepted; otherwise if any name matches any 'r'
    # pattern it is rejected; otherwise it is accepted.

    # Don't have more than one filter line active at once: only one gets used.

    # Run vgscan after you change this parameter to ensure that
    # the cache file gets regenerated (see below).
    # If it doesn't do what you expect, check the output of 'vgscan -vvvv'.

    # By default we accept every block device:
    filter = [ "a/*/" ]

    # Exclude the cdrom drive
    # filter = [ "r|/dev/cdrom|" ]

    # When testing I like to work with just loopback devices:
    # filter = [ "a|loop/", "r/*/" ]

    # Or maybe all loops and ide drives except hdc:
    # filter = [ "a|loop|", "r|/dev/hdc|", "a|/dev/ide|", "r|.*|" ]

    # Use anchors if you want to be really specific
    # filter = [ "a|^/dev/hda8$", "r/*/" ]

    # The results of the filtering are cached on disk to avoid
    # rescanning dud devices (which can take a very long time).
    # By default this cache is stored in the /etc/lvm/cache directory
    # in a file called '.cache'.
    # It is safe to delete the contents: the tools regenerate it.
    # (The old setting 'cache' is still respected if neither of
    # these new ones is present.)
    cache_dir = "/etc/lvm/cache"
    cache_file_prefix = ""

    # You can turn off writing this cache file by setting this to 0.
    write_cache_state = 1

```

```

# Advanced settings.

# List of pairs of additional acceptable block device types found
# in /proc/devices with maximum (non-zero) number of partitions.
# types = [ "fd", 16 ]

# If sysfs is mounted (2.6 kernels) restrict device scanning to
# the block devices it believes are valid.
# 1 enables; 0 disables.
sysfs_scan = 1

# By default, LVM2 will ignore devices used as components of
# software RAID (md) devices by looking for md superblocks.
# 1 enables; 0 disables.
md_component_detection = 1

# By default, if a PV is placed directly upon an md device, LVM2
# will align its data blocks with the the chunk_size exposed in sysfs.
# 1 enables; 0 disables.
md_chunk_alignment = 1

# If, while scanning the system for PVs, LVM2 encounters a device-mapper
# device that has its I/O suspended, it waits for it to become accessible.
# Set this to 1 to skip such devices. This should only be needed
# in recovery situations.
ignore_suspended_devices = 0
}

# This section that allows you to configure the nature of the
# information that LVM2 reports.
log {

    # Controls the messages sent to stdout or stderr.
    # There are three levels of verbosity, 3 being the most verbose.
    verbose = 0

    # Should we send log messages through syslog?
    # 1 is yes; 0 is no.
    syslog = 1

    # Should we log error and debug messages to a file?
    # By default there is no log file.
    #file = "/var/log/lvm2.log"

    # Should we overwrite the log file each time the program is run?
    # By default we append.
    overwrite = 0

    # What level of log messages should we send to the log file and/or syslog?
    # There are 6 syslog-like log levels currently in use - 2 to 7 inclusive.
    # 7 is the most verbose (LOG_DEBUG).
    level = 0

    # Format of output messages
    # Whether or not (1 or 0) to indent messages according to their severity
    indent = 1

    # Whether or not (1 or 0) to display the command name on each line output
    command_names = 0

    # A prefix to use before the message text (but after the command name,
    # if selected). Default is two spaces, so you can see/grep the severity
    # of each message.
    prefix = "  "

    # To make the messages look similar to the original LVM tools use:
    # indent = 0

```

```
# command_names = 1
# prefix = " -- "

# Set this if you want log messages during activation.
# Don't use this in low memory situations (can deadlock).
# activation = 0
}

# Configuration of metadata backups and archiving. In LVM2 when we
# talk about a 'backup' we mean making a copy of the metadata for the
# *current* system. The 'archive' contains old metadata configurations.
# Backups are stored in a human readable text format.
backup {

    # Should we maintain a backup of the current metadata configuration ?
    # Use 1 for Yes; 0 for No.
    # Think very hard before turning this off!
    backup = 1

    # Where shall we keep it ?
    # Remember to back up this directory regularly!
    backup_dir = "/etc/lvm/backup"

    # Should we maintain an archive of old metadata configurations.
    # Use 1 for Yes; 0 for No.
    # On by default. Think very hard before turning this off.
    archive = 1

