Red Hat Enterprise Linux 9

配置和管理逻辑卷

配置和管理 LVM 逻辑卷指南

Last Updated: 2022-07-12

Red Hat Enterprise Linux 9 配置和管理逻辑卷

配置和管理 LVM 逻辑卷指南

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本文档提供如何在 Red Hat Enterprise Linux 9 中管理 LVM 逻辑卷的说明。

目录

让开源更具包容性

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请让我们了解如何改进文档。

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要通过 Bugzilla 提交反馈，请创建一个新的 ticket：
  1. 进入 Bugzilla 网站。
  2. 在 Component 中选择 **Documentation**。
  3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
  4. 点 **Submit Bug**。

第 1 章 逻辑卷管理概述

逻辑卷管理(LVM)在物理存储上创建抽象层，帮助您创建逻辑存储卷。这比直接使用物理存储的方式具有更大的灵活性。

此外，硬件存储配置在软件中是隐藏的，因此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

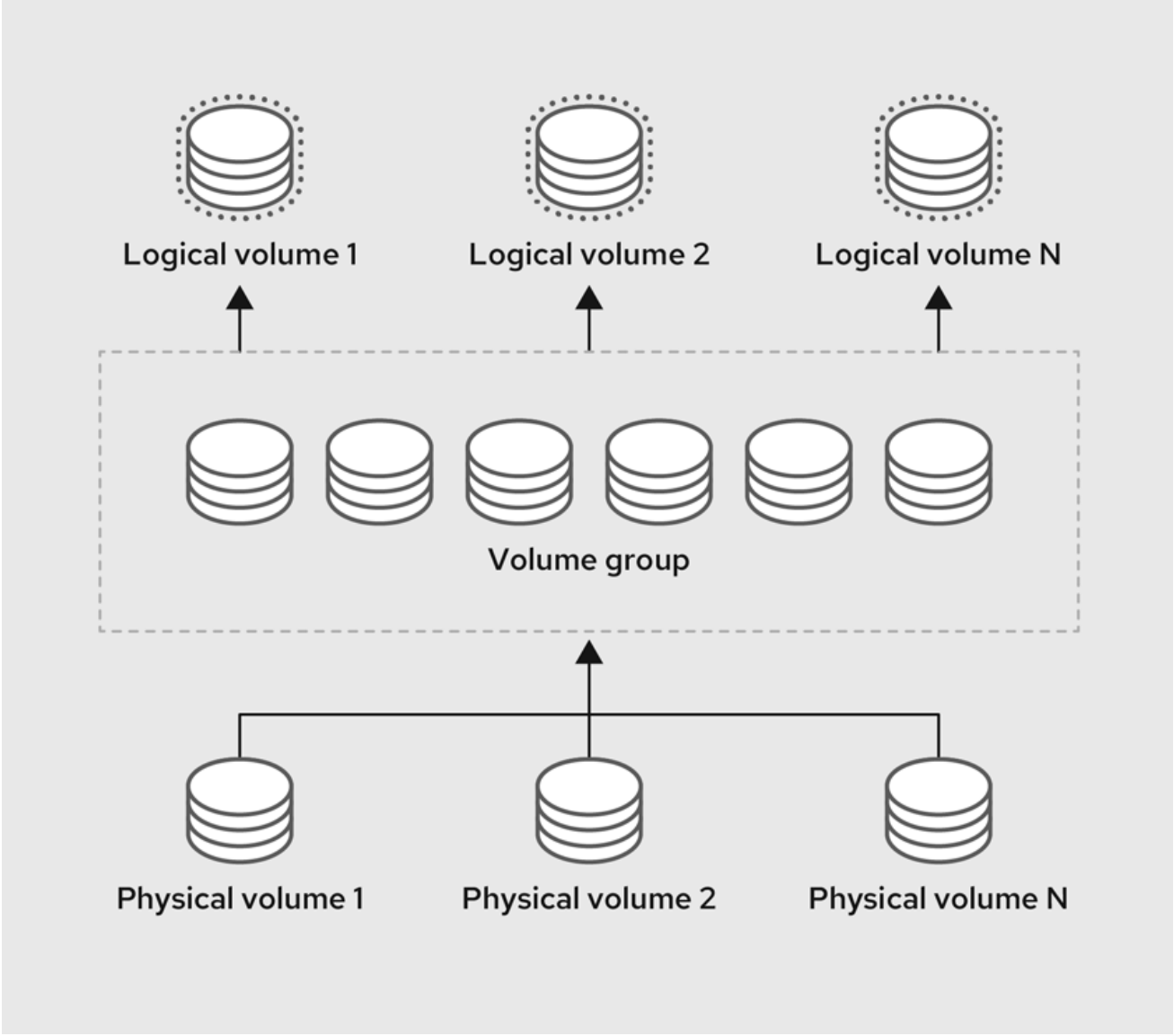
卷组(VG)是物理卷(PV)的集合，它会创建一个磁盘空间池，从中可以分配逻辑卷。如需更多信息，请参阅管理 LVM 卷组。

逻辑卷

逻辑卷代表可挂载的存储设备。如需更多信息，请参阅管理 LVM 逻辑卷。

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。

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。

在线数据重新定位

部署更新、更快或者更弹性的存储子系统，可以在系统活跃时移动数据。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。

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。

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。

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。

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。

精简卷

逻辑卷可以使用精简模式置备。这可让您创建大于可用物理空间的逻辑卷。

缓存卷

缓存逻辑卷使用快速块设备，如 SSD 驱动器，以提高更大、较慢的块设备的性能。

第 2 章 管理 LVM 物理卷

物理卷(PV)是 LVM 要使用的分区或整个磁盘。要将设备用于 LVM 逻辑卷，必须将设备初始化为物理卷。

如果您将整个磁盘作为您的物理卷使用，那么磁盘就不能有分区表。对于 DOS 磁盘分区，应该使用 **fdisk** 或 **cfdisk** 命令或对等命令将分区 id 设置为 0x8e。对于整个磁盘设备，分区表必须被删除，这样会有效地破坏磁盘中的所有数据。您可以以 root 用户身份运行 **dd if=/dev/zero of=*<PhysicalVolume>* bs=512 count=1** 来把第一个扇区写为 0 来删除一个已存在的分区表。

2.1. 物理卷概述

将块设备初始化为物理卷会在接近设备起始的位置放置一个标签。下面描述了 LVM 标签：

- LVM 标签为物理设备提供正确的标识和设备排序。未标记的非 LVM 设备可以在重新引导后更改名称，具体取决于系统在启动过程中发现它们的顺序。LVM 标签在重新引导时具有持久性并在整个集群中可用。
- LVM 标签可将该设备识别为 LVM 物理卷。它包含一个随机唯一标识符，即物理卷的 UUID。它还以字节为单位保存块设备的大小，并记录 LVM 元数据存储在该设备中的位置。
- 默认情况下，LVM 标签是放在第二个 512 字节扇区。您可以在创建物理卷时将标签放在前 4 个扇区的任意一个扇区，从而覆盖此默认设置。如果需要，LVM 卷可与其它使用这些扇区的用户共同存在。

