



# Red Hat Enterprise Linux 8

## 配置和管理逻辑卷

配置和管理 LVM





## Legal Notice

Copyright © 2025 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, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## Abstract

逻辑卷管理器(LVM)是一种存储虚拟化软件,旨在增强物理存储设备的管理和灵活性。通过抽象物理硬件, LVM 允许您动态创建、调整大小和删除虚拟存储设备。在这个框架中,物理卷(PV)代表分组在一起的原始存储设备,以组成卷组(VG)。在此 VG 中, LVM 分配空间来创建逻辑卷(LV)。LV 是文件系统、数据库或应用程序可以使用的虚拟块存储设备。

# Table of Contents

|   |           |
|---|-----------|
| 对红帽文档提供反馈 .....                             | 5         |
| <b>第 1 章 逻辑卷管理概述 .....</b>                  | <b>6</b>  |
| 1.1. LVM 构架 .....                           | 6         |
| 1.2. LVM 的优点 .....                          | 7         |
| <b>第 2 章 管理 LVM 物理卷 .....</b>               | <b>9</b>  |
| 2.1. 创建 LVM 物理卷 .....                       | 9         |
| 2.2. 删除 LVM 物理卷 .....                       | 9         |
| 2.3. 在 WEB 控制台中创建逻辑卷 .....                  | 10        |
| 2.4. 在 WEB 控制台中格式化逻辑卷 .....                 | 12        |
| 2.5. 在 WEB 控制台中重新定义逻辑卷大小 .....              | 15        |
| 2.6. 其他资源 .....                             | 17        |
| <b>第 3 章 管理 LVM 卷组 .....</b>                | <b>18</b> |
| 3.1. 创建 LVM 卷组 .....                        | 18        |
| 3.2. 在 WEB 控制台中创建卷组 .....                   | 19        |
| 3.3. 重命名 LVM 卷组 .....                       | 20        |
| 3.4. 扩展 LVM 卷组 .....                        | 21        |
| 3.5. 合并 LVM 卷组 .....                        | 22        |
| 3.6. 从卷组中删除物理卷 .....                        | 22        |
| 3.7. 分割 LVM 卷组 .....                        | 23        |
| 3.8. 将卷组移动到另一个系统中 .....                     | 24        |
| 3.9. 删除 LVM 卷组 .....                        | 26        |
| 3.10. 在集群环境中删除 LVM 卷组 .....                 | 26        |
| <b>第 4 章 基本逻辑卷管理 .....</b>                  | <b>28</b> |
| 4.1. 逻辑卷功能概述 .....                          | 28        |
| 4.2. 创建逻辑卷 .....                            | 28        |
| 4.3. 重新调整逻辑卷大小 .....                        | 34        |
| 4.4. 重命名逻辑卷 .....                           | 41        |
| 4.5. 删除逻辑卷 .....                            | 41        |
| 4.6. 激活逻辑卷 .....                            | 42        |
| 4.7. 停用逻辑卷 .....                            | 43        |
| <b>第 5 章 高级逻辑卷管理 .....</b>                  | <b>45</b> |
| 5.1. 管理逻辑卷快照 .....                          | 45        |
| 5.2. 缓存逻辑卷 .....                            | 51        |
| 5.3. 创建自定义精简池 .....                         | 54        |
| <b>第 6 章 使用快照管理系统升级 .....</b>               | <b>56</b> |
| 6.1. BOOM 过程概述 .....                        | 56        |
| 6.2. 使用 BOOM BOOT MANAGER 升级至另一个版本 .....    | 56        |
| 6.3. 在 RED HAT ENTERPRISE LINUX 版本间切换 ..... | 60        |
| 6.4. 成功升级后删除逻辑卷快照 .....                     | 60        |
| 6.5. 在升级后创建回滚引导条目 .....                     | 61        |
| <b>第 7 章 自定义 LVM 报告 .....</b>               | <b>64</b> |
| 7.1. 控制 LVM 显示的格式 .....                     | 64        |
| 7.2. 为 LVM 显示指定单元 .....                     | 65        |
| 7.3. 自定义 LVM 配置文件 .....                     | 67        |
| 7.4. 定义 LVM 选择标准 .....                      | 68        |

|   |            |
|---|------------|
| <b>第 8 章 在共享存储上配置 LVM</b>                   | <b>70</b>  |
| 8.1. 为虚拟机磁盘配置 LVM                           | 70         |
| 8.2. 将 LVM 配置为在一台机器上使用 SAN 磁盘               | 70         |
| 8.3. 配置 LVM，以使用 SAN 磁盘进行故障转移                | 71         |
| 8.4. 配置 LVM，来在多台机器间共享 SAN 磁盘                | 71         |
| 8.5. 使用 STORAGE RHEL 系统角色创建共享的 LVM 设备       | 72         |
| <b>第 9 章 配置 RAID 逻辑卷</b>                    | <b>74</b>  |
| 9.1. RAID 级别和线性支持                           | 74         |
| 9.2. LVM RAID 片段类型                          | 75         |
| 9.3. 创建 RAID0 的参数                           | 77         |
| 9.4. 创建 RAID 逻辑卷                            | 77         |
| 9.5. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池  | 78         |
| 9.6. 创建 RAID0 条带化逻辑卷                        | 79         |
| 9.7. 使用 STORAGE RHEL 系统角色为 RAID LVM 卷配置条带大小 | 80         |
| 9.8. 软数据崩溃                                  | 81         |
| 9.9. 创建具有 DM 完整性的 RAID 逻辑卷                  | 82         |
| 9.10. 将 RAID 逻辑卷转换为另一个 RAID 级别              | 84         |
| 9.11. 将线性设备转换为 RAID 逻辑卷                     | 85         |
| 9.12. 将 LVM RAID1 逻辑卷转换为 LVM 线性逻辑卷          | 86         |
| 9.13. 将镜像 LVM 设备转换为 RAID1 逻辑卷               | 86         |
| 9.14. 更改现有 RAID1 设备中的镜像数                    | 87         |
| 9.15. 将 RAID 镜像分离为一个独立的逻辑卷                  | 89         |
| 9.16. 分割和合并 RAID 镜像                         | 90         |
| 9.17. 将 RAID 失败策略设置为 ALLOCATE               | 91         |
| 9.18. 将 RAID 失败策略设置为 WARN                   | 92         |
| 9.19. 替换正常工作的 RAID 设备                       | 93         |
| 9.20. 在逻辑卷中替换失败的 RAID 设备                    | 95         |
| 9.21. 检查 RAID 逻辑卷中数据的一致性                    | 97         |
| 9.22. RAID1 逻辑卷上的 I/O 操作                    | 98         |
| 9.23. 重塑 RAID 卷                             | 99         |
| 9.24. 在 RAID 逻辑卷中更改区域大小                     | 101        |
| <b>第 10 章 限制 LVM 设备可见性和用法</b>               | <b>104</b> |
| 10.1. LVM 过滤的持久性标识符                         | 104        |
| 10.2. LVM 设备过滤器                             | 104        |
| <b>第 11 章 控制 LVM 分配</b>                     | <b>107</b> |
| 11.1. 从指定设备分配扩展                             | 107        |
| 11.2. LVM 分配策略                              | 109        |
| 11.3. 防止在物理卷中分配                             | 110        |
| <b>第 12 章 使用标签对 LVM 对象进行分组</b>              | <b>111</b> |
| 12.1. LVM 对象标签                              | 111        |
| 12.2. 列出 LVM 标签                             | 111        |
| 12.3. 向 LVM 对象添加标签                          | 111        |
| 12.4. 从 LVM 对象中删除标签                         | 112        |
| 12.5. 定义 LVM 主机标签                           | 112        |
| 12.6. 使用标签控制逻辑卷激活                           | 113        |
| <b>第 13 章 LVM 故障排除</b>                      | <b>114</b> |
| 13.1. 在 LVM 中收集诊断数据                         | 114        |
| 13.2. 显示有关失败的 LVM 设备的信息                     | 115        |
| 13.3. 从卷组中删除丢失的 LVM 物理卷                     | 116        |

|   |     |
|---|-----|
| 13.4. 查找丢失的 LVM 物理卷的元数据                                   | 117 |
| 13.5. 在 LVM 物理卷中恢复元数据                                     | 117 |
| 13.6. LVM 输出中的轮询错误  | 119 |
| 13.7. 防止创建 LVM 卷时出现循环错误                                   | 119 |
| 13.8. LVM 元数据及其在磁盘上的位置                                    | 120 |
| 13.9. 从磁盘中提取 VG 元数据                                       | 121 |
| 13.10. 将提取的元数据保存到文件中                                      | 123 |
| 13.11. 使用 PVCREATE 和 VGCFGRESTORE 命令修复带有损坏的 LVM 标头和元数据的磁盘 | 123 |
| 13.12. 使用 PVCK 命令修复带有损坏的 LVM 标头和元数据的磁盘                    | 125 |
| 13.13. LVM RAID 故障排除                                      | 126 |
| 13.14. 对多路径 LVM 设备进行重复的物理卷警告进行故障排除                        | 129 |





## 对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

### 通过 Jira 提交反馈（需要帐户）

1. 登录到 [Jira](#) 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

## 第 1 章 逻辑卷管理概述

逻辑卷管理器(LVM)在物理存储上创建抽象层，可帮助您创建逻辑存储卷。与直接物理存储使用相比，这提供了更大的灵活性。

另外，硬件存储配置在软件中是隐藏的，因此您可以在不停止应用程序或卸载文件系统的情况下调整大小和移动。这可降低操作成本。

### 1.1. LVM 构架

以下是 LVM 组件：

#### 物理卷

物理卷(PV)是指定为 LVM 使用的分区或整个磁盘。如需更多信息，请参阅[管理 LVM 物理卷](#)。

#### 卷组

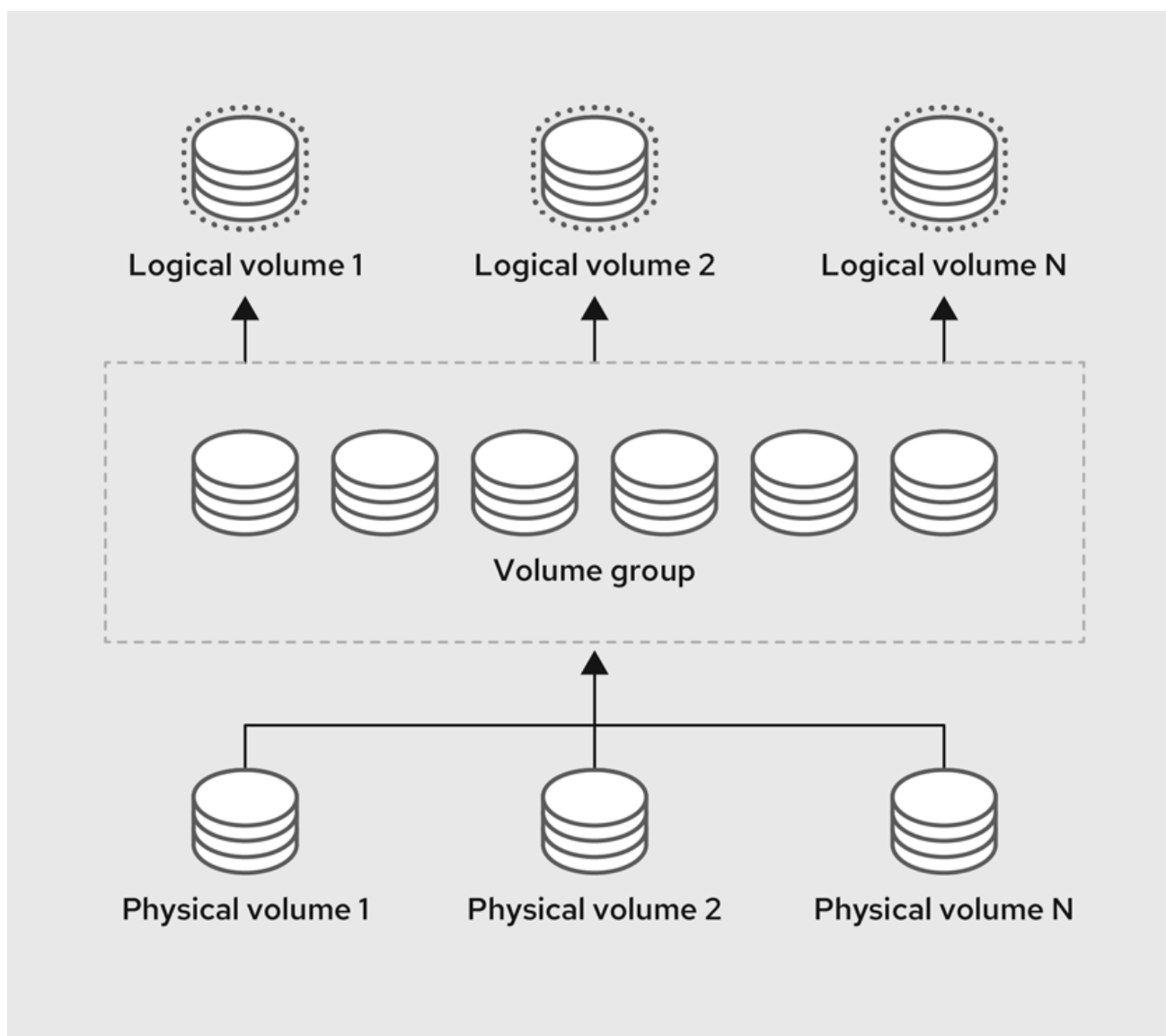
卷组(VG)是物理卷(PV)的集合，它会创建一个磁盘空间池，您可以分配逻辑卷。如需更多信息，请参阅[管理 LVM 卷组](#)。

#### 逻辑卷

逻辑卷代表可用的存储设备。如需更多信息，请参阅[基本逻辑卷管理](#)和[高级逻辑卷管理](#)。

下图显示了 LVM 的组件：

图 1.1. LVM 逻辑卷组件



## 1.2. LVM 的优点

与直接使用物理存储相比，逻辑卷具有以下优势：

### 灵活的容量

使用逻辑卷时，您可以将设备和分区聚合到一个逻辑卷中。借助此功能，文件系统可以扩展到多个设备中，就像它们是一个单一的大型设备一样。

### 方便设备命名

逻辑卷可以使用用户定义的名称和自定义名称进行管理。

### 存储卷大小

您可以使用简单的软件命令扩展逻辑卷或减小逻辑卷大小，而无需重新格式化和重新分区基础设备。如需更多信息，[请参阅重新定义逻辑卷大小](#)。

### 在线数据重新定位

要部署更新、更快或更弹性的存储子系统，您可以在系统处于活动状态时，使用 **pvmove** 命令移动数据。在磁盘处于使用状态时可以重新分配磁盘。例如，您可以在删除热插拔磁盘前将其清空。有关如何迁移数据的更多信息，请参阅 **pvmove** 手册页，以及 [从卷组中删除物理卷](#)。

### 条带化卷

您可以创建一个在两个或者多个设备间条带化分布数据的逻辑卷。这可显著提高吞吐量。如需更多信息，[请参阅创建条带逻辑卷](#)。

## RAID 卷

逻辑卷为您对数据配置 RAID 提供了一种便捷的方式。这可防止设备故障并提高性能。如需更多信息，[请参阅配置 RAID 逻辑卷](#)。

## 卷快照

您可以对数据进行快照（逻辑卷在一个特点时间点上的副本）用于一致性备份或测试更改的影响，而不影响实际数据。如需更多信息，[请参阅管理逻辑卷快照](#)。

## 精简卷

逻辑卷可以是精简调配的。这可让您创建大于可用物理空间的逻辑卷。如需更多信息，[请参阅创建精简逻辑卷](#)。

## Caching

缓存使用 SSD 等快速设备来缓存逻辑卷中的数据，从而提高性能。如需更多信息，[请参阅缓存逻辑卷](#)。

## 其他资源

- [自定义 LVM 报告](#)

## 第 2 章 管理 LVM 物理卷

物理卷(PV)是物理存储设备或者 LVM 使用的存储设备中的分区。

在初始化过程中，LVM 磁盘标签和元数据被写入该设备，允许 LVM 在逻辑卷管理方案中跟踪和管理它。



### 注意

在初始化后，您无法增加元数据的大小。如果您需要较大的元数据，您必须在初始化过程中设置适当的大小。

初始化过程完成后，您可以将 PV 分配给卷组(VG)。您可以将这个 VG 分成逻辑卷(LV)，它们是操作系统和应用程序可用于存储的虚拟块设备。

为确保最佳性能，请将整个磁盘作为单个 PV 进行分区以供 LVM 使用。

### 2.1. 创建 LVM 物理卷

您可以使用 **pvcreate** 命令初始化物理卷 LVM 使用情况。

#### 先决条件

- 管理访问权限。
- 已安装 **lvms2** 软件包。

#### 流程

1. 确定您要用作物理卷的存储设备。要列出所有可用存储设备，请使用：

```
$ lsblk
```

2. 创建 LVM 物理卷：

```
# pvcreate /dev/sdb
```

使用您要初始化的设备名称替换 `/dev/sdb`。

#### 验证步骤

- 显示创建的物理卷：

```
# pvs

PV          VG  Fmt Attr PSize PFree
/dev/sdb    lvm2 a-- 28.87g 13.87g
```

#### 其他资源

- **pvcreate (8)**、**pvdplay (8)**、**pvs (8)**、**pvscan (8)** 和 **lvm (8)** man page

### 2.2. 删除 LVM 物理卷

您可以使用 **pvremove** 命令删除物理卷以供 LVM 使用。

### 先决条件

- 管理访问权限。

### 流程

1. 列出物理卷以识别您要删除的设备：

```
# pvs

PV          VG Fmt Attr PSize PFree
/dev/sdb1   lvm2 --- 28.87g 28.87g
```

2. 删除物理卷：

```
# pvremove /dev/sdb1
```

将 `/dev/sdb1` 替换为与物理卷关联的设备名称。

### 注意

如果您的物理卷是卷组的一部分，则需要首先将其从卷组中删除。

- 如果卷组包含多个物理卷，请使用 **vgreduce** 命令：

```
# vgreduce VolumeGroupName /dev/sdb1
```

将 `VolumeGroupName` 替换为卷组的名称。将 `/dev/sdb1` 替换为设备名称。

- 如果您的卷组只包含一个物理卷，请使用 **vgremove** 命令：

```
# vgremove VolumeGroupName
```

将 `VolumeGroupName` 替换为卷组的名称。

### 验证

- 验证物理卷是否已删除：

```
# pvs
```

### 其他资源

- 系统中 **pvremove (8)** 手册页

## 2.3. 在 WEB 控制台中创建逻辑卷

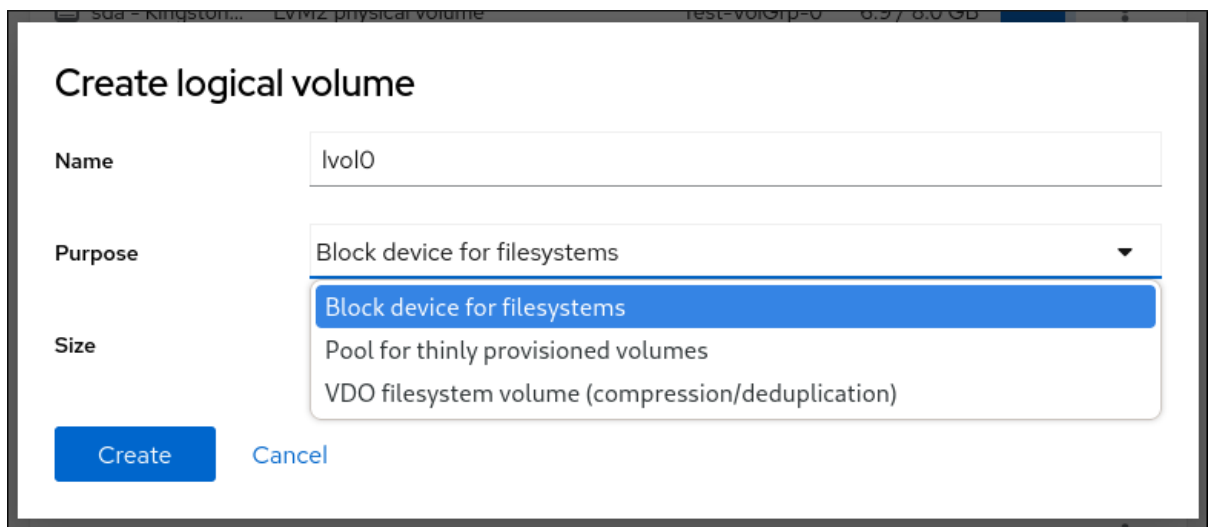
逻辑卷作为物理驱动器使用。您可以使用 RHEL 8 web 控制台在卷组中创建 LVM 逻辑卷。

### 先决条件

- 已安装 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- 已创建卷组。

## 流程

1. 登录到 RHEL 8 web 控制台。  
详情请参阅 [Web 控制台的日志记录](#)。
2. 点 **Storage**。
3. 在 **Storage** 表中，点您要在其中创建逻辑卷的卷组。
4. 在 **Logical volume group** 页面中，滚动到 **LVM2 logical volumes** 部分，然后单击 **Create new logical volume**。
5. 在 **Name** 字段中输入新逻辑卷的名称。不要在名称中包含空格。
6. 在 **Purpose** 下拉菜单中，选择 **Block device for filesystems**。  
此配置允许您创建一个逻辑卷，其最大卷大小等于卷组中所含所有驱动器的总和。



**Create logical volume**

Name:

Purpose: Block device for filesystems ▼

Size: Block device for filesystems  
Pool for thinly provisioned volumes  
VDO filesystem volume (compression/deduplication)

7. 定义逻辑卷的大小。考虑：
  - 使用这个逻辑卷的系统所需的空間。
  - 您要创建的逻辑卷数量。

您可以选择不使用整个空间。如果需要，您可以稍后增大逻辑卷。

### Create logical volume

Name

rhel-logical-volume

Purpose

Block device for filesystems

Size

16.0

GB

Create

Cancel

8. 点 **Create**。
- 逻辑卷被创建。要使用逻辑卷，您必须格式化并挂载卷。

验证

- 在 **Logical volume** 页面中，滚动到 **LVM2 logical volumes** 部分，并验证是否列出了新逻辑卷。

### LVM2 volume group

Add physical volume

Name

Test-VolGrp-0

edit

UUID

pYf9eO-7nwg-ms96-LbmM-AYBf-puBq-jpjetg

Capacity

8.01 GB, 7.46 GiB, 8011120640 bytes

Physical volumes

sda

Kingston DT 101 II (001372997BD5F941C63402DA)

3.7 / 8.0

### LVM2 logical volumes

Create new

| ID         | Type             | Location |  |
|------------|------------------|----------|--|
| Test-Vol-0 | Unformatted data | 3.70 GB  |  |

Unformatted data

Format

LVM2 logical volume

Shrink

Grow

Deactivate

Delete

2.4. 在 WEB 控制台中格式化逻辑卷

逻辑卷作为物理驱动器使用。要使用它们，您必须使用文件系统格式化它们。

**警告**

格式化逻辑卷会删除卷上的所有数据。

12



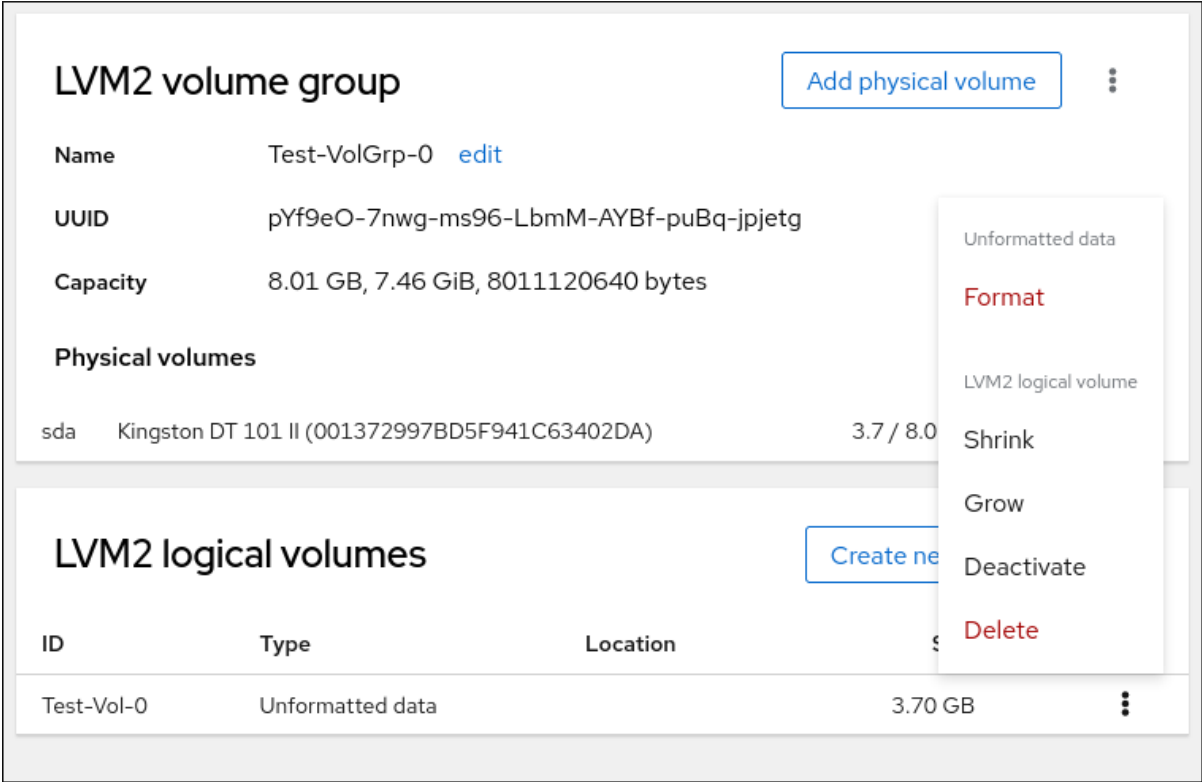
您选择的文件系统决定了可用于逻辑卷的配置参数。例如，XFS 文件系统不支持缩小卷。

先决条件

- 已安装 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- 已创建的逻辑卷。
- 您有对系统的 root 访问权限。


流程

1. 登录到 RHEL 8 web 控制台。  
详情请参阅 [Web 控制台的日志记录](#)。
2. 点 **Storage**。
3. 在 **Storage** 表中，点创建了逻辑卷的卷组。
4. 在 **Logical volume group** 页面中，滚动到 **LVM2 logical volumes** 部分。
5. 点击您要格式的卷组旁的菜单按钮 **⋮**。
6. 从下拉菜单中选择 **Format**。



7. 在 **Name** 字段中输入文件系统的名称。

8. 在 **Mount Point** 字段中添加挂载路径。

 **Format /dev/rhel-volume-group/rhel-logical-volume**

|                      |  |
|----------------------|--|
| <b>Name</b>          | <input type="text" value="rhel-fs"/>   |
| <b>Mount point</b>   | <input type="text" value="/media"/>  |
| <b>Type</b>          | XFS (recommended) ▼  |
| <b>Overwrite</b>     | <input type="checkbox"/> Overwrite existing data with zeros (slower)   |
| <b>Encryption</b>    | No encryption ▼  |
| <b>At boot</b>       | Mount without waiting, ignore failure ▼  |
|                      | <input checked="" type="checkbox"/> Mounts in parallel with services<br><input checked="" type="checkbox"/> Boot still succeeds when filesystem does not mount |
| <b>Mount options</b> | <input type="checkbox"/> Mount read only<br><input type="checkbox"/> Custom mount options  |

Formatting erases all data on a storage device.