    # Where should archived files go ?
    # Remember to back up this directory regularly!
    archive_dir = "/etc/lvm/archive"

    # What is the minimum number of archive files you wish to keep ?
    retain_min = 10

    # What is the minimum time you wish to keep an archive file for ?
    retain_days = 30
}

# Settings for the running LVM2 in shell (readline) mode.
shell {

    # Number of lines of history to store in ~/.lvm_history
    history_size = 100
}

# Miscellaneous global LVM2 settings
global {
    library_dir = "/usr/lib64"

    # The file creation mask for any files and directories created.
    # Interpreted as octal if the first digit is zero.
    umask = 077

    # Allow other users to read the files
    #umask = 022

    # Enabling test mode means that no changes to the on disk metadata
    # will be made. Equivalent to having the -t option on every
    # command. Defaults to off.
    test = 0

    # Default value for --units argument
    units = "h"

    # Whether or not to communicate with the kernel device-mapper.
```

```

# Set to 0 if you want to use the tools to manipulate LVM metadata
# without activating any logical volumes.
# If the device-mapper kernel driver is not present in your kernel
# setting this to 0 should suppress the error messages.
activation = 1

# If we can't communicate with device-mapper, should we try running
# the LVM1 tools?
# This option only applies to 2.4 kernels and is provided to help you
# switch between device-mapper kernels and LVM1 kernels.
# The LVM1 tools need to be installed with .lvm1 suffices
# e.g. vgscan.lvm1 and they will stop working after you start using
# the new lvm2 on-disk metadata format.
# The default value is set when the tools are built.
# fallback_to_lvm1 = 0

# The default metadata format that commands should use - "lvm1" or "lvm2".
# The command line override is -M1 or -M2.
# Defaults to "lvm1" if compiled in, else "lvm2".
# format = "lvm1"

# Location of proc filesystem
proc = "/proc"

# Type of locking to use. Defaults to local file-based locking (1).
# Turn locking off by setting to 0 (dangerous: risks metadata corruption
# if LVM2 commands get run concurrently).
# Type 2 uses the external shared library locking_library.
# Type 3 uses built-in clustered locking.
locking_type = 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"

# Other entries can go here to allow you to load shared libraries
# e.g. if support for LVM1 metadata was compiled as a shared library use
# format_libraries = "liblvm2format1.so"
# Full pathnames can be given.

# Search this directory first for shared libraries.
# library_dir = "/lib"

# The external locking library to load if locking_type is set to 2.
# locking_library = "liblvm2clusterlock.so"
}

activation {
# How to fill in missing stripes if activating an incomplete volume.
# Using "error" will make inaccessible parts of the device return
# I/O errors on access. You can instead use a device path, in which
# case, that device will be used to in place of missing stripes.
# But note that using anything other than "error" with mirrored
# or snapshotted volumes is likely to result in data corruption.

```

```

missing_stripe_filler = "error"

# How much stack (in KB) to reserve for use while devices suspended
reserved_stack = 256

# How much memory (in KB) to reserve for use while devices suspended
reserved_memory = 8192

# Nice value used while devices suspended
process_priority = -18

# If volume_list is defined, each LV is only activated if there is a
# match against the list.
# "vgname" and "vgname/lvname" are matched exactly.
# "@tag" matches any tag set in the LV or VG.
# "@*" matches if any tag defined on the host is also set in the LV or VG
#
# volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]

# Size (in KB) of each copy operation when mirroring
mirror_region_size = 512

# Setting to use when there is no readahead value stored in the metadata.
#
# "none" - Disable readahead.
# "auto" - Use default value chosen by kernel.
readahead = "auto"