下面描述了 LVM 元数据：

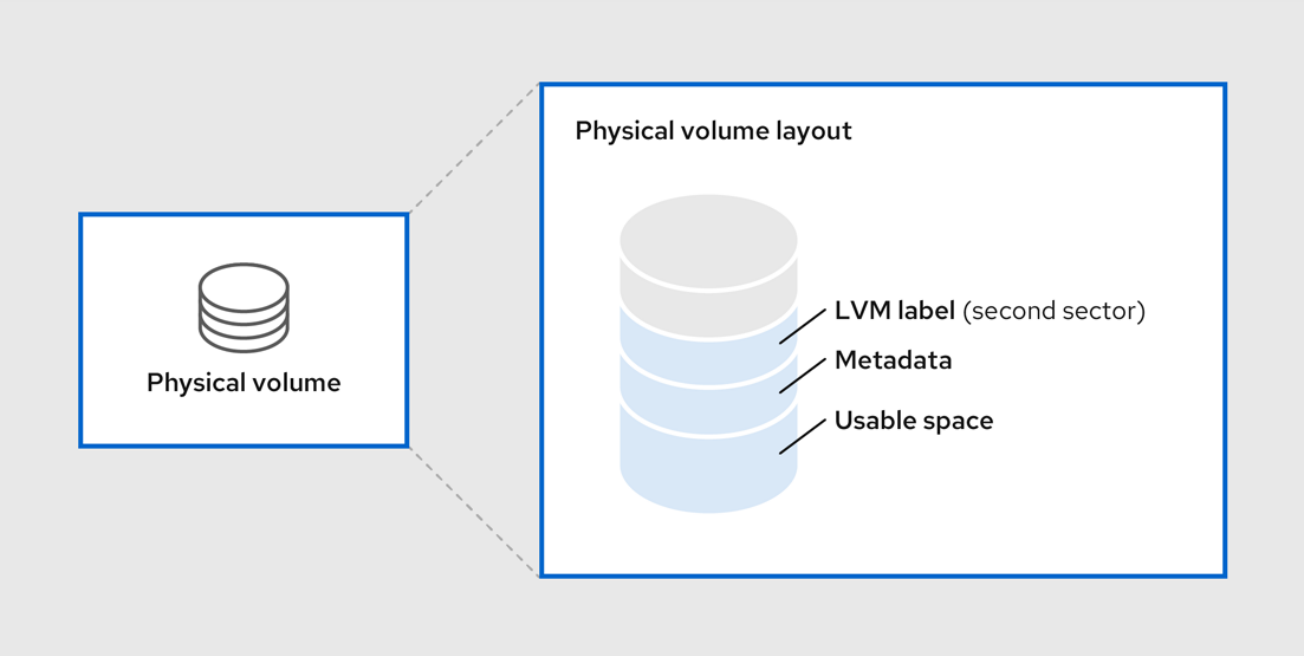
- LVM 元数据包含您系统中 LVM 卷组的配置详情。默认情况下，卷组中的每个物理卷的元数据区域都会保留一个一样的元数据副本。LVM 元数据很小，它以 ASCII 格式保存。
- 目前 LVM 允许您在每个物理卷中存储 0、1 或 2 个元数据副本。默认为 1 个副本。当您在物理卷中配置元数据副本数后，您将无法再更改该号码。第一个副本保存在设备的起始位置，紧随在标签后面。如果有第二个副本，会将其放在设备的末尾。如果您不小心写入了不同于您想要写入的磁盘覆盖了磁盘起始部分，那么您可以使用在设备末尾的元数据的第二个副本恢复元数据。

下图说明了 LVM 物理卷的布局。LVM 标签在第二个扇区，接下来是元数据区域，后面是设备的可用空间。

注意

在 Linux 内核和整个文档中，扇区的大小被视为 512 字节。

图 2.1. 物理卷布局



其他资源

- 一个磁盘上的多个分区

2.2. 一个磁盘上的多个分区

您可以使用 LVM 从磁盘分区中创建物理卷(PV)。

红帽建议您创建一个覆盖整个磁盘的单一分区，将其标记为 LVM 物理卷，理由如下：

方便管理

如果每个真实磁盘只出现一次，那么在系统中追踪硬件就比较容易。特别是当磁盘失败时。

条带化性能

LVM 无法告知两个物理卷位于同一个物理磁盘中。如果您在两个物理卷位于同一物理磁盘时创建了条状逻辑卷，那么条带就可能在同一磁盘的不同分区中。这可能会降低性能，而不是提高性能。

RAID 冗余

LVM 无法确定两个物理卷是否位于同一设备中。如果您在位于同一设备上的两个物理卷上创建 RAID 逻辑卷，则性能和容错可能会丢失。

虽然不建议您这样做，但在某些情况下可能需要将磁盘分成独立的 LVM 物理卷。例如：在有多个磁盘的系统中，当您要将现有系统迁移到 LVM 卷时，可能需要将数据在分区间转移。另外，如果您有一个很大的磁盘，并且因为管理的原因想要有一个以上卷组，那么对磁盘进行分区是很必要的。如果您的磁盘有一个以上的分区，且这些分区在同一卷组中，在创建卷时指定逻辑卷中应包含哪些分区。

请注意，虽然 LVM 支持将非分区磁盘用作物理卷，但建议创建一个全磁盘分区，因为在混合操作系统环境中创建不含分区的 PV 可能会有问题。其他操作系统可能会认为这些设备为空，并覆盖驱动器开头的 PV 标签。

2.3. 创建 LVM 物理卷

这个步骤描述了如何创建和标记 LVM 物理卷（PV）。

在此过程中，将 */dev/vdb1*、*/dev/vdb2* 和 */dev/vdb3* 替换为您系统中的可用存储设备。

先决条件

- 已安装 **lvm2** 软件包。

步骤

1. 使用以空格分隔的设备名称作为 **pvcreate** 命令的参数来创建多个物理卷：

```
# pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Physical volume "/dev/vdb3" successfully created.
```