9. 在 **Type** 下拉菜单中选择一个文件系统：

- **XFS** 文件系统支持大的逻辑卷，在不停止工作的情况下在线切换物理驱动器，并可以增大现有的文件系统。如果您没有不同的首选项，请保留这个文件系统。  
XFS 不支持缩小使用 XFS 文件系统格式的卷大小

- **ext4** 文件系统支持：

- 逻辑卷
- 在不中断的情况下在线切换物理驱动器
- 增大文件系统
- 缩小文件系统

- 如果您希望 RHEL web 控制台使用零重写整个磁盘，请选择 **Overwrite existing data with zeros** 复选框。使用这个选项较慢，因为程序必须经过整个磁盘，但它更为安全。如果磁盘包含任何数据且需要覆盖数据，则使用这个选项。

如果您没有选择 **Overwrite existing data with zeros** 复选框，RHEL web 控制台只重写磁盘头。这提高了格式化速度。

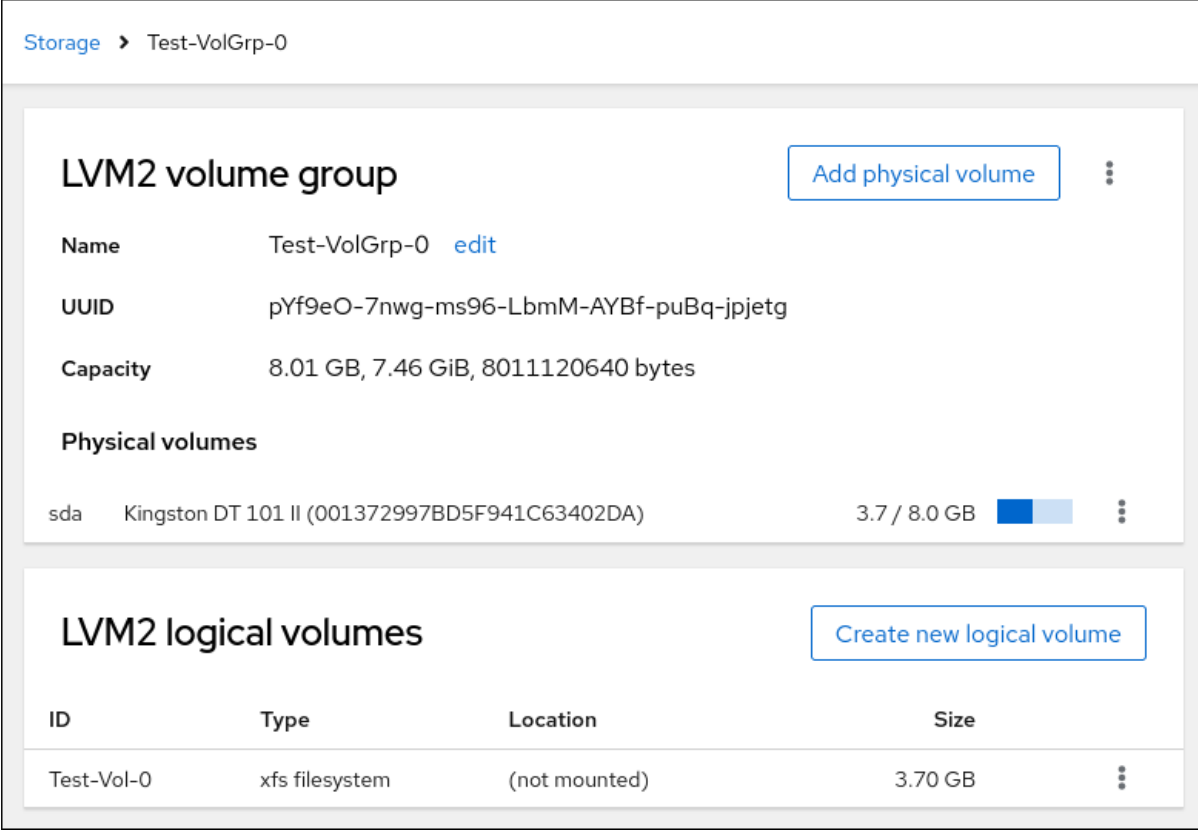
- 如果要在逻辑卷上启用它，请在 **Encryption** 下拉菜单中选择加密的类型。  
您可以选择具有 LUKS1 (Linux Unified Key Setup) 或 LUKS2 加密的版本，其允许您使用密码短语加密卷。
- 在 **At boot** 下拉菜单中，选择您希望逻辑卷在系统引导后何时挂载。
- 选择所需的 **挂载选项**。

## 14. 格式化逻辑卷：

- 如果要格式化卷并立即挂载它，请单击 **Format and mount**。
- 如果要格式化卷，而不挂载它，请单击 **Format only**。  
根据卷大小以及选择格式化选项，格式化可能需要几分钟。

## 验证

1. 在 **Logical volume group** 页面中，滚动到 **LVM2 logical volumes** 部分，然后点逻辑卷，来检查详情和其它选项。



Storage > Test-VolGrp-0

### LVM2 volume group

[Add physical volume](#)

**Name** Test-VolGrp-0 [edit](#)

**UUID** pYf9eO-7nwg-ms96-LbmM-AYBf-puBq-jpjetg

**Capacity** 8.01 GB, 7.46 GiB, 8011120640 bytes

**Physical volumes**

|     |   |              |  |
|-----|---|--------------|--|
| sda | Kingston DT 101 II (001372997BD5F941C63402DA) | 3.7 / 8.0 GB |  |
|-----|---|--------------|--|

### LVM2 logical volumes

[Create new logical volume](#)

| ID         | Type           | Location      | Size    |
|------------|----------------|---------------|---------|
| Test-Vol-0 | xfs filesystem | (not mounted) | 3.70 GB |

2. 如果您选择了 **Format only** 选项，点逻辑卷行末尾的菜单按钮，然后选择 **Mount** 来使用逻辑卷。

## 2.5. 在 WEB 控制台中重新定义逻辑卷大小

您可以在 RHEL 8 web 控制台中扩展或减少逻辑卷。示例流程演示了如何在不使卷离线的情况下增大和缩小逻辑卷的大小。

**警告**


您不能减少包含 GFS2 或者 XFS 文件系统的卷。

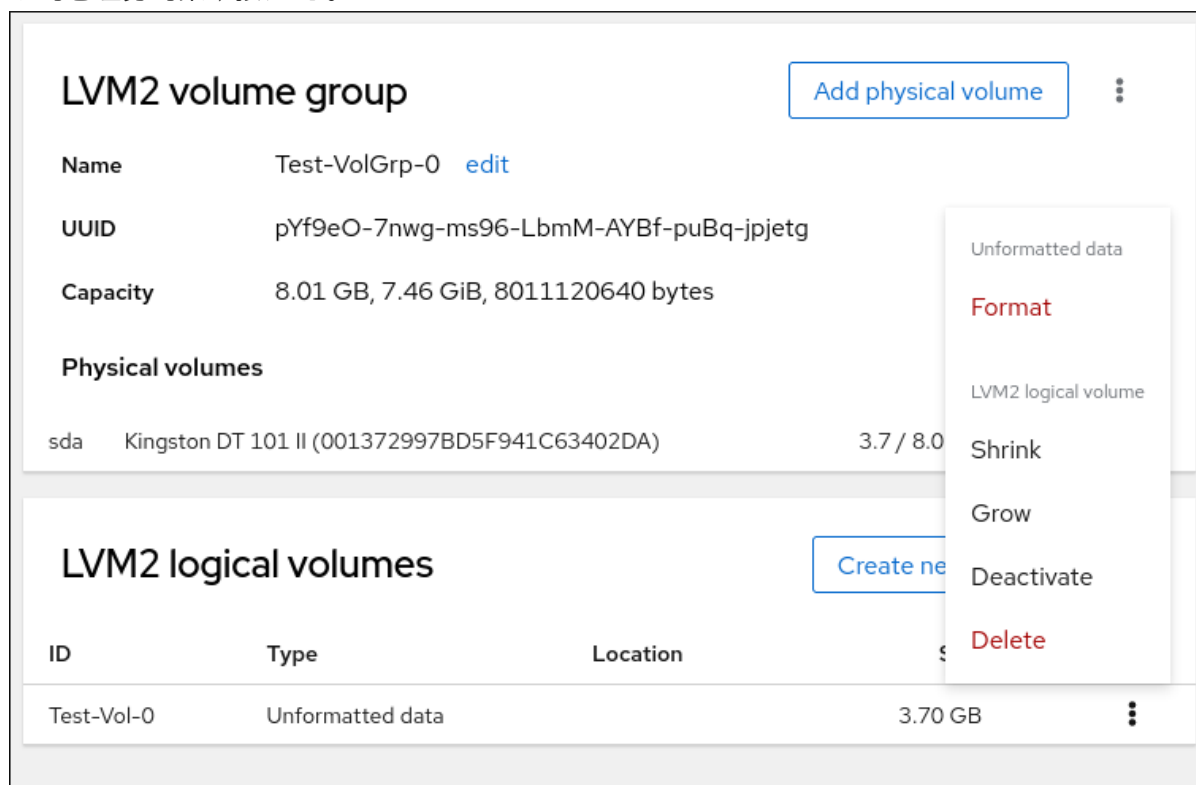
## 先决条件

- 已安装 RHEL 8 web 控制台。

- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- 现有逻辑卷包含支持重新定义逻辑卷大小的文件系统。

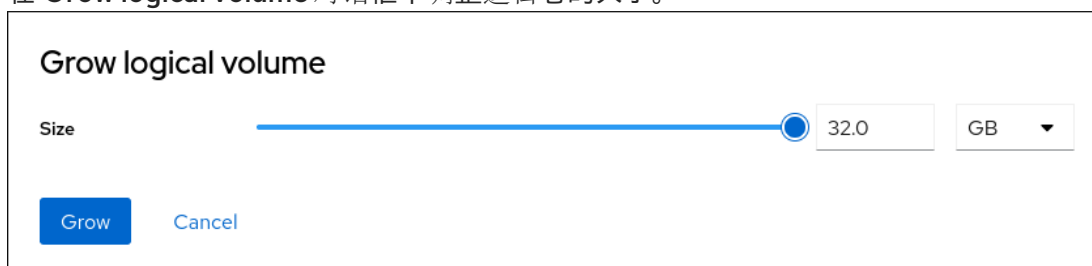
## 流程

1. 登录到 RHEL web 控制台。
2. 点 **Storage**。
3. 在 **Storage** 表中，点创建了逻辑卷的卷组。
4. 在 **Logical Volume group** 页面中，滚动到 **LVM2 logical volumes** 部分，然后点击您要调整其大小的卷组旁的菜单按钮 。

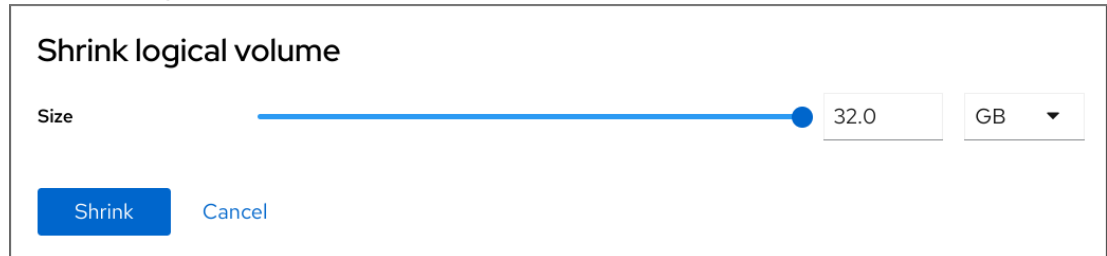


5. 在菜单中，选择 **Grow** 或 **Shrink** 来调整卷的大小：

- 增大卷：
  - a. 选择 **Grow** 来增加卷的大小。
  - b. 在 **Grow logical volume** 对话框中调整逻辑卷的大小。



- c. 点 **Grow**。  
LVM 增大逻辑卷，而不会导致系统中断。
- 缩小卷：
  - a. 选择 **Shrink** 以减少卷的大小。
  - b. 在 **Shrink logical volume** 对话框中调整逻辑卷的大小。



- c. 点 **Shrink**。  
LVM 缩小逻辑卷，而不会导致系统中断。

## 2.6. 其他资源

- **pvcreate (8)** 手册页。
- [使用 parted 在磁盘创建分区表](#)。
- 您系统上的 **parted (8)** 手册页

## 第 3 章 管理 LVM 卷组

您可以创建并使用卷组(VG)来管理并调整多个物理卷(PV)到单个存储实体。

扩展是您可以在 LVM 中分配的最小空间单位。物理扩展(PE)和逻辑扩展(LE)的默认大小为 4 MiB，您可以配置。所有扩展的大小都相同。

当您在 VG 中创建逻辑卷(LV)时，LVM 会在 PV 上分配物理扩展。LV 中的逻辑扩展与 VG 中的物理扩展对应一对一。您不需要指定创建 LV 的 PE。LVM 将找到可用的 PE，并将它们划分为创建请求大小的 LV。

在 VG 中，您可以创建多个 LV，每个 LV 像一个传统分区一样，但可以跨物理卷跨越物理卷并动态调整大小。vgs 可以自动管理磁盘空间的分配。

### 3.1. 创建 LVM 卷组

您可以使用 **vgcreate** 命令创建卷组(VG)。您可以调整非常大或非常小的卷的扩展大小，以优化性能和存储效率。您可以在创建 VG 时指定扩展大小。要更改扩展大小，您必须重新创建卷组。

#### 先决条件

- 管理访问权限。
- 已安装 **lvm2** 软件包。
- 创建一个或多个物理卷。有关创建物理卷的更多信息，[请参阅创建 LVM 物理卷](#)。

#### 流程

1. 列出并识别您要包含在 VG 中的 PV：

```
# pvs
```

2. 创建 VG：

```
# vgcreate VolumeGroupName PhysicalVolumeName1 PhysicalVolumeName2
```

使用您要创建的卷组名称替换 *VolumeGroupName*。将 *PhysicalVolumeName* 替换为 PV 的名称。

要在创建 VG 时指定扩展大小，请使用 **-s *ExtentSize*** 选项。将 *ExtentSize* 替换为扩展的大小。如果没有提供大小后缀，命令默认为 MB。

#### 验证

- 验证是否已创建 VG：

```
# vgs

VG                #PV #LV #SN Attr   VSize VFree
VolumeGroupName  1  0  0 wz--n- 28.87g 28.87g
```

#### 其他资源

- 您的系统上的 **vgcreate (8)**, **vgs (8)**, 和 **pvs (8)** man page

### 3.2. 在 WEB 控制台中创建卷组

从一个或多个物理驱动器或其它存储设备创建卷组。

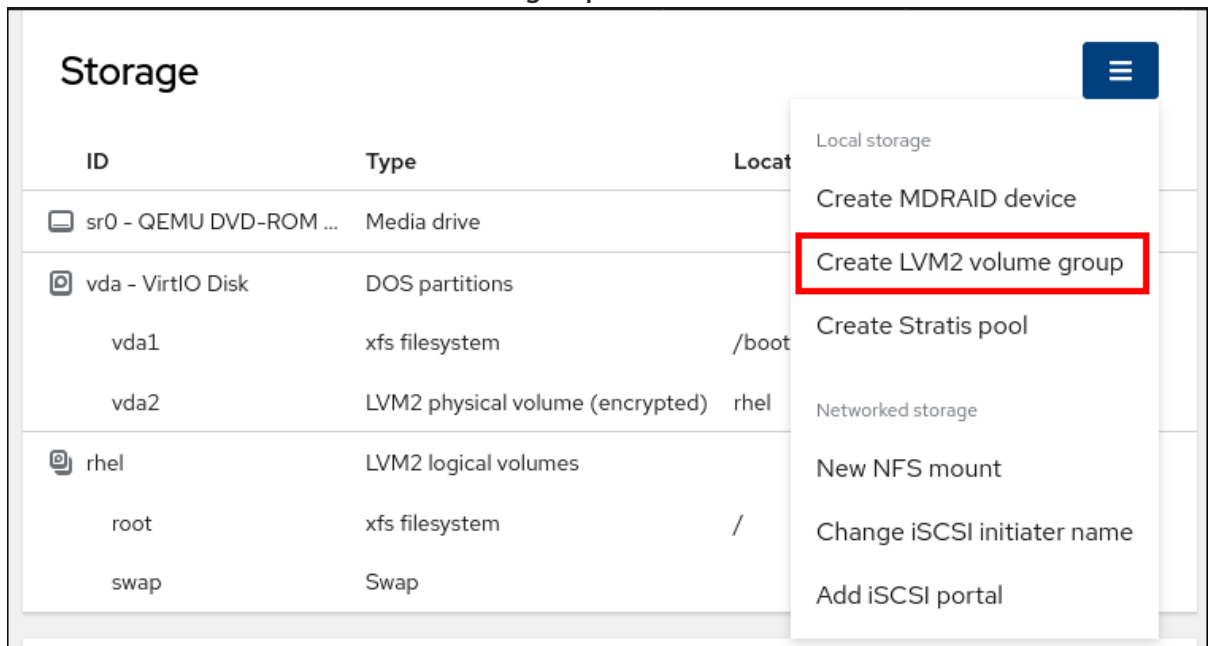
从卷组创建逻辑卷。每个卷组都可以包括多个逻辑卷。

#### 先决条件

- 已安装 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- 要创建卷组的物理驱动器或其他类型的存储设备。

#### 流程

1. 登录到 RHEL 8 web 控制台。  
详情请参阅 [Web 控制台的日志记录](#)。
2. 点 **Storage**。
3. 在 **Storage** 表中，点菜单按钮。
4. 从下拉菜单中选择 **Create LVM2 volume group**。



5. 在 **Name** 字段中输入卷组的名称。名称不得包含空格。
6. 选择您要组合的驱动器来创建卷组。

## Create volume group

Name

Disks

|                                     |                                 |                     |
|-------------------------------------|---------------------------------|---------------------|
| <input checked="" type="checkbox"/> | 16.0 GB RAID device raid-device | /dev/md/raid-device |
| <input checked="" type="checkbox"/> | 16.0 GB VirtIO Disk             | /dev/vdb            |

Create

Cancel

RHEL web 控制台仅显示未使用的块设备。如果您没有在列表中看到设备，请确保它没有被系统使用，或者将其格式化为空且未使用。使用的设备包括，例如：

- 使用文件系统格式化的设备
- 另一个卷组中的物理卷
- 物理卷是另一个软件 RAID 设备的成员

7. 点 **Create**。  
已创建卷组。

### 验证

- 在 **Storage** 页面中，检查新卷组是否列在 **Storage** 表中。

## 3.3. 重命名 LVM 卷组

您可以使用 **vgrename** 命令重命名卷组(VG)。

### 先决条件

- 管理访问权限。
- 已安装 **lvm2** 软件包。
- 创建一个或多个物理卷。有关创建物理卷的更多信息，请参阅[创建 LVM 物理卷](#)。
- 已创建卷组。有关创建卷组的详情，请参考 [第 3.1 节“创建 LVM 卷组”](#)。

### 流程

1. 列出并识别您要重命名的 VG：

```
# vgs
```

2. 重命名 VG：

```
# vgrename OldVolumeGroupName NewVolumeGroupName
```



将 *OldVolumeGroupName* 替换为 VG 的名称。将 *NewVolumeGroupName* 替换为 VG 的新名称。

## 验证

- 验证 VG 是否具有新名称：

```
# vgs

VG                #PV #LV #SN Attr   VSize VFree
NewVolumeGroupName 1   0   0 wz--n- 28.87g 28.87g
```

## 其他资源

- vgrename (8), vgs (8)** man pages

## 3.4. 扩展 LVM 卷组

您可以使用 **vgextend** 命令将物理卷(PV)添加到卷组(VG)中。

## 先决条件

- 管理访问权限。
- 已安装 **lvm2** 软件包。
- 创建一个或多个物理卷。有关创建物理卷的更多信息，[请参阅创建 LVM 物理卷](#)。
- 已创建卷组。有关创建卷组的详情，请参考 [第 3.1 节“创建 LVM 卷组”](#)。

## 流程

- 列出并识别您要扩展的 VG：

```
# vgs
```

- 列出并识别您要添加到 VG 中的 PV：

```
# pvs
```

- 扩展 VG：

```
# vgextend VolumeGroupName PhysicalVolumeName
```

将 *VolumeGroupName* 替换为 VG 的名称。将 *PhysicalVolumeName* 替换为 PV 的名称。

## 验证

- 验证 VG 现在是否包含新 PV：

```
# pvs

PV      VG      Fmt Attr PSize PFree
```

```
/dev/sda VolumeGroupName lvm2 a-- 28.87g 28.87g
/dev/sdd VolumeGroupName lvm2 a-- 1.88g 1.88g
```

## 其他资源

- **vgextend (8), vgs (8), pvs (8)** man pages

## 3.5. 合并 LVM 卷组

您可以将两个现有的卷组(VG)与 **vgmerge** 命令合并。源卷将合并到目标卷中。

### 先决条件

- 管理访问权限。
- 已安装 **lvm2** 软件包。
- 创建一个或多个物理卷。有关创建物理卷的更多信息，[请参阅创建 LVM 物理卷](#)。
- 创建两个或多个卷组。有关创建卷组的详情，请参考 [第 3.1 节“创建 LVM 卷组”](#)。

### 流程

1. 列出并识别您要合并的 VG：

```
# vgs

VG          #PV #LV #SN Attr   VSize VFree
VolumeGroupName1  1  0  0 wz--n- 28.87g 28.87g
VolumeGroupName2  1  0  0 wz--n- 1.88g 1.88g
```

2. 将源 VG 合并到目标 VG 中：

```
# vgmerge VolumeGroupName2 VolumeGroupName1
```

将 *VolumeGroupName2* 替换为源 VG 的名称。将 *VolumeGroupName1* 替换为目标 VG 的名称。

### 验证

- 验证 VG 现在是否包含新 PV：

```
# vgs

VG          #PV #LV #SN Attr   VSize VFree
VolumeGroupName1  2  0  0 wz--n- 30.75g 30.75g
```

## 其他资源

- 系统中 **vgmerge (8)** 手册页

## 3.6. 从卷组中删除物理卷

要从卷组(VG)中删除未使用的物理卷(PV)，请使用 **vgreduce** 命令。**vgreduce** 命令通过删除一个或多个空物理卷来缩小卷组的容量。这样就可以使不同的卷组自由使用那些物理卷，或者将其从系统中删除。

## 流程

1. 如果物理卷仍在使用，则将数据从同一卷组中迁移到另一个物理卷中：

```
# pvmove /dev/vdb3
/dev/vdb3: Moved: 2.0%
...
/dev/vdb3: Moved: 79.2%
...
/dev/vdb3: Moved: 100.0%
```

2. 如果现有卷组中的其他物理卷上没有足够的空闲扩展：

- a. 从 `/dev/vdb4` 创建一个新物理卷：

```
# pvcreate /dev/vdb4
Physical volume "/dev/vdb4" successfully created
```

- b. 将新创建的物理卷添加到卷组中：

```
# vgextend VolumeGroupName /dev/vdb4
Volume group "VolumeGroupName" successfully extended
```

- c. 将数据从 `/dev/vdb3` 移到 `/dev/vdb4` 中：

```
# pvmove /dev/vdb3 /dev/vdb4
/dev/vdb3: Moved: 33.33%
/dev/vdb3: Moved: 100.00%
```

3. 从卷组中删除物理卷 `/dev/vdb3`:

```
# vgreduce VolumeGroupName /dev/vdb3
Removed "/dev/vdb3" from volume group "VolumeGroupName"
```

## 验证

- 验证 `/dev/vdb3` 物理卷是否已从 `VolumeGroupName` 卷组中删除：

```
# pvs
PV          VG          Fmt Attr PSize  PFree  Used
/dev/vdb1 VolumeGroupName lvm2 a-- 1020.00m 0      1020.00m
/dev/vdb2 VolumeGroupName lvm2 a-- 1020.00m 0      1020.00m
/dev/vdb3          lvm2 a-- 1020.00m 1008.00m 12.00m
```

## 其他资源

- 在您的系统中 **vgreduce (8)**, **pvmove (8)**, 和 **pvs (8)** man page

## 3.7. 分割 LVM 卷组

如果在物理卷中有足够的空闲空间，就可在不添加新磁盘的情况下创建新的卷组。

在初始设置中，卷组 `VolumeGroupName1` 由 `/dev/vdb1`、`/dev/vdb2` 和 `/dev/vdb3` 组成。完成此步骤后，卷组 `VolumeGroupName1` 将包含 `/dev/vdb1` 和 `/dev/vdb2`，第二个卷组 `VolumeGroupName2` 将包含 `/dev/vdb3`。

### 先决条件

- 卷组中有足够的空间。使用 **vgscan** 命令确定卷组中当前有多少可用空间。
- 根据现有物理卷中的可用容量，使用 **pvmove** 命令将所有使用的物理区块移动到其他物理卷。如需更多信息，请参阅[从卷组中删除物理卷](#)。

### 流程

1. 将现有卷组 `VolumeGroupName1` 分成新卷组 `VolumeGroupName2`：

```
# vgsplit VolumeGroupName1 VolumeGroupName2 /dev/vdb3
Volume group "VolumeGroupName2" successfully split from "VolumeGroupName1"
```



#### 注意

如果您使用现有卷组创建了逻辑卷，请使用以下命令取消激活逻辑卷：

```
# lvchange -a n /dev/VolumeGroupName1/LogicalVolumeName
```

2. 查看两个卷组的属性：

```
# vgs
VG                #PV #LV #SN Attr   VSize  VFree
VolumeGroupName1  2  1  0 wz--n- 34.30G 10.80G
VolumeGroupName2  1  0  0 wz--n- 17.15G 17.15G
```

### 验证

- 验证新创建的卷组 `VolumeGroupName2` 是否由 `/dev/vdb3` 物理卷组成：

```
# pvs
PV          VG                Fmt  Attr  PSize    PFree    Used
/dev/vdb1  VolumeGroupName1  lvm2  a--   1020.00m    0    1020.00m
/dev/vdb2  VolumeGroupName1  lvm2  a--   1020.00m    0    1020.00m
/dev/vdb3  VolumeGroupName2  lvm2  a--   1020.00m 1008.00m  12.00m
```

### 其他资源

- **vgsplit (8)**, **vgs (8)**, 和 **pvs (8)** man page

## 3.8. 将卷组移动到另一个系统中

您可以使用以下命令将整个 LVM 卷组(VG)移到另一个系统中：

### vgexport

在现有系统上使用此命令使系统无法访问不活跃的 VG。一旦 VG 无法访问，您可以分离其物理卷 (PV)。

## vgimport

在其他系统上使用以下命令使在旧系统中不活跃的 VG 在新系统中可访问。

## 先决条件

- 没有用户正在访问您要移动的卷组中活动卷上的文件。

## 流程

1. 卸载 *LogicalVolumeName* 逻辑卷：

```
# umount /dev/mnt/LogicalVolumeName
```

2. 停用激活卷组中的所有逻辑卷，这可防止卷组上的任何进一步活动：

```
# vgchange -an VolumeGroupName
vgchange -- volume group "VolumeGroupName" successfully deactivated
```

3. 导出卷组以防止您要将其从中删除的系统访问：

```
# vgexport VolumeGroupName
vgexport -- volume group "VolumeGroupName" successfully exported
```

4. 查看导出的卷组：

```
# pvscan
PV /dev/sda1   is in exported VG VolumeGroupName [17.15 GB / 7.15 GB free]
PV /dev/sdc1   is in exported VG VolumeGroupName [17.15 GB / 15.15 GB free]
PV /dev/sdd1   is in exported VG VolumeGroupName [17.15 GB / 15.15 GB free]
...
```

5. 关闭您的系统，拔出组成卷组的磁盘，并将其连接到新系统。

6. 将磁盘插入新系统，并导入卷组使其可以被新系统访问：

```
# vgimport VolumeGroupName
```



### 注意

您可以使用 **vgimport** 命令的 **--force** 参数导入缺少物理卷的卷组，然后运行 **vgreduce --removemissing** 命令。

7. 激活卷组：

```
# vgchange -ay VolumeGroupName
```

8. 挂载文件系统使其可使用：

```
# mkdir -p /mnt/VolumeGroupName/users
# mount /dev/VolumeGroupName/users /mnt/VolumeGroupName/users
```

#### 其他资源

- **vgimport (8)**, **vgexport (8)**, 和 **vgchange (8)** man page

### 3.9. 删除 LVM 卷组

您可以使用 **vgremove** 命令删除现有卷组。只能删除不包含逻辑卷的卷组。

#### 先决条件

- 管理访问权限。

#### 流程

1. 确定卷组不包含逻辑卷：

```
# vgs -o vg_name,lv_count VolumeGroupName

VG          #LV
VolumeGroupName 0
```

将 *VolumeGroupName* 替换为卷组的名称。

2. 删除卷组：

```
# vgremove VolumeGroupName
```

将 *VolumeGroupName* 替换为卷组的名称。

#### 其他资源

- **vgs (8)**, **vgremove (8)** 手册页

### 3.10. 在集群环境中删除 LVM 卷组

在集群环境中，LVM 使用 `lockspace <qualifier>` 协调对在多个机器之间共享的卷组的访问。在删除卷组前，您必须停止 **锁定空间**，以确保其他节点在删除过程中没有尝试访问或修改它。

#### 先决条件

- 管理访问权限。
- 卷组没有包含逻辑卷。

#### 流程

1. 确定卷组不包含逻辑卷：

```
# vgs -o vg_name,lv_count VolumeGroupName
```

```
VG          #LV  
VolumeGroupName 0
```

将 *VolumeGroupName* 替换为卷组的名称。

2. 停止所有节点上的 锁定空间，但您要删除卷组的节点除外：

```
# vgchange --lockstop VolumeGroupName
```

将 *VolumeGroupName* 替换为卷组的名称，并等待锁定停止。

3. 删除卷组：

```
# vgremove VolumeGroupName
```

将 *VolumeGroupName* 替换为卷组的名称。

## 其他资源

- **vgremove (8), vgchange (8)** man page

## 第 4 章 基本逻辑卷管理

使用 LVM，您可以执行以下任务：

- 创建新的逻辑卷以扩展您系统的存储功能
- 扩展现有卷和精简池，以适应增长的数据
- 重命名卷以更好地机构
- 缩减卷以释放未使用的空间
- 在不再需要时安全地删除卷
- 激活或停用卷以控制系统对其数据的访问

### 4.1. 逻辑卷功能概述

使用逻辑卷管理器(LVM)，您可以以灵活、高效的方式管理磁盘存储，使传统分区方案无法提供。以下是用于存储管理和优化的关键 LVM 功能概述。

#### 连接

连接涉及将一个或多个物理卷中的空间合并成单逻辑卷，从而有效地合并物理存储。

#### 条带

条带通过将数据分布到多个物理卷来优化数据 I/O 效率。这个方法通过允许并行 I/O 操作来提高顺序读取和写入的性能。

#### RAID

LVM 支持 RAID 0、1、4、5、6 和 10。当您创建 RAID 逻辑卷时，LVM 会创建一个元数据子卷，它是阵列中的每个数据或奇偶校验子卷的大小的一个区块。

#### 精简置备

精简配置允许创建逻辑卷大于可用物理存储。通过精简配置，系统会根据实际使用情况动态分配存储，而不是分配预先确定的前期。

#### 快照

使用 LVM 快照，您可以创建逻辑卷的时间点副本。快照可以为空启动。当原始逻辑卷中所做的更改时，快照会通过写时复制(CoW)捕获预先更改状态，仅随着更改来保留原始逻辑卷的状态。

#### Caching

LVM 支持在较慢的块设备中使用快速块设备（比如 SSD 驱动器）作为写入或者写入缓存。用户可以创建缓存逻辑卷来提高其现有逻辑卷的性能，或者创建由小而快速的设备组成的新缓存逻辑卷，再加上一个大型、较慢的设备。

### 4.2. 创建逻辑卷

LVM 通过将物理层抽象到可根据您的需要创建和调整的逻辑卷，为处理磁盘存储提供了一种灵活的方法。

#### 4.2.1. 创建线性（厚）逻辑卷

使用线性逻辑卷(LV)，您可以将多个物理存储单元合并到一个虚拟存储空间中。您可以轻松地扩展或减少线性 LV，以适应数据要求。



## 先决条件

- 管理访问权限。
- 已安装 **lvm2** 软件包。
- 已创建卷组。如需更多信息，请参阅[创建 LVM 卷组](#)。

## 流程

1. 列出卷组的名称及其大小：

```
# vgs -o vg_name,vg_size

VG          VSize
VolumeGroupName 30.75g
```

2. 创建线性 LV：

```
# lvcreate --name LogicalVolumeName --size VolumeSize VolumeGroupName
```

使用 LV 的名称替换 *LogicalVolumeName*。使用 LV 的大小替换 *VolumeSize*。如果没有大小后缀，命令默认为 MB。将 *VolumeGroupName* 替换为卷组的名称。

## 验证

- 验证是否已创建线性 LV：

```
# lvs -o lv_name,seg_type

LV          Type
LogicalVolumeName  linear
```

## 其他资源

- **vgs (8)**, **lvs (8)**, **lvcreate (8)** man pages

### 4.2.2. 使用 **storage** RHEL 系统角色创建或者调整逻辑卷大小

使用 **storage** 角色执行以下任务：

- 在由多个磁盘组成的卷组中创建 LVM 逻辑卷
- 在 LVM 上调整现有文件系统大小
- 以池总大小的百分比表示 LVM 卷大小

如果卷组不存在，角色会创建它。如果逻辑卷在卷组中存在，如果其大小与 playbook 中指定的内容不匹配，则会调整其大小。

如果您要缩小逻辑卷，以避免数据丢失，则您必须确保该逻辑卷上的文件系统没有使用要缩小的逻辑卷中的空间。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

## 流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Create logical volume
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_pools:
          - name: myvg
            disks:
              - sda
              - sdb
              - sdc
            volumes:
              - name: mylv
                size: 2G
                fs_type: ext4
                mount_point: /mnt/data
```

示例 playbook 中指定的设置包括以下内容：

### size: <size>

您必须使用单位（如 GiB）或百分比（例如 60%）来指定大小。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 验证指定的卷是否已创建或已调整为请求的大小：

```
# ansible managed-node-01.example.com -m command -a 'lvs myvg'
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

### 4.2.3. 创建条带逻辑卷

使用条状逻辑卷(LV)，您可以在多个物理卷(PV)之间分发数据，从而可能会同时使用多个磁盘的读取和写入速度。

在创建条带 LV 时，务必要考虑条带数目和大小。条带数是分布数据的 PV 数量。增加条带数可以通过同时使用多个磁盘来提高性能。条带大小是在移至下一个磁盘前写入条状磁盘的数据块大小，并以 KB 为单位指定。最佳条带大小取决于您的工作负载和文件系统块大小。默认值为 64KB，可以调整。

## 先决条件

- 管理访问权限。

## 流程

1. 列出卷组的名称及其大小：

```
# vgs -o vg_name,vg_size

VG          VSize
VolumeGroupName 30.75g
```

2. 创建条带 LV：

```
# lvcreate --stripes NumberOfStripes --stripesize StripeSize --size LogicalVolumeSize --name LogicalVolumeName VolumeGroupName
```

用条带数替换 *NumberOfStripes*。将 *StripeSize* 替换为条带大小（以 KB 为单位）。**--stripesize** 不是一个必需的选项。如果没有指定条带大小，则默认为 64KB。使用 LV 的名称替换 *LogicalVolumeName*。将 *VolumeGroupName* 替换为卷组的名称。

## 验证

- 验证创建了条状 LV：

```
# lvs -o lv_name,seg_type

LV          Type
LogicalVolumeName striped
```

## 其他资源

- **vgs (8)** **lvs (8)**, **lvcreate (8)** man pages

### 4.2.4. 创建 RAID 逻辑卷

RAID 逻辑卷可让您使用多个磁盘来实现冗余和性能。LVM 支持各种 RAID 级别，包括 RAID0、RAID1、RAID4、RAID5、RAID6 和 RAID10。

通过 LVM，您可以创建条带 RAID (RAID0、RAID4、RAID5、RAID6)、镜像 RAID (RAID1)或两者的组合 (RAID10)。

RAID 4、RAID 5 和 RAID 6 通过存储奇偶校验数据来提供容错功能，这些数据可用于在磁盘失败时重建丢失的信息。

在创建 RAID LV 时，将每个条带放在一个单独的 PV 中。条带数等于卷组(VG)中 PV 数量。

表 4.1. 最小 RAID 配置要求

| RAID 级别 | 类型    | 奇偶校验             | 最小设备数 | 最小条带号 |
|---------|-------|------------------|-------|-------|
| RAID0   | 条带    | None             | 2     | 2     |
| RAID1   | 镜像    | None             | 2     | -     |
| RAID4   | 条带    | 使用第一个设备存储奇偶校验    | 3     | 2     |
| RAID5   | 条带    | 使用额外的设备来存储奇偶校验   | 3     | 2     |
| RAID6   | 条带    | 使用两个额外的设备来存储奇偶校验 | 5     | 3     |
| RAID10  | 条带和镜像 | None             | 4     | 2     |

先决条件

- 管理访问权限.