# 'mirror_image_fault_policy' and 'mirror_log_fault_policy' define
# how a device failure affecting a mirror is handled.
# A mirror is composed of mirror images (copies) and a log.
# A disk log ensures that a mirror does not need to be re-synced
# (all copies made the same) every time a machine reboots or crashes.
#
# In the event of a failure, the specified policy will be used to
# determine what happens:
#
# "remove" - Simply remove the faulty device and run without it. If
#             the log device fails, the mirror would convert to using
#             an in-memory log. This means the mirror will not
#             remember its sync status across crashes/reboots and
#             the entire mirror will be re-synced. If a
#             mirror image fails, the mirror will convert to a
#             non-mirrored device if there is only one remaining good
#             copy.
#
# "allocate" - Remove the faulty device and try to allocate space on
#              a new device to be a replacement for the failed device.
#              Using this policy for the log is fast and maintains the
#              ability to remember sync state through crashes/reboots.
#              Using this policy for a mirror device is slow, as it
#              requires the mirror to resynchronize the devices, but it
#              will preserve the mirror characteristic of the device.
#              This policy acts like "remove" if no suitable device and
#              space can be allocated for the replacement.
#              Currently this is not implemented properly and behaves
#              similarly to:
#
# "allocate_anywhere" - Operates like "allocate", but it does not
#                       require that the new space being allocated be on a
#                       device is not part of the mirror. For a log device
#                       failure, this could mean that the log is allocated on
#                       the same device as a mirror device. For a mirror
#                       device, this could mean that the mirror device is
#                       allocated on the same device as another mirror device.
#                       This policy would not be wise for mirror devices
#                       because it would break the redundant nature of the

```

```

#           mirror. This policy acts like "remove" if no suitable
#           device and space can be allocated for the replacement.

mirror_log_fault_policy = "allocate"
mirror_device_fault_policy = "remove"
}

#####
# Advanced section #
#####

# Metadata settings
#
# metadata {
#   # Default number of copies of metadata to hold on each PV. 0, 1 or 2.
#   # You might want to override it from the command line with 0
#   # when running pvcreate on new PVs which are to be added to large VGs.

#   # pvmetadacopies = 1

#   # Approximate default size of on-disk metadata areas in sectors.
#   # You should increase this if you have large volume groups or
#   # you want to retain a large on-disk history of your metadata changes.

#   # pvmetadatasize = 255

#   # List of directories holding live copies of text format metadata.
#   # These directories must not be on logical volumes!
#   # It's possible to use LVM2 with a couple of directories here,
#   # preferably on different (non-LV) filesystems, and with no other
#   # on-disk metadata (pvmetadacopies = 0). Or this can be in
#   # addition to on-disk metadata areas.
#   # The feature was originally added to simplify testing and is not
#   # supported under low memory situations - the machine could lock up.
#   #
#   # Never edit any files in these directories by hand unless you
#   # you are absolutely sure you know what you are doing! Use
#   # the supplied toolset to make changes (e.g. vgcfgrestore).

#   # dirs = [ "/etc/lvm/metadata", "/mnt/disk2/lvm/metadata2" ]
#}

# Event daemon
#
dmeventd {
#   # mirror_library is the library used when monitoring a mirror device.
#   #
#   # "libdevmapper-event-lvm2mirror.so" attempts to recover from
#   # failures. It removes failed devices from a volume group and
#   # reconfigures a mirror as necessary. If no mirror library is
#   # provided, mirrors are not monitored through dmeventd.

#   mirror_library = "libdevmapper-event-lvm2mirror.so"

#   # snapshot_library is the library used when monitoring a snapshot device.
#   #
#   # "libdevmapper-event-lvm2snapshot.so" monitors the filling of
#   # snapshots and emits a warning through syslog, when the use of
#   # snapshot exceeds 80%. The warning is repeated when 85%, 90% and
#   # 95% of the snapshot are filled.

#   snapshot_library = "libdevmapper-event-lvm2snapshot.so"
}

```

附錄 C. LVM 物件標籤 (Object Tags)

LVM 標籤是個可用來將類型相同的 LVM2 物件組織在一起的字串。標籤可被連至像是實體卷冊、卷冊群組，以及邏輯卷冊的物件。標籤可在叢集配置中被連至主機。快照 (Snapshot) 無法被標記。

標籤可代替 PV、VG 或 LV 引數在指令列上提供。標籤應以一個 @ 來作為字首以避免意義不明確。各個標籤都是透過將它取代為持有該標籤、類型基於它在指令列上的位置來斷定的所有物件來擴充的。

LVM 標籤為使用 [A-Za-z0-9_+.-] 達 128 字元的字串。它們不可以連字符號作為起始。

只有卷冊群組中的物件可被標記。實體卷冊若由卷冊群組中被移除掉的話，它們便會失去它們的標籤；這是因為標籤會被作為是卷冊群組 metadata 的一部分來儲存，因此當某個實體卷冊被移除時，它們也會跟著被刪除掉。快照無法被標記。

下列指令列出了所有標有著 database 標籤的邏輯卷冊。