   这会在 */dev/vdb1*、*/dev/vdb2* 和 */dev/vdb3* 上放置一个标签，将它们标记为属于 LVM 的物理卷。

2. 根据您的要求，使用以下命令之一查看创建的物理卷：

   a. **pvdisplay** 命令，它为每个物理卷提供详细的多行输出。它以固定格式显示物理属性，如大小、扩展、卷组和其他选项：

```
# pvdisplay
--- NEW Physical volume ---
  PV Name               /dev/vdb1
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb2
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb3
  VG Name
  PV Size               1.00 GiB
[..]
```

   b. **pvs** 命令提供了可以对其进行格式配置的物理卷信息，每行显示一个物理卷。

```
# pvs
  PV         VG  Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
/dev/vdb3        lvm2           1020.00m   0
```

   c. **pvscan** 命令扫描系统中所有支持的物理卷 LVM 块设备。您可以在 **lvm.conf** 文件中定义过滤器，以便这个命令避免扫描特定物理卷：

```
# pvscan
  PV /dev/vdb1                    lvm2 [1.00 GiB]
  PV /dev/vdb2                    lvm2 [1.00 GiB]
  PV /dev/vdb3                    lvm2 [1.00 GiB]
```

-

其他资源

- **pvcreate(8)**, **pvdisplay(8)**, **pvs(8)**, **pvscan(8)**, 和 **lvm(8)** man page

2.4. 删除 LVM 物理卷

如果 LVM 不再需要某个设备，您可以使用 **pvremove** 命令删除 LVM 标签。执行 **pvremove** 命令会将空物理卷上的 LVM 元数据归零。

步骤

1. 删除物理卷：

```
# pvremove /dev/vdb3
Labels on physical volume "/dev/vdb3" successfully wiped.
```

2. 查看现有物理卷并验证是否已删除所需卷：

```
# pvs
  PV         VG   Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
```

如果您要删除的物理卷当前是卷组的一部分，则必须使用 **vgreduce** 命令将其从卷组中删除。如需更多信息，请参阅从卷组中删除物理卷

其他资源

- **pvremove(8)** 手册页

第 3 章 管理 LVM 卷组

卷组(VG)是物理卷(PV)的集合，它会创建一个磁盘空间池，从中可以分配逻辑卷。

在卷组中，可用于分配的磁盘空间被分成固定大小的单元，我们称之为扩展。一个扩展就是可被分配的最小空间单位。在物理卷中，扩展被称为物理扩展。

逻辑卷被分配成与物理卷扩展大小相同的逻辑扩展。因此卷组中的所有逻辑卷的扩展大小都是一样的。卷组将逻辑扩展与物理扩展匹配。

3.1. 创建 LVM 卷组

此流程描述了如何使用 */dev/vdb1* 和 */dev/vdb2* 物理卷创建 LVM 卷组(VG) *myvg*。

先决条件

- 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。

步骤

1. 创建卷组：

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
 Volume group "myvg" successfully created.
```

   这将创建一个名为 *myvg* 的 VG。PV */dev/vdb1* 和 */dev/vdb2* 是 *myvg* VG 的基础存储级别。

2. 根据您的要求，使用以下命令之一查看创建的卷组：

   a. **vgs** 命令以可配置的形式提供卷组信息，每行显示一个卷组：

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  myvg   4   1   0 wz--n-   3.98g 1008.00m
```

   b. **vgdisplay** 命令以固定格式显示卷组属性，如大小、区块、物理卷数量和其他选项。以下示例显示了卷组 *myvg* 的 **vgdisplay** 命令的输出。如果没有指定卷组，则会显示所有现有卷组：

```
# vgdisplay myvg _ --- Volume group --- VG Name _myvg
  System ID
  Format                lvm2
  Metadata Areas        4
  Metadata Sequence No  6
  VG Access             read/write
[..]
```

   c. **vgscan** 命令为卷组扫描系统中所有受支持的 LVM 块设备：

```
# vgscan
  Found volume group "myvg" using metadata type lvm2
```

3. 可选：通过添加一个或多个可用物理卷来提高卷组容量：

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

其他资源

- **pvcreate(8)**, **vgextend(8)**, **vgdisplay(8)**, **vgs(8)**, **vgscan(8)**, 和 **lvm(8)** man page

3.2. 合并 LVM 卷组

要将两个卷组合并为一个卷组，请使用 **vgmerge** 命令。如果这两个卷的物理扩展大小相等，且两个卷组的物理卷和逻辑卷的描述符合目的卷组的限制，您可以将一个不活跃的"源"卷与一个活跃或者不活跃的"目标"卷合并。

步骤

- 将不活跃卷组 *数据库* 合并到活跃或者不活跃的卷组 *myvg* 中，提供详细的运行时信息：

```
# vgmerge -v myvg databases
```

其他资源

- **vgmerge(8)** 手册页

3.3. 从卷组中删除物理卷

要从卷组中删除未使用的物理卷，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。

步骤

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中没有足够的可用扩展：

   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 *myvg* 卷组：

```
# vgextend myvg /dev/vdb4
  Volume group "myvg" 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 myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

验证

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

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

其他资源

- **vgreduce(8)**, **pvmove(8)**, 和 **pvs(8)** man pages

3.4. 分割 LVM 卷组

这个步骤描述了如何分割现有卷组。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 *myvg* 由 */dev/vdb1*、*/dev/vdb2* 和 */dev/vdb3* 组成。完成此步骤后，卷组 *myvg* 将包含 */dev/vdb1* 和 */dev/vdb2*，第二个卷组 *yourvg* 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。

步骤

1. 将现有卷组 *myvg* 拆分到新卷组 *yourvg* ：

```
# vgsplit myvg yourvg /dev/vdb3
  Volume group "yourvg" successfully split from "myvg"
```

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/myvg/mylv
```

有关创建逻辑卷的更多信息，请参阅管理 LVM 逻辑卷。

2. 查看两个卷组的属性：

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

验证

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

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

其他资源

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

3.5. 重命名 LVM 卷组

此流程将现有卷组 *myvg* 重命名为 *myvg1*。

步骤

1. 取消激活卷组。如果是一个集群卷组，在每个这样的节点上使用以下命令取消激活它的所有节点上的卷组：

```
# vgchange --activate n myvg
```

2. 重命名现有卷组：

```
# vgrename myvg myvg1
Volume group "myvg" successfully renamed to "myvg1"
```

您还可以通过指定设备的完整路径来重命名卷组：

```
# vgrename /dev/myvg /dev/myvg1
```

其他资源

- **vgrename(8)** 手册页

第 4 章 管理 LVM 逻辑卷

逻辑卷是文件系统、数据库或应用可以使用的虚拟块存储设备。要创建 LVM 逻辑卷，物理卷(PV)合并为一个卷组(VG)。这会创建一个磁盘空间池，用于分配 LVM 逻辑卷（LV）。

4.1. 逻辑卷概述

管理员可以在不损坏数据的情况下增大或缩小逻辑卷，这与标准磁盘分区不同。如果卷组中的物理卷位于不同的驱动器或者 RAID 阵列中，那么管理员也可以跨存储设备分配逻辑卷。