流程

1. 列出卷组的名称及其大小：

```
# vgs -o vg_name,vg_size
VG          VSize
VolumeGroupName 30.75g
```

2. 创建 RAID LV：

- 要创建条带 raid，请使用：

```
# lvcreate --type raid/level --stripes NumberOfStripes --stripesize StripeSize --size Size --name LogicalVolumeName VolumeGroupName
```

使用 RAID 级别 0、4、5 或 6 替换 *level*。用条带数替换 *NumberOfStripes*。将 *StripeSize* 替换为条带大小（以 KB 为单位）。使用 LV 的大小替换 *Size*。使用 LV 的名称替换 *LogicalVolumeName*。

- 要创建镜像的 RAID，请使用：

```
# lvcreate --type raid1 --mirrors MirrorsNumber --size Size --name LogicalVolumeName
VolumeGroupName
```

将 *MirrorsNumber* 替换为镜像数量。使用 LV 的大小替换 *Size*。使用 LV 的名称替换 *LogicalVolumeName*。

- 要创建镜像和条带 RAID，请使用：

```
# lvcreate --type raid10 --mirrors MirrorsNumber --stripes NumberOfStripes --stripesize
StripeSize --size Size --name LogicalVolumeName VolumeGroupName
```

将 *MirrorsNumber* 替换为镜像数量。用条带数替换 *NumberOfStripes*。将 *StripeSize* 替换为条带大小（以 KB 为单位）。使用 LV 的大小替换 *Size*。使用 LV 的名称替换 *LogicalVolumeName*。

## 验证

- 验证 RAID LV 是否已创建：

```
# lvs -o lv_name,seg_type

LV          Type
LogicalVolumeName  raid0
```

## 其他资源

- **lvmraid(7), vgs(8), lvs(8), lvcreate(8)** man pages

### 4.2.5. 创建精简逻辑卷

在精简配置下，分配卷组(VG)的物理扩展(PE)，以创建具有特定物理大小的精简池。然后，根据虚拟大小从这个精简池中分配逻辑卷(LV)，不受池的物理容量的限制。因此，当所有精简 LV 的总虚拟大小超过精简池的物理容量时，每个精简 LV 的虚拟大小可能会超过精简池的物理容量。因此，务必要密切监控逻辑和物理使用情况，以避免耗尽空间和中断。

精简配置可根据需要分配空间，降低初始成本并提高资源利用率，从而优化存储效率。但是，在使用精简 LV 时，请注意以下缺陷：

- 不当的丢弃处理可能会阻止未使用的存储空间的发行版本，从而导致空间随着时间的推移完全分配。
- 在带有快照的文件系统上，写时复制(CoW)操作可能会较慢。
- 数据块可以在多个文件系统间术语，导致随机访问限制。

## 先决条件

- 管理访问权限。

- 您已创建了物理卷。如需更多信息，请参阅[创建 LVM 物理卷](#)。
- 您已创建了一个卷组。如需更多信息，请参阅[创建 LVM 卷组](#)。
- 您已创建了逻辑卷。如需更多信息，请参阅[创建逻辑卷](#)。

## 流程

1. 列出卷组的名称及其大小：

```
# vgs -o vg_name,vg_size

VG          VSize
VolumeGroupName 30.75g
```

2. 创建精简池：

```
# lvcreate --type thin-pool --size PoolSize --name ThinPoolName VolumeGroupName
```

将 *PoolSize* 替换为精简池可以使用的最大磁盘空间量。将 *ThinPoolName* 替换为精简池的名称。将 *VolumeGroupName* 替换为卷组的名称。

3. 创建精简 LV：

```
# lvcreate --type thin --virtualsize MaxVolumeSize --name ThinVolumeName --thinpool
ThinPoolName VolumeGroupName
```

将 *MaxVolumeSize* 替换为卷可在精简池中增大的最大大小。将 *ThinPoolName* 替换为精简池的名称。将 *VolumeGroupName* 替换为卷组的名称。



### 注意

您可以在同一精简池中创建其他精简 LV。

## 验证

- 验证是否创建了 thin LV：

```
# lvs -o lv_name,seg_type

LV          Type
ThinPoolName thin-pool
ThinVolumeName thin
```

## 其他资源

- [lvs \(8\)](#), [lvcreate \(8\)](#) man pages

## 4.3. 重新调整逻辑卷大小

使用逻辑卷管理器(LVM)，您可以根据需要重新定义逻辑卷(LV)大小，而不影响存储的数据。

### 4.3.1. 扩展线性逻辑卷

您可以使用 **lvextend** 命令扩展线性（厚）LV 及其快照。

### 先决条件

- 管理访问权限。

### 流程

1. 确定您的卷组有足够的空间来扩展 LV：

```
# lvs -o lv_name,lv_size,vg_name,vg_size,vg_free
LV          LSize  VG          VSize VFree
LogicalVolumeName 1.49g VolumeGroupName 30.75g 29.11g
```

2. 扩展线性 LV 并调整文件系统大小：

```
# lvextend --size +AdditionalSize --resizefs VolumeGroupName/LogicalVolumeName
```

使用添加到 LV 的空间替换 *AdditionalSize*。默认测量单位是 MB，但您可以指定其他单元。将 *VolumeGroupName* 替换为卷组的名称。使用精简卷的名称替换 *LogicalVolumeName*。

### 验证

- 验证线性 LV 是否已扩展：

```
# lvs -o lv_name,lv_size
LV          LSize
NewLogicalVolumeName 6.49g
```

## 4.3.2. 扩展精简逻辑卷

您可以使用 **lvextend** 命令扩展精简逻辑卷(LV)。

### 先决条件

- 管理访问权限。

### 流程

1. 确定精简池有足够的空间供您计划添加的数据：

```
# lvs -o lv_name,lv_size,data_percent
LV          LSize  Data%
MyThinPool  20.10g 3.21
ThinVolumeName 1.10g 4.88
```

2. 扩展 thin LV 并调整文件系统大小：

```
# lvextend --size +AdditionalSize --resizefs VolumeGroupName/ThinVolumeName
```

使用添加到 LV 的空间替换 *AdditionalSize*。默认测量单位是 MB，但您可以指定其他单元。将 *VolumeGroupName* 替换为卷组的名称。将 *ThinVolumeName* 替换为精简卷的名称。

## 验证

- 验证 thin LV 是否已扩展：

```
# lvs -o lv_name,lv_size,data_percent

LV          LSize  Data%
MyThinPool   20.10g 3.21
ThinVolumeName 6.10g 0.43
```

### 4.3.3. 扩展精简池

精简逻辑卷的虚拟大小可能会超过精简池的物理容量，从而导致过度配置。要防止空间不足，您必须监控并定期扩展精简池的容量。

**data\_percent** 指标指示精简池当前使用的已分配数据空间的百分比。**metadata\_percent** 指标反映了用于存储元数据的空间百分比，这对在精简池中管理映射非常重要。

监控这些指标对于确保有效的存储管理和避免容量问题至关重要。

LVM 提供了根据需要手动扩展数据或元数据容量的选项。另外，您可以启用监控并自动扩展精简池。

#### 4.3.3.1. 手动扩展精简池

逻辑卷管理器(LVM)提供手动扩展数据段、元数据段或精简池的选项。

##### 4.3.3.1.1. 扩展精简池

您可以使用 **lvextend** 命令扩展精简池。

## 先决条件

- 管理访问权限。

## 流程

1. 显示使用的数据和元数据空间：

```
# lvs -o lv_name,seg_type,data_percent,metadata_percent

LV          Type    Data% Meta%
ThinPoolName thin-pool 97.66 26.86
ThinVolumeName thin    48.80
```

2. 扩展精简池：

```
# lvextend -L Size VolumeGroupName/ThinPoolName
```

使用您的精简池的新大小替换 *Size*。将 *VolumeGroupName* 替换为卷组的名称。将 *ThinPoolName* 替换为精简池的名称。

数据大小将扩展。如果需要，元数据大小将扩展。

## 验证



- 验证精简池是否已扩展：

```
# lvs -o lv_name,seg_type,data_percent,metadata_percent

LV          Type   Data% Meta%
ThinPoolName thin-pool 24.41 16.93
ThinVolumeName thin    24.41
```

#### 其他资源

- **lvs (8), lvextend (8)** man page
- **LVs -o help**

#### 4.3.3.1.2. 扩展精简池数据片段

您可以使用 **lvextend** 命令扩展 **data\_percent** 段。

#### 先决条件

- 管理访问权限。

#### 流程

1. 显示 **data\_percent** 片段：

```
# lvs -o lv_name,seg_type,data_percent

LV          Type   Data%
ThinPoolName thin-pool 93.87
```

2. 扩展 **data\_percent** 片段：

```
# lvextend -L Size VolumeGroupName/ThinPoolName_tdata
```

将 *Size* 替换为您的数据片段的大小。将 *VolumeGroupName* 替换为卷组的名称。将 *ThinPoolName* 替换为精简池的名称。

#### 验证

- 验证 **data\_percent** 片段是否已扩展：

```
# lvs -o lv_name,seg_type,data_percent

LV          Type   Data%
ThinPoolName thin-pool 40.23
```

#### 其他资源

- **lvs (8), lvextend (8)** man page
- **LVs -o help**

#### 4.3.3.1.3. 扩展精简池元数据片段

您可以使用 **lvextend** 命令扩展 **metadata\_percent** 段。

##### 先决条件

- 管理访问权限。

##### 流程

1. 显示 **metadata\_percent** 片段：

```
# lvs -o lv_name,seg_type,metadata_percent

LV          Type    Meta%
ThinPoolName thin-pool 75.00
```

2. 扩展 **metadata\_percent** 片段：

```
# lvextend -L Size VolumeGroupName/ThinPoolName_tmeta
```

使用元数据片段的大小替换 *Size*。将 *VolumeGroupName* 替换为卷组的名称。将 *ThinPoolName* 替换为精简池的名称。

##### 验证

- 验证 **metadata\_percent** 片段是否已扩展：

```
# lvs -o lv_name,seg_type,metadata_percent

LV          Type    Meta%
ThinPoolName thin-pool 0.19
```

##### 其他资源

- **lvs (8)**, **lvextend (8)** man page
- **LVs -o help**

#### 4.3.3.2. 自动扩展精简池

您可以通过启用监控和设置 **thin\_pool\_autoextend\_threshold** 和 **thin\_pool\_autoextend\_percent** 配置参数来自动扩展精简池。

##### 先决条件

- 管理访问权限。

##### 流程

1. 检查精简池是否已监控：

```
# lvs -o lv_name,vg_name,seg_monitor
```

| LV           | VG              | Monitor       |
|--------------|-----------------|---------------|
| ThinPoolName | VolumeGroupName | not monitored |

2. 使用 **dmeventd** 守护进程启用精简池监控：

```
# lvchange --monitor y VolumeGroupName/ThinPoolName
```

将 *VolumeGroupName* 替换为卷组的名称。将 *ThinPoolName* 替换为精简池的名称。

3. 以 **root** 用户身份，在您选择的编辑器中打开 **/etc/lvm/lvm.conf** 文件。
4. 取消注释 **thin\_pool\_autoextend\_threshold** 和 **thin\_pool\_autoextend\_percent** 行，并将每个参数设置为所需的值：

```
thin_pool_autoextend_threshold = 70
thin_pool_autoextend_percent = 20
```

**thin\_pool\_autoextend\_threshold** 决定 LVM 开始自动扩展精简池的百分比。例如，将其设置为 70 表示 LVM 会在达到 70% 容量时尝试扩展精简池。

**thin\_pool\_autoextend\_percent** 根据精简池达到阈值时应扩展的百分比来指定。例如，将其设置为 20 表示精简池将增加其当前大小的 20%。

5. 保存更改并退出编辑器。
6. 重启 **lvm2-monitor**：

```
# systemctl restart lvm2-monitor
```

## 其他资源

- **lvs (8), lvchange (8), dmeventd (8)** man pages

### 4.3.4. 缩小逻辑卷

当 LV 的大小缩小时，释放的逻辑扩展将返回到卷组，然后可供其他 LV 使用。



#### 警告

存储在减少的区域中的数据会丢失。在继续操作前，始终备份数据并调整文件系统的大小。

## 先决条件

- 管理访问权限。

## 流程

1. 列出逻辑卷及其卷组：

```
# lvs -o lv_name,vg_name,lv_size

LV          VG          LSize
LogicalVolumeName  VolumeGroupName 6.49g
```

2. 检查挂载逻辑卷的位置：

```
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/LogicalVolumeName

SOURCE                                TARGET
/dev/mapper/VolumeGroupName-NewLogicalVolumeName /MountPoint
```

将 `/dev/VolumeGroupName/LogicalVolumeName` 替换为逻辑卷的路径。

3. 卸载逻辑卷：

```
# umount /MountPoint
```

将 `/MountPoint` 替换为您的逻辑卷的挂载点。

4. 检查并修复所有文件系统错误：

```
# e2fsck -f /dev/VolumeGroupName/LogicalVolumeName
```

5. 重新定义 LV 和文件系统大小：

```
# lvreduce --size TargetSize --resizefs VolumeGroupName/LogicalVolumeName
```

使用 LV 的新大小替换 `TargetSize`。将 `VolumeGroupName/LogicalVolumeName` 替换为逻辑卷的路径。

6. 重新挂载文件系统：

```
# mount -o remount /MountPoint
```

将 `/MountPoint` 替换为您的文件系统的挂载点。

## 验证

1. 验证文件系统的空间使用情况：

```
# df -hT /MountPoint/

Filesystem                                Type  Size  Used Avail Use% Mounted on
/dev/mapper/VolumeGroupName-NewLogicalVolumeName ext4  2.9G  139K  2.7G   1%  /MountPoint
```

将 `/MountPoint` 替换为您的逻辑卷的挂载点。

2. 验证 LV 的大小：

```
# lvs -o lv_name,lv_size
```

```

LV          LSize
NewLogicalVolumeName 4.00g

```

## 4.4. 重命名逻辑卷

您可以使用 **lvrename** 命令重命名现有逻辑卷，包括快照。

### 先决条件

- 管理访问权限。

### 流程

1. 列出逻辑卷及其卷组：

```

# lvs -o lv_name,vg_name

LV          VG
LogicalVolumeName VolumeGroupName

```

2. 重命名逻辑卷：

```

# lvrename VolumeGroupName/LogicalVolumeName
VolumeGroupName/NewLogicalVolumeName

```

将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。使用新逻辑卷名称替换 *NewLogicalVolumeName*。

### 验证

- 验证逻辑卷是否已重命名：

```

# lvs -o lv_name
LV
NewLogicalVolumeName

```

### 其他资源

- 系统中 **lvrename** (8) 手册页

## 4.5. 删除逻辑卷

您可以使用 **lvremove** 命令删除现有逻辑卷，包括快照。

### 先决条件

- 管理访问权限。

### 流程

1. 列出逻辑卷及其路径：

```
# lvs -o lv_name,lv_path
```

```
LV          Path
LogicalVolumeName /dev/VolumeGroupName/LogicalVolumeName
```

2. 检查挂载逻辑卷的位置：

```
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/LogicalVolumeName
```

```
SOURCE          TARGET
/dev/mapper/VolumeGroupName-LogicalVolumeName /MountPoint
```

将 `/dev/VolumeGroupName/LogicalVolumeName` 替换为逻辑卷的路径。

3. 卸载逻辑卷：

```
# umount /MountPoint
```

将 `/MountPoint` 替换为您的逻辑卷的挂载点。

4. 删除逻辑卷：

```
# lvremove VolumeGroupName/LogicalVolumeName
```

将 `VolumeGroupName/LogicalVolumeName` 替换为逻辑卷的路径。

## 其他资源

- **LVs (8), lvremove (8)** man page

## 4.6. 激活逻辑卷

您可以使用 **lvchange** 命令激活逻辑卷。

### 先决条件

- 管理访问权限。

### 流程

1. 列出逻辑卷、卷组及其路径：

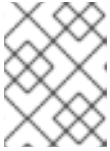
```
# lvs -o lv_name,vg_name,lv_path
```

```
LV          VG          Path
LogicalVolumeName VolumeGroupName VolumeGroupName/LogicalVolumeName
```

2. 激活逻辑卷：

```
# lvchange --activate y VolumeGroupName/LogicalVolumeName
```

将 `VolumeGroupName` 替换为卷组的名称。使用逻辑卷的名称替换 `LogicalVolumeName`。



### 注意

当激活作为另一个 LV 快照创建的精简 LV 时，您可能需要使用 **ignoreactivationskip** 选项来激活它。

### 验证

- 验证 LV 是否活跃：

```
# lvsdisplay VolumeGroupName/LogicalVolumeName

...
LV Status          available
```

将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。

### 其他资源

- **lvs (8)**, **lvchange (8)**, **lvdisplay (8)** man pages

## 4.7. 停用逻辑卷

默认情况下，当您创建逻辑卷时，它处于活动状态。您可以使用 **lvchange** 命令停用逻辑卷。



### 警告

取消激活带有活跃挂载或正在使用的逻辑卷可能会导致数据不一致和系统错误。

### 先决条件

- 管理访问权限。

### 流程

1. 列出逻辑卷、卷组及其路径：

```
# lvs -o lv_name,vg_name,lv_path

LV          VG          Path
LogicalVolumeName VolumeGroupName /dev/VolumeGroupName/LogicalVolumeName
```

2. 检查挂载逻辑卷的位置：

```
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/LogicalVolumeName

SOURCE                                TARGET
/dev/mapper/VolumeGroupName-LogicalVolumeName /MountPoint
```

将 */dev/VolumeGroupName/LogicalVolumeName* 替换为逻辑卷的路径。

### 3. 卸载逻辑卷：

```
# umount /MountPoint
```

将 */MountPoint* 替换为您的逻辑卷的挂载点。

### 4. 取消激活逻辑卷：

```
# lvchange --activate n VolumeGroupName/LogicalVolumeName
```

将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。

## 验证

- 验证 LV 是否未激活：

```
# lvsdisplay VolumeGroupName/LogicalVolumeName
```

```
...  
LV Status          NOT available
```

将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。

## 其他资源

- lvs (8) lvchange (8), lvsdisplay (8)** man pages



## 第 5 章 高级逻辑卷管理

LVM 包括如下高级功能：

- 快照，它们是逻辑卷的时间点副本(LV)
- 缓存，您可以使用更快的存储作为较慢存储的缓存
- 创建自定义精简池

### 5.1. 管理逻辑卷快照

快照是逻辑卷(LV)，可在特定时间点镜像另一个 LV 的内容。

#### 5.1.1. 了解逻辑卷快照

当您创建快照时，您要创建一个作为另一个 LV 的时点副本的新 LV。最初，快照 LV 没有包含实际数据。相反，它会在创建快照时引用原始 LV 的数据块。



#### 警告

定期监控快照的存储使用情况非常重要。如果快照达到其分配的空间 100%，它将无效。

必须在填写快照之前扩展快照。这可以通过使用 `lvextend` 命令手动完成，或者通过 `/etc/lvm/lvm.conf` 文件自动完成。

#### thick LV 快照

当原始 LV 上的数据发生变化时，写时复制(CoW)系统会在进行更改前将原始数据复制到快照。这样，快照的大小仅随着更改而增长，在创建快照时存储原始卷的状态。密集快照是一种 LV，需要您预先分配某些存储空间。以后可以扩展或减少这个数量，但您应该考虑您要原始 LV 进行的更改类型。这有助于您避免通过分配太多空间来避免资源浪费，或者需要频繁增加快照大小（如果分配过小）。

#### 精简 LV 快照

精简快照是从现有精简置备的 LV 创建的 LV 类型。精简快照不需要分配额外的空间。最初，原始 LV 及其快照共享相同的数据块。当更改原始 LV 时，它会将新数据写入不同的块，而快照将继续引用原始块，并在创建快照时保留 LV 数据的点点视图。

精简配置是通过根据需要分配磁盘空间来高效地优化和管理存储。这意味着您可以创建多个 LV，而无需为每个 LV 预先分配大量存储。存储在精简池中的所有 LV 中共享，使其更有效地使用资源。精简池按需为其 LV 分配空间。

#### 在 thick 和 thin LV 快照之间进行选择

密集或精简 LV 快照之间的选择直接由您要执行快照的 LV 类型决定。如果您的原始 LV 是厚的 LV，则您的快照将是厚的。如果您的原始 LV 是精简的，则您的快照将是精简的。

#### 5.1.2. 管理厚逻辑卷快照

当您创建厚 LV 快照时，务必要考虑存储要求和快照的预期期限。您需要根据原始卷的预期更改来为其分配足够的存储。快照必须在预期生命周期内捕获更改，但它不能超过原始 LV 的大小。如果您期望更改率较低，则较小的快照大小为 10%-15%。对于具有高变化率的 LV，您可能需要分配 30% 或更多。



### 重要

必须在填写快照之前扩展快照。如果快照达到其分配的空间 100%，则会无效。您可以使用 **lvs -o lv\_name,data\_percent,origin** 命令监控快照容量。

#### 5.1.2.1. 创建厚逻辑卷快照

您可以使用 **lvcreate** 命令创建厚 LV 快照。

#### 先决条件

- 管理访问权限。
- 您已创建了物理卷。如需更多信息，请参阅[创建 LVM 物理卷](#)。
- 您已创建了一个卷组。如需更多信息，请参阅[创建 LVM 卷组](#)。
- 您已创建了逻辑卷。如需更多信息，请参阅[创建逻辑卷](#)。

#### 流程

1. 确定您要创建快照的 LV：

```
# lvs -o vg_name,lv_name,lv_size

VG          LV          LSize
VolumeGroupName LogicalVolumeName 10.00g
```

快照的大小不能超过 LV 的大小。

2. 创建厚 LV 快照：

```
# lvcreate --snapshot --size SnapshotSize --name SnapshotName
VolumeGroupName/LogicalVolumeName
```

使用您要为快照分配的大小（如 10G）替换 *SnapshotSize*。使用您要提供给快照逻辑卷的名称替换 *SnapshotName*。使用包含原始逻辑卷的卷组名称替换 *VolumeGroupName*。使用您要为其创建快照的逻辑卷的名称替换 *LogicalVolumeName*。

#### 验证

- 验证快照是否已创建：

```
# lvs -o lv_name,origin

LV          Origin
LogicalVolumeName
SnapshotName LogicalVolumeName
```

#### 其他资源

- **lvcreate (8)** 和 **lvs (8)** 手册页

#### 5.1.2.2. 手动扩展逻辑卷快照

如果快照达到其分配的空间 100%，则会无效。必须在填写快照之前扩展快照。这可以通过使用 **lvextend** 命令手动完成此操作。

##### 先决条件

- 管理访问权限。

##### 流程

1. 列出卷组、逻辑卷、快照的源卷、使用百分比和大小：

```
# lvs -o vg_name,lv_name,origin,data_percent,lv_size
VG          LV          Origin      Data%  LSize
VolumeGroupName LogicalVolumeName          10.00g
VolumeGroupName SnapshotName    LogicalVolumeName 82.00  5.00g
```

2. 扩展 thick-provisioned 快照：

```
# lvextend --size +AdditionalSize VolumeGroupName/SnapshotName
```

使用添加到快照中的空间（如 +1G）替换 *AdditionalSize*。将 *VolumeGroupName* 替换为卷组的名称。使用快照的名称替换 *SnapshotName*。

##### 验证

- 验证 LV 是否已扩展：

```
# lvs -o vg_name,lv_name,origin,data_percent,lv_size
VG          LV          Origin      Data%  LSize
VolumeGroupName LogicalVolumeName          10.00g
VolumeGroupName SnapshotName    LogicalVolumeName 68.33  6.00g
```

#### 5.1.2.3. 自动扩展厚逻辑卷快照

如果快照达到其分配的空间 100%，则会无效。必须在填写快照之前扩展快照。这可以自动完成。

##### 先决条件

- 管理访问权限。

##### 流程

1. 以 **root** 用户身份，在您选择的编辑器中打开 **/etc/lvm/lvm.conf** 文件。
2. 取消注释 **snapshot\_autoextend\_threshold** 和 **snapshot\_autoextend\_percent** 行，并将每个参数设置为所需的值：

```
snapshot_autoextend_threshold = 70
snapshot_autoextend_percent = 20
```

**snapshot\_autoextend\_threshold** 决定 LVM 开始自动扩展快照的百分比。例如，将参数设置为 70 表示 LVM 会在达到 70% 容量时尝试扩展快照。

**snapshot\_autoextend\_percent** 指定快照在达到阈值时应扩展的百分比。例如，将参数设置为 20 表示快照将增加其当前大小的 20%。

3. 保存更改并退出编辑器。
4. 重启 **lvm2-monitor** :

```
# systemctl restart lvm2-monitor
```

#### 5.1.2.4. 合并厚逻辑卷快照

您可以将厚 LV 快照合并到创建快照的原始逻辑卷中。合并过程意味着原始 LV 恢复为创建快照时所处的状态。合并完成后，会删除快照。



#### 注意

如果任一活动，原始 LV 和快照 LV 之间的合并就会发布。只有在 LV 被重新激活且没有使用后，才会继续。

#### 先决条件

- 管理访问权限。

#### 流程

1. 列出 LV、其卷组及其路径：

```
# lvs -o lv_name,vg_name,lv_path
```

| LV                | VG              | Path                                   |
|-------------------|-----------------|--|
| LogicalVolumeName | VolumeGroupName | /dev/VolumeGroupName/LogicalVolumeName |
| SnapshotName      | VolumeGroupName | /dev/VolumeGroupName/SnapshotName      |

2. 检查挂载 LV 的位置：

```
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/LogicalVolumeName
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/SnapshotName
```

将 `/dev/VolumeGroupName/LogicalVolumeName` 替换为逻辑卷的路径。将 `/dev/VolumeGroupName/SnapshotName` 替换为快照的路径。

3. 卸载 LV：

```
# umount /LogicalVolume/MountPoint
# umount /Snapshot/MountPoint
```

将 `/LogicalVolume/MountPoint` 替换为逻辑卷的挂载点。将 `/Snapshot/MountPoint` 替换为您的快照的挂载点。

#### 4. 取消激活 LV：

```
# lvchange --activate n VolumeGroupName/LogicalVolumeName
# lvchange --activate n VolumeGroupName/SnapshotName
```

将 `VolumeGroupName` 替换为卷组的名称。使用逻辑卷的名称替换 `LogicalVolumeName`。将 `SnapshotName` 替换为您的快照的名称。

#### 5. 将 thick LV 快照合并到原始卷中：

```
# lvconvert --merge SnapshotName
```

使用快照的名称替换 `SnapshotName`。

#### 6. 激活 LV：

```
# lvchange --activate y VolumeGroupName/LogicalVolumeName
```

将 `VolumeGroupName` 替换为卷组的名称。使用逻辑卷的名称替换 `LogicalVolumeName`。

#### 7. 挂载 LV：

```
# umount /LogicalVolume/MountPoint
```

将 `/LogicalVolume/MountPoint` 替换为逻辑卷的挂载点。

### 验证

- 验证快照是否已删除：

```
# lvs -o lv_name
```

### 其他资源

- [lvconvert \(8\), lvs \(8\) 手册页](#)

## 5.1.3. 管理精简逻辑卷快照

当存储效率是优先级时，精简置备适合。存储空间动态分配减少了初始存储成本，并最大程度提高可用存储资源的使用。在具有动态工作负载或存储随时间增长的环境中，精简配置允许灵活性。它使存储系统能够适应变化的需求，而无需大量预先分配存储空间。使用动态分配时，有可能进行过度置备，其中所有 LV 的总大小可能会超过精简池的物理大小，假设不会同时使用所有空间。

### 5.1.3.1. 创建精简逻辑卷快照

您可以使用 **lvcreate** 命令创建精简 LV 快照。在创建精简 LV 快照时，请避免指定快照大小。包含 `size` 参数会导致创建 thick 快照。

### 先决条件

- 管理访问权限。
- 您已创建了物理卷。如需更多信息，请参阅[创建 LVM 物理卷](#)。
- 您已创建了一个卷组。如需更多信息，请参阅[创建 LVM 卷组](#)。
- 您已创建了逻辑卷。如需更多信息，请参阅[创建逻辑卷](#)。