```
lvs @database
```

C.1. 新增和移除物件標籤

若要新增或移除實體卷冊的標籤，請使用 `pvchange` 指令的 `--addtag` 或 `--deltag` 選項。

若要新增或移除卷冊群組的標籤，請使用 `vgchange` 或 `vgcreate` 指令的 `--addtag` 或 `--deltag` 選項。

若要新增或移除邏輯卷冊的標籤，請使用 `lvchange` 或 `lvcreate` 指令的 `--addtag` 或 `--deltag` 選項。

C.2. 主機標籤 (Host Tags)

In a cluster configuration, you can define host tags in the configuration files. If you set `hosttags = 1` in the `tags` section, a host tag is automatically defined using the machine's hostname. This allow you to use a common configuration file which can be replicated on all your machines so they hold identical copies of the file, but the behavior can differ between machines according to the hostname.

For information on the configuration files, see [附錄 B, LVM 配置檔案](#).

針對於各個主機標籤，會有個額外的配置檔案被讀取（若它存在的話）：`lvm_hosttag.conf`。若該檔案定義了新的標籤的話，那麼額外的配置檔案便會被附加至需要被讀取的檔案清單中。

比方說，每當主機名稱為 `host1` 時，下列配置檔案中的項目就會定義 `tag1` 以及 `tag2`。

```
tags { tag1 { } tag2 { host_list = ["host1"] } }
```

C.3. 利用標籤來控制啓用

您可在配置檔案中指定只有特定邏輯卷冊可在該主機上被啓用。比方說，下列項目被作為是一個啓用請求（例如 `vgchange -ay`）的過濾器，並且只會啓用 `vg1/lvo10` 和該主機上在 `metadata` 中含有 `database` 標籤的邏輯卷冊或卷冊群組。

```
activation { volume_list = ["vg1/lvo10", "@database" ] }
```

There is a special match "@" that causes a match only if any metadata tag matches any host tag on that machine.

讓我們思考另一個當叢集中所有的機器在配置檔案中都含有下列項目的範例:

```
tags { hosttags = 1 }
```

若您只希望在 db2 主機上啟用 vg1/lvo12 的話, 請進行下列步驟:

1. 在叢集中的任何主機上執行 `lvchange --addtag @db2 vg1/lvo12`。
2. 執行 `lvchange -ay vg1/lvo12`。

這個方法需要將主機名稱儲存在卷冊群組 metadata 中。

附錄 D. LVM 卷冊群組 Metadata

卷冊群組的配置詳情又被稱為是 metadata。就預設值，卷冊群組中所有實體卷冊中的每個 metadata 區域中都會有個 metadata 的副本。LVM 卷冊群組 metadata 非常小並且被儲存為 ASCII。

若有個卷冊群組包含著許多實體卷冊，含有許多重複的 metadata 副本是非常沒有效率的。您可透過使用 `pvcreate` 指令的 `--metadatacopies 0` 選項來建立一個實體卷冊並且不建立任何 metadata 的副本。一旦您選擇了實體卷冊將會包含的 metadata 副本數量後，您之後便無法再針對它進行變更。不過請注意，不管任何時候，每個卷冊群組都必須包含著至少一個實體卷冊以及一個 metadata 區域（除非您使用了一項能讓您將卷冊群組 metadata 儲存在檔案系統中的進階配置設定）。若您打算在未來將卷冊群組切割的話，所有卷冊群組就都需要至少一個 metadata 副本。

核心的 metadata 是以 ASCII 來儲存的。metadata 區域是個循環緩衝（circular buffer）。新的 metadata 會被附加至較舊的 metadata 然後指向它起始的指標（pointer）將會被更新。

您可透過使用 `pvcreate` 指令的 `--metadatasize` 選項來指定 metadata 區域的大小。預設的大小對於含有許多邏輯卷冊或實體卷冊的卷冊群組來說太小了。

D.1. 實體卷冊標籤（Physical Volume Label）

就預設值，`pvcreate` 指令會將實體卷冊標籤放置在第二個 512 位元組的磁區中。這個標籤亦可被選擇性地放置於前四個磁區中的任何一個，因為掃描實體卷冊標籤的 LVM 工具會檢查前四個磁區。實體卷冊的標籤是以 LABELONE 這個字串作為起始的。

實體卷冊標籤包含著：

- 實體卷冊的 UUID
- 區塊裝置的大小（以位元組為單位）
- 無終結（NULL-terminated）的資料區域位置清單
- 無終結的 metadata 區域位置清單

Metadata 的位置是以偏差值和大小（單位為位元組）來儲存的。標籤中可放置 15 個左右的位置，不過 LVM 工具目前只使用了 3 個：一個單獨的資料區域加上兩個 metadata 區域。

D.2. Metadata 內容

卷冊群組 metadata 中包含著：

- 有關於它如何以及何時被建立的相關資訊
- 有關於卷冊群組的資訊

卷冊群組資訊包含著：

- 名稱和特殊 id
- 一個每當 metadata 被更新時便會跟著遞增的版本號碼
- 任何屬性：可讀取/寫入？可重設大小？
- 任何它所可能包含的實體卷冊和邏輯卷冊數量上的管理限制
- 扇區大小（以磁區 [sector] 為單位，定義為 512 個位元組）
- 一系列未經順序排序、構成卷冊群組的實體卷冊，各個都含有：

- 它的 UUID，用來測定包含著它的區塊裝置
- 任何屬性，例如實體卷冊是否可被分配
- 實體卷冊中第一個扇區起始的 offset（單位為磁區 [sector]）
- 扇區數量
- 一系列未經順序排序的邏輯卷冊。各個都含有
 - 一系列經過順序排序的邏輯卷冊區段（logical volume segment）。Metadata 會針對於各個區段來包含一個映射以套用至經過排序的實體卷冊區段或是邏輯卷冊區段

D.3. Metadata 範例

下列顯示了一個稱為 myvg 的卷冊群組的 LVM 卷冊群組 metadata 範例。