如果您缩小逻辑卷到比卷中数据所需的容量小的容量时，则可能会丢失数据。此外，某些文件系统无法缩小。为确保最大的灵活性，创建逻辑卷以满足您当前的需求，并使超额存储容量保持未分配。您可以根据需要安全地扩展逻辑卷使用未分配空间。

重要

在 AMD、Intel、ARM 和 IBM Power Systems 服务器中,引导装载程序无法读取 LVM 卷。您必须为您的 **/boot** 分区创建一个标准的非 LVM 磁盘分区。在 IBM Z 中， **zipl** 引导装载程序通过线性映射在 LVM 逻辑卷中支持 **/boot**。默认情况下，安装过程总是在 LVM 卷中创建 / 和 swap 分区，物理卷中有一个单独的 **/boot** 分区。

以下是不同类型的逻辑卷：

线性卷

线性卷将来自一个或多个物理卷的空间集合到一个逻辑卷中。例如：如果您有两个 60GB 的磁盘，您可以创建一个 120GB 的逻辑卷。物理存储是连在一起的。

条带化逻辑卷

当您向 LVM 逻辑卷写入数据时，文件系统会在基本物理卷之间部署数据。您可以通过创建一个条状逻辑卷来控制将数据写入物理卷的方法。对于大量连续的读取和写入，这样可以提高数据输入/输出的效率。

条带化通过以 round-robin 模式向预定数目的物理卷写入数据来提高性能。使用条带，I/O 可以并行执行。在某些情况下，这可能会为条带中的每个额外物理卷增加近线性能。

RAID 逻辑卷

LVM 支持 RAID 0、1、4、5、6 和 10。RAID 逻辑卷不是集群感知的。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。

精简配置的逻辑卷（精简卷）

使用精简配置的逻辑卷，您可以创建大于可用物理存储的逻辑卷。通过创建精简配置的卷集合，系统可以分配您使用的内容，而不是分配请求的完整存储量

快照卷

LVM 快照功能提供在特定时间创建设备的虚拟镜像且不会造成服务中断的功能。在提取快照后，当对原始设备进行修改时，快照功能会生成有变化的数据区域的副本，以便重建该设备的状态。

精简配置的快照卷

使用精简配置的快照卷，可以有更多虚拟设备存储在同一个数据卷中。精简置备的快照很有用，因为您不会复制在给定时间要捕获的所有数据。

缓存卷

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 LVM 逻辑卷

此流程描述了如何从 *myvg* 卷组中创建 *mylv* LVM 逻辑卷(LV)，该组使用 */dev/vdb1*、*/dev/vdb2* 和 */dev/vdb3* 物理卷创建。

先决条件

- 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。

步骤

1. 创建逻辑卷：

```
# lvcreate -n mylv -L 500M myvg
```

使用 **-n** 选项将 LV 名称设置为 *mylv*，并使用 **-L** 选项以 Mb 为单位设置 LV 的大小，但可以使用任何其他单元。默认情况下 LV 类型是线性的，但用户可以使用 **--type** 选项指定所需的类型。

重要

如果 VG 没有足够数量的可用物理扩展用于请求的大小和类型，该命令将失败。

2. 根据您的要求，使用以下命令之一查看创建的逻辑卷：

   a. **lvs** 命令提供了可以对其进行格式配置的逻辑卷信息，每行显示一个逻辑卷。

```
# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
 mylv myvg -wi-ao----   500.00m
```

   b. **lvdisplay** 命令以固定格式显示逻辑卷属性，如大小、布局和映射：

```
# lvdisplay -v /dev/myvg/mylv
  --- Logical volume ---
  LV Path                /dev/myvg/mylv
  LV Name                mylv
  VG Name                myvg
  LV UUID                YTnAk6-kMlT-c4pG-HBFZ-Bx7t-ePMk-7YjhaM
  LV Write Access        read/write
[..]
```

   c. **lvscan** 命令扫描系统中所有逻辑卷并列出它们：

```
# lvscan
 ACTIVE                 '/dev/myvg/mylv' [500.00 MiB] inherit
```

3. 在逻辑卷中创建文件系统。以下命令在逻辑卷中创建 **xfs** 文件系统：

```
# mkfs.xfs /dev/myvg/mylv
meta-data=/dev/myvg/mylv       isize=512    agcount=4, agsize=32000 blks
         =                     sectsz=512   attr=2, projid32bit=1
```

```
        =                       crc=1        finobt=1, sparse=1, rmapbt=0
        =                       reflink=1
data     =                      bsize=4096   blocks=128000, imaxpct=25
        =                       sunit=0      swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0, ftype=1
log      =internal log          bsize=4096   blocks=1368, version=2
        =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                  extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

4. 挂载逻辑卷并报告文件系统磁盘空间使用情况：

```
# mount /dev/myvg/mylv /mnt

# df -h
Filesystem              1K-blocks  Used  Available Use% Mounted on

/dev/mapper/myvg-mylv   506528    29388  477140    6%   /mnt
```

其他资源

- **lvcreate(8)**, **lvdisplay(8)**, **lvs(8)**, **lvscan(8)**, **lvm(8)** 和 **mkfs.xfs(8)** man page

4.3. 重命名 LVM 逻辑卷

这个步骤描述了如何将现有逻辑卷 *mylv* 重命名为 *mylv1*。

步骤

1. 如果逻辑卷当前已挂载，卸载该卷：

```
# umount /mnt
```

使用挂载点替换 */mnt*。

2. 如果在集群环境中存在逻辑卷，则在所有其激活的节点上取消激活逻辑卷。对每个这样的节点运行以下命令：

```
# lvchange --activate n myvg/mylv
```

3. 重命名现有逻辑卷：

```
# lvrename myvg mylv mylv1
Logical volume "mylv" successfully renamed to "mylv1"
```

您还可以通过指定设备的完整路径来重命名逻辑卷：

```
# lvrename /dev/myvg/mylv /dev/myvg/mylv1
```

其他资源

- **lvrename(8)** 手册页

4.4. 从逻辑卷中删除磁盘

这个步骤描述了如何从现有逻辑卷中删除磁盘，替换磁盘或者将磁盘用作不同卷的一部分。

要删除磁盘，您必须首先将 LVM 物理卷中的扩展移动到不同的磁盘或者一组磁盘中。

步骤

1. 在使用 LV 时查看物理卷的已用和可用空间：

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

2. 将数据移到其他物理卷中：

   a. 如果现有卷组中的其他物理卷中有足够的可用扩展，请使用以下命令移动数据：

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

   b. 如果现有卷组中的其他物理卷上没有足够的可用扩展，请使用以下命令来添加新物理卷，使用新创建的物理卷扩展卷组，并将数据移动到此物理卷中：

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

# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended

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

3. 删除物理卷：

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

   如果逻辑卷包含失败的物理卷，您就无法使用该逻辑卷。要从卷组中删除缺少的物理卷，如果缺少的物理卷上没有分配逻辑卷，您可以使用 **vgreduce** 命令的 **--removemissing** 参数：