## 流程

1. 确定您要创建快照的 LV：

```
# lvs -o lv_name,vg_name,pool_lv,lv_size

LV          VG          Pool    LSize
PoolName    VolumeGroupName    152.00m
ThinVolumeName  VolumeGroupName PoolName 100.00m
```

2. 创建 thin LV 快照：

```
# lvcreate --snapshot --name SnapshotName VolumeGroupName/ThinVolumeName
```

使用您要提供给快照逻辑卷的名称替换 *SnapshotName*。使用包含原始逻辑卷的卷组名称替换 *VolumeGroupName*。使用您要为其创建快照的精简逻辑卷的名称替换 *ThinVolumeName*。

## 验证

- 验证快照是否已创建：

```
# lvs -o lv_name,origin

LV          Origin
PoolName
SnapshotName  ThinVolumeName
ThinVolumeName
```

## 其他资源

- [lvcreate \(8\)](#) 和 [lvs \(8\)](#) 手册页

### 5.1.3.2. 合并精简逻辑卷快照

您可以将 thin LV 快照合并到创建快照的原始逻辑卷中。合并过程意味着原始 LV 恢复为创建快照时所处的状态。合并完成后，会删除快照。

## 先决条件

- 管理访问权限。

## 流程

1. 列出 LV、其卷组及其路径：

```
# lvs -o lv_name,vg_name,lv_path
```

| LV               | VG              | Path                                  |
|------------------|-----------------|---------------------------------------|
| ThinPoolName     | VolumeGroupName |                                       |
| ThinSnapshotName | VolumeGroupName | /dev/VolumeGroupName/ThinSnapshotName |
| ThinVolumeName   | VolumeGroupName | /dev/VolumeGroupName/ThinVolumeName   |

2. 检查原始 LV 挂载的位置：

```
# findmnt -o SOURCE,TARGET /dev/VolumeGroupName/ThinVolumeName
```

将 *VolumeGroupName/ThinVolumeName* 替换为逻辑卷的路径。

3. 卸载 LV：

```
# umount /ThinLogicalVolume/MountPoint
```

将 */ThinLogicalVolume/MountPoint* 替换为逻辑卷的挂载点。将 */ThinSnapshot/MountPoint* 替换为您的快照的挂载点。

4. 取消激活 LV：

```
# lvchange --activate n VolumeGroupName/ThinLogicalVolumeName
```

将 *VolumeGroupName* 替换为卷组的名称。将 *ThinLogicalVolumeName* 替换为逻辑卷的名称。

5. 将 thin LV 快照合并到源中：

```
# lvconvert --mergethin VolumeGroupName/ThinSnapshotName
```

将 *VolumeGroupName* 替换为卷组的名称。将 *ThinSnapshotName* 替换为快照的名称。

6. 挂载 LV：

```
# umount /ThinLogicalVolume/MountPoint
```

将 */ThinLogicalVolume/MountPoint* 替换为逻辑卷的挂载点。

## 验证

- 验证原始 LV 是否已合并：

```
# lvs -o lv_name
```

## 其他资源

- **lvremove (8), lvs (8)** 手册页

## 5.2. 缓存逻辑卷

您可以使用 **dm-cache** 或 **dm-writecache** 目标缓存逻辑卷。

**DM-cache** 使用更快的存储设备(SSD)作为较慢的存储设备(HDD)的缓存。它缓存读取和写入数据，从而优化频繁使用的数据的访问时间。在混合工作负载环境中，增强读写操作可带来显著的性能改进。

**DM-writecache** 使用更快的存储介质(SSD)来临时保存写入数据来优化写操作，然后再将其提交到主存储设备(HDD)。对于写入性能可能会减慢数据传输过程的写密集型应用程序来说，这非常有用。

### 5.2.1. 使用 dm-cache 缓存逻辑卷

当使用 **dm-cache** 缓存 LV 时，会创建一个缓存池。缓存池是组合了缓存数据的 LV，它存储了实际缓存的内容，以及缓存元数据，跟踪缓存中存储的内容。然后，这个池与特定 LV 关联，以缓存其数据。

**DM-cache** 以两种块类型为目标：经常访问（热）块移动到缓存中，而不太频繁访问（折叠）块保留在较慢的设备上。

#### 先决条件

- 管理访问权限。

#### 流程

1. 显示您要缓存的 LV 及其卷组：

```
# lvs -o lv_name,vg_name
LV          VG
LogicalVolumeName  VolumeGroupName
```

2. 创建缓存池：

```
# lvcreate --type cache-pool --name CachePoolName --size Size VolumeGroupName
/FastDevicePath
```

将 *CachePoolName* 替换为缓存池的名称。使用缓存池的大小替换 *Size*。将 *VolumeGroupName* 替换为卷组的名称。将 */FastDevicePath* 替换为快速设备的路径，如 SSD 或 NVME。

3. 将缓存池附加到 LV：

```
# lvconvert --type cache --cachepool VolumeGroupName/CachePoolName
VolumeGroupName/LogicalVolumeName
```

#### 验证

- 验证 LV 现在是否已缓存：

```
# lvs -o lv_name,pool_lv
LV          Pool
LogicalVolumeName  [CachePoolName_cpools]
```

#### 其他资源

- **lvcreate (8)**, **lvconvert (8)**, **lvs (8)** man pages

### 5.2.2. 使用 dm-writecache 缓存逻辑卷



当使用 **dm-writecache** 缓存 LV 时，会创建逻辑卷和物理存储设备之间的缓存层。**DM-writecache** 通过将写入操作临时存储在更快的存储介质中（如 SSD）来进行操作，然后再将其写回主存储设备，从而优化写入密集型工作负载。

## 先决条件

- 管理访问权限。

## 流程

1. 显示您要缓存的逻辑卷及其卷组：

```
# lvs -o lv_name,vg_name
LV          VG
LogicalVolumeName  VolumeGroupName
```

2. 创建缓存卷：

```
# lvcreate --name CacheVolumeName --size Size VolumeGroupName /FastDevicePath
```

将 *CacheVolumeName* 替换为缓存卷的名称。使用缓存池的大小替换 *Size*。将 *VolumeGroupName* 替换为卷组的名称。将 */FastDevicePath* 替换为快速设备的路径，如 SSD 或 NVME。

3. 将缓存卷附加到 LV：

```
# lvconvert --type writecache --cachevol CacheVolumeName
VolumeGroupName/LogicalVolumeName
```

将 *CacheVolumeName* 替换为缓存卷的名称。将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。

## 验证

- 验证 LV 现在是否已缓存：

```
# lvs -o lv_name,pool_lv
LV          Pool
LogicalVolumeName  [CacheVolumeName_cvol]
```

## 其他资源

- **lvcreate (8)**, **lvconvert (8)**, **lvs (8)** man pages

## 5.2.3. 取消逻辑卷的缓存

使用两种主要方法从 LV 中删除缓存。

### 分割

您可以从 LV 中分离缓存，但保留缓存卷本身。在这种情况下，LV 将不再受益于缓存机制，但缓存卷及其数据将保持不变。虽然缓存卷被保留，但缓存中的数据无法被重复使用，并将在下次在缓存设置中使用时擦除。

## 取消缓存

您可以从 LV 中分离缓存，并完全删除缓存卷。此操作有效销毁缓存，从而释放空间。

### 先决条件

- 管理访问权限。

### 流程

1. 显示缓存的 LV：

```
# lvs -o lv_name,pool_lv,vg_name
```

| LV                | Pool                   | VG              |
|-------------------|------------------------|-----------------|
| LogicalVolumeName | [CacheVolumeName_cvol] | VolumeGroupName |

2. 分离或删除缓存的卷：

- 要分离缓存的卷，请使用：

```
# lvconvert --splitcache VolumeGroupName/LogicalVolumeName
```

- 要分离和删除缓存的卷，请使用：

```
# lvconvert --uncache VolumeGroupName/LogicalVolumeName
```

将 *VolumeGroupName* 替换为卷组的名称。使用逻辑卷的名称替换 *LogicalVolumeName*。

### 验证

- 验证 LV 是否没有缓存：

```
# lvs -o lv_name,pool_lv
```

### 其他资源

- **lvconvert (8), lvs (8)** 手册页

## 5.3. 创建自定义精简池

您可以创建自定义精简池来更好地控制存储。

### 先决条件

- 管理访问权限。

### 流程

1. 显示可用卷组：

```
# vgs -o vg_name
```

```
VG
VolumeGroupName
```

2. 列出可用的设备：

```
# lsblk
```

3. 创建 LV 来保存精简池数据：

```
# lvcreate --name ThinPoolDataName --size Size VolumeGroupName /DevicePath
```

将 *ThinPoolDataName* 替换为您的精简池 data LV 的名称。使用 LV 的大小替换 *Size*。将 *VolumeGroupName* 替换为您的卷组的名称。

4. 创建 LV 来保存精简池元数据：

```
# lvcreate --name ThinPoolMetadataName --size Size VolumeGroupName /DevicePath
```

5. 将 LV 合并到精简池中：

```
# lvconvert --type thin-pool --poolmetadata ThinPoolMetadataName
VolumeGroupName/ThinPoolDataName
```

## 验证

1. 验证是否已创建自定义精简池：

```
# lvs -o lv_name,seg_type

LV          Type
ThinPoolDataName thin-pool
```

## 其他资源

- **vgs (8)** **lvs (8)**, **lvcreate (8)** man pages

## 第 6 章 使用快照管理系统升级

执行 Red Hat Enterprise Linux 系统的可回滚升级，以返回到操作系统的早期版本。您可以使用 **Boom Boot Manager** 和 **Leapp** 操作系统现代化框架。

在执行操作系统升级前，请考虑以下方面：

- 带有快照的系统升级不适用于系统树中的多个文件系统，例如一个独立的 **/var** 或 **/usr** 分区。
- 带有快照的系统升级不适用于 Red Hat Update Infrastructure (RHUI) 系统。考虑创建虚拟机的快照 (VM)，而不是使用 Boom 工具。

### 6.1. BOOM 过程概述

使用 Boom Boot Manager 创建引导条目，以便您可以从 GRUB 引导装载程序菜单中选择并访问这些条目。创建引导条目简化了准备可回滚升级的过程。

以下引导条目是升级和回滚过程的一部分：

- **升级引导条目**  
引导 Leapp 升级环境。使用 **leapp** 实用程序创建和管理此引导条目。**leapp** 升级过程会自动删除这个条目。
- **Red Hat Enterprise Linux 8 引导条目**  
引导升级系统环境。使用 **leapp** 实用程序在成功升级后创建此引导条目。
- **快照引导条目**  
引导原始系统的快照。在成功或不成功升级尝试后，使用它来检查并测试以前的操作系统状态。在升级操作系统前，请使用 **boom** 命令创建此引导条目。
- **回滚引导条目**  
引导原始系统环境，并将任何升级回滚到以前的系统状态。在启动升级过程的回滚时，请使用 **boom** 命令创建此引导条目。

#### 其他资源

- **boom (1)** 系统中的 man page

### 6.2. 使用 BOOM BOOT MANAGER 升级至另一个版本

使用 Boom Boot Manager 执行 Red Hat Enterprise Linux 操作系统的升级。

#### 先决条件

- 您正在运行 Red Hat Enterprise Linux 7.9。
- 已安装 **lvm2-python-boom** 软件包的当前版本（版本 **lvm2-python-boom-1.2-2.el7\_9.5** 或更高版本）。
- 您有足够的空间用于快照。根据原始安装的大小进行大小估计。列出所有挂载的逻辑卷。
- 您已安装了 **leapp** 软件包。
- 您已启用了软件存储库。



## 注意

其他文件系统可能包括 `/usr` 或 `/var`。

## 流程

### 1. 创建您的 `root` 逻辑卷快照：

- 如果您的 `root` 文件系统使用精简配置，请创建一个精简快照：

```
# lvcreate -s rhel/root -kn -n root_snapshot_before_changes
Logical volume "root_snapshot_before_changes" created.
```

在这里：

- **-s** 创建快照。
  - **rhel/root** 将文件系统复制到逻辑卷。
  - **-kn** 在引导时自动激活 LV。
  - **-n root\_snapshot\_before\_changes** 显示快照的名称。  
在创建精简快照时，不要定义快照大小。快照从精简池中分配。
- 如果您的 `root` 文件系统使用 `thick` 置备，请创建一个 `thick` 快照：

```
# lvcreate -s rhel/root -n root_snapshot_before_changes -L 25g
Logical volume "root_snapshot_before_changes" created.
```

在这个命令中：

- **-s** 创建快照。
  - **rhel/root** 将文件系统复制到逻辑卷。
  - **-n root\_snapshot\_before\_changes** 显示快照的名称。
  - **-L 25g** 是快照大小。根据原始安装的大小进行大小估计。  
在创建厚快照时，定义可保存升级过程中所有更改的快照大小。



## 重要

创建的快照不包括任何其他系统更改。

### 2. 使用 GRUB 引导装载程序启用 **boom**

```
# grub2-mkconfig > /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-1160.118.1.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-1160.118.1.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-1160.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-1160.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-f9f6209866c743739757658d1a4850b2
Found initrd image: /boot/initramfs-0-rescue-f9f6209866c743739757658d1a4850b2.img
done
```

### 3. 创建配置集：

```
# boom profile create --from-host --uname-pattern el7
Created profile with os_id f150f3d:
OS ID: "f150f3d6693495254255d46e20ecf5c690ec3262",
Name: "Red Hat Enterprise Linux Server", Short name: "rhel",
Version: "7.9 (Maipo)", Version ID: "7.9",
Kernel pattern: "/vmlinuz-%{version}", Initramfs pattern: "/initramfs-%{version}.img",
Root options (LVM2): "rd.lvm.lv=%{lvm_root_lv}",
Root options (BTRFS): "rootflags=%{btrfs_subvolume}",
Options: "root=%{root_device} ro %{root_opts}",
Title: "%{os_name} %{os_version_id} (%{version})",
Optional keys: "grub_users grub_arg grub_class id", UTS release pattern: "el7"
```

### 4. 使用原始引导镜像的备份副本创建原始系统的快照引导条目：

```
# boom create --backup --title "Root LV snapshot before changes" --rootlv
rhel/root_snapshot_before_changes
Created entry with boot_id bfef767:
title Root LV snapshot before changes
machine-id 7d70d7fcc6884be19987956d0897da31
version 3.10.0-1160.114.2.el7.x86_64
linux /vmlinuz-3.10.0-1160.114.2.el7.x86_64.boom0
initrd /initramfs-3.10.0-1160.114.2.el7.x86_64.img.boom0
options root=/dev/rhel/root_snapshot_before_changes ro
rd.lvm.lv=rhel/root_snapshot_before_changes
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

在这里：

- **--title "Root LV snapshot before changes"** 是引导条目的名称，它在系统启动期间在引导条目列表中显示。
- **--rootlv** 是与新引导条目对应的根逻辑卷。  
完成上一步后，您有一个引导条目，允许在升级前访问原始系统。

### 5. 使用 Leapp 工具升级到 Red Hat Enterprise Linux 8：

```
# leapp upgrade
==> Processing phase `configuration_phase`
====> * ipu_workflow_config
      IPU workflow config actor
==> Processing phase `FactsCollection`
...
=====
REPORT OVERVIEW
=====

Upgrade has been inhibited due to the following problems:
1. Btrfs has been removed from RHEL8
2. Missing required answers in the answer file
```

HIGH and MEDIUM severity reports:

1. Packages available in excluded repositories will not be installed
2. GRUB core will be automatically updated during the upgrade
3. Difference in Python versions and support in RHEL 8
4. chrony using default configuration

Reports summary:

```
Errors:          0
Inhibitors:      2
HIGH severity reports:  3
MEDIUM severity reports: 1
LOW severity reports:   3
INFO severity reports:  4
```

Before continuing consult the full report:

A report has been generated at /var/log/leapp/leapp-report.json  
A report has been generated at /var/log/leapp/leapp-report.txt

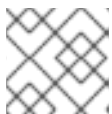
```
=====
                        END OF REPORT OVERVIEW
=====
```

检查并解决 **leapp upgrade** 命令报告所指示的任何阻碍。有关报告的详细信息，请参阅 [从命令行评估可升级性](#)。

#### 6. 重启到升级引导条目：

```
# leapp upgrade --reboot
==> Processing phase `configuration_phase`
=====> * ipu_workflow_config
      IPU workflow config actor
==> Processing phase `FactsCollection`
...
```

从 GRUB 引导屏幕中选择 **Red Hat Enterprise Linux Upgrade Initramfs** 条目。



#### 注意

Red Hat Enterprise Linux 8 不提供 GRUB 引导屏幕中的 Snapshots 子菜单。

#### 验证

- 完成升级后，系统会自动重启。GRUB 屏幕显示升级(Red Hat Enterprise Linux 8)和之前可用的操作系统版本。升级的系统版本是默认选择。

#### 其他资源

- **boom (1)** 系统中的 man page
- [什么是 BOOM 以及如何安装它？](#)（红帽知识库）
- [如何创建 BOOM 引导条目](#)（红帽知识库）
- [Leapp 工具对从 RHEL 7 原位升级到 RHEL 8 所需的数据](#)

## 6.3. 在 RED HAT ENTERPRISE LINUX 版本间切换

同时访问您机器上当前和以前的 Red Hat Enterprise Linux 版本。使用 **Boom Boot Manager** 访问不同的操作系统版本会降低与升级操作系统相关的风险，并有助于减少硬件停机时间。有了这种在环境之间切换的能力，您可以：

- 以并排的方式快速比较这两个环境。
- 在评估升级结果时在环境之间切换。
- 恢复文件系统的较早内容。
- 在升级的主机运行期间，继续访问旧系统。
- 随时停止并恢复更新过程，即使更新本身正在运行。

### 流程

1. 重启系统：

```
# reboot
```

2. 从 GRUB 引导装载程序屏幕中选择所需的引导条目。

### 验证

- 验证所选引导卷是否已显示：

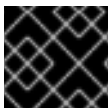
```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos1)/vmlinuz-3.10.0-1160.118.1.el7.x86_64.boom0
root=/dev/rhel/root_snapshot_before_changes ro
rd.lvm.lv=rhel/root_snapshot_before_changes
```

### 其他资源

- **boom (1)** 系统中的 man page

## 6.4. 成功升级后删除逻辑卷快照

如果您使用 Boom Boot Manager 成功升级了您的系统，您可以删除快照引导条目和逻辑卷(LV)快照，以使用升级的系统。



### 重要

删除后，您无法使用 LV 快照执行任何进一步的操作。

### 先决条件

- 您最近已 [使用 Boom Boot Manager](#) 将 Red Hat Enterprise Linux 升级到更新的版本。

### 流程

1. 从 GRUB 引导装载程序屏幕引导到 Red Hat Enterprise Linux 8。



2. 系统加载后，查看可用的引导条目。以下输出显示 boom 引导条目列表中的快照引导条目：

```
# boom list
WARNING - Options for BootEntry(boot_id=cae29bf) do not match OsProfile: marking read-only
BootID Version Name RootDevice
e0252ad 3.10.0-1160.118.1.el7.x86_64 Red Hat Enterprise Linux Server /dev/rhel/root_snapshot_before_changes
611ad14 3.10.0-1160.118.1.el7.x86_64 Red Hat Enterprise Linux Server /dev/mapper/rhel-root
3bfed71- 3.10.0-1160.el7.x86_64 Red Hat Enterprise Linux Server /dev/mapper/rhel-root
_cae29bf 4.18.0-513.24.1.el8_9.x86_64 Red Hat Enterprise Linux /dev/mapper/rhel-root
```

警告是正常的，您无法使用 boom 更改或删除这些条目，因为 **内核**和 **grubby** 软件包管理它们。

3. 使用 **BootID** 值删除快照条目：

```
# boom delete --boot-id e0252ad
Deleted 1 entry
```

这会从 GRUB 菜单中删除引导条目。

4. 删除 LV 快照：

```
# lvremove rhel/root_snapshot_before_changes
Do you really want to remove active logical volume rhel/root_snapshot_before_changes?
[y/n]: y
Logical volume "root_snapshot_before_changes" successfully removed
```

5. 完成剩余的升级后任务。详情请参阅 [从 RHEL 7 升级到 RHEL 8](#)。

## 其他资源

- **boom (1)** 系统中的 man page

## 6.5. 在升级后创建回滚引导条目

要在未成功升级后将操作系统升级恢复回系统以前的状态，请使用回滚引导条目。如果您发现升级环境有问题，例如与内部软件不兼容，这也很有用。

要准备回滚引导条目，请使用快照环境。

### 先决条件

- 您有一个快照。有关创建快照的说明，请参阅 [使用 Boom 引导管理器升级到另一个版本](#)。

### 流程

1. 将快照与原始卷合并：

```
# lvconvert --merge rhel/root_snapshot_before_changes
Logical volume rhel/root_snapshot_before_changes contains a filesystem in use.
Delaying merge since snapshot is open.
```

Merging of thin snapshot rhel/root\_snapshot\_before\_changes will occur on next activation of rhel/root.



### 警告

合并快照后，您必须继续此流程中所有剩余步骤，以防止数据丢失。

## 2. 为合并的快照创建一个回滚引导条目：

```
# boom create --backup --title "RHEL Rollback" --rootlv rhel/root
Created entry with boot_id 1e6d298:
title RHEL Rollback
machine-id f9f6209866c743739757658d1a4850b2
version 3.10.0-1160.118.1.el7.x86_64
linux /vmlinuz-3.10.0-1160.118.1.el7.x86_64.boom0
initrd /initramfs-3.10.0-1160.118.1.el7.x86_64.img.boom0
options root=/dev/rhel/root ro rd.lvm.lv=rhel/root
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

## 3. 重启机器以恢复操作系统状态：

```
# reboot
```

- 系统重启后，从 GRUB 屏幕中选择 **RHEL Rollback** 引导条目。
- 当 **root** 逻辑卷处于活动状态时，系统会自动启动快照合并操作。



### 重要

当合并操作启动时，快照卷不再可用。在成功引导 **RHEL Rollback** 引导条目后，**Root LV 快照引导条目** 不再工作。合并快照逻辑卷会破坏 Root LV 快照，并恢复原始卷之前的状态。

## 4. 完成合并操作后，删除未使用的条目，并恢复原始引导条目：

- 从 **/boot** 文件系统中删除未使用的 Red Hat Enterprise Linux 8 引导条目，并重建 **grub.cfg** 文件以使更改生效：

```
# rm -f /boot/loader/entries/*.el8*
```

```
# rm -f /boot/*.el8*
```

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-1160.118.1.el7.x86_64.boom0
```

```
....
done
```

- b. 恢复原始的 Red Hat Enterprise Linux 引导条目：

```
# new-kernel-pkg --update $(uname -r)
```

5. 成功回滚回系统后，删除 **boom** 快照，并回滚引导条目：

```
# boom list -o+title
BootID  Version          Name                                RootDevice          Title
a49fb09 3.10.0-1160.118.1.el7.x86_64 Red Hat Enterprise Linux Server /dev/mapper/rhel-
root      Red Hat Enterprise Linux (3.10.0-1160.118.1.el7.x86_64) 8.9 (Ootpa)
1bb11e4 3.10.0-1160.el7.x86_64 Red Hat Enterprise Linux Server /dev/mapper/rhel-root
Red Hat Enterprise Linux (3.10.0-1160.el7.x86_64) 8.9 (Ootpa)
e0252ad 3.10.0-1160.118.1.el7.x86_64 Red Hat Enterprise Linux Server
/dev/rhel/root_snapshot_before_changes Root LV snapshot before changes
1e6d298 3.10.0-1160.118.1.el7.x86_64 Red Hat Enterprise Linux Server /dev/rhel/root
RHEL Rollback

# boom delete e0252ad
Deleted 1 entry
# boom delete 1e6d298
Deleted 1 entry
```

## 其他资源

- **boom (1)** 系统中的 man page

## 第 7 章 自定义 LVM 报告

LVM 提供了广泛的配置和命令行选项来生成自定义报告。您可以对输出进行排序，指定单位、使用选择条件，并更新 **lvm.conf** 文件以自定义 LVM 报告。

### 7.1. 控制 LVM 显示的格式

当您使用没有附加选项的 **pvs**、**lvs** 或 **vgs** 命令时，您会看到以默认排序顺序显示的默认字段集合。**pvs** 命令的默认字段包括以下信息，按物理卷名称排序：

```
# pvs
PV      VG      Fmt  Attr PSize PFree
/dev/vdb1 VolumeGroupName lvm2  a--  17.14G 17.14G
/dev/vdb2 VolumeGroupName lvm2  a--  17.14G 17.09G
/dev/vdb3 VolumeGroupName lvm2  a--  17.14G 17.14G
```

#### PV

物理卷名称。

#### VG

卷组名称。

#### Fmt

物理卷的元数据格式：**lvm2** 或 **lvm1**。

#### Attr

物理卷的状态：(a)- 可分配或(x)- 导出。

#### PSize

物理卷的大小。

#### PFree

物理卷中剩余的可用空间。

#### 显示自定义字段

要显示与默认字段不同的字段，请使用 **-o** 选项。下面的例子只显示物理卷的名称、大小和可用空间：

```
# pvs -o pv_name,pv_size,pv_free
PV      PSize PFree
/dev/vdb1 17.14G 17.14G
/dev/vdb2 17.14G 17.09G
/dev/vdb3 17.14G 17.14G
```

#### 对 LVM 显示进行排序

要根据特定条件对结果进行排序，请使用 **-O** 选项。以下示例按其物理卷的空闲空间以升序排序条目：

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
PV      PSize PFree
/dev/vdb2 17.14G 17.09G
/dev/vdb1 17.14G 17.14G
/dev/vdb3 17.14G 17.14G
```

要按降序排列结果，请使用 **-O** 选项和 **-** 字符：

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
PV      PSize PFree
/dev/vdb1 17.14G 17.14G
/dev/vdb3 17.14G 17.14G
/dev/vdb2 17.14G 17.09G
```

## 其他资源

- [lvmreport \(7\)](#), [lvs \(8\)](#), [vgs \(8\)](#), 和 [pvs \(8\)](#) man page
- [为 LVM 显示指定单元](#)
- [自定义 LVM 配置文件](#)

## 7.2. 为 LVM 显示指定单元

您可以通过指定 LVM display 命令的 **- units** 参数来查看基础 2 或基本 10 单位的 LVM 设备的大小。请查看下表以了解所有参数：

| 单元类型    | 描述                      | 可用选项   | default   |
|---------|-------------------------|--|---|
| 基数 2 单位 | 以 2 的指数显示单位 (1024 的倍数)。 | <b>b</b> : Bytes.<br><b>s</b> : Sectors, 512 字节每个。<br><b>k</b> : kibibytes.<br><b>m</b> : <b>Mebibytes</b> .<br><b>g</b> : <b>Gibibytes</b> .<br><b>t</b> : <b>Tebibytes</b> .<br><b>p</b> : <b>Pebibytes</b> .<br><b>e</b> : <b>Exbibytes</b> .<br><b>h</b> : 使用最合适的单元是人类可读的。<br><b>r</b> : 使用舍入指示符的 <b>Human-readable</b> 与 <b>h</b> 类似, 使用舍入前缀 <b>&lt;</b> 或 <b>&gt;</b> 来指示 LVM 将显示的大小舍入到最接近的单元。 | <b>r</b> (当未指定单元时)。您可以通过在 <b>/etc/lvm/lvm.conf</b> 文件的全局部分中设置 <b>units</b> 参数来覆盖默认设置。 |

| 单元类型     | 描述   | 可用选项  | default |
|----------|--|---|---------|
| 基本 10 单元 | 以 1000 的倍数显示单位。  | <b>b</b> : Bytes.<br><b>S</b> : Sectors, 512 字节每个。<br><b>K</b> : Kilobytes.<br><b>M</b> : <b>Megabytes</b> .<br><b>G</b> : <b>Gigabytes</b> .<br><b>T</b> : <b>Terabytes</b> .<br><b>P</b> : <b>Petabytes</b> .<br><b>E</b> : <b>Exabytes</b> .<br><b>H</b> : 使用最合适的单元是人类可读的。<br><b>R</b> : 使用舍入指示符的 <b>Human</b> 可读, 与带有舍入前缀 <b>&lt; &lt; &lt;</b> 或 <b>&gt;</b> 的 <b>H</b> 类似, 以指示 LVM 将显示的大小舍入到最接近的单元。 | N/A     |
| 自定义单元    | 将数量与基本 2 或基础 10 单元相结合。例如, 若要以 4MB 为单位显示结果, 可使用 <b>4m</b> 。 | N/A   | N/A     |

- 如果您没有为单元指定值, 则默认使用人类可读的格式(**r**)。以下 **vgs** 命令以人类可读的格式显示 VG 的大小。使用最合适的单元, 并且舍入指示符 **< < <** 显示实际大小是一个估计值, 它小于 931 gibibytes。

```
# vgs myvg
VG   #PV #LV #SN Attr VSize   VFree
myvg 1   1   0 wz-n <931.00g <930.00g
```

- 以下 **pvs** 命令以 **/dev/vdb** 物理卷的 base 2 gibibyte 单位显示输出：

```
# pvs --units g /dev/vdb
PV      VG   Fmt Attr PSize  PFree
/dev/vdb myvg lvm2 a-- 931.00g 930.00g
```

- 以下 **pvs** 命令以 **/dev/vdb** 物理卷的 base 10GB 单位显示输出：

```
# pvs --units G /dev/vdb
PV      VG   Fmt Attr PSize  PFree
/dev/vdb myvg lvm2 a-- 999.65G 998.58G
```

- 以下 **pvs** 命令以 512 字节扇区显示输出：

■

```
# pvs --units s
PV      VG      Fmt Attr PSize   PFree
/dev/vdb myvg   lvm2 a-- 1952440320S 1950343168S
```

- 您可以为 LVM display 命令指定自定义单元。下面的例子以 4MB 为单位为单位显示 **pvs** 命令的输出：

```
# pvs --units 4m
PV      VG      Fmt Attr PSize   PFree
/dev/vdb myvg   lvm2 a-- 238335.00U 238079.00U
```

### 7.3. 自定义 LVM 配置文件

您可以通过编辑 **lvm.conf** 文件来自定义 LVM 配置，具体取决于具体存储和系统要求。例如，您可以编辑 **lvm.conf** 文件来修改过滤设置、配置卷组自动激活、管理精简池或自动扩展快照。

#### 流程

1. 在您选择的编辑器中打开 **lvm.conf** 文件。
2. 通过取消注释和修改要修改默认显示值的设置来自定义 **lvm.conf** 文件。
  - 要自定义您在 **lvs** 输出中看到的字段，取消注释 **lvs\_cols** 参数并修改它：

```
lvs_cols="lv_name,vg_name,lv_attr"
```

- 要隐藏 **pvs**、**vgs** 和 **lvs** 命令的空字段，取消注释 **compact\_output=1** 设置：