```
# 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.e15 #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-1DHq-1MPs-EjoE-Oo18-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-2R1V-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-j06s-RAVx-UREWOG"
            device = "/dev/sdc"     # Hint only

            status = ["ALLOCATABLE"]

```

```

        dev_size = 35964301      # 17.1491 Gigabytes
        pe_start = 384
        pe_count = 4390 # 17.1484 Gigabytes
    }

    pv3 {
        id = "hG1Uwi-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-Ur90F9"
        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
            ]
        }
    }
}
}

```

附錄 E. 修訂歷史

修訂 5.4-1-0 Tue Aug 18 2009

Steven Levine slevine@redhat.com

更換了叢集邏輯卷冊管理員 (Cluster Logical Volume Manager) , 修正了該文件中的錯誤。

已修正: #510273

確認缺少了叢集卷冊群組的圖示支援。

已修正: #515742

針對於大於 1.5TB 的鏡像預設值增加鏡像區域大小。

已修正: #491028

修正了各種字彙以及格式上的錯誤。

已修正: #491028

修正了各種字彙以及格式上的錯誤。

已修正: #504028

提供了有關於在叢集中建立鏡像的新部份以及相關文件; 一般技術修正。

已修正: #494007

清楚描述了 snapshot 大小的設定。

已修正: #504028

新增了在叢集中建立鏡像卷冊的範例。

已修正: #518567

新增了確認叢集中所有節點皆可存取共享儲存裝置的警告。

已修正: #510920

修正了錯字。

索引

A

- activating logical volumes
 - individual nodes, 40
- activating volume groups, 27
 - individual nodes, 27
 - local node only, 27
- administrative procedures,
- allocation
 - policy, 23
 - preventing, 22
- archive file, 17, 28

B

- backup
 - file, 17
 - metadata, 17, 28
- backup file, 28
- block device
 - scanning, 21

C

- cache file
 - building, 25
- cluster environment, 2, 15
- CLVM
 - definition, 2
- clvmd daemon, 3
- command line units, 19
- configuration examples,
- creating
 - logical volume, 29
 - logical volume, example, 51
 - LVM volumes in a cluster, 15
 - physical volumes, 20
 - striped logical volume, example, 52
 - volume group, clustered, 24
 - volume groups, 23
- creating LVM volumes
 - overview, 16

D

- data relocation, online, 39
- deactivating volume groups, 27
 - exclusive on one node, 27
 - local node only, 27
- device numbers
 - major, 33
 - minor, 33
 - persistent, 33
- device path names, 19
- device scan filters, 39

- device size, maximum, 24
- device special file directory, 24
- display
 - sorting output, 47
- displaying
 - logical volumes, 35, 45
 - physical volumes, 22, 43
 - volume groups, 25, 44

E

- extent
 - allocation, 23
 - definition, 8, 23

F

- failed devices
 - displaying, 61
- feedback, viii, viii
- file system
 - growing on a logical volume, 16
- filters, 39

G

- growing file system
 - logical volume, 16

H

- help display, 20

I

- initializing
 - partitions, 21
 - physical volumes, 21
- Insufficient Free Extents message, 66

L

- linear logical volume
 - converting to mirrored, 33
 - creation, 29
 - definition, 9
- logging, 17
- logical volume
 - administration, general, 29
 - changing parameters, 34
 - creation, 29
 - creation example, 51
 - definition, 1, 8
 - displaying, 35, 40, 45
 - exclusive access, 40
 - extending, 35
 - growing, 35
 - linear, 29
 - local access, 40