```
# vgreduce --removemissing myvg
```

其他资源

- **pvmove(8)**, **vgextend(8)**, **vereduce(8)**, 和 **pvs(8)** man 页

4.5. 删除 LVM 逻辑卷

此流程描述了如何从卷组 *myvg* 中删除现有逻辑卷 */dev/myvg/mylv1*。

步骤

1. 如果逻辑卷当前已挂载，卸载该卷：

```
# umount /mnt
```

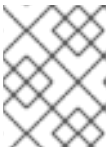
2. 如果在集群环境中存在逻辑卷，则在所有其激活的节点上取消激活逻辑卷。对每个这样的节点运行以下命令：

```
# lvchange --activate n vg-name/lv-name
```

3. 使用 **lvremove** 实用程序删除逻辑卷：

```
# lvremove /dev/myvg/mylv1

Do you really want to remove active logical volume "mylv1"? [y/n]: y
Logical volume "mylv1" successfully removed
```



注意

在这种情况下，逻辑卷还没有被取消激活。如果您在删除逻辑卷前明确取消激活了逻辑卷，则无法看到验证您是否要删除活跃逻辑卷的提示信息。

其他资源

- **lvremove(8)** 手册页

4.6. 删除 LVM 卷组

这个步骤描述了如何删除现有卷组。

先决条件

- 卷组没有包含逻辑卷。要从卷组中删除逻辑卷，请参阅删除 LVM 逻辑卷。

步骤

1. 如果卷组存在于集群的环境中，在所有节点上停止卷组的**锁定空间**。在除您要删除的节点外的所有节点上使用以下命令：

```
# vgchange --lockstop vg-name
```

等待锁定停止。

2. 删除卷组：

```
# vgremove vg-name
  Volume group "vg-name" successfully removed
```

其他资源

- **vgremove(8)** 手册页

第 5 章 修改逻辑卷的大小

创建逻辑卷后，您可以修改卷的大小。

5.1. 增大逻辑卷和文件系统

这个步骤描述了如何扩展逻辑卷并在同一逻辑卷中增大文件系统。

要增大逻辑卷的大小，请使用 **lvextend** 命令。当扩展逻辑卷时，可以指定您想要增大的量，或者指定扩展它需要达到的大小。

先决条件

1. 您有一个现有逻辑卷(LV)，其中包含一个文件系统。使用 **df -Th** 命令确定文件系统类型。
   有关创建 LV 和文件系统的更多信息，请参阅创建 LVM 逻辑卷。
2. 卷组中有足够的空间来扩展 LV 和文件系统。使用 **vgs -o name,vgfree** 命令确定可用空间。

步骤

1. 可选：如果卷组的空间不足以增大 LV，请使用以下命令向卷组中添加新物理卷：

   ```
   # vgextend myvg /dev/vdb3
   Physical volume "/dev/vdb3" successfully created.
   Volume group "myvg" successfully extended
   ```

   如需更多信息，请参阅创建 LVM 卷组。

2. 现在卷组足够大，根据您的要求执行以下步骤：

   a. 要使用提供的大小扩展 LV，请使用以下命令：

   ```
   # lvextend -L 3G /dev/myvg/mylv
   Size of logical volume myvg/mylv changed from 2.00 GiB (512 extents) to 3.00 GiB (768 extents).
   Logical volume myvg/mylv successfully resized.
   ```

注意

您可以使用 **lvextend** 命令的 **-r** 选项扩展逻辑卷并通过单个命令重新定义基础文件系统大小：

```
# lvextend -r -L 3G /dev/myvg/mylv
```

警告

您还可以使用带有相同参数的 **lvresize** 命令扩展逻辑卷，但这个命令不能保证意外收缩。

b. 要扩展 *mylv* 逻辑卷使其占据 *myvg* 卷组中所有未分配的空间，请使用以下命令：

```
# lvextend -l +100%FREE /dev/myvg/mylv
 Size of logical volume myvg/mylv changed from 10.00 GiB (2560 extents) to 6.35 TiB (1665465 extents).
 Logical volume myvg/mylv successfully resized.
```

与 **lvcreate** 命令一样，您可以使用 **lvextend** 命令的 **-l** 参数来指定扩展数目，从而增大逻辑卷的大小。您还可以使用此参数指定卷组的比例或者卷组中剩余空间的比例。

3. 如果您没有在 **lvextend** 命令中使用 **r** 选项来扩展 LV 并使用单个命令重新定义文件系统大小，请使用以下命令重新定义逻辑卷上的文件系统大小：

```
xfs_growfs /mnt/mnt1/
meta-data=/dev/mapper/myvg-mylv  isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 262144 to 524288
```

注意

如果没有 **-D** 选项，**xfs_growfs** 将文件系统增大到底层设备支持的最大大小。如需更多信息，请参阅增加 XFS 文件系统的大小。

有关重新定义 ext4 文件系统大小的信息，请参阅重新定义 ext4 文件系统大小。

验证

- 使用以下命令验证文件系统是否在增长：

```
# df -Th
Filesystem           Type      Size  Used Avail Use% Mounted on
devtmpfs             devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                tmpfs     1.9G  8.6M  1.9G   1% /run
tmpfs                tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/rhel-root xfs        45G  3.7G   42G   9% /
/dev/vda1            xfs      1014M  369M  646M  37% /boot
tmpfs                tmpfs     374M     0  374M   0% /run/user/0
/dev/mapper/myvg-mylv xfs       2.0G   47M  2.0G   3% /mnt/mnt1
```

其他资源

- **vgextend(8)**, **lvextend(8)**, 和 **xfs_growfs(8)** man 页

5.2. 缩小逻辑卷

您可以使用 **lvreduce** 命令来减小逻辑卷的大小。

注意

GFS2 或者 XFS 文件系统不支持缩小，因此您无法缩小包含 GFS2 或者 XFS 文件系统的逻辑卷大小。

如果您要缩小的逻辑卷包含一个文件系统，为了防止数据丢失，必须确定该文件系统没有使用将被缩小的逻辑卷中的空间。因此，建议您在逻辑卷包含文件系统时使用 **lvreduce** 命令的 **--resizefs** 选项。

当您使用这个选项时，**lvreduce** 命令会在缩小逻辑卷前尝试缩小文件系统。如果缩小文件系统失败，就像文件系统已满或者文件系统不支持缩小一样，则 **lvreduce** 命令将失败，且不会尝试缩小逻辑卷。

警告

在大多数情况下，**lvreduce** 命令会警告可能的数据丢失，并要求进行确认。但是，您不应该依赖于这些确认提示来防止数据丢失，因为在某些情况下，您不会看到这些提示信息，比如当逻辑卷不活跃或者没有使用 **--resizefs** 选项时。

请注意，使用 **lvreduce** 命令的 **--test** 选项不指示操作是安全的，因为此选项不会检查文件系统或测试文件系统大小。

步骤

- 要将 *myvg* 卷组中 *mylv* 逻辑卷缩小到 64MB，请使用以下命令：

```
# lvreduce --resizefs -L 64M myvg/mylv
fsck from util-linux 2.37.2
/dev/mapper/myvg-mylv: clean, 11/25688 files, 4800/102400 blocks
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mapper/myvg-mylv to 65536 (1k) blocks.
The filesystem on /dev/mapper/myvg-mylv is now 65536 (1k) blocks long.