```
compact_output = 1
```

- 要将 gigabytes 设置为 **pvs**、**vgs** 和 **lvs** 命令的默认单元，请将 **units = "r"** 设置替换为 **units = "G"**：

```
units = "G"
```

3. 确保已取消注释 **lvm.conf** 文件的对应部分。例如，要修改 **lvs\_cols** 参数，必须取消注释 **report** 部分：

```
report {
...
}
```

#### 验证

- 在修改 **lvm.conf** 文件后查看更改的值：

```
# lvmconfig --typeconfig diff
```

#### 其他资源

- 在您的系统中的 **lvm.conf (5)** 手册页

## 7.4. 定义 LVM 选择标准

选择标准是 **<field> <operator> <value>** 形式的一组语句，它使用比较运算符来定义特定字段的值。然后，处理或显示与选择条件匹配的对象。对象可以是物理卷(PV)、卷组(VG)或逻辑卷(LV)。语句通过逻辑和分组运算符进行组合。

要定义选择标准，请使用 **-S** 或 **--select** 选项，后跟一个或多个语句。

**-S** 选项的工作原理是通过描述要处理的对象，而不是命名每个对象。这在处理许多对象时很有用，当搜索有复杂特征集合的对象时，很难查找和单独命名每个对象。也可以使用 **-S** 选项作为快捷方式，以避免输入许多名称。

要查看完整的字段和可能运算符集，请使用 **lvs -S help** 命令。使用任何报告或处理命令替换 **lvs** 来查看该命令的详情：

- 报告命令包括 **pvs,vgs,lvs,pvdisplay,vgdisplay,lvdisplay** 和 **dmsetup info -c**。
- 处理命令包括 **pvchange,vgchange,lvchange,vgimport,vgexport,vgremove** 和 **lvremove**。

使用 **pvs** 命令选择条件示例

- 以下 **pvs** 命令示例只显示带有名称为 **nvme** 的物理卷：

```
# pvs -S name=~nvme
PV      Fmt Attr PSize PFree
/dev/nvme2n1 lvm2 --- 1.00g 1.00g
```

- 以下 **pvs** 命令示例只显示 **myvg** 卷组中的物理设备：

```
# pvs -S vg_name=myvg
PV      VG  Fmt Attr PSize  PFree
/dev/vdb1 myvg lvm2 a-- 1020.00m 396.00m
/dev/vdb2 myvg lvm2 a-- 1020.00m 896.00m
```

使用 **lvs** 命令的选择条件示例

- 以下 **lvs** 命令示例只显示大小大于 100m 但小于 200m 的逻辑卷：

```
# lvs -S 'size > 100m && size < 200m'
LV VG Attr   LSize Cpy%Sync
rr myvg rwi-a-r--- 120.00m 100.00
```

- 以下 **lvs** 命令示例只显示名为 **lv01** 的逻辑卷，以及 0 到 2 之间的任何数字：

```
# lvs -S name=~lv01[02]
LV VG Attr   LSize
lv00 myvg -wi-a----- 100.00m
lv02 myvg -wi----- 100.00m
```

- 以下 **lvs** 命令示例只显示带有 **raid1** segment 类型的逻辑卷：

```
# lvs -S segtype=raid1
LV VG Attr   LSize Cpy%Sync
rr myvg rwi-a-r--- 120.00m 100.00
```



## 高级示例

您可以将选择条件与其他选项合并。

- 以下 **lvchange** 命令示例将特定的标签 **mytag** 添加到活跃的逻辑卷中：

```
# lvchange --addtag mytag -S active=1
Logical volume myvg/mylv changed.
Logical volume myvg/lvol0 changed.
Logical volume myvg/lvol1 changed.
Logical volume myvg/rr changed.
```

- 以下 **lvs** 命令示例显示所有名称不匹配 **\_pmspare** 的逻辑卷，并将默认标头改为自定义卷：

```
# lvs -a -o lv_name,vg_name,attr,size,pool_lv,origin,role -S 'name!~_pmspare'
LV      VG      Attr      LSize Pool Origin Role
thin1   example Vwi-a-tz-- 2.00g tp      public,origin,thinorigin
thin1s   example Vwi---tz-- 2.00g tp thin1 public,snapshot,thinsnapshot
thin2   example Vwi-a-tz-- 3.00g tp      public
tp       example twi-aotz-- 1.00g      private
[tp_tdata] example Twi-ao---- 1.00g      private,thin,pool,data
[tp_tmeta] example ewi-ao---- 4.00m      private,thin,pool,metadata
```

- 下面的 **lvchange** 命令示例标记了在正常激活命令中跳过 **role=thinsnapshot** 和 **origin=thin1** 的逻辑卷：

```
# lvchange --setactivationskip n -S 'role=thinsnapshot && origin=thin1'
Logical volume myvg/thin1s changed.
```

- 以下 **lvs** 命令示例只显示与所有这三个条件匹配的逻辑卷：

- 名称包含 **\_tmeta**。
- 角色是 **元数据**。
- 大小小于或等于 4m。

```
# lvs -a -S 'name=~_tmeta && role=metadata && size <= 4m'
LV      VG      Attr      LSize
[tp_tmeta] myvg ewi-ao---- 4.00m
```

## 其他资源

- 系统中 **lvmreport (7)** 手册页

## 第 8 章 在共享存储上配置 LVM

共享存储是可同时被多个节点访问的存储。您可以使用 LVM 管理共享存储。共享存储通常用于集群和高可用性设置，对于共享存储如何出现在系统上有两种常见场景：

- LVM 设备被附加到主机，并传给客户机虚拟机使用。在这种情况下，设备永远不会被主机使用，只被客户机虚拟机使用。
- 机器被附加到存储区域网络(SAN)，例如使用光纤通道，而 SAN LUN 对多台机器可见：

### 8.1. 为虚拟机磁盘配置 LVM

要防止虚拟机存储暴露给主机，您可以配置 LVM 设备访问和 LVM 系统 ID。您可以通过从主机中排除有问题的设备来实现此目的，这样可确保主机上的 LVM 看不到或不使用传给客户虚拟机的设备。您可以通过在 VG 中设置 LVM 系统 ID 以匹配客户虚拟机，来防止偶然使用主机上虚拟机的 VG。

#### 流程

1. 在 **lvm.conf** 文件中，过滤要排除的设备的路径：

```
filter = [ "r|^path_to_device$" ]
```

2. 可选：您可以进一步保护 LVM 设备：

- a. 在主机和虚拟机的 **lvm.conf** 文件中设置 LVM 系统 ID 功能：

```
system_id_source = "uname"
```

- b. 设置 VG 的系统 ID 以匹配虚拟机系统 ID。这样可确保只有客户虚拟机能够激活 VG：

```
$ vgchange --systemid <VM_system_id> <VM_vg_name>
```

### 8.2. 将 LVM 配置为在一台机器上使用 SAN 磁盘

要防止 SAN LUN 被错误的机器使用，请在所有机器上的 **lvm.conf** 过滤器中排除这些磁盘，要使用它们的机器除外。

您还可以通过在所有机器上配置系统 ID，并在 VG 中设置系统 ID 来匹配使用它的机器，来保护它不被错误的机器使用。

#### 流程

1. 在 **lvm.conf** 文件中过滤设备的路径以排除该设备：

```
filter = [ "r|^path_to_device$" ]
```

2. 在 **lvm.conf** 文件中设置 LVM 系统 ID 功能：

```
system_id_source = "uname"
```

3. 设置 VG 的系统 ID，以匹配使用此 VG 的机器的系统 ID：

■

```
$ vgchange --systemid <system_id> <vg_name>
```

### 8.3. 配置 LVM，以使用 SAN 磁盘进行故障转移

您可以将 LUN 配置为在机器间移动，例如用于故障转移目的。您可以通过配置 **lvm.conf** 过滤器来在所有可能使用它们的机器上包含 LUN，并在每台机器上配置 LVM 系统 ID 来设置 LVM。

以下流程描述了初始 LVM 配置，来为故障转移设置 LVM，并在机器之间移动 VG，您需要配置 **pacemaker** 并将自动修改 VG 的系统 ID 的 LVM 激活资源代理，来匹配可以使用 VG 的机器的系统 ID。如需更多信息，请参阅 [配置和管理高可用性集群](#)。

#### 流程

1. 在 **lvm.conf** 文件中，过滤要排除的设备的路径：

```
filter = [ "a|^path_to_device$" ]
```

2. 在所有机器上的 **lvm.conf** 文件中设置 LVM 系统 ID 功能：

```
system_id_source = "uname"
```

### 8.4. 配置 LVM，来在多台机器间共享 SAN 磁盘

使用 **lvmlockd** 守护进程和锁管理器（如 **dlm** 或 **sanlock**），您可以启用从多台机器访问 SAN 磁盘上的共享 VG。具体命令可能会因使用的锁管理器和操作系统而异。下面的流程描述了配置 LVM 来在多个机器间共享 SAN 磁盘所需的步骤的概述。



#### 警告

在使用 **pacemaker** 时，必须使用 [配置和管理高可用性集群](#) 中所示的 **pacemaker** 步骤配置和启动系统。

#### 流程

1. 配置 **lvm.conf** 过滤器，来包含将使用它们的所有机器的 LUN：

```
filter = [ "a|^path_to_device$" ]
```

2. 配置 **lvm.conf** 文件，以便在所有机器上使用 **lvmlockd** 守护进程：

```
use_lvmlockd=1
```

3. 在所有机器上启动 **lvmlockd** 守护进程文件。
4. 启动锁管理器，以与 **lvmlockd** 一起使用，如所有机器上的 **dlm** 或 **sanlock**。
5. 使用命令 **vgcreate --shared** 创建一个新的共享 VG。

6. 在所有机器上使用 **vgchange --lockstart** 和 **vgchange --lockstop** 命令启动和停止访问现有的共享 VG。

## 其他资源

- 系统中 **lvmlockd** (8) 手册页

## 8.5. 使用 STORAGE RHEL 系统角色创建共享的 LVM 设备

如果您希望多个系统同时访问同一个存储，则您可以使用 **storage** RHEL 系统角色创建共享的 LVM 设备。

这可带来以下显著优点：

- 资源共享
- 管理存储资源方面的灵活性
- 存储管理任务的简化

## 先决条件

- [您已准备好控制节点和受管节点](#)
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- **lvmlockd** 在受管节点上配置。如需更多信息，请参阅 [配置 LVM，以在多台机器之间共享 SAN 磁盘](#)。

## 流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  become: true
  tasks:
    - name: Create shared LVM device
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_pools:
          - name: vg1
            disks: /dev/vdb
            type: lvm
            shared: true
            state: present
            volumes:
              - name: lv1
                size: 4g
```

```
mount_point: /opt/test1
storage_safe_mode: false
storage_use_partitions: true
```

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

### 其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录

## 第 9 章 配置 RAID 逻辑卷

您可以使用逻辑卷管理器(LVM)创建和管理 Redundant Array of Independent Disks (RAID)卷。LVM 支持 RAID 0、1、4、5、6 和 10。LVM RAID 卷有以下特征：

- LVM 会创建和管理利用多设备 (MD) 内核驱动程序的 RAID 逻辑卷。
- 您可以从阵列中临时分割 RAID1 镜像，并在之后将其合并到阵列中。
- LVM RAID 卷支持快照。
- RAID 逻辑卷不是集群感知的。您可以只在一台机器中创建和激活 RAID 逻辑卷，但不能在多台机器中同时激活它们。
- 当您创建 RAID 逻辑卷 (LV) 时，LVM 会创建一个元数据子卷，它是阵列中的每个数据或奇偶校验子卷的大小的一个区块。例如，创建一个双向 RAID1 阵列会导致两个元数据子卷(`lv_rmeta_0` 和 `lv_rmeta_1`)和两个数据子卷(`lv_rimage_0` 和 `lv_rimage_1`)。
- 在 RAID LV 中添加完整性可减少或防止软崩溃。

### 9.1. RAID 级别和线性支持

以下是 RAID 支持的配置，包括级别 0、1、4、5、6、10 和线性：

#### 0 级

RAID 级别 0，通常称为条带化的数据映射技术。这意味着，要写入阵列的数据被分成条块，并在阵列的成员磁盘中写入，这样可以在成本低的情况下提供高的 I/O 性能，但不提供冗余。

RAID 级别 0 的实现只在成员设备间条状分布到阵列中最小设备的大小。就是说，如果您有多个设备，它们的大小稍有不同，那么每个设备的大小都被视为与最小设备的大小相同。因此，级别 0 阵列的常见存储容量是所有磁盘的总容量。如果成员磁盘具有不同的大小，RAID 0 将使用可用区使用这些磁盘的所有空间。

#### 1 级

RAID 级别 1 或称为镜像 (mirroring)，通过将相同数据写入阵列的每个磁盘来提供冗余，在每个磁盘上保留“镜像”副本。因为其简单且数据高度可用，RAID 1 仍然被广泛使用。级别 1 需要两个或者多个磁盘，它提供了很好的数据可靠性，提高了需要读取的应用程序的性能，但是成本相对高。

为了实现数据可靠性，需要向阵列中的所有磁盘写入相同的信息，所以 RAID 1 的成本会很高。与基于奇偶校验的其他级别（如级别 5）相比，空间的利用效率较低。然而，对空间利用率的牺牲提供了高性能：基于奇偶校验的 RAID 级别会消耗大量 CPU 资源以便获得奇偶校验，而 RAID 级别 1 只是一次向多个 RAID 成员中写入同样数据，其对 CPU 的消耗较小。因此，在使用软件 RAID 的系统中，或系统中有其他操作需要大量使用 CPU 资源时，RAID 1 可能会比使用基于奇偶校验的 RAID 级别的性能更好。

级别 1 阵列的存储容量等于硬件 RAID 中最小镜像硬盘或者软件 RAID 中最小镜像分区的容量相同。级别 1 所提供的冗余性是所有 RAID 级别中最高的，因为阵列只需要在有一个成员可以正常工作的情况下就可以提供数据。

#### 级别 4

级别 4 使用单一磁盘驱动器中的奇偶校验来保护数据。奇偶校验信息根据阵列中其余成员磁盘的内容计算。然后当阵列中的一个磁盘失败时，这个信息就可以被用来重建数据。然后，在出现问题的磁盘被替换前，使用被重建的数据就可以满足 I/O 的请求。在磁盘被替换后，可以在上面重新生成数据。因为专用奇偶校验磁盘代表所有写交易到 RAID 阵列的固有瓶颈，因此在没有写回缓存等技术的情况下，级别 4 很少被使用。或者在特定情况下，系统管理员有意设计具有这个瓶颈的软件 RAID 设备，比如当阵列使用数据填充后没有写入事务的数组。因此，Anaconda 中并没有提供 RAID 4 这个选项。

但是，如果需要，用户可以手动创建它。

硬件 RAID 4 的存储容量等于分区数量减一乘以最小成员分区的容量。RAID 4 阵列的性能是非对称的，即读的性能会好于写的性能。这是因为，写操作会在生成奇偶校验时消耗额外的 CPU 和主内存带宽，然后在将实际数据写入磁盘时也会消耗额外的总线带宽，因为您不仅写数据，而且还写奇偶校验。读操作只需要读取数据而不是奇偶校验，除非该阵列处于降级状态。因此，在正常操作条件下，对于相同数量的数据传输，读操作会对驱动器和计算机总线产生较少的流量。

5 级

这是最常见的 RAID 类型。通过在一个阵列的所有成员磁盘中分布奇偶校验，RAID 5 解除了级别 4 中原有的写入瓶颈。唯一性能瓶颈是奇偶校验计算过程本身。现代 CPU 可以非常快速地计算奇偶校验。但是，如果您在 RAID 5 阵列中有大量磁盘，以便在所有设备间合并数据传输速度非常高，则奇偶校验计算可能会成为瓶颈。  
5 级具有非对称性能，读性能显著提高。RAID 5 的存储容量的计算方法与级别 4 的计算方法是一样的。

级别 6

如果数据的冗余性和保护性比性能更重要，且无法接受 RAID 1 的空间利用率低的问题，则通常会选择使用级别 6。级别 6 使用一个复杂的奇偶校验方式，可以在阵列中出现任意两个磁盘失败的情况下进行恢复。因为使用的奇偶校验方式比较复杂，软件 RAID 设备会对 CPU 造成较大负担，同时对写操作造成更大的负担。因此，与级别 4 和 5 相比，级别 6 的性能不对称性更严重。  
RAID 6 阵列的总容量与 RAID 5 和 4 类似，但您必须从额外奇偶校验存储空间和设备数中减去 2 个设备（而不是 1 个）。

级别 10

这个 RAID 级别将级别 0 的性能优势与级别 1 的冗余合并。它还可减少在具有多于两个设备的 1 级阵列中发现的一些空间。对于 10 级，可以创建一个 3 个驱动器阵列，来仅存储每块数据的 2 个副本，然后允许整个阵列的大小为最小设备的 1.5 倍，而不是只等于最小设备（这与 3 设备 1 级阵列类似）。与 RAID 级别 6 相比，计算奇偶校验对 CPU 的消耗较少，但空间效率较低。  
在安装过程中，不支持创建 RAID 10。您可在安装后手动创建。

线性 RAID

线性 RAID 是创建更大的虚拟驱动器的一组驱动器。  
在线性 RAID 中，块会被从一个成员驱动器中按顺序分配，只有在第一个完全填充时才会进入下一个驱动器。这个分组方法不会提供性能优势，因为 I/O 操作不太可能在不同成员间同时进行。线性 RAID 也不提供冗余性，并会降低可靠性。如果有任何一个成员驱动器失败，则无法使用整个阵列，数据可能会丢失。该容量是所有成员磁盘的总量。

9.2. LVM RAID 片段类型

要创建 RAID 逻辑卷，您可以使用 **lvcreate** 命令的 **--type** 参数指定 RAID 类型。对于大多数用户，指定五个可用主类型之一 **raid1**、**raid4**、**raid5**、**raid6** 和 **raid10** 应足够。

下表描述了可能的 RAID 片段类型。

表 9.1. LVM RAID 片段类型

| 片段类型  | 描述   |
|-------|--|
| raid1 | RAID1 镜像。当您指定 <b>-m</b> 参数而不指定条带时，这是 <b>lvcreate</b> 命令的 <b>--type</b> 参数的默认值。 |

| 片段类型            | 描述  |
|-----------------|---|
| <b>raid4</b>    | RAID4 专用奇偶校验磁盘。   |
| <b>raid5_la</b> | <ul style="list-style-type: none"> <li>RAID5 左非对称。</li> <li>轮转奇偶校验 0 并分配数据。</li> </ul>  |
| <b>raid5_ra</b> | <ul style="list-style-type: none"> <li>RAID5 右非对称。</li> <li>轮转奇偶校验 N 并分配数据。</li> </ul>  |
| <b>raid5_ls</b> | <ul style="list-style-type: none"> <li>RAID5 左对称。</li> <li>它与 <b>raid5</b> 相同。</li> <li>使用数据重启轮转奇偶校验 0。</li> </ul>                                    |
| <b>raid5_rs</b> | <ul style="list-style-type: none"> <li>RAID5 右对称。</li> <li>使用数据重启轮转奇偶校验 N。</li> </ul>   |
| <b>raid6_zr</b> | <ul style="list-style-type: none"> <li>RAID6 零重启。</li> <li>它与 <b>raid6</b> 相同。</li> <li>数据重启时的旋转奇偶校验零（从左到右）。</li> </ul>                               |
| <b>raid6_nr</b> | <ul style="list-style-type: none"> <li>RAID6 N 重启。</li> <li>数据重启时的旋转奇偶校验 N（从左到右）。</li> </ul>  |
| <b>raid6_nc</b> | <ul style="list-style-type: none"> <li>RAID6 N 继续。</li> <li>数据持续的旋转奇偶校验 N（从左到右）。</li> </ul>   |
| <b>raid10</b>   | <ul style="list-style-type: none"> <li>条状镜像。如果您指定了 <b>-m</b> 参数以及大于 1 的条带数，则这是 <b>lvcreate</b> 命令的 <b>--type</b> 参数的默认值。</li> <li>镜像集合的条带。</li> </ul> |



| 片段类型             | 描述   |
|------------------|--|
| raid0/raid0_meta | 条带。RAID0 以条带大小的单位在多个数据子卷间分布逻辑卷数据。这可以提高性能。如果任何数据子卷失败，逻辑卷数据将会丢失。 |

9.3. 创建 RAID0 的参数

您可以使用 `lvcreate --type raid0[meta] --stripes Stripes --stripesize StripeSize VolumeGroup [PhysicalVolumePath]` 命令创建 RAID0 条状逻辑卷。

下表描述了不同的参数，您可以在创建 RAID0 条状逻辑卷时使用它们。

表 9.2. 创建 RAID0 条状逻辑卷的参数

| 参数  | 描述   |
|---|--|
| <code>--type raid0[_meta]</code>            | 指定 <b>raid0</b> 创建一个没有元数据卷的 RAID0 卷。指定 <b>raid0_meta</b> 创建带有元数据卷的 RAID0 卷。因为 RAID0 是不方便的，因此它不会存储任何镜像数据块作为 RAID1/10，或者计算并存储任何奇偶校验块，因为 RAID4/5/6。因此，它不需要元数据卷来保持有关镜像或奇偶校验块重新同步进程的状态。在从 RAID0 转换到 RAID4/5/6/10 时，元数据卷是强制的。指定 <b>raid0_meta</b> 预先分配这些元数据卷，以防止相应的分配失败。 |
| <code>--stripes <i>Stripes</i></code>       | 指定在其中分割逻辑卷的设备数。  |
| <code>--stripesize <i>StripeSize</i></code> | 以 KB 为单位指定每个条的大小。这是在移动到下一个设备前写入一个设备的数据量。   |
| <i>卷组</i>                                   | 指定要使用的卷组。  |
| <i>PhysicalVolumePath</i>                   | 指定要使用的设备。如果没有指定，LVM 会选择 <i>Stripes</i> 选项指定的设备数，每个条带一个。  |

9.4. 创建 RAID 逻辑卷

您可以根据您为 `-m` 参数指定的值，来创建具有不同副本数的 RAID1 阵列。同样，您可以使用 `-i` 参数为 RAID 0、4、5、6 和 10 逻辑卷指定条带数。您还可以使用 `-l` 参数指定条带大小。下面的步骤描述了创建不同类型的 RAID 逻辑卷的不同方法。

流程

- 创建一个双向 RAID。以下命令在卷组 *my\_vg* 中创建一个名为 *my\_lv* 的双向 RAID1 阵列，大小为 1G：

```
# lvcreate --type raid1 -m 1 -L 1G -n my_lv my_vg
Logical volume "my_lv" created.
```

- 使用条带创建 RAID5 阵列。以下命令在卷组 `my_vg` 中创建具有三个条带的 RAID5 阵列和一个隐式奇偶校验驱动器，名为 `my_lv`，大小为 1G。请注意，您可以指定与 LVM 条带卷类似的条带数。自动添加正确奇偶校验驱动器数。

```
# lvcreate --type raid5 -i 3 -L 1G -n my_lv my_vg
```

- 使用条带创建 RAID6 阵列。以下命令在卷组 `my_vg` 中创建具有三个 3 个条带的 RAID6 阵列，以及名为 `my_lv` 的两个隐式奇偶校验驱动器，大小为 1G：

```
# lvcreate --type raid6 -i 3 -L 1G -n my_lv my_vg
```

## 验证

- 显示 LVM 设备 `my_vg/my_lv`，它是一个双向 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       6.25  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(0)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rmeta_0]   /dev/sde1(256)
[my_lv_rmeta_1]   /dev/sdf1(0)
```

## 其他资源

- 系统中 **lvcreate** (8) 和 **lvraid** (7) 手册页

## 9.5. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池

使用 **storage** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 上配置带有 RAID 的 LVM 池。您可以使用可用参数设置一个 Ansible playbook，来配置带有 RAID 的 LVM 池。

### 先决条件

- [您已准备好控制节点和受管节点](#)
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

### 流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure LVM pool with RAID
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_safe_mode: false
```

```

storage_pools:
  - name: my_pool
    type: lvm
    disks: [sdh, sdi]
    raid_level: raid1
    volumes:
      - name: my_volume
        size: "1 GiB"
        mount_point: "/mnt/app/shared"
        fs_type: xfs
        state: present

```

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件。

## 2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

## 3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 验证您的池是否在 RAID 中：

```
# ansible managed-node-01.example.com -m command -a 'lsblk'
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录
- [管理 RAID](#)

## 9.6. 创建 RAID0 条带化逻辑卷

RAID0 逻辑卷以条的大小为单位，将逻辑卷数据分散到多个数据子卷中。下面的步骤创建了一个名为 `mylv` 的 LVM RAID0 逻辑卷，该逻辑卷在磁盘间条状分布数据。

### 先决条件

1. 您已创建了三个或者多个物理卷。有关创建物理卷的更多信息，[请参阅创建 LVM 物理卷](#)。
2. 您已创建了卷组。如需更多信息，[请参阅创建 LVM 卷组](#)。

### 流程

1. 从现有卷组中创建 RAID0 逻辑卷。以下命令从卷组 `myvg` 中创建 RAID0 卷 `mylv`，大小为 2G，有三个条带，条带大小为 4kB：

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv my_vg
Rounding size 2.00 GiB (512 extents) up to stripe boundary size 2.00 GiB(513 extents).
Logical volume "mylv" created.
```

2. 在 RAID0 逻辑卷中创建文件系统。以下命令在逻辑卷中创建 ext4 文件系统：

```
# mkfs.ext4 /dev/my_vg/mylv
```

3. 挂载逻辑卷并报告文件系统磁盘空间使用情况：

```
# mount /dev/my_vg/mylv /mnt

# df
Filesystem            1K-blocks    Used Available Use% Mounted on
/dev/mapper/my_vg-mylv 2002684    6168 1875072   1% /mnt
```

## 验证

- 查看创建的 RAID0 剥离的逻辑卷：

```
# lvs -a -o +devices,segtype my_vg
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert Devices Type
mylv my_vg rwi-a-r--- 2.00g mylv_rimage_0(0),mylv_rimage_1(0),mylv_rimage_2(0) raid0
[mylv_rimage_0] my_vg iwi-aor--- 684.00m /dev/sdf1(0) linear
[mylv_rimage_1] my_vg iwi-aor--- 684.00m /dev/sdg1(0) linear
[mylv_rimage_2] my_vg iwi-aor--- 684.00m /dev/sdh1(0) linear
```

## 9.7. 使用 STORAGE RHEL 系统角色为 RAID LVM 卷配置条带大小

使用 **storage** 系统角色，您可以使用 Red Hat Ansible Automation Platform 为 RHEL 上的 RAID LVM 卷配置条带大小。您可以使用可用参数设置一个 Ansible playbook，来配置带有 RAID 的 LVM 池。

### 先决条件

- [您已准备好控制节点和受管节点](#)
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

### 流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure stripe size for RAID LVM volumes
      ansible.builtin.include_role:
```

```

name: redhat.rhel_system_roles.storage
vars:
  storage_safe_mode: false
  storage_pools:
    - name: my_pool
      type: lvm
      disks: [sdh, sdi]
      volumes:
        - name: my_volume
          size: "1 GiB"
          mount_point: "/mnt/app/shared"
          fs_type: xfs
          raid_level: raid0
          raid_stripe_size: "256 KiB"
          state: present

```

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件。

## 2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

## 3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 验证条带大小是否已设置为所需的大小：

```
# ansible managed-node-01.example.com -m command -a 'lvs -o+stripesize /dev/my_pool/my_volume'
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录
- [管理 RAID](#)

## 9.8. 软数据崩溃

数据存储中的软崩溃意味着，从存储设备中检索的数据与写入到那个设备中的数据不同。错误的数据可以在存储设备中无限期存在。在检索并尝试使用此数据前，您可能不会发现此损坏的数据。

根据配置类型，一个独立磁盘的冗余阵列 (RAID) 逻辑卷 (LV) 可防止设备失败时数据丢失。如果由 RAID 阵列组成的设备失败，可以从 RAID LV 一部分的其他设备中恢复数据。但是 RAID 配置不能保证数据本身的完整性。软崩溃、静默崩溃、软错误和静默错误用来描述，即使系统和软件仍继续按预期工作，但数据已损坏的情况的术语。

当创建一个具有 DM 完整性或在现有 RAID LV 中添加完整性的 RAID LV 时，请考虑以下点：

- 完整性元数据需要额外的存储空间。对于每个 RAID 镜像，每个 500MB 数据都需要 4MB 的额外存储空间，因为添加到数据的校验和。
- 添加 DM 完整性会因为访问数时延迟而影响到性能，有些 RAID 的配置会比其他 RAID 配置受到的影响更大。RAID1 配置通常比 RAID5 或其变体提供更好的性能。
- RAID 完整性块的大小也会影响性能。配置更大的 RAID 完整块可提供更好的性能。但是，一个较小的 RAID 完整性块可以提供更好的兼容性。
- 完整性有两种模式：**位图 (bitmap)** 或**日志 (journal)**。**bitmap** 完整性模式通常比 **journal** 模式提供更好的性能。

## 提示

如果您遇到性能问题，请使用带有完整性的 RAID1，或者测试特定 RAID 配置的性能以确保它满足您的要求。

## 9.9. 创建具有 DM 完整性的 RAID 逻辑卷

当您创建带有设备映射器(DM)完整性的 RAID LV 或者在现有 RAID 逻辑卷(LV)中添加完整性时，它会降低因为软崩溃而丢失数据的风险。在使用 LV 前，等待完整性同步和 RAID 元数据完成。否则，在后台进行的初始化可能会影响 LV 的性能。

设备映射程序(DM)完整性与 RAID 级别 1、4、5、6 和 10 一起使用，用于缓解或防止软崩溃导致数据丢失。RAID 层可确保非破坏的数据副本可以修复软崩溃错误。

## 流程

1. 创建具有 DM 完整性的 RAID LV。以下示例在 *my\_vg* 卷组中创建一个名为 *test-lv* 的 RAID LV，可用大小为 256M 和 RAID 级别 1：

```
# lvcreate --type raid1 --raidintegrity y -L 256M -n test-lv my_vg
Creating integrity metadata LV test-lv_rimage_0_imeta with size 8.00 MiB.
Logical volume "test-lv_rimage_0_imeta" created.
Creating integrity metadata LV test-lv_rimage_1_imeta with size 8.00 MiB.
Logical volume "test-lv_rimage_1_imeta" created.
Logical volume "test-lv" created.
```



## 注意

要在现有 RAID LV 中添加 DM 完整性，请使用以下命令：

```
# lvconvert --raidintegrity y my_vg/test-lv
```

在 RAID LV 中添加完整性限制了您可以在那个 RAID LV 上执行的一些操作。

2. 可选：在执行某些操作前删除完整性。

```
# lvconvert --raidintegrity n my_vg/test-lv
Logical volume my_vg/test-lv has removed integrity.
```