- lvs display arguments, 45
- mirrored, 31
- reducing, 37
- removing, 34
- renaming, 34
- resizing, 34
- shrinking, 37
- snapshot, 38
- striped, 31
- lvchange command, 34
- lvconvert command, 33
- lvcreate command, 29
- lvdisplay command, 35
- lvextend command, 35
- LVM
 - architecture overview, 1
 - clustered, 2
 - components, 2,
 - custom report format, 40
 - directory structure, 24
 - help, 20
 - history, 1
 - label, 7
 - logging, 17
 - logical volume administration, 29
 - physical volume administration, 20
 - physical volume, definition, 7
 - volume group, definition, 8
- LVM1, 1
- LVM2, 1
- lvmdiskscan command, 21
- lvreduce command, 34, 37
- lvremove command, 34
- lvrename command, 34
- lvs command, 40, 45
 - display arguments, 45
- lvscan command, 35

M

- man page display, 20
- metadata
 - backup, 17, 28
 - recovery, 64
- mirrored logical volume
 - clustered, 57
 - converting to linear, 33
 - creation, 31
 - definition, 11
 - failure recovery, 62
 - reconfiguration, 33

O

- online data relocation, 39

P

- partition type, setting, 20
- partitions
 - multiple, 8
- path names, 19
- persistent device numbers, 33
- physical extent
 - preventing allocation, 22
- physical volume
 - adding to a volume group, 24
 - administration, general, 20
 - creating, 20
 - definition, 7
 - display, 43
 - displaying, 22, 40
 - illustration, 7
 - initializing, 21
 - layout, 7
 - pvs display arguments, 43
 - recovery, 66
 - removing, 23
 - removing from volume group, 26
 - removing lost volume, 66
 - resizing, 23
- pvddisplay command, 22
- pvmove command, 39
- pvremove command, 23
- pvresize command, 23
- pvs command, 40
 - display arguments, 43
- pvscommand, 22

R

- removing
 - disk from a logical volume, 55
 - logical volume, 34
 - physical volumes, 23
- renaming
 - logical volume, 34
 - volume group, 28
- report format, LVM devices, 40
- resizing
 - logical volume, 34
 - physical volume, 23

S

- scanning
 - block devices, 21
- scanning devices, filters, 39
- snapshot logical volume
 - creation, 38
- snapshot volume
 - definition, 12

striped logical volume

- creation, 31
- creation example, 52
- definition, 10
- extending, 36
- growing, 36

T

troubleshooting,

U

units, command line, 19

V

verbose output, 19

vgcfbackup command, 28

vgcfrestore command, 28

vgchange command, 26

vgcreate command, 23, 24

vgdisplay command, 25

vgexport command, 28

vgextend command, 24

vgimport command, 28

vgmerge command, 27

vgmknodes command, 29

vgreduce command, 26

vgrename command, 28

vgs command, 40

- display arguments, 44

vgscan command, 25

vgsplit command, 27

volume group

- activating, 27
- administration, general, 23
- changing parameters, 26
- combining, 27
- creating, 23
- creating in a cluster, 24
- deactivating, 27
- definition, 8
- displaying, 25, 40, 44
- extending, 24
- growing, 24
- merging, 27
- moving between systems, 28
- reducing, 26
- removing, 27
- renaming, 28
- shrinking, 26
- splitting, 27
 - example procedure, 53
- vgs display arguments, 44