Size of logical volume myvg/mylv changed from 100.00 MiB (25 extents) to 64.00 MiB (16 extents).
 Logical volume myvg/mylv successfully resized.
```

  在本例中，*mylv* 包含一个文件系统，该命令可调整逻辑卷的大小。

- 在调整大小值前指定 **-** 符号表示该值将从逻辑卷的实际大小中减小。要将逻辑卷缩小到 64MB，请使用以下命令：

```
# lvreduce --resizefs -L -64M myvg/mylv
```

其他资源

- **lvreduce(8)** man 页

第 6 章 逻辑卷快照

使用 LVM 快照功能，您可以创建一个卷的虚拟镜像，例如 */dev/sda*，而不会造成服务中断。

6.1. 快照卷概述

当您在进行快照后修改了原始卷（origin）时，快照功能会对修改的区域进行复制，以便可以重建卷的状态。当您创建快照时，仍可对原始卷进行完全的读写访问。

因为快照只复制创建快照后更改的数据区域，快照功能需要的存储量可以保持最少。例如，对于很少更新的原始卷，原始容量的 3-5% 就足以进行快照维护。它并不是一个备份过程的替代。快照副本是虚拟副本，并不是实际的介质备份。

预留用来存储原始卷更改的空间的大小取决于快照的大小。例如：如果您创建了一个快照，然后完全覆盖了原始卷，则快照必须至少与原始卷大小相同才可以保存相应的更改。您应该定期监控快照的大小。例如，一个多数用于读取的、短时间存在的卷（如 **/usr**）快照的空间会比一个长时间存在的，包括大量写操作的卷（如 **/home**）的快照要小。

如果快照已满，则快照会变得无效，因为它无法跟踪原始卷上的更改。但是，您可以将 LVM 配置为在其使用超过 **snapshot_autoextend_threshold** 值时自动扩展快照，以避免快照无效。快照可以完全重新定义，您可以执行以下操作：

- 如果您有存储容量，则可以增大快照卷以避免丢失快照。
- 如果您发现快照卷超过您的需要，您可以减小卷的大小来为其它逻辑卷最大限度腾出空间。

精简快照卷具有以下优点：

- 最典型的是，您需要在不中断实时数据更新的情况下对逻辑卷进行备份时，可以选择使用快照。
- 您可以在快照文件系统中执行 **fsck** 命令检查文件系统的完整性，并决定原始文件系统是否需要修复。
- 因为快照是可读/写的，如果您需要使用生产环境的数据来测试应用程序时，可以对数据进行快照，然后针对快照执行所需的测试，而不会影响到实际的数据。
- 您可以为 Red Hat Virtualization 创建 LVM 卷。您可以使用 LVM 快照来创建虚拟客户机镜像的快照。这些快照可方便修改现有客户虚拟机或者使用最小附加存储创建新客户虚拟机。

6.2. 创建原始卷的快照

使用 **lvcreate** 命令以及 **-s** 或 **--size** 参数，后跟创建原始卷的快照所需的大小。卷的快照是可写的。默认情况下，与精简配置的快照相比，在正常激活命令中会使用原始卷激活快照卷。LVM 不支持创建大于原始卷大小和卷所需的元数据大小的快照卷。如果您指定大于这个卷的快照卷，LVM 会创建一个原始卷大小所需的快照卷。

注意

集群中的节点不支持 LVM 快照。您不能在共享卷组中创建快照卷。然而，如果您需要在共享逻辑卷中创建一致的数据备份，您可以单独激活该卷，然后创建快照。

下面的步骤创建了一个名为 *origin* 的原始逻辑卷，以及一个名为 *snap* 的这个原始卷的快照卷。

先决条件

- 您已创建了卷组 *vg001*。如需更多信息，请参阅创建 LVM 卷组。

步骤

1. 从卷组 *vg001* 中创建名为 *origin* 的逻辑卷：

```
# lvcreate -L 1G -n origin vg001
Logical volume "origin" created.
```

2. 创建名为 *snap* 的 */dev/vg001/origin* 的快照逻辑卷，大小为 *100 MB* ：

```
# lvcreate --size 100M --name snap --snapshot /dev/vg001/origin
  Logical volume "snap" created.
```

如果原始逻辑卷包含一个文件系统，您可以在任意目录中挂载快照逻辑卷，以便访问文件系统的内容，并在不断更新原始文件系统时进行备份。

3. 显示原始卷以及正在使用的快照卷的比例：

```
# lvs -a -o +devices
  LV      VG    Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---   1.00g                                                     /dev/sde1(0)
  snap   vg001  swi-a-s--- 100.00m      origin 0.00                                    /dev/sde1(256)
```

您可以使用 **lvdisplay */dev/vg001/origin*** 命令，显示逻辑卷 */dev/vg001/origin* 的状态，包括所有快照逻辑卷和它们的状态。

警告

因为快照在原始卷有变化时会增大，所以通常使用 **lvs** 命令监控快照卷的比例以确定它没有被填满是很重要的。使用了 100% 的快照会完全丢失，因为对原始卷中未更改的部分的写入无法在不破坏快照的情况下无法成功。

4. 您可以将 LVM 配置为在其使用超过 **snapshot_autoextend_threshold** 值时自动扩展快照，以避免容量达到 100%。查看来自 **/etc/lvm.conf** 文件中的 **snapshot_autoextend_threshold** 和 **snapshot_autoextend_percent** 选项的现有值，并根据要求对其进行编辑。
在以下示例中，根据您的要求将 **snapshot_autoextend_threshold** 选项设置为一个小于 100 的值，**snapshot_autoextend_percent** 选项的值如下：

```
# vi /etc/lvm.conf
snapshot_autoextend_threshold = 70
snapshot_autoextend_percent = 20
```

您还可以执行以下命令手动扩展此快照：