## 验证

- 查看有关添加的 DM 完整性的信息：
  - 查看在 `my_vg` 卷组中创建的 `test-lv` RAID LV 的信息：

```
# lvs -a my_vg
LV          VG      Attr      LSize  Origin              Cpy%Sync
test-lv     my_vg   rwi-a-r--- 256.00m              2.10
[test-lv_rimage_0] my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig] 93.75
[test-lv_rimage_0_imeta] my_vg ewi-ao---- 8.00m
[test-lv_rimage_0_iorig] my_vg -wi-ao---- 256.00m
[test-lv_rimage_1] my_vg gwi-aor--- 256.00m [test-lv_rimage_1_iorig] 85.94
[...]
```

下面描述了此输出的不同选项：

**g 属性**

它是 `Attr` 列下的属性列表，表示 RAID 镜像使用完整性。完整性将校验和存储在 `_imeta` RAID LV 中。

**Cpy%Sync 列**

它指示顶层 RAID LV 和每个 RAID 镜像的同步进度。

**RAID 镜像**

它通过 `raid_image_N` 在 LV 列中指示。

**LV 列**

它确保对顶层 RAID LV 和每个 RAID 镜像显示 100% 同步进度。

- 显示每个 RAID LV 的类型：

```
# lvs -a my_vg -o+segtype
LV          VG      Attr      LSize  Origin              Cpy%Sync Type
test-lv     my_vg   rwi-a-r--- 256.00m              87.96  raid1
[test-lv_rimage_0] my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig] 100.00
integrity
[test-lv_rimage_0_imeta] my_vg ewi-ao---- 8.00m              linear
[test-lv_rimage_0_iorig] my_vg -wi-ao---- 256.00m              linear
[test-lv_rimage_1] my_vg gwi-aor--- 256.00m [test-lv_rimage_1_iorig] 100.00
integrity
[...]
```

- 有一个增量的计数器，它计算在每个 RAID 镜像上检测到的不匹配数。查看 `my_vg/test-lv` 下的 `rimage_0` 检测到的数据不匹配：

```
# lvs -o+integritymismatches my_vg/test-lv_rimage_0
LV          VG      Attr      LSize  Origin              Cpy%Sync IntegMismatches
[test-lv_rimage_0] my_vg gwi-aor--- 256.00m [test-lv_rimage_0_iorig] 100.00
0
```

在这个示例中，完整性还没有检测到任何不匹配的数据，因此 `IntegMismatches` 计数器会显示 0 (0)。

- 查看 `/var/log/messages` 日志文件中的数据完整性信息，如下例所示：

**例 9.1. 内核消息日志中的 dm-integrity 不匹配示例**

```
device-mapper: integrity: dm-12: Checksum failed at sector 0x24e7
```

**例 9.2. 来自内核消息日志的 dm-integrity 数据更正示例**

```
md/raid1:mdX: 读取错误修正(8 扇区位在 dm-16 上的 9448)
```

**其他资源**

- 系统中 **lvcreate (8)** 和 **lvraid (7)** 手册页

**9.10. 将 RAID 逻辑卷转换为另一个 RAID 级别**

LVM 支持 RAID 接管，这意味着将 RAID 逻辑卷从一个 RAID 级别转换为另一个 RAID 级别，例如从 RAID 5 转换到 RAID 6。您可以更改 RAID 级别，以增加或减少设备故障的恢复能力。

**流程**

1. 创建 RAID 逻辑卷：

```
# lvcreate --type raid5 -i 3 -L 500M -n my_lv my_vg
Using default stripesize 64.00 KiB.
Rounding size 500.00 MiB (125 extents) up to stripe boundary size 504.00 MiB (126
extents).
Logical volume "my_lv" created.
```

2. 查看 RAID 逻辑卷：

```
# lvs -a -o +devices,segtype
LV          VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy% Sync
Convert Devices                                     Type
my_lv       my_vg       rwi-a-r--- 504.00m                               100.00
my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0),my_lv_rimage_3(0) raid5
[my_lv_rimage_0] my_vg       iwi-aor--- 168.00m
/dev/sda(1)                               linear
```

3. 将 RAID 逻辑卷转换为另一个 RAID 级别：

```
# lvconvert --type raid6 my_vg/my_lv
Using default stripesize 64.00 KiB.
Replaced LV type raid6 (same as raid6_zr) with possible type raid6_ls_6.
Repeat this command to convert to raid6 after an interim conversion has finished.
Are you sure you want to convert raid5 LV my_vg/my_lv to raid6_ls_6 type? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

4. 可选：如果这个命令提示重复转换，请运行：

```
# lvconvert --type raid6 my_vg/my_lv
```



## 验证

1. 查看具有转换的 RAID 级别的 RAID 逻辑卷：

```
# lvs -a -o +devices,segtype
LV          VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices                                     Type
my_lv       my_vg          rwi-a-r--- 504.00m                100.00
my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0),my_lv_rimage_3(0),my_lv_rimage_4(0) raid6
[my_lv_rimage_0] my_vg          iwi-aor--- 172.00m
/dev/sda(1)                                     linear
```

## 其他资源

- 系统中 **lvconvert (8)** 和 **lvraid (8)** 手册页

## 9.11. 将线性设备转换为 RAID 逻辑卷

您可以将现有的线性逻辑卷转换为 RAID 逻辑卷。要执行此操作，请使用 **lvconvert** 命令的 **--type** 参数。

RAID 逻辑卷由元数据和数据子卷对组成。当您线性设备转换为 RAID1 阵列时，它会创建一个新的元数据子卷，并将其与线性卷所在的同一物理卷中的原始逻辑卷相关联。其他镜像添加到 metadata/data 子卷对中。如果无法将与原始逻辑卷配对的元数据镜像放在同一个物理卷上，则 **lvconvert** 将失败。

## 流程

1. 查看需要转换的逻辑卷设备：

```
# lvs -a -o name,copy_percent,devices my_vg
LV      Copy%  Devices
my_lv    /dev/sde1(0)
```

2. 将线性逻辑卷转换为 RAID 设备。以下命令将卷组 `my_vg` 中的线性逻辑卷 `my_lv` 转换为双向 RAID1 阵列：

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
Are you sure you want to convert linear LV my_vg/my_lv to raid1 with 2 images enhancing
resilience? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

## 验证

- 确定逻辑卷是否转换为 RAID 设备：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       6.25   my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0] /dev/sde1(0)
[my_lv_rimage_1] /dev/sdf1(1)
[my_lv_rmeta_0]  /dev/sde1(256)
[my_lv_rmeta_1]  /dev/sdf1(0)
```

## 其他资源

- 系统上的 **lvconvert** (8) 手册页

## 9.12. 将 LVM RAID1 逻辑卷转换为 LVM 线性逻辑卷

您可以将现有的 RAID1 LVM 逻辑卷转换为 LVM 线性逻辑卷。要执行此操作，请使用 **lvconvert** 命令并指定 **-m0** 参数。这会删除所有 RAID 数据子卷以及构成 RAID 阵列的所有 RAID 元数据子卷，保留顶层 RAID1 镜像作为线性逻辑卷。

### 流程

1. 显示现有 LVM RAID1 逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
```

2. 将现有的 RAID1 LVM 逻辑卷转换为 LVM 线性逻辑卷。以下命令将 LVM RAID1 逻辑卷 *my\_vg/my\_lv* 转换为 LVM 线性设备：

```
# lvconvert -m0 my_vg/my_lv
Are you sure you want to convert raid1 LV my_vg/my_lv to type linear losing all resilience?
[y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

当您 将 LVM RAID1 逻辑卷转换成 LVM 线性卷，您可以指定要删除的物理卷。在以下示例中，**lvconvert** 命令指定要删除 */dev/sde1*，保留 */dev/sdf1* 作为组成线性设备的物理卷：

```
# lvconvert -m0 my_vg/my_lv /dev/sde1
```

### 验证

- 验证 RAID1 逻辑卷是否转换为 LVM 线性设备：

```
# lvs -a -o name,copy_percent,devices my_vg
LV  Copy%  Devices
my_lv  /dev/sdf1(1)
```

## 其他资源

- 系统上的 **lvconvert** (8) 手册页

## 9.13. 将镜像 LVM 设备转换为 RAID1 逻辑卷

您可以将片段类型为 *mirror* 的现有镜像 LVM 设备转换为 RAID1 LVM 设备。要执行此操作，请使用带有 **--type raid1** 参数的 **lvconvert** 命令。这会将名为 **mimage** 的镜像子卷重命名为名为 **rimage** 的 RAID 子卷。

另外，它还会删除镜像日志，并为同一物理卷上的数据子卷创建名为 **rmeta** 的元数据子卷，来作为响应的数据子卷。

## 流程

1. 查看镜像逻辑卷 *my\_vg/my\_lv* 的布局：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       15.20  my_lv_mimage_0(0),my_lv_mimage_1(0)
[my_lv_mimage_0]  /dev/sde1(0)
[my_lv_mimage_1]  /dev/sdf1(0)
[my_lv_mlog]      /dev/sdd1(0)
```

2. 将镜像逻辑卷 *my\_vg/my\_lv* 转换为 RAID1 逻辑卷：

```
# lvconvert --type raid1 my_vg/my_lv
Are you sure you want to convert mirror LV my_vg/my_lv to raid1 type? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

## 验证

- 验证镜像逻辑卷是否转换为 RAID1 逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(0)
[my_lv_rimage_1]  /dev/sdf1(0)
[my_lv_rmeta_0]   /dev/sde1(125)
[my_lv_rmeta_1]   /dev/sdf1(125)
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 9.14. 更改现有 RAID1 设备中的镜像数

您可以更改现有 RAID1 阵列中的镜像数量，类似于更改 LVM 镜像实现中的镜像数量。

当您使用 **lvconvert** 命令将镜像添加到 RAID1 逻辑卷时，您可以执行以下操作：

- 指定生成的设备的镜像总数，
- 要添加到该设备的镜像数量，以及
- 可以指定新元数据/数据镜像对所在的物理卷。

## 流程

1. 显示 LVM 设备 *my\_vg/my\_lv*，它是一个双向 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
```

```

LV          Copy% Devices
my_lv       6.25  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0] /dev/sde1(0)
[my_lv_rimage_1] /dev/sdf1(1)
[my_lv_rmeta_0]  /dev/sde1(256)
[my_lv_rmeta_1]  /dev/sdf1(0)

```

元数据子卷（名为 **rmeta**）始终与它们的数据子卷 **rimage** 存在于同一物理设备上。元数据/数据子卷对不会与 RAID 阵列中另一元数据/数据子卷对创建在同一物理卷上（除非您指定了 **--alloc anywhere**）。

2. 将双向 RAID1 逻辑卷 *my\_vg/my\_lv* 转换为三向 RAID1 逻辑卷：

```

# lvconvert -m 2 my_vg/my_lv
Are you sure you want to convert raid1 LV my_vg/my_lv to 3 images enhancing resilience?
[y/n]: y
Logical volume my_vg/my_lv successfully converted.

```

以下是更改现有 RAID1 设备中的镜像数的几个示例：

- 您还可以在 RAID 中添加镜像时要使用的物理卷。以下命令通过指定用于阵列的物理卷 */dev/sdd1*，将双向 RAID1 逻辑卷 *my\_vg/my\_lv* 转换为三向 RAID1 逻辑卷：

```
# lvconvert -m 2 my_vg/my_lv /dev/sdd1
```

- 将三向 RAID1 逻辑卷转换成双向 RAID1 逻辑卷：

```

# lvconvert -m1 my_vg/my_lv
Are you sure you want to convert raid1 LV my_vg/my_lv to 2 images reducing resilience?
[y/n]: y
Logical volume my_vg/my_lv successfully converted.

```

- 通过指定物理卷 */dev/sde1*，其中包含要删除的镜像，将三向 RAID1 逻辑卷转换成双向 RAID1 逻辑卷：

```
# lvconvert -m1 my_vg/my_lv /dev/sde1
```

另外，当您删除镜像及其关联的元数据子卷时，任何高数字镜像都会被切换以填充插槽。如果您从包含 **lv\_rimage\_0**、**lv\_rimage\_1** 和 **lv\_rimage\_2** 的三向 RAID1 阵列中删除 **lv\_rimage\_1**，则会产生一个由 **lv\_rimage\_0** 和 **lv\_rimage\_1** 组成的 RAID1 阵列。子卷 **lv\_rimage\_2** 将重命名并接管空插槽，成为 **lv\_rimage\_1**。

## 验证

- 在更改现有 RAID1 设备中的镜像数后查看 RAID1 设备：

```

# lvs -a -o name,copy_percent,devices my_vg
LV Cpy%Sync Devices
my_lv 100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sdd1(1)
[my_lv_rimage_1] /dev/sde1(1)
[my_lv_rimage_2] /dev/sdf1(1)

```

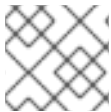
```
[my_lv_rmeta_0] /dev/sdd1(0)
[my_lv_rmeta_1] /dev/sde1(0)
[my_lv_rmeta_2] /dev/sdf1(0)
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 9.15. 将 RAID 镜像分离为一个独立的逻辑卷

您可以分离 RAID 逻辑卷的镜像形成新的逻辑卷。和您从现有 RAID1 逻辑卷中删除 RAID 镜像一样，当您从设备的中间部分删除 RAID 数据子卷（及其关联的元数据子卷）时，会使用数字高的镜像来填充空的位置。因此，构成 RAID 阵列的逻辑卷中的索引号会是一个意外的整数序列。



### 注意

如果 RAID1 阵列还没有同步，您就无法分离 RAID 镜像。

## 流程

1. 显示 LVM 设备 *my\_vg/my\_lv*，它是一个双向 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       12.00  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] /dev/sdf1(1)
[my_lv_rmeta_0] /dev/sde1(0)
[my_lv_rmeta_1] /dev/sdf1(0)
```

2. 将 RAID 镜像分成一个单独的逻辑卷：

- 以下示例将双向 RAID1 逻辑卷 *my\_lv* 分成两个线性逻辑卷 *my\_lv* 和 *new*：

```
# lvconvert --splitmirror 1 -n new my_vg/my_lv
Are you sure you want to split raid1 LV my_vg/my_lv losing all resilience? [y/n]: y
```

- 以下示例将一个三向 RAID1 逻辑卷 *my\_lv* 分成一个双向 RAID1 逻辑卷 *my\_lv* 以及一个线性逻辑卷 *new*：

```
# lvconvert --splitmirror 1 -n new my_vg/my_lv
```

## 验证

- 在分离 RAID 逻辑卷的镜像后查看逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV  Copy%  Devices
my_lv /dev/sde1(1)
new  /dev/sdf1(1)
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 9.16. 分割和合并 RAID 镜像

您可以使用 **lvconvert** 命令的 **--trackchanges** 参数（使用 **--trackchanges** 参数和 **lvconvert** 命令的 **--splitmirrors** 参数），临时分离 RAID1 阵列的镜像以进行只读使用。这可让您以后将镜像合并到阵列中，同时只重新同步那些自镜像被分割后更改的阵列的部分。

当您使用 **--trackchanges** 参数分离 RAID 镜像时，您可以指定要分离的镜像，但您无法更改要分离的卷名称。另外，得到的卷有以下限制：

- 创建的新卷为只读。
- 不能调整新卷的大小。
- 不能重命名剩余的数组。
- 不能调整剩余的数组大小。
- 您可以独立激活新卷和剩余的阵列。

您可以合并分离的镜像。当您合并镜像时，只有自镜像分割后更改的阵列部分会被重新同步。

### 流程

1. 创建 RAID 逻辑卷：

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
Logical volume "my_lv" created
```

2. 可选：查看创建的 RAID 逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdc1(1)
[my_lv_rimage_2]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdc1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
```

3. 从创建的 RAID 逻辑卷中分割镜像，并跟踪对剩余的阵列的更改：

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
my_lv_rimage_2 split from my_lv for read-only purposes.
Use 'lvconvert --merge my_vg/my_lv_rimage_2' to merge back into my_lv
```

4. 可选：在分割镜像后查看逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sdc1(1)
```

```
[my_lv_rimage_1]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sdc1(0)
[my_lv_rmeta_1]     /dev/sdd1(0)
```

5. 将卷合并回阵列中：

```
# lvconvert --merge my_vg/my_lv_rimage_1
my_vg/my_lv_rimage_1 successfully merged back into my_vg/my_lv
```

## 验证

- 查看合并的逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]    /dev/sdc1(1)
[my_lv_rimage_1]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sdc1(0)
[my_lv_rmeta_1]     /dev/sdd1(0)
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 9.17. 将 RAID 失败策略设置为 ALLOCATE

您可以在 `/etc/lvm/lvm.conf` 文件中将 **raid\_fault\_policy** 字段设置为 **allocate** 参数。使用这个首选项，系统会尝试使用卷组中的备用设备替换失败的设备。如果没有备用设备，系统日志会包含此信息。

## 流程

1. 查看 RAID 逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg

LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]    /dev/sdb1(1)
[my_lv_rimage_1]    /dev/sdc1(1)
[my_lv_rimage_2]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sdb1(0)
[my_lv_rmeta_1]     /dev/sdc1(0)
[my_lv_rmeta_2]     /dev/sdd1(0)
```

2. 如果 `/dev/sdb` 设备失败，请查看 RAID 逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg

/dev/sdb: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
```

```
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
```

```
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]    [unknown](1)
[my_lv_rimage_1]    /dev/sdc1(1)
[...]
```

如果 `/dev/sdb` 设备失败，您还可以查看系统日志中的错误消息。

- 在 `lvm.conf` 文件中，将 `raid_fault_policy` 字段设置为 `allocate`：

```
# vi /etc/lvm/lvm.conf
raid_fault_policy = "allocate"
```



### 注意

如果将 `raid_fault_policy` 设置为 `allocate`，但没有备用设备，则分配会失败，逻辑卷保留原样。如果分配失败，您可以使用 `lvconvert --repair` 命令修复和替换失败的设备。如需更多信息，请参阅 [在逻辑卷中替换失败的 RAID 设备](#)。

## 验证

- 验证失败的设备现在是否被卷组中的新设备替换了：

```
# lvs -a -o name,copy_percent,devices my_vg
Couldn't find device with uuid 3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
LV          Copy%  Devices
lv          100.00 lv_rimage_0(0),lv_rimage_1(0),lv_rimage_2(0)
[lv_rimage_0]    /dev/sdh1(1)
[lv_rimage_1]    /dev/sdc1(1)
[lv_rimage_2]    /dev/sdd1(1)
[lv_rmeta_0]     /dev/sdh1(0)
[lv_rmeta_1]     /dev/sdc1(0)
[lv_rmeta_2]     /dev/sdd1(0)
```



### 注意

即使失败的设备现在被替换了，但显示仍然表示 LVM 没有找到失败的设备，因为该设备还没有从卷组中删除。您可以通过执行 `vgreduce --removemissing my_vg` 命令从卷组中删除失败的设备。

## 其他资源

- 在您的系统中的 `lvm.conf (5)` 手册页

## 9.18. 将 RAID 失败策略设置为 WARN

您可以在 `lvm.conf` 文件中将 `raid_fault_policy` 字段设置为 `warn` 参数。有了这个首选项，系统会在系统日志中添加了一条指示失败设备的警告。根据警告，您可以确定后续步骤。

默认情况下，`lvm.conf` 中 `raid_fault_policy` 字段的值是 `warn`。



## 流程

1. 查看 RAID 逻辑卷：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdc1(1)
[my_lv_rimage_2]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdc1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
```

2. 在 `lvm.conf` 文件中将 `raid_fault_policy` 字段设置为 `warn`：

```
# vi /etc/lvm/lvm.conf
# This configuration option has an automatic default value.
raid_fault_policy = "warn"
```

3. 如果 `/dev/sdb` 设备失败，请查看系统日志以显示错误消息：

```
# grep lvm /var/log/messages

Apr 14 18:48:59 virt-506 kernel: sd 25:0:0:0: rejecting I/O to offline device
Apr 14 18:48:59 virt-506 kernel: I/O error, dev sdb, sector 8200 op 0x1:(WRITE) flags
0x20800 phys_seg 0 prio class 2
[...]
Apr 14 18:48:59 virt-506 dmeventd[91060]: WARNING: VG my_vg is missing PV 9R2TVV-
bwfn-Bdyj-Gucu-1p4F-qJ2Q-82kCAF (last written to /dev/sdb).
Apr 14 18:48:59 virt-506 dmeventd[91060]: WARNING: Couldn't find device with uuid
9R2TVV-bwfn-Bdyj-Gucu-1p4F-qJ2Q-82kCAF.
Apr 14 18:48:59 virt-506 dmeventd[91060]: Use 'lvconvert --repair my_vg/ly_lv' to replace
failed device.
```

如果 `/dev/sdb` 设备失败，系统日志会显示错误消息。在这种情况下，LVM 将不会自动尝试通过替换其中一个镜像修复 RAID 设备。如果设备失败，您可以使用 **lvconvert** 命令的 **--repair** 参数替换该设备。如需更多信息，请参阅 [在逻辑卷中替换失败的 RAID 设备](#)。

## 其他资源

- 在您的系统中的 **lvm.conf (5)** 手册页

## 9.19. 替换正常工作的 RAID 设备

您可以使用 **lvconvert** 命令的 **--replace** 参数替换逻辑卷中正常工作的 RAID 设备。

**警告**

如果 RAID 设备失败，以下命令无法工作。

**先决条件**

- RAID 设备没有失败。

**流程**

1. 创建 RAID1 阵列：

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
Logical volume "my_lv" created
```

2. 检查创建的 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sdb1(1)
[my_lv_rimage_1] /dev/sdb2(1)
[my_lv_rimage_2] /dev/sdc1(1)
[my_lv_rmeta_0]  /dev/sdb1(0)
[my_lv_rmeta_1] /dev/sdb2(0)
[my_lv_rmeta_2] /dev/sdc1(0)
```

3. 根据您的要求，使用以下任一方法替换 RAID 设备：

- a. 通过指定要替换的物理卷来替换 RAID1 设备：

```
# lvconvert --replace /dev/sdb2 my_vg/my_lv
```

- b. 通过指定要用于替换的物理卷来替换 RAID1 设备：

```
# lvconvert --replace /dev/sdb1 my_vg/my_lv /dev/sdd1
```

- c. 通过指定多个 replace 参数来一次替换多个 RAID 设备：

```
# lvconvert --replace /dev/sdb1 --replace /dev/sdc1 my_vg/my_lv
```

**验证**

1. 在指定要替换的物理卷后检查 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       37.50 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sdb1(1)
```

```
[my_lv_rimage_1]    /dev/sdc2(1)
[my_lv_rimage_2]    /dev/sdc1(1)
[my_lv_rmeta_0]     /dev/sdb1(0)
[my_lv_rmeta_1]     /dev/sdc2(0)
[my_lv_rmeta_2]     /dev/sdc1(0)
```

2. 在指定要用于替换的物理卷后检查 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       28.00  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdd1(0)
```

3. 一次替换多个 RAID 设备后检查 RAID1 阵列：

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       60.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdd1(1)
[my_lv_rimage_2]  /dev/sde1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdd1(0)
[my_lv_rmeta_2]   /dev/sde1(0)
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 9.20. 在逻辑卷中替换失败的 RAID 设备

RAID 与传统的 LVM 镜像不同。如果是 LVM 镜像，请删除失败的设备。否则，当 RAID 阵列继续使用失败的设备运行时，镜像逻辑卷将挂起。对于 RAID1 以外的 RAID 级别，删除设备意味着转换到较低 RAID 级别，例如从 RAID6 转换到 RAID5，或者从 RAID4 或 RAID5 转换到 RAID0。

您可以使用 **lvconvert** 命令的 **--repair** 参数替换 RAID 逻辑卷中作为物理卷的故障设备，而不是删除失败的设备并分配一个替换品。

## 先决条件

- 卷组包含一个物理卷，它有足够的可用容量替换失败的设备。  
如果卷组中没有有足够可用扩展的物理卷，请使用 **vgextend** 程序添加一个新的、足够大的物理卷。

## 流程

1. 查看 RAID 逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
LV          Cpy%Sync Devices
my_lv       100.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
```

```
[my_lv_rimage_0]    /dev/sde1(1)
[my_lv_rimage_1]    /dev/sdc1(1)
[my_lv_rimage_2]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sde1(0)
[my_lv_rmeta_1]     /dev/sdc1(0)
[my_lv_rmeta_2]     /dev/sdd1(0)
```

2. 在 `/dev/sdc` 设备失败后查看 RAID 逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
/dev/sdc: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
LV          Cpy%Sync Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]    /dev/sde1(1)
[my_lv_rimage_1]    [unknown](1)
[my_lv_rimage_2]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sde1(0)
[my_lv_rmeta_1]     [unknown](0)
[my_lv_rmeta_2]     /dev/sdd1(0)
```

3. 替换失败的设备：

```
# lvconvert --repair my_vg/my_lv
/dev/sdc: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
Faulty devices in my_vg/my_lv successfully replaced.
```

4. 可选：手动指定替换失败设备的物理卷：

```
# lvconvert --repair my_vg/my_lv replacement_pv
```

5. 使用替换检查逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg

/dev/sdc: open failed: No such device or address
/dev/sdc1: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
LV          Cpy%Sync Devices
my_lv       43.79  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]    /dev/sde1(1)
[my_lv_rimage_1]    /dev/sdb1(1)
[my_lv_rimage_2]    /dev/sdd1(1)
[my_lv_rmeta_0]     /dev/sde1(0)
[my_lv_rmeta_1]     /dev/sdb1(0)
```

```
[my_lv_rmeta_2] /dev/sdd1(0)
```

在您从卷组中删除失败的设备前，LVM 工具仍然指示 LVM 无法找到失败的设备。

6. 从卷组中删除失败的设备：

```
# vgreduce --removemissing my_vg
```

## 验证

1. 删除失败的设备后查看可用的物理卷：

```
# pvscan
PV /dev/sde1 VG rhel_virt-506 lvm2 [<7.00 GiB / 0 free]
PV /dev/sdb1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
PV /dev/sdd1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
PV /dev/sdd1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
```

2. 替换失败的设备后检查逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] /dev/sdb1(1)
[my_lv_rimage_2] /dev/sdd1(1)
[my_lv_rmeta_0] /dev/sde1(0)
[my_lv_rmeta_1] /dev/sdb1(0)
[my_lv_rmeta_2] /dev/sdd1(0)
```

## 其他资源

- 系统中 **lvconvert (8)** 和 **vgreduce (8)** 手册页

## 9.21. 检查 RAID 逻辑卷中数据的一致性

LVM 提供对 RAID 逻辑卷的清理支持。RAID 清理是读取阵列中的所有数据和奇偶校验块的过程，并检查它们是否是分配的。**lvchange --syncaction repair** 命令对阵列启动后台同步操作。

## 流程

1. 可选：通过设置以下选项之一来控制 RAID 逻辑卷初始化的速度：

- **--maxrecoveryrate Rate[bBsSkKmMgG]** 为 RAID 逻辑卷设置最大恢复率，使其不会加快 I/O 操作。
- **--minrecoveryrate Rate[bBsSkKmMgG]** 设置 RAID 逻辑卷的最小恢复率，以确保 sync 操作的 I/O 达到最小吞吐量，即使存在大量小 I/O

```
# lvchange --maxrecoveryrate 4K my_vg/my_lv
Logical volume _my_vg/my_lv_changed.
```

使用恢复率值替换 4K，它是阵列中每个设备的每秒数量。如果没有后缀，选项会假定为 kiB/每秒/每个设备。

```
# lvchange --syncaction repair my_vg/my_lv
```

当您执行 RAID 清理操作时，**sync** 操作所需的后台 I/O 可能会排挤 LVM 设备的其它 I/O，如对卷组元数据的更新。这可能导致其它 LVM 操作速度下降。



### 注意

您还可以在创建 RAID 设备时使用这些最大值和最小 I/O 速率。例如，**lvcreate --type raid10 -i 2 -m 1 -L 10G --maxrecoveryrate 128 -n my\_lv my\_vg** 创建一个双向 RAID10 阵列 **my\_lv**，它位于卷组 **my\_vg** 中，其大小为 10G，最大恢复率为 128 kiB/sec/device。

2. 显示阵列中未修复的数量的差异，没有修复它们：

```
# lvchange --syncaction check my_vg/my_lv
```

此命令对阵列启动后台同步操作。

3. 可选：查看 **var/log/syslog** 文件中的内核消息。
4. 修正阵列中的差异：

```
# lvchange --syncaction repair my_vg/my_lv
```

这个命令修复或者替换 RAID 逻辑卷中失败的设备。您可以在执行此命令后，在 **var/log/syslog** 文件查看内核消息。

## 验证

1. 显示有关清理操作的信息：

```
# lvs -o +raid_sync_action,raid_mismatch_count my_vg/my_lv
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
SyncAction Mismatches
my_lv my_vg rwi-a-r--- 500.00m 100.00 idle 0
```

## 其他资源

- 系统中 **lvchange** (8) 和 **lvraid** (7) 手册页
- [最小和最大 I/O 速率选项](#)

## 9.22. RAID1 逻辑卷上的 I/O 操作

您可以使用 **lvchange** 命令的 **--writemostly** 和 **--writebehind** 参数控制 RAID1 逻辑卷中设备的 I/O 操作。以下是使用这些参数的格式：

```
--[raid]writemostly PhysicalVolume[:{t|y|n}]
```

将 RAID1 逻辑卷中的一个设备标记为 **write-mostly**，并避免对这些驱动器的所有读操作（除非有必要）。设置此参数会使驱动器中的 I/O 操作数量保持最小。

使用 **lvchange --writemostly /dev/sdb my\_vg/my\_lv** 命令来设置此参数。

您可以使用以下方法设置 **writemostly** 属性：

**:y**

默认情况下，对于逻辑卷中指定的物理卷，**writemostly** 属性的值是 **yes**。

**:n**

要删除 **writemostly** 标志，请将 **:n** 附加到物理卷上。

**:t**

要切换 **writemostly** 属性的值，请指定 **--writemostly** 参数。

您可以在单个命令中多次使用此参数，例如 **lvchange --writemostly /dev/sdd1:n --writemostly /dev/sdb1:t --writemostly /dev/sdc1:y my\_vg/my\_lv**。因此，可以一次为逻辑卷中的 **所有物理卷** 切换写属性。

### **--[raid]writebehind IOCCount**

将待处理写的最大数量标记为 **writemostly**。这些是适用于 RAID1 逻辑卷中设备的写操作的数量。超过这个参数值后，在 RAID 阵列通知所有写操作完成前，对组成设备的所有写操作都会同步完成。

您可以使用 **lvchange --writebehind 100 my\_vg/my\_lv** 命令来设置此参数。将 **writemostly** 属性的值设置为零来清除首选项。使用这个设置，系统可以任意选择值。

## 9.23. 重塑 RAID 卷

RAID 重塑意味着在不更改 RAID 级别的情况下更改 RAID 逻辑卷的属性。您可以更改的一些属性包括 RAID 布局、条带大小和条带数目。

### 流程

1. 创建 RAID 逻辑卷：

```
# lvcreate --type raid5 -i 2 -L 500M -n my_lv my_vg

Using default stripesize 64.00 KiB.
Rounding size 500.00 MiB (125 extents) up to stripe boundary size 504.00 MiB (126
extents).
Logical volume "my_lv" created.
```