```
# lvextend -L+100M /dev/vg001/snap
```

注意

此功能需要卷组中的未分配空间。快照的自动扩展不会将快照卷增加到超过计算的快照所需的最大值。一旦快照增长到足够大来覆盖原始数据后，便不会再监控它是否发生了自动扩展。

其他资源

- **lvcreate(8)**, **lvextend(8)**, and **lvs(8)** man page
- **/etc/lvm/lvm.conf** 文件

6.3. 将快照合并到其原始卷

使用 **lvconvert** 命令和 **--merge** 选项将快照合并到原始（原始）卷中。如果您丢失了数据或文件，或者需要将其系统恢复到以前的状态，则可以执行系统回滚。在合并快照卷后，结果逻辑卷带有原始卷的名称、副号码和 UUID。在合并过程中，对原始卷的读取和写入将会被指向要合并的快照。当合并完成后，会删除合并的快照。

如果原始卷和快照卷没有打开并激活，则合并会立即启动。否则，会在激活原始卷或快照被激活，且两者都关闭时合并会开始。在激活原始卷后，您可以将快照合并到无法关闭的原始卷中（如 **root** 文件系统）。

步骤

1. 合并快照卷。以下命令将快照卷 *vg001/snap* 合并到原始卷 *中* ：

```
# lvconvert --merge vg001/snap
Merging of volume vg001/snap started.
  vg001/origin: Merged: 100.00%
```

2. 查看原始卷：

```
# lvs -a -o +devices
  LV     VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001 owi-a-s---  1.00g                                       /dev/sde1(0)
```

其他资源

- **lvconvert(8)** man page

第 7 章 创建和管理精简配置的卷（精简卷）

Red Hat Enterprise Linux 支持精简配置的快照卷和逻辑卷。

逻辑卷和快照卷可以被精简置备：

- 使用精简配置的逻辑卷，您可以创建大于可用物理存储的逻辑卷。
- 使用精简配置的快照卷，您可以将更多虚拟设备存储在同一个数据卷中

7.1. 精简配置概述

许多现代存储堆栈现在提供在密集置备和精简置备之间进行选择的功能：

- 厚置备提供了块存储的传统方式分配块，无论其实际使用情况如何。
- 精简配置可以置备更大的块存储池，其大小可能大于存储数据的物理设备，从而造成过度置备。可能会出现过度置备，因为在实际使用单个块前不会分配各个块。如果您有多个共享同一池的精简配置设备，则可以过度置备这些设备。

通过使用精简置备，您可以过量使用物理存储，而是可以管理称为精简池的空闲空间池。当应用程序需要时，您可以将这个精简池分配给任意数量的设备。您可以在需要时动态扩展精简池，以便有效分配存储空间。

例如：如果 10 个用户为每个应用程序请求一个 100GB 文件系统，您可以为每个用户创建一个 100GB 文件系统，但其后端的实际存储可以小于这个大小，它在需要时才使用实际的存储。

注意

在使用精简配置时，务必要监控存储池，并在可用物理空间耗尽前添加更多容量。

以下是使用精简配置的设备的一些优点：

- 您可以创建大于可用物理存储的逻辑卷。
- 您可以将更多虚拟设备存储在同一数据卷中。
- 您可以创建可逻辑地增长的文件系统来支持数据要求，并且返回未使用的块供池中任意文件系统使用

以下是使用精简配置的设备的潜在缺陷：

- 精简配置的卷存在可用物理存储的固有风险。如果您过度置备底层存储，则可能因为缺少可用的物理存储会导致停机。例如，如果您使用 1T 物理存储创建了一个 10T 的精简置备存储，则该卷将在 1T 耗尽后不可用或不可写入数据。
- 如果在精简置备后卷没有向层发送 discard 信号，则使用量的核算就不准确。例如，在没有使用 **-o discard mount** 选项的情况下放置了一个文件系统，且没有在精简置备的设备之上运行 **fstrim**，则永远不会取消分配之前使用的存储。在这种情况下，即使实际上并没有使用，也会出现整个置备数量被使用完的情况
- 您必须监控逻辑和物理空间的使用情况，以避免出现用尽可用空间的情况。
- 在带有快照的文件系统上，Copy on Write (CoW) 操作可能会较慢。

- 数据块可能会在多个文件系统上相互混合，从而导致出现随机的底层存储访问限制的问题（即使它没有向最终用户显示这种方式）。

7.2. 创建精简配置的逻辑卷

使用精简配置的逻辑卷，您可以创建大于可用物理存储的逻辑卷。通过创建精简配置的卷集合，系统可以分配您使用的内容，而不是分配请求的完整存储量

使用 **lvcreate** 命令的 **-T** 或 **--thin** 选项，您可以创建一个精简池或精简卷。您也可以使用 **lvcreate** 命令的 **-T** 选项，通过单个命令同时创建精简池和精简卷。这个步骤描述了如何创建和增大精简置备逻辑卷。

先决条件

- 您已创建了卷组。如需更多信息，请参阅创建 LVM 卷组。

步骤

1. 创建精简池：

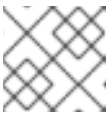
```
# lvcreate -L 100M -T vg001/mythinpool
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

请注意：由于您要创建物理空间池，您必须指定池的大小。**lvcreate** 命令的 **-T** 选项不使用参数；它确定要从添加的其他选项中创建哪种设备类型。您还可以使用额外参数创建精简池，如下例所示：

- 您还可以使用 **lvcreate** 命令的 **--thinpool** 参数创建精简池。与 **-T** 选项不同，**--thinpool** 参数要求您指定您要创建的精简池逻辑卷的名称。以下示例使用 **--thinpool** 参数在卷组 *vg001* 中创建名为 *mythinpool* 的精简池，大小为 *100M* ：

```
# lvcreate -L 100M --thinpool mythinpool vg001
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

- 随着条带的创建支持，您可以使用 **-i** 和 **-I** 选项创建条带。以下命令在卷组 *vg001* 中创建一个名为 *thinpool* 的 *100M* 精简池，其中有两个 *64 kB* 条带，块大小为 *256 kB*。它还创建一个名为 *vg001/thinvolume* 的 *1T* 精简卷。



注意

确定卷组中有两个有足够可用空间的物理卷，或者您无法创建精简池。

```
# lvcreate -i 2 -I 64 -c 256 -L 100M -T vg001/thinpool -V 1T --name thinvolume
```

2. 创建精简卷：

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
```

```
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

在这种情况下，您为卷所指定的虚拟大小需要比包括它的池要大。您还可以使用额外参数创建精简卷，如下例所示：

- 要创建精简卷和精简池，请使用 **lvcreate** 命令的 **-T** 选项并指定大小和虚拟大小参数：

```
# lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger
automatic extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