2. 查看 RAID 逻辑卷：

```
# lvs -a -o +devices

LV          VG   Attr      LSize   Pool   Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
my_lv       my_vg rwi-a-r--- 504.00m                100.00
my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] my_vg iwi-aor--- 252.00m                /dev/sda(1)
[my_lv_rimage_1] my_vg iwi-aor--- 252.00m                /dev/sdb(1)
[my_lv_rimage_2] my_vg iwi-aor--- 252.00m                /dev/sdc(1)
[my_lv_rmeta_0]  my_vg ewi-aor--- 4.00m                /dev/sda(0)
[my_lv_rmeta_1]  my_vg ewi-aor--- 4.00m                /dev/sdb(0)
[my_lv_rmeta_2]  my_vg ewi-aor--- 4.00m                /dev/sdc(0)
```

3. 可选：查看 RAID 逻辑卷的 **stripes** 镜像和 **stripesize**：

```
# lvs -o stripes my_vg/my_lv
#Str
3
```

```
# lvs -o stripesize my_vg/my_lv
Stripe
64.00k
```

4. 根据您的要求，使用以下方法修改 RAID 逻辑卷的属性：

- a. 修改 RAID 逻辑卷的 **stripes** 镜像：

```
# lvconvert --stripes 3 my_vg/my_lv
Using default stripesize 64.00 KiB.
WARNING: Adding stripes to active logical volume my_vg/my_lv will grow it from 126 to
189 extents!
Run "lvresize -l126 my_vg/my_lv" to shrink it or use the additional capacity.
Are you sure you want to add 1 images to raid5 LV my_vg/my_lv? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

- b. 修改 RAID 逻辑卷的 **stripesize**：

```
# lvconvert --stripesize 128k my_vg/my_lv
Converting stripesize 64.00 KiB of raid5 LV my_vg/my_lv to 128.00 KiB.
Are you sure you want to convert raid5 LV my_vg/my_lv? [y/n]: y
Logical volume my_vg/my_lv successfully converted.
```

- c. 修改 **maxrecoveryrate** 和 **minrecoveryrate** 属性：

```
# lvchange --maxrecoveryrate 4M my_vg/my_lv
Logical volume my_vg/my_lv changed.
```

```
# lvchange --minrecoveryrate 1M my_vg/my_lv
Logical volume my_vg/my_lv changed.
```

- d. 修改 **syncaction** 属性：

```
# lvchange --syncaction check my_vg/my_lv
```

- e. 修改 **writemostly** 和 **writebehind** 属性：

```
# lvchange --writemostly /dev/sdb my_vg/my_lv
Logical volume my_vg/my_lv changed.
```

```
# lvchange --writebehind 100 my_vg/my_lv
Logical volume my_vg/my_lv changed.
```

## 验证

1. 查看 RAID 逻辑卷的 **stripes** 镜像和 **stripesize**：



```
# lvs -o stripes my_vg/my_lv
#Str
4
```

```
# lvs -o stripesize my_vg/my_lv
Stripe
128.00k
```

2. 修改 **maxrecoveryrate** 属性后，查看 RAID 逻辑卷：

```
# lvs -a -o +raid_max_recovery_rate
LV          VG      Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
Convert MaxSync
my_lv        my_vg   rwi-a-r--- 10.00g                100.00    4096
[my_lv_rimage_0] my_vg iwi-aor--- 10.00g
[...]
```

3. 修改 **minrecoveryrate** 属性后，查看 RAID 逻辑卷：

```
# lvs -a -o +raid_min_recovery_rate
LV          VG      Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
Convert MinSync
my_lv        my_vg   rwi-a-r--- 10.00g                100.00    1024
[my_lv_rimage_0] my_vg iwi-aor--- 10.00g
[...]
```

4. 修改 **syncaction** 属性后，查看 RAID 逻辑卷：

```
# lvs -a
LV          VG      Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
Convert
my_lv        my_vg   rwi-a-r--- 10.00g                2.66
[my_lv_rimage_0] my_vg iwi-aor--- 10.00g
[...]
```

## 其他资源

- 系统中 **lvconvert (8)** 和 **lvraid (8)** 手册页

## 9.24. 在 RAID 逻辑卷中更改区域大小

当您创建 RAID 逻辑卷时，`/etc/lvm/lvm.conf` 文件中的 **raid\_region\_size** 参数代表 RAID 逻辑卷的区域大小。创建 RAID 逻辑卷后，您可以更改卷的区域大小。此参数定义跟踪脏或干净状态的粒度。位图中的脏位定义 RAID 卷脏关闭（例如系统故障）后要同步的工作集。

如果将 **raid\_region\_size** 设置为更高的值，它会减小位图的大小以及拥塞。但它会影响重新同步区域期间的 **write** 操作，因为写入 RAID 会延迟，直到同步区域完成为止。

## 流程

1. 创建 RAID 逻辑卷：

```
# lvcreate --type raid1 -m 1 -L 10G test
Logical volume "lvol0" created.
```

## 2. 查看 RAID 逻辑卷：

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

| LV  | VG   | Attr       | LSize  | Pool | Origin | Data% | Meta% | Move   | Log | Cpy% | Sync | Convert      |
|---|------|------------|--------|------|--------|-------|-------|--------|-----|------|------|--------------|
| Devices                                   |      |            | Region |      |        |       |       |        |     |      |      |              |
| lvol0                                     | test | rwi-a-r--- | 10.00g |      |        |       |       | 100.00 |     |      |      |              |
| lvol0_rimage_0(0),lvol0_rimage_1(0) 2.00m |      |            |        |      |        |       |       |        |     |      |      |              |
| [lvol0_rimage_0]                          | test | iwi-aor--- | 10.00g |      |        |       |       |        |     |      |      | /dev/sde1(1) |
| 0   |      |            |        |      |        |       |       |        |     |      |      |              |
| [lvol0_rimage_1]                          | test | iwi-aor--- | 10.00g |      |        |       |       |        |     |      |      | /dev/sdf1(1) |
| 0   |      |            |        |      |        |       |       |        |     |      |      |              |
| [lvol0_rmeta_0]                           | test | ewi-aor--- | 4.00m  |      |        |       |       |        |     |      |      | /dev/sde1(0) |
| 0   |      |            |        |      |        |       |       |        |     |      |      |              |
| [lvol0_rmeta_1]                           | test | ewi-aor--- | 4.00m  |      |        |       |       |        |     |      |      |              |

**Region** 列表示 `raid_region_size` 参数的值。

## 3. 可选：查看 **raid\_region\_size** 参数的值：

```
# cat /etc/lvm/lvm.conf | grep raid_region_size

# Configuration option activation/raid_region_size.
# raid_region_size = 2048
```

## 4. 更改 RAID 逻辑卷的区域大小：

```
# lvconvert -R 4096K my_vg/my_lv

Do you really want to change the region_size 512.00 KiB of LV my_vg/my_lv to 4.00 MiB?
[y/n]: y
Changed region size on RAID LV my_vg/my_lv to 4.00 MiB.
```

## 5. 重新同步 RAID 逻辑卷：

```
# lvchange --resync my_vg/my_lv

Do you really want to deactivate logical volume my_vg/my_lv to resync it? [y/n]: y
```

## 验证

### 1. 查看 RAID 逻辑卷：

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

| LV  | VG   | Attr       | LSize  | Pool | Origin | Data% | Meta% | Move | Log | Cpy% | Sync | Convert      |
|---|------|------------|--------|------|--------|-------|-------|------|-----|------|------|--------------|
| Devices                                   |      |            | Region |      |        |       |       |      |     |      |      |              |
| lvol0                                     | test | rwi-a-r--- | 10.00g |      |        |       |       | 6.25 |     |      |      |              |
| lvol0_rimage_0(0),lvol0_rimage_1(0) 4.00m |      |            |        |      |        |       |       |      |     |      |      |              |
| [lvol0_rimage_0]                          | test | iwi-aor--- | 10.00g |      |        |       |       |      |     |      |      | /dev/sde1(1) |
| 0   |      |            |        |      |        |       |       |      |     |      |      |              |

```
[lvol0_rimage_1] test iwi-aor--- 10.00g      /dev/sdf1(1)
0
[lvol0_rmeta_0] test ewi-aor--- 4.00m        /dev/sde1(0)
0
```

**Region** 列表示 **raid\_region\_size** 参数的更改值。

2. 查看 **lvm.conf** 文件中 **raid\_region\_size** 参数的值：

```
# cat /etc/lvm/lvm.conf | grep raid_region_size

# Configuration option activation/raid_region_size.
# raid_region_size = 4096
```

## 其他资源

- 系统上的 **lvconvert (8)** 手册页

## 第 10 章 限制 LVM 设备可见性和用法

您可以通过控制 LVM 可扫描的设备来限制逻辑卷管理器(LVM)可用的设备。

要调整 LVM 设备扫描的配置，请编辑 `/etc/lvm/lvm.conf` 文件中的 LVM 设备过滤器设置。`lvm.conf` 文件中的过滤器由一系列简单的正则表达式组成。系统会将这些表达式应用于 `/dev` 目录中的每个设备名称，以确定是否接受或拒绝每个检测到的块设备。

### 10.1. LVM 过滤的持久性标识符

传统的 Linux 设备名称（如 `/dev/sda`）可能会在系统修改和重启过程中改变。持久性命名属性(PNA),如全球识别符(WWID)、通用唯一标识符(UUID)) 和路径名称都基于存储设备的唯一特征，并对硬件配置中的更改有弹性。这使得它们在系统重启后更稳定和可预测。

在 LVM 过滤中持久设备标识符的实现可增强 LVM 配置的稳定性和可靠性。它还降低了与设备名称的动态性质关联的系统引导失败的风险。

#### 其他资源

- [持久性命名属性](#)
- [当本地磁盘名称不是持久的时，如何配置 lvm 过滤器？](#)（红帽知识库）

### 10.2. LVM 设备过滤器

Logical Volume Manager (LVM)设备过滤器是设备名称模式列表。您可以用它来指定系统可以评估设备的一组强制标准，并将其视为可以有效地与 LVM 一起使用。LVM 设备过滤器可让您控制 LVM 使用哪个设备。这有助于防止意外数据丢失或未授权访问存储设备。

#### 10.2.1. LVM 设备过滤器模式特征

LVM 设备过滤器的模式采用正则表达式的形式。正则表达式使用一个字符分割，前面为 **a** 表示接收，为 **r** 表示拒绝。匹配设备的列表中的第一个正则表达式决定了 LVM 接受还是拒绝（忽略）一个特定设备。然后，LVM 会在匹配设备路径的列表中查找初始正则表达式。LVM 使用这个正则表达式来确定是否应该使用 **a** 结果批准设备，或使用 **r** 结果拒绝设备。

如果单个设备有多个路径名称，LVM 会根据它们列出的顺序访问这些路径名称。在任何 **r** 模式之前，如果至少有一个路径名称与一个 **a** 模式匹配，则 LVM 会批准设备。但是，如果在发现 **a** 模式之前所有路径名称都与 **r** 模式一致，则设备将被拒绝。

与模式不匹配的路径名称不会影响设备的批准状态。如果没有与设备的模式相对应的路径名，LVM 仍然会批准设备。

对于系统上的每个设备，`udev` 规则会生成多个符号链接。目录包含符号链接，如 `/dev/disk/by-id/`、`/dev/disk/by-uuid/`、`/dev/disk/by-path/`，以确保系统上的每个设备都可以通过多个路径名称访问。

要拒绝过滤器中的设备，与该特定设备关联的所有路径名称必须与对应的拒绝 **r** 表达式匹配。但是，识别要拒绝的所有可能路径名称可能会有一定难度。因此，这就是使用一系列特定的 **a** 表达式，后跟一个拒绝所有其它路径的 `r|.*` 表达式来创建专门接受某些路径并拒绝所有其他路径的过滤的原因。

在过滤器中定义特定设备时，为该设备使用符号链接名称，而不是内核名称。设备的内核名称可能会改变，如 `/dev/sda`，而某些符号链接名称不会改变，如 `/dev/disk/by-id/wwn-*`。

默认设备过滤器接受所有连接到系统的设备。理想的用户配置的设备过滤器接受一个或多个模式，并拒绝所有其他模式。例如，模式列表以 `r|.*/` 结尾。

您可以在 `lvm.conf` 文件的 `devices/filter` 和 `devices/global_filter` 配置字段中找到 LVM 设备过滤器配置。`devices/filter` 和 `devices/global_filter` 配置字段是等同的。

### 其他资源

- 在您的系统中的 [lvm.conf \(5\)](#) 手册页

### 10.2.2. LVM 设备过滤器配置示例

以下示例显示过滤器配置来控制 LVM 扫描并稍后使用的设备。要在 `lvm.conf` 文件中配置设备过滤器，请参阅 [应用 LVM 设备过滤器配置](#)



#### 注意

在处理复制或克隆的 PV 时，您可能会遇到重复的物理卷(PV)警告。您可以设置过滤器来解决这个问题。请参阅 [防止重复 PV 警告的 LVM 设备过滤器示例](#) 中的过滤器配置示例。

- 要扫描所有设备，请输入：

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

- 要删除 `cdrom` 设备以避免驱动器中无介质时的延迟，请输入：

```
filter = [ "r!^/dev/cdrom$" ]
```

- 要添加所有 loop 设备，并删除所有其他设备，请输入：

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

- 要添加所有循环和 SCSI 设备，并删除所有其他块设备，请输入：

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

- 要在第一个 SCSI 驱动器上只添加分区 8，并删除所有其他块设备，请输入：

```
filter = [ "a!^/dev/sda8$|", "r|.*/" ]
```

- 要从 WWID 以及所有多路径设备标识的特定设备和中添加所有分区，请输入：

```
filter = [ "a|/dev/disk/by-id/<disk-id>.|", "a|/dev/mapper/mpath.|", "r|.*/" ]
```

命令还会删除任何其他块设备。

### 其他资源

- 在您的系统中的 [lvm.conf \(5\)](#) 手册页
- [应用 LVM 设备过滤器配置](#)

- [防止重复 PV 警告的 LVM 设备过滤器示例](#)

### 10.2.3. 应用 LVM 设备过滤器配置

您可以通过在 **lvm.conf** 配置文件中设置过滤器来控制 LVM 扫描哪个设备。

#### 先决条件

- 您已准备了要使用的设备过滤器模式。

#### 流程

1. 使用以下命令测试设备过滤器模式，而不实际修改 **/etc/lvm/lvm.conf** 文件。以下包括一个过滤器配置示例。

```
# lvs --config 'devices{ filter = [ "a|/dev/emcpower.|", "r|.|" ] }'
```

2. 在 **/etc/lvm/lvm.conf** 文件的配置部分 **devices** 中添加设备过滤器模式：

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

3. 重启时只扫描所需的设备：

```
# dracut --force --verbose
```

这个命令重建 **initramfs** 文件系统，以便 LVM 在重启时只扫描必要的设备。

## 第 11 章 控制 LVM 分配

默认情况下，卷组使用 **normal** 分配策略。这会根据常识性规则分配物理扩展，如不在同一个物理卷上放置并行条带。您可以使用 **vgcreate** 命令的 **--alloc** 参数指定不同的分配策略(**contiguous**、**anywhere** 或 **cling**)。通常，只在特殊情况下需要指定非通常或非标准扩展分配时，才需要不同于 **normal** 的分配策略。

### 11.1. 从指定设备分配扩展

您可以使用命令行末尾的 **device** 参数以及 **lvcreate** 和 **lvconvert** 命令来限制从特定的设备分配。您可以为每个设备指定实际的扩展范围，以提供更多控制。命令通过使用指定的物理卷(PV)作为参数，来只为新逻辑卷(LV)分配扩展。它使用每个 PV 的可用扩展，直到它们耗尽，然后使用列出的下一个 PV 中的扩展。如果所有列出的 PV 上没有足够的空间用于请求的 LV 大小，则命令失败。请注意，命令只从指定的 PV 分配。RAID LV 对单独的 raid 镜像或单独的条带使用顺序 PV。如果 PV 对于整个 raid 镜像不够大，则生成的设备使用不是完全可预测的。

#### 流程

1. 创建卷组(VG)：

```
# vgcreate <vg_name> <PV> ...
```

其中：

- **<vg\_name>** 是 VG 的名称。
- **<PV>** 是 PV。

2. 您可以分配 PV 来创建不同的卷类型，如 **linear** 或 **raid**：

- a. 分配扩展以创建线性卷：

```
# lvcreate -n <lv_name> -L <lv_size> <vg_name> [ <PV> ... ]
```

其中：

- **<lv\_name>** 是 LV 的名称。
- **<lv\_size>** 是 LV 的大小。默认单位是 MB。
- **<vg\_name>** 是 VG 的名称。
- **[ <PV ...> ]** 是 PV。

您可以在命令行上指定其中一个 PV、所有 PV 或 **none**：

- 如果您指定一个 PV，则会从其中为那个 LV 分配扩展。



#### 注意

如果 PV 对于整个 LV 没有足够的空闲扩展，则 **lvcreate** 失败。

- 如果您指定了两个 PV，则将从其中一个 PV 或两者的组合为那个 LV 分配扩展。

- 如果没有指定任何 PV，则将从 VG 中的一个 PV 或者 VG 中所有 PV 的组合分配扩展。



### 注意

在这些情况下，LVM 可能无法使用所有命名的 PV 或可用的 PV。如果第一个 PV 对于整个 LV 有足够的空闲扩展，则可能不会使用其他 PV。但是，如果第一个 PV 没有一组空闲扩展的分配大小，则 LV 可能会从第一个 PV 中分配一部分，从第二个 PV 中分配一部分。

#### 例 11.1. 从一个 PV 中分配扩展

在这个示例中，**lv1** 扩展将从 **sda** 分配。

```
# lvcreate -n lv1 -L1G vg /dev/sda
```

#### 例 11.2. 从两个 PV 中分配扩展

在这个示例中，**lv2** 扩展将从 **sda** 或 **sdb** 或两者的组合中分配。

```
# lvcreate -n lv2 L1G vg /dev/sda /dev/sdb
```

#### 例 11.3. 分配扩展，而不指定 PV

在本例中，**lv3** 扩展将从 VG 中的一个 PV 或 VG 中所有 PV 的组合中分配。

```
# lvcreate -n lv3 -L1G vg
```

或者

- b. 分配扩展以创建 raid 卷：

```
# lvcreate --type <segment_type> -m <mirror_images> -n <lv_name> -L <lv_size>  
<vg_name> [ <PV> ... ]
```

其中：

- **<segment\_type>** 是指定的片段类型（如 **raid5**、**mirror**、**snapshot**）。
- **<mirror\_images>** 使用创建一个带有指定的镜像数的 **raid1** 或镜像 LV。例如，**-m 1** 会产生一个带有两个镜像的 **raid1** LV。
- **<lv\_name>** 是 LV 的名称。
- **<lv\_size>** 是 LV 的大小。默认单位是 MB。
- **<vg\_name>** 是 VG 的名称。
- **<[PV ...]>** 是 PV。



第一个 raid 镜像将从第一个 PV 中分配，第二个 raid 镜像将从第二个 PV 中分配，以此类推。

#### 例 11.4. 从两个 PV 中分配 raid 镜像

在这个示例中，**lv4** 第一个 raid 镜像将从 **sda** 中分配，第二个镜像将从 **sdb** 中分配。

```
# lvcreate --type raid1 -m 1 -n lv4 -L1G vg /dev/sda /dev/sdb
```

#### 例 11.5. 从三个 PV 中分配 raid 镜像

在本例中，**lv5** 第一个 raid 镜像将从 **sda** 中分配，第二个镜像将从 **sdb** 中分配，第三个镜像将从 **sd c** 中分配。

```
# lvcreate --type raid1 -m 2 -n lv5 -L1G vg /dev/sda /dev/sdb /dev/sdc
```

### 其他资源

- 系统中 **lvcreate** (8), **lvconvert** (8), 和 **lvraid** (7) man page

## 11.2. LVM 分配策略

当 LVM 操作必须为一个或多个逻辑卷(LV)分配物理扩展时，分配过程如下：

- 为考虑考虑，生成卷组中未分配的物理扩展的完整集合。如果您在命令行末尾提供了任何范围的物理扩展，则只考虑指定物理卷(PV)上此范围内未分配的物理扩展。
- 每个分配策略依次尝试，从最严格的策略(**contiguous**)开始，以使用 **--alloc** 选项指定的分配策略结束，或者设为特定 LV 或卷组(VG)的默认分配策略。对于每个策略，从需要填充的空 LV 空间的最低编号的逻辑扩展开始工作，根据分配策略施加的限制，来分配尽可能多的空间。如果需要更多空间，LVM 会进入下一个策略。

分配策略的限制如下：

- **contiguous** 策略要求任何逻辑扩展的物理位置紧挨着前一个逻辑扩展的物理位置，但 LV 的第一个逻辑扩展除外。  
当 LV 为条带或镜像时，**contiguous** 分配限制将独立应用于每个需要空间的条带或 raid 镜像。
- **cling** 分配策略要求将用于任何逻辑扩展的 PV 添加到现有 LV 中，该 LV 之前已被该 LV 中至少一个逻辑扩展使用。
- **normal** 分配策略不会选择一个共享同一 PV 的物理扩展，因为逻辑扩展已分配给在那个并行 LV 中相同偏移处的一个并行 LV（即，不同的条带或 raid 镜像）。
- 如果有足够的空闲扩展来满足分配请求，但 **normal** 分配策略将不使用它们，而 **anywhere** 分配策略将使用它们，即使这通过在同一 PV 上放置两个条带降低了性能。

您可以使用 **vgchange** 命令更改分配策略。



## 注意

未来的更新可能会根据定义的分配策略在布局行为中引入代码更改。例如：如果您在命令行中提供两个空物理卷，它们有相同数量的可用物理扩展可用于分配，LVM 当前会以它们列出的顺序处理它们，但不保证在将来的版本中这个行为不会有变化。如果您需要特定 LV 的一个特定布局，请通过 **lvcreate** 和 **lvconvert** 步骤序列构建它，这样应用到每个步骤的分配策略就不会让 LVM 自行决定布局。

## 11.3. 防止在物理卷中分配

您可以使用 **pvchange** 命令防止在一个或多个物理卷的空闲空间上分配物理扩展。如果有磁盘错误或者要删除物理卷，则可能需要这样做。

### 流程

- 使用以下命令禁止在 **device\_name** 上分配物理扩展：

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

您还可以使用 **pvchange** 命令的 **-xy** 参数允许之前被禁止的分配。

### 其他资源

- 系统中 **pvchange (8)** 手册页

## 第 12 章 使用标签对 LVM 对象进行分组

您可以为逻辑卷管理器(LVM)对象分配标签来分组它们。使用这个功能，您可以按照组对 LVM 行为（如激活）进行自动控制。您还可以使用标签而不是 LVM 对象参数。

### 12.1. LVM 对象标签

逻辑卷管理器(LVM)标记同一类型的 LVM 对象。您可以将标签附加到对象，如物理卷、卷组和逻辑卷，以及集群配置中的主机。

为避免歧义，请在每个标签前加上 @。标签可以通过使用拥有该标签的所有对象以及通过该标签在命令行上的位置所期望的类型替换标签来进行扩展。

LVM 标签是最多 1024 个字符的字符串。LVM 标签不能以连字符开头。

有效的标签仅由有限的字符范围组成。允许的字符是 **A-Z a-z 0-9 \_ + . - / = ! : # & amp; .**

只能标记卷组中的对象。如果从卷组中移除了标签，则物理卷会丢失标签；这是因为标签作为卷组元数据的一部分存储，并在物理卷被删除时删除。

您可以将一些命令应用到具有相同标签的所有卷组(VG)、逻辑卷(LV)或物理卷(PV)。当您可以替换 VG、LV 或 PV 名称的标签名称时，给定命令的手册页显示语法，如 **VG|Tag**、**LV|Tag** 或 **PV|Tag**。

### 12.2. 列出 LVM 标签

以下示例演示了如何列出 LVM 标签。

#### 流程

- 使用以下命令列出所有带有 **数据库** 标签的逻辑卷：

```
# lvs @database
```

- 使用以下命令列出当前活跃的主机标签：

```
# lvm tags
```

### 12.3. 向 LVM 对象添加标签

您可以使用 **--addtag** 选项和各种卷管理命令向 LVM 对象添加标签来对它们进行分组。

#### 先决条件

- 已安装 **lvm2** 软件包。

#### 流程

- 要向现有 PV 添加标签，请使用：

```
# pvchange --addtag <@tag> <PV>
```

- 要向现有 VG 添加标签，请使用：

```
# vgchange --addtag <@tag> <VG>
```

- 要在创建过程中向 VG 添加标签，请使用：

```
# vgcreate --addtag <@tag> <VG>
```

- 要向现有 LV 添加标签，请使用：

```
# lvchange --addtag <@tag> <LV>
```

- 要在创建过程中向 LV 添加标签，请使用：

```
# lvcreate --addtag <@tag> ...
```

## 12.4. 从 LVM 对象中删除标签

如果您不再希望保持 LVM 对象分组，您可以使用 **--deltag** 选项和各种卷管理命令从对象中删除标签。

### 先决条件

- **lvm2** 软件包已安装。
- 您已在物理卷(PV)、卷组(VG)或逻辑卷(LV)上创建了标签。

### 流程

- 要从现有 PV 中删除标签，请使用：

```
# pvchange --deltag @tag PV
```

- 要从现有 VG 中删除标签，请使用：

```
# vgchange --deltag @tag VG
```

- 要从现有 LV 中删除标签，请使用：

```
# lvchange --deltag @tag LV
```

## 12.5. 定义 LVM 主机标签

这个步骤描述了如何在集群配置中定义 LVM 主机标签。您可以在 配置文件中定义主机标签。

### 流程

- 在 **tags** 部分中设置 **hosttags = 1**，以使用计算机的主机名自动定义主机标签。  
这样，您可以使用一个通用配置文件，该文件可在所有计算机上复制，使它们拥有文件的相同副本，但其行为会根据主机名在计算机之间有所不同。

对于每个主机标签，如果存在额外的配置文件，则会读取它：**lvm\_hosttag.conf**。如果该文件定义了新标签，则其他配置文件将附加到要读取的文件列表中。

例如，配置文件中以下条目始终定义了 **tag1**，并在主机名是 **host1** 时定义 **tag2**：

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

## 12.6. 使用标签控制逻辑卷激活

这个步骤描述了如何在主机上激活特定逻辑卷的配置文件中指定。

### 流程

例如，以下条目充当激活请求的过滤器（如 **vgchange -ay**），仅激活 **vg1/lvol0** 以及具有该主机上元数据中的 **数据库** 标签的任何逻辑卷或卷组：

```
activation { volume_list = ["vg1/lvol0", "@database" ] }
```

特殊匹配 **@\***，仅当任何元数据标签匹配该机器上的任何主机标签时才使匹配。

另外，请考虑集群中每台机器在配置文件中有以下条目的情况：

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

如果您想仅在主机 **db2** 上激活 **vg1/lvol2**，请执行以下操作：

1. 从集群中的任何主机运行 **lvchange --addtag @db2 vg1/lvol2**。
2. 运行 **lvchange -ay vg1/lvol2**。

这个解决方案涉及将主机名存储在卷组元数据中。

## 第 13 章 LVM 故障排除

您可以使用逻辑卷管理器(LVM)工具来排除 LVM 卷和组群中的各种问题。

### 13.1. 在 LVM 中收集诊断数据

如果 LVM 命令没有按预期工作，您可以使用以下方法收集诊断信息。

#### 流程

- 使用以下方法收集不同类型的诊断数据：
  - 向任何 LVM 命令添加 **-v** 参数，以提高命令输出的详细程度。添加更多的 **v** 会进一步增加输出的详细程度。最多允许 4 个这样的 **v**，例如 **-vvvv**。
  - 在 **/etc/lvm/lvm.conf** 配置文件的 **log** 部分中，增加 **level** 选项的值。这会导致 LVM 在系统日志中提供更多详情。
  - 如果问题与逻辑卷激活有关，请启用 LVM 在激活过程中记录信息：
    - i. 在 **/etc/lvm/lvm.conf** 配置文件的 **log** 部分中设置 **activation = 1** 选项。
    - ii. 使用 **-vvvv** 选项执行 LVM 命令。
    - iii. 检查命令输出。
    - iv. 将 **activation** 选项重置为 **0**。  
如果您没有将选项重置为 **0**，则系统在内存不足时可能会变得无响应。
  - 为诊断显示信息转储：

```
# lvmdump
```
  - 显示附加系统信息：

```
# lvs -v
```

```
# pvs --all
```

```
# dmsetup info --columns
```
  - 检查 **/etc/lvm/backup/** 目录中的最后一个 LVM 元数据备份，并在 **/etc/lvm/archive/** 目录中检查存档版本。
  - 检查当前的配置信息：

```
# lvmconfig
```
  - 检查 **/run/lvm/hints** 缓存文件以获取哪些设备上具有物理卷的记录。

#### 其他资源

- 系统中 **lvmdump (8)** 手册页

## 13.2. 显示有关失败的 LVM 设备的信息

有关失败的逻辑卷管理器(LVM)卷的故障排除信息可帮助您确定失败的原因。您可以检查以下最常见 LVM 卷失败的示例。

### 例 13.1. 失败的卷组

在本例中，组成卷组 *myvg* 的设备之一失败。然后卷组可用性取决于故障类型。例如，如果涉及到 RAID 卷，卷组仍可用。您还可以查看有关失败的设备的信息。

```
# vgs --options +devices
/dev/vdb1: open failed: No such device or address
/dev/vdb1: open failed: No such device or address
WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to
/dev/sdb1).
WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed
devices.

VG   #PV #LV #SN Attr   VSize VFree Devices
myvg  2  2  0 wz-pn- <3.64t <3.60t [unknown](0)
myvg  2  2  0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/vdb1(0)
```

### 例 13.2. 逻辑卷失败

在这个示例中，其中一个设备失败了。这可能是卷组中逻辑卷失败的原因。命令输出显示失败的逻辑卷。

```
# lvs --all --options +devices

/dev/vdb1: open failed: No such device or address
/dev/vdb1: open failed: No such device or address
WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to
/dev/sdb1).
WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed
devices.

LV   VG Attr   LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert Devices
mylv myvg -wi-a---p- 20.00g                [unknown](0)
[unknown](5120),/dev/sdc1(0)
```

### 例 13.3. RAID 逻辑卷的失败的镜像

下面的例子显示了 RAID 逻辑卷的镜像失败时，**pvs** 和 **lvs** 工具的命令输出。逻辑卷仍可使用。

```
# pvs

Error reading device /dev/sdc1 at 0 length 4.

Error reading device /dev/sdc1 at 4096 length 4.
```

Couldn't find device with uuid b2J8oD-vdjw-tGCA-ema3-iXob-Jc6M-TC07Rn.

WARNING: Couldn't find all devices for LV myvg/my\_raid1\_rimage\_1 while checking used and assumed devices.

WARNING: Couldn't find all devices for LV myvg/my\_raid1\_rmeta\_1 while checking used and assumed devices.

| PV        | VG         | Fmt  | Attr | PSize    | PFree    |
|-----------|------------|------|------|----------|----------|
| /dev/sda2 | rhel_bp-01 | lvm2 | a--  | <464.76g | 4.00m    |
| /dev/sdb1 | myvg       | lvm2 | a--  | <836.69g | 736.68g  |
| /dev/sdd1 | myvg       | lvm2 | a--  | <836.69g | <836.69g |
| /dev/sde1 | myvg       | lvm2 | a--  | <836.69g | <836.69g |
| [unknown] | myvg       | lvm2 | a-m  | <836.69g | 736.68g  |

```
# lvs -a --options name,vgname,attr,size,devices myvg
```

Couldn't find device with uuid b2J8oD-vdjw-tGCA-ema3-iXob-Jc6M-TC07Rn.

WARNING: Couldn't find all devices for LV myvg/my\_raid1\_rimage\_1 while checking used and assumed devices.

WARNING: Couldn't find all devices for LV myvg/my\_raid1\_rmeta\_1 while checking used and assumed devices.

| LV                  | VG   | Attr       | LSize   | Devices                                   |
|---------------------|------|------------|---------|---|
| my_raid1            | myvg | rwi-a-r-p- | 100.00g | my_raid1_rimage_0(0),my_raid1_rimage_1(0) |
| [my_raid1_rimage_0] | myvg | iwi-aor--- | 100.00g | /dev/sdb1(1)                              |
| [my_raid1_rimage_1] | myvg | lwi-aor-p- | 100.00g | [unknown](1)                              |
| [my_raid1_rmeta_0]  | myvg | ewi-aor--- | 4.00m   | /dev/sdb1(0)                              |
| [my_raid1_rmeta_1]  | myvg | ewi-aor-p- | 4.00m   | [unknown](0)                              |

### 13.3. 从卷组中删除丢失的 LVM 物理卷

如果物理卷失败，您可以激活卷组中剩余的物理卷，并从卷组中删除所有使用该物理卷的逻辑卷。

#### 流程

1. 激活卷组中剩余的物理卷：

```
# vgchange --activate y --partial myvg
```

2. 检查要删除哪些逻辑卷：

```
# vgreduce --removemissing --test myvg
```

3. 从卷组中删除所有使用丢失的物理卷的逻辑卷：

```
# vgreduce --removemissing --force myvg
```

4. 可选：如果意外删除要保留的逻辑卷，您可以撤销 **vgreduce** 操作：



```
# vgcfgrestore myvg
```



### 警告

如果您删除了精简池，LVM 无法撤销操作。

## 13.4. 查找丢失的 LVM 物理卷的元数据

如果意外覆盖或者破坏了卷组物理卷元数据区域，您会得到出错信息表示元数据区域不正确，或者系统无法使用特定的 UUID 找到物理卷。

这个过程找到丢失或者损坏的物理卷的最新归档元数据。

### 流程

1. 查找包含物理卷的卷组元数据文件。归档的元数据文件位于 `/etc/lvm/archive/volume-group-name_backup-number.vg` 路径中：

```
# cat /etc/lvm/archive/myvg_00000-1248998876.vg
```

使用备份号替换 00000-1248998876。选择该卷组最高数字最后已知的有效元数据文件。

2. 找到物理卷的 UUID。使用以下任一方法。

- 列出逻辑卷：

```
# lvs --all --options +devices
```

```
Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5SK'.
```

- 检查归档的元数据文件。在卷组配置的 **physical\_volumes** 部分中，查找标记为 **id =** 的 UUID。

- 使用 **--partial** 选项取消激活卷组：

```
# vgchange --activate n --partial myvg
```

```
PARTIAL MODE. Incomplete logical volumes will be processed.
```

```
WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
```

```
WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/vdb1).
```

```
0 logical volume(s) in volume group "myvg" now active
```

## 13.5. 在 LVM 物理卷中恢复元数据

这个过程恢复被损坏或者替换为新设备的物理卷的元数据。您可以通过重写物理卷的元数据区域从物理卷中恢复数据。

**警告**

不要在正常的 LVM 逻辑卷中尝试这个步骤。如果您指定了不正确的 UUID，将会丢失您的数据。

**先决条件**

- 您已找出丢失的物理卷的元数据。详情请查看[查找缺少的 LVM 物理卷的元数据](#)。

**流程**

1. 恢复物理卷中的元数据：

```
# pvcreate --uuid physical-volume-uuid \ --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \ block-device
```

**注意**

该命令只覆盖 LVM 元数据区域，不会影响现有的数据区域。

**例 13.4. 在 `/dev/vdb1` 上恢复物理卷**

以下示例使用以下属性将 `/dev/vdb1` 设备标记为物理卷：

- **FmGRh3-zhok-iVi8-7qTD-S5BI-MAEN-NYM5Sk** 的 UUID
- **VG\_00050.vg** 中包含的元数据信息，它是卷组最新的好归档元数据。

```
# pvcreate --uuid "FmGRh3-zhok-iVi8-7qTD-S5BI-MAEN-NYM5Sk" \ --restorefile
/etc/lvm/archive/VG_00050.vg \ /dev/vdb1
```

```
...
Physical volume "/dev/vdb1" successfully created
```

2. 恢复卷组的元数据：

```
# vgcfgrestore myvg

Restored volume group myvg
```

3. 显示卷组中的逻辑卷：

```
# lvs --all --options +devices myvg
```

逻辑卷目前不活跃。例如：

```

LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
mylv myvg -wi--- 300.00G                /dev/vdb1 (0),/dev/vdb1(0)
mylv myvg -wi--- 300.00G                /dev/vdb1 (34728),/dev/vdb1(0)

```

4. 如果逻辑卷的片段类型是 RAID，则重新同步逻辑卷：

```
# lvchange --resync myvg/mylv
```

5. 激活逻辑卷：

```
# lvchange --activate y myvg/mylv
```

6. 如果磁盘中的 LVM 元数据至少使用了覆盖其数据的空间，这个过程可以恢复物理卷。如果覆盖元数据的数据超过了元数据区域，则该卷中的数据可能会受到影响。您可能能够使用 **fsck** 命令恢复这些数据。

## 验证

- 显示活跃逻辑卷：

```

# lvs --all --options +devices

LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
mylv myvg -wi--- 300.00G                /dev/vdb1 (0),/dev/vdb1(0)
mylv myvg -wi--- 300.00G                /dev/vdb1 (34728),/dev/vdb1(0)

```

## 13.6. LVM 输出中的轮询错误

LVM 命令报告卷组中的空间使用情况，将报告的编号舍入到 **2** 个十进制位置，以提供人类可读的输出。这包括 **vgdisplay** 和 **vgs** 实用程序。

因此，报告的剩余空间值可能大于卷组中物理扩展提供的内容。如果您试图根据报告可用空间的大小创建逻辑卷，则可能会遇到以下错误：

```
Insufficient free extents
```

要临时解决这个问题，您必须检查卷组中可用物理扩展的数量，即可用空间的具体值。然后您可以使用扩展数目成功创建逻辑卷。

## 13.7. 防止创建 LVM 卷时出现循环错误

在创建 LVM 逻辑卷时，您可以指定逻辑卷的逻辑扩展数目以避免循环错误。

### 流程

1. 在卷组中找到可用物理扩展数目：

```
# vgdisplay myvg
```

### 例 13.5. 卷组中可用扩展

例如：以下卷组有 8780 可用物理扩展：

```

--- Volume group ---
VG Name          myvg
System ID
Format           lvm2
Metadata Areas    4
Metadata Sequence No 6
VG Access         read/write
[...]
Free PE / Size    8780 / 34.30 GB

```

2. 创建逻辑卷。以扩展而不是字节为单位输入卷大小。

#### 例 13.6. 通过指定扩展数目来创建逻辑卷

```
# lvcreate --extents 8780 --name mylv myvg
```

#### 例 13.7. 创建逻辑卷以占据所有剩余空间

另外，您可以扩展逻辑卷使其使用卷组中剩余的可用空间的比例。例如：

```
# lvcreate --extents 100%FREE --name mylv myvg
```

### 验证

- 检查卷组现在使用的扩展数目：

```
# vgs --options +vg_free_count,vg_extent_count

VG   #PV #LV #SN Attr   VSize  VFree Free #Ext
myvg 2  1  0 wz--n- 34.30G  0   0  8780
```

## 13.8. LVM 元数据及其在磁盘上的位置

提供不同偏移和大小的 LVM 标头和元数据区域。

默认 LVM 磁盘标头：

- 可在 **label\_header** 和 **pv\_header** 结构中找到。
- 在磁盘的第二个 512 字节扇区中。请注意，如果在创建物理卷(PV)时指定了非默认位置，则标头也可以在第一个或第三个扇区中。

标准 LVM 元数据区域：

- 从磁盘开始的头 4096 个字节。
- 从磁盘开始的结尾 1 MiB。
- 以包含 **mda\_header** 结构的 512 字节扇区开头。

元数据文本区域在 **mda\_header** 扇区之后开始，并直到元数据区域的末尾。LVM VG 元数据文本以循环方式写入到元数据文本区域中。**mda\_header** 指向文本区域中最新 VG 元数据的位置。

您可以使用 **# pvck --dump headers /dev/sda** 命令打印磁盘中的 LVM 标头。此命令打印 **label\_header**、**pv\_header**、**mda\_header** 以及元数据文本的位置（如果发现的话）。错误字段使用 **CHECK** 前缀打印。

LVM 元数据区偏移将与创建 PV 的机器的页大小匹配，因此元数据区域也可以从磁盘开头的 8K、16K 或 64K 开始。

在创建 PV 时可以指定更大或更小的元数据区域，在这种情况下，元数据区域可能会在 1 MiB 之外的位置结束。**pv\_header** 指定元数据区域的大小。

在创建 PV 时，可选择在磁盘末尾启用第二个元数据区域。**pv\_header** 包含元数据区域的位置。

## 13.9. 从磁盘中提取 VG 元数据

根据您的情况，选择以下流程之一，来从磁盘中提取 VG 元数据。有关如何保存提取的元数据的详情，请参考 [将提取的元数据保存在文件中](#)。



### 注意

要进行修复，您可以使用 **/etc/lvm/backup/** 中的备份文件，而无需从磁盘中提取元数据。

### 流程

- 打印有效的 **mda\_header** 中提到的当前元数据文本：

```
# pvck --dump metadata <disk>
```

#### 例 13.8. 有效的 **mda\_header** 中的元数据文本

```
# pvck --dump metadata /dev/sdb
metadata text at 172032 crc Oxc627522f # vgrame test segno 59
---
<raw metadata from disk>
---
```

- 根据找到的有效的 **mda\_header**，打印元数据区域中找到的所有元数据副本的位置：

```
# pvck --dump metadata_all <disk>
```

#### 例 13.9. 元数据区域中元数据副本的位置

```
# pvck --dump metadata_all /dev/sdb
metadata at 4608 length 815 crc 29fcd7ab vg test seqno 1 id FaCsSz-1ZZn-mTO4-Xl4i-
zb6G-BYat-u53Fzv
metadata at 5632 length 1144 crc 50ea61c3 vg test seqno 2 id FaCsSz-1ZZn-mTO4-
Xl4i-zb6G-BYat-u53Fzv
metadata at 7168 length 1450 crc 5652ea55 vg test seqno 3 id FaCsSz-1ZZn-mTO4-
Xl4i-zb6G-BYat-u53Fzv
```

- 搜索元数据区域中所有的元数据副本，而不使用 **mda\_header**，例如，如果标头丢失或损坏：

```
# pvck --dump metadata_search <disk>
```

#### 例 13.10. 不使用 **mda\_header** 的元数据区域中的元数据副本

```
# pvck --dump metadata_search /dev/sdb
Searching for metadata at offset 4096 size 1044480
metadata at 4608 length 815 crc 29fcd7ab vg test seqno 1 id FaCsSz-1ZZn-mTO4-Xl4i-
zb6G-BYat-u53Fzv
metadata at 5632 length 1144 crc 50ea61c3 vg test seqno 2 id FaCsSz-1ZZn-mTO4-
Xl4i-zb6G-BYat-u53Fzv
metadata at 7168 length 1450 crc 5652ea55 vg test seqno 3 id FaCsSz-1ZZn-mTO4-
Xl4i-zb6G-BYat-u53Fzv
```

- 在 **dump** 命令中包含 **-v** 选项，以显示每个元数据副本的描述：

```
# pvck --dump metadata -v <disk>
```

#### 例 13.11. 显示元数据的每个副本的描述

```
# pvck --dump metadata -v /dev/sdb
metadata text at 199680 crc 0x628cf243 # vgrame my_vg seqno 40
---
my_vg {
id = "dmEbPi-gsgx-VbvS-Uaia-HczM-iu32-Rb7iOf"
seqno = 40
format = "lvm2"
status = ["RESIZEABLE", "READ", "WRITE"]
flags = []
extent_size = 8192
max_lv = 0
max_pv = 0
metadata_copies = 0

physical_volumes {

pv0 {
id = "8gn0is-Hj8p-njgs-NM19-wuL9-mcB3-kUDiOQ"
device = "/dev/sda"

device_id_type = "sys_wwid"
device_id = "naa.6001405e635dbaab125476d88030a196"
status = ["ALLOCATABLE"]
flags = []
dev_size = 125829120
pe_start = 8192
pe_count = 15359
}

pv1 {
```

```

id = "E9qChJ-5EIL-HVEp-rc7d-U5Fg-fHxL-2QLyID"
device = "/dev/sdb"

device_id_type = "sys_wwid"
device_id = "naa.6001405f3f9396fddcd4012a50029a90"
status = ["ALLOCATABLE"]
flags = []
dev_size = 125829120
pe_start = 8192
pe_count = 15359
}

```

此文件可用于修复。默认情况下，第一个元数据区域用于转储元数据。如果磁盘在磁盘末尾处有第二个元数据区域，则您可以使用 **--settings "mda\_num=2"** 选项来将第二个元数据区域用于转储元数据。

### 13.10. 将提取的元数据保存到文件中

如果您需要使用转储的元数据进行修复，则需要使用 **-f** 选项和 **--settings** 选项将提取的元数据保存到文件中。

#### 流程

- 如果将 **-f <filename>** 添加到 **--dump metadata** 中，则原始元数据被写入到指定的文件中。您可以使用此文件进行修复。
- 如果将 **-f <filename>** 添加到 **--dump metadata\_all** 或 **--dump metadata\_search** 中，则所有位置的原始元数据都被写入到指定的文件中。
- 要保存 **--dump metadata\_all|metadata\_search** add **--settings "metadata\_offset=<offset>"** 中的一个元数据文本的实例，其中 **<offset>** 来自列表输出 "metadata at <offset>"。

#### 例 13.12. 命令的输出

```

# pvck --dump metadata_search --settings metadata_offset=5632 -f meta.txt /dev/sdb
Searching for metadata at offset 4096 size 1044480
metadata at 5632 length 1144 crc 50ea61c3 vg test seqno 2 id FaCsSz-1ZZn-mTO4-
XI4i-zb6G-BYat-u53Fvx
# head -2 meta.txt
test {
id = "FaCsSz-1ZZn-mTO4-XI4i-zb6G-BYat-u53Fvx"

```

### 13.11. 使用 **PVCREATE** 和 **VGCFGRESTORE** 命令修复带有损坏的 LVM 标头和元数据的磁盘

您可以恢复损坏的或者使用新设备替换的物理卷上的元数据和标头。您可以通过重写物理卷的元数据区域从物理卷中恢复数据。



### 警告

这些指令应当谨慎使用，只有在您熟悉每个命令的含义、卷的当前布局、您需要实现的布局以及备份元数据文件的内容时才应使用。这些命令可能会损坏数据，因此建议您联系红帽全球支持服务来帮助进行故障排除。

## 先决条件

- 您已找出丢失的物理卷的元数据。详情请查看[查找缺少的 LVM 物理卷的元数据](#)。

## 流程

1. 收集 **pvcreate** 和 **vgcfgrestore** 命令需要的以下信息。您可以通过运行 **# pvs -o+uuid** 命令收集有关磁盘和 UUID 的信息。
  - **metadata-file** 是 VG 的最新元数据备份文件的路径，例如：**/etc/lvm/backup/<vg-name>**
  - **VG-name** 是有损坏或缺失 PV 的 VG 的名称。
  - 此设备上损坏的 PV 的 **UUID** 是从 **# pvs -i+uuid** 命令的输出中获得的值。
  - **disk** 是 PV 所在磁盘的名称，例如 **/dev/sdb**。请注意，这是正确的磁盘，或寻求帮助，否则以下这些步骤可能导致数据丢失。
2. 在磁盘上重新创建 LVM 标头：

```
# pvcreate --restorefile <metadata-file> --uuid <UUID> <disk>
```

（可选）验证标头是否有效：

```
# pvck --dump headers <disk>
```

3. 恢复磁盘上的 VG 元数据：

```
# vgcfgrestore --file <metadata-file> <vg-name>
```

（可选）验证元数据是否已恢复：

```
# pvck --dump metadata <disk>
```

如果没有 VG 的元数据备份文件，您可以使用[将提取的元数据保存到文件](#)中的流程来获得。

## 验证

- 要验证新物理卷是否完整，且卷组是否正常工作，请检查以下命令的输出：

```
# vgs
```

## 其他资源



- [pvck \(8\)手册页](#)
- [从物理卷中提取 LVM 元数据备份](#)
- [如何在线修复物理卷上的元数据？](#)（红帽知识库）
- [如果组成卷组的一个物理卷失败，如何在 Red Hat Enterprise Linux 上恢复卷组？](#)（红帽知识库）

## 13.12. 使用 PVCK 命令修复带有损坏的 LVM 标头和元数据的磁盘

这是 [使用 pvcreate 和 vgcfgrestore 命令修复带有损坏的 LVM 标头和元数据的磁盘](#) 的替代方法。有些情况下，**pvcreate** 和 **vgcfgrestore** 命令可能无法正常工作。这个方法更针对损坏的磁盘。

此方法使用 **pvck --dump** 提取的元数据输入文件，或者 **/etc/lvm/backup** 中的备份文件。在可能的情况下，使用 **pvck --dump** 从同一 VG 中的其他 PV 中保存的元数据，或者从 PV 上的第二个元数据区域中保存的元数据。如需更多信息，请参阅 [将提取的元数据保存到文件中](#)。

### 流程

- 修复磁盘上的标头和元数据：

```
# pvck --repair -f <metadata-file> <disk>
```

其中

- **<metadata-file>** 是包含 VG 的最新元数据的文件。这可以是 **/etc/lvm/backup/vg-name**，也可以是包含 **pvck --dump metadata\_search** 命令输出中的原始元数据文本的文件。
- **<disk>** 是 PV 所在的磁盘的名称，例如 **/dev/sdb**。要防止数据丢失，请验证是否为正确的磁盘。如果您不确定磁盘是否正确，请联系红帽支持团队。



### 注意

如果元数据文件是一个备份文件，则 **pvck --repair** 应在 VG 中保存元数据的每个 PV 上运行。如果元数据文件是从另一个 PV 中提取的原始元数据，则仅需要在损坏的 PV 上运行 **pvck --repair**。

### 验证

- 要检查新物理卷是否完整，且卷组是否工作正常，请检查以下命令的输出：

```
# vgs <vgname>
```

```
# pvs <pvname>
```

```
# lvs <lvname>
```

### 其他资源

- [pvck \(8\)手册页](#)
- [从物理卷中提取 LVM 元数据备份](#)。

- [如何在线修复物理卷上的元数据？](#)（红帽知识库）
- [如果组成卷组的一个物理卷失败，如何在 Red Hat Enterprise Linux 上恢复卷组？](#)（红帽知识库）

## 13.13. LVM RAID 故障排除

您可以对 LVM RAID 设备中的多个问题进行故障排除，修正数据错误、恢复设备或者替换失败的设备。

### 13.13.1. 检查 RAID 逻辑卷中数据的一致性

LVM 提供对 RAID 逻辑卷的清理支持。RAID 清理是读取阵列中的所有数据和奇偶校验块的过程，并检查它们是否是分配的。**lvchange --syncaction repair** 命令对阵列启动后台同步操作。

#### 流程

1. 可选：通过设置以下选项之一来控制 RAID 逻辑卷初始化的速度：

- **--maxrecoveryrate Rate[bBsSkKmMgG]** 为 RAID 逻辑卷设置最大恢复率，使其不会驱逐正常的 I/O 操作。
- **--minrecoveryrate Rate[bBsSkKmMgG]** 设置 RAID 逻辑卷的最小恢复率，以确保 sync 操作的 I/O 达到最小吞吐量，即使存在大量标准 I/O 时

```
# lvchange --maxrecoveryrate 4K my_vg/my_lv
Logical volume _my_vg/my_lv_changed.
```

使用恢复率值替换 4K，它是阵列中每个设备的每秒的量。如果没有后缀，选项会假定为 kiB/每秒/每个设备。

```
# lvchange --syncaction repair my_vg/my_lv
```

当您执行 RAID 清理操作时，**sync** 操作所需的后台 I/O 可能会排挤 LVM 设备的其它 I/O，如对卷组元数据的更新。这可能导致其它 LVM 操作速度下降。



#### 注意

您还可以在创建 RAID 设备时使用这些最大和最小 I/O 速率。例如，**lvcreate -type raid10 -i 2 -m 1 -L 10G --maxrecoveryrate 128 -n my\_lv my\_vg** 创建一个双向 RAID10 阵列 my\_lv，它在卷组 my\_vg 中，有大小为 10G，最大恢复率为 128 kiB/sec/device 的 3 个条带。

2. 显示阵列中未修复的数量的数量，没有修复它们：

```
# lvchange --syncaction check my_vg/my_lv
```

此命令对阵列启动后台同步操作。

3. 可选：查看 **var/log/syslog** 文件中的内核消息。
4. 修正阵列中的差异：

```
# lvchange --syncaction repair my_vg/my_lv
```

这个命令修复或者替换 RAID 逻辑卷中失败的设备。您可以在执行此命令后，在 **var/log/syslog** 文件查看内核消息。

## 验证

1. 显示有关清理操作的信息：

```
# lvs -o +raid_sync_action,raid_mismatch_count my_vg/my_lv
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
SyncAction Mismatches
my_lv my_vg rwi-a-r--- 500.00m 100.00 idle 0
```

## 其他资源

- 您系统上的 **lvchange (8)** 和 **lvraid (7)** 手册页
- [最小和最大 I/O 速率选项](#)

### 13.13.2. 在逻辑卷中替换失败的 RAID 设备

RAID 与传统的 LVM 镜像不同。如果是 LVM 镜像，请删除失败的设备。否则，当 RAID 阵列继续使用失败的设备运行时，镜像逻辑卷将挂起。对于 RAID1 以外的 RAID 级别，删除设备意味着转换到较低 RAID 级别，例如从 RAID6 转换到 RAID5，或者从 RAID4 或 RAID5 转换到 RAID0。

您可以使用 **lvconvert** 命令的 **--repair** 参数替换 RAID 逻辑卷中作为物理卷的故障设备，而不是删除失败的设备并分配一个替换品。

## 先决条件

- 卷组包含一个物理卷，它有足够的可用容量替换失败的设备。  
如果卷组中没有有足够可用扩展的物理卷，请使用 **vgextend** 程序添加一个新的、足够大的物理卷。

## 流程

1. 查看 RAID 逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
LV Cpy%Sync Devices
my_lv 100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] /dev/sdc1(1)
[my_lv_rimage_2] /dev/sdd1(1)
[my_lv_rmeta_0] /dev/sde1(0)
[my_lv_rmeta_1] /dev/sdc1(0)
[my_lv_rmeta_2] /dev/sdd1(0)
```

2. 在 **/dev/sdc** 设备失败后查看 RAID 逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
/dev/sdc: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
```

```
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and assumed devices.
```

```
LV          Cpy%Sync Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] [unknown](1)
[my_lv_rimage_2] /dev/sdd1(1)
[my_lv_rmeta_0] /dev/sde1(0)
[my_lv_rmeta_1] [unknown](0)
[my_lv_rmeta_2] /dev/sdd1(0)
```

### 3. 替换失败的设备：

```
# lvconvert --repair my_vg/my_lv
```

```
/dev/sdc: open failed: No such device or address
```

```
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
```

```
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and assumed devices.
```

```
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and assumed devices.
```

```
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
```

```
Faulty devices in my_vg/my_lv successfully replaced.
```

### 4. 可选：手动指定替换失败设备的物理卷：

```
# lvconvert --repair my_vg/my_lv replacement_pv
```

### 5. 使用替换检查逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
```

```
/dev/sdc: open failed: No such device or address
```

```
/dev/sdc1: open failed: No such device or address
```

```
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
```

```
LV          Cpy%Sync Devices
```

```
my_lv       43.79 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
```

```
[my_lv_rimage_0] /dev/sde1(1)
```

```
[my_lv_rimage_1] /dev/sdb1(1)
```

```
[my_lv_rimage_2] /dev/sdd1(1)
```

```
[my_lv_rmeta_0] /dev/sde1(0)
```

```
[my_lv_rmeta_1] /dev/sdb1(0)
```

```
[my_lv_rmeta_2] /dev/sdd1(0)
```

在您从卷组中删除失败的设备前，LVM 工具仍然指示 LVM 无法找到失败的设备。

### 6. 从卷组中删除失败的设备：

```
# vgreduce --removemissing my_vg
```

## 验证

#### 1. 删除失败的设备后查看可用的物理卷：

```
# pvscan
```

```
PV /dev/sde1 VG rhel_virt-506 lvm2 [<7.00 GiB / 0 free]
PV /dev/sdb1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
PV /dev/sdd1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
PV /dev/sdd1 VG my_vg lvm2 [<60.00 GiB / 59.50 GiB free]
```

2. 替换失败的设备后检查逻辑卷：

```
# lvs --all --options name,copy_percent,devices my_vg
my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]      /dev/sde1(1)
  [my_lv_rimage_1]      /dev/sdb1(1)
  [my_lv_rimage_2]      /dev/sdd1(1)
  [my_lv_rmeta_0]       /dev/sde1(0)
  [my_lv_rmeta_1]       /dev/sdb1(0)
  [my_lv_rmeta_2]       /dev/sdd1(0)
```

其他资源

- 您系统上的 **lvconvert (8)** 和 **vgreduce (8)** 手册页

13.14. 对多路径 LVM 设备进行重复的物理卷警告进行故障排除

当将 LVM 与多路径存储搭配使用时，列出卷组或者逻辑卷的 LVM 命令可能会显示如下信息：

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/dm-5 not /dev/sdd
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowerb not /dev/sde
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sddlmap not /dev/sdf
```

您可以排除这些警告来了解 LVM 显示它们的原因，或者隐藏警告信息。

13.14.1. 重复 PV 警告的根本原因

当设备映射器多路径(DM 多路径)、EMC PowerPath 或 Hitachi Dynamic Link Manager (HDLM)等多路径软件管理系统上的存储设备时，每个路径都会作为不同的 SCSI 设备注册。

然后多路径软件会创建一个映射到这些独立路径的新设备。因为每个 LUN 在 **/dev** 目录中有多个指向同一底层数据的设备节点，所以所有设备节点都包含相同的 LVM 元数据。

表 13.1. 不同多路径软件的设备映射示例

| 多路径软件         | 到 LUN 的 SCSI 路径                   | 多路径设备映射到路径  |
|---------------|-----------------------------------|---|
| DM Multipath  | <b>/dev/sdb</b> 和 <b>/dev/sdc</b> | <b>/dev/mapper/mpath1</b> 或 <b>/dev/mapper/mpatha</b> |
| EMC PowerPath |                                   | <b>/dev/emcpowera</b>                                 |
| HDLM          |                                   | <b>/dev/sddlmap</b>                                   |

由于多个设备节点，LVM 工具会多次查找相同的元数据，并将其作为重复报告。

### 13.14.2. 重复 PV 警告的情况

LVM 在以下任一情况下显示重复的 PV 警告：

#### 指向同一设备的单路径

输出中显示的两个设备都是指向同一设备的单一路径。

以下示例显示一个重复的 PV 警告，在该示例中重复的设备是同一设备的单一路径。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sdd not /dev/sdf
```

如果您使用 **multipath -ll** 命令列出当前的 DM 多路径拓扑，您可以在相同的多路径映射中找到 **/dev/sdd** 和 **/dev/sdf**。

这些重复的信息只是警告，并不意味着 LVM 操作失败。相反，它们会提醒您 LVM 只使用其中一个设备作为物理卷，并忽略其它设备。

如果消息表示 LVM 选择了不正确的设备，或者警告是否对用户造成破坏，您可以应用过滤器。过滤器将 LVM 配置为只搜索物理卷所需的设备，并将任何基本路径留给多路径设备。因此，不再会出现警告。

#### 多路径映射

输出中显示的两个设备都是多路径映射。

以下示例显示两个设备都有重复的 PV 警告，它们是多路径映射。重复的物理卷位于两个不同的设备中，而不是位于同一设备的两个不同路径中。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/mapper/mpatha not /dev/mapper/mpathc
```

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowera not /dev/emcpowerh
```

对于同一设备上的单一路径的设备来说，这种情形比重复的警告更为严重。这些警告通常意味着机器正在访问它不应该访问的设备：例如：LUN 克隆或镜像(mirror)。

除非您明确知道您应该从机器中删除哪些设备，否则这个情况可能无法恢复。红帽建议联系红帽技术支持来解决这个问题。

### 13.14.3. 防止重复 PV 警告的 LVM 设备过滤器示例

以下示例显示 LVM 设备过滤器，它避免了由单一逻辑单元(LUN)由多个存储路径导致的重复物理卷警告。

您可以为逻辑卷管理器(LVM)配置过滤器来检查所有设备的元数据。元数据包括其上有根卷组的本地硬盘，以及任何多路径设备。通过拒绝到多路径设备的底层路径（如 **/dev/sdb**、**/dev/sdd**），您可以避免这些重复的 PV 警告，因为 LVM 在多路径设备上只查找每个唯一元数据区域一次。

- 要接受第一块硬盘上的第二个分区以及任何设备映射器(DM)多路径设备，并拒绝任何其它设备，请输入：

```
filter = [ "a|/dev/sda2$|", "a|/dev/mapper/mpath.*|", "r|.*)" ]
```

- 要接受所有 HP SmartArray 控制器和任何 EMC PowerPath 设备，请输入：

■

```
filter = [ "a|/dev/cciss/.*", "a|/dev/emcpower.*", "r|.*)" ]
```

- 要接受第一个 IDE 驱动器上的任何分区，以及任何多路径设备，请输入：

```
filter = [ "a|/dev/hda.*", "a|/dev/mapper/mpath.*", "r|.*)" ]
```

## 其他资源

- [LVM 设备过滤器配置示例](#)

### 13.14.4. 其他资源

- [限制 LVM 设备可见性和用法](#)
- [LVM 设备过滤器](#)