- 要使用剩余的可用空间来创建精简卷和精简池，请使用 **100%FREE** 选项：

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool -n thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of data.
  Logical volume "thinvolume" created.
```

- 要将现有逻辑卷转换为精简池卷，请使用 **lvconvert** 命令的 **--thinpool** 参数。您还必须使用 **--poolmetadata** 参数和 **--thinpool** 参数将现有逻辑卷转换为精简池卷的元数据卷。
以下示例将卷组 *vg001* 中的现有逻辑卷 *lv1* 转换为精简池卷，并将卷组 *vg001* 中现有逻辑卷 *lv2* 转换为那个精简池卷的元数据卷：

```
# lvconvert --thinpool vg001/lv1 --poolmetadata vg001/lv2
  Converted vg001/lv1 to thin pool.
```

注意

将逻辑卷转换成精简池卷或精简池元数据卷会破坏逻辑卷的内容，因为 **lvconvert** 不会保留设备的内容，而是覆盖其内容。

- 默认情况下，**lvcreate** 命令根据以下公式设置精简池元数据逻辑卷的大小：

```
Pool_LV_size / Pool_LV_chunk_size * 64
```

如果您有大量快照，或者您的精简池中有小的块，因此以后会大量增加精简池的大小，您可能需要用 **lvcreate** 命令的 **--poolmetadatasize** 参数来增加精简池的元数据卷的默认值。精简池元数据逻辑卷所支持的值在 2MiB 到 16GiB 之间。

以下示例演示了如何增加精简池元数据卷的默认值：

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool --poolmetadatasize 16M -n
thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "thinvolume" created.
```

3. 查看创建的精简池和精简卷：

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 100.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 100.00m
/dev/sda(1)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

4. 可选：使用 **lvextend** 命令扩展精简池的大小。但是您无法缩小精简池的大小。

注意

如果您在创建精简池和精简卷时使用 **-l 100%FREE** 参数，则此命令会失败。

以下命令调整了一个已存在的大小为 *100M* 的精简池，将其大小增加 *100M*。

```
# lvextend -L+100M vg001/mythinpool
  Size of logical volume vg001/mythinpool_tdata changed from 100.00 MiB (25 extents) to
200.00 MiB (50 extents).
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (200.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.

  Logical volume vg001/mythinpool successfully resized
```

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 200.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(1)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(27)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

5. 可选：要重命名精简池和精简卷，请使用以下命令：

```
# lvrename vg001/mythinpool vg001/mythinpool1
  Renamed "mythinpool" to "mythinpool1" in volume group "vg001"

# lvrename vg001/thinvolume vg001/thinvolume1
  Renamed "thinvolume" to "thinvolume1" in volume group "vg001"
```

重命名后查看精简池和精简卷：

```
# lvs
  LV          VG       Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
mythinpool1 vg001    twi-a-tz 100.00m                    0.00
thinvolume1 vg001    Vwi-a-tz   1.00g mythinpool1         0.00
```

6. 可选：要删除精简池，请使用以下命令：

```
# lvremove -f vg001/mythinpool1
  Logical volume "thinvolume1" successfully removed.
  Logical volume "mythinpool1" successfully removed.
```

其他资源

- **lvcreate(8)**, **lvrename(8)**, **lvs(8)**, and **lvconvert(8)** man page

7.3. 精简配置的快照卷

Red Hat Enterprise Linux 支持精简配置的快照卷。精简逻辑卷的快照还创建一个精简逻辑卷(LV)。精简快照卷具有与其它精简卷相同的特征。您可以独立激活卷、扩展卷、重新命名卷、删除卷、甚至快照卷。

注意

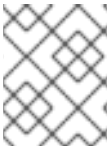
与所有 LVM 快照卷以及所有精简卷类似，集群中的节点不支持精简快照卷。快照卷必须在一个集群节点中完全激活。

传统的快照必须为创建的每个快照分配新空间，在对原始卷进行更改时保留数据。但是精简置备快照与原始卷共享相同的空间。thin LV 的快照效率更高，因为对 thin LV 及其任何的通用块是共享的。您可以创建 thin LV 的快照或其他精简快照。对于递归快照的通用块在精简池中也是共享的。

精简快照卷提供以下优点：

- 增加原始卷的快照数量对性能的影响不大。
- 使用精简快照卷可以减少磁盘的使用量，因为只有新数据会被写入，且不会复制到每个快照中。
- 精简快照卷不需要和原始卷被同时激活（传统快照有这个要求）。
- 从快照恢复原始卷时，不需要合并精简快照。您可以删除原始源并使用快照替代。传统快照具有单独的卷，用于存储必须要复制回去的改变，即，与原始快照合并以进行重置。
- 与传统的快照相比，允许的快照数量限制有显著提高。

虽然使用精简快照卷有很多优点，但在有些情况下，传统的 LVM 快照卷功能可能更适合您的需要。您可以将传统快照用于所有类型的卷。但是，要使用精简快照，您需要使用精简置备。



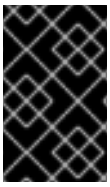
注意

您不能限制精简快照卷的大小 ; 如果需要，快照将使用精简池中所有空间。一般说来，在决定使用什么快照格式时，您应该考虑具体的要求。

默认情况下，在正常激活命令中会跳过精简快照卷。

7.4. 创建精简配置的快照卷

使用精简配置的快照卷，您可以将更多虚拟设备存储在同一个数据卷中



重要

在创建精简快照卷时，不要指定卷的大小。如果指定了 size 参数，则创建的快照不会是一个精简快照卷，也不会使用精简池来存储数据。例如： **lvcreate -s vg/thinvolume -L10M** 命令将不会创建精简快照，即使原始卷是一个精简卷。

可为精简配置的原始卷创建精简快照，也可针对不是精简置备的原始卷创建精简快照。下面的步骤描述了创建精简置备快照卷的不同方法。

先决条件

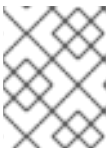
- 您已创建了精简配置的逻辑卷。如需更多信息，请参阅创建精简配置的逻辑卷。

步骤

- 创建精简配置的快照卷。以下命令创建一个精简置备的快照卷，它被命名为精简置备的逻辑卷 *vg001/thinvolume* 的 *mysnapshot1*：

```
# lvcreate -s --name mysnapshot1 vg001/thinvolume
  Logical volume "mysnapshot1" created
```

```
# lvs
  LV          VG      Attr     LSize   Pool       Origin     Data%  Move Log Copy%  Convert
  mysnapshot1 vg001   Vwi-a-tz   1.00g mythinpool thinvolume   0.00
  mythinpool  vg001   twi-a-tz 100.00m                        0.00
  thinvolume  vg001   Vwi-a-tz   1.00g mythinpool             0.00
```



注意

在使用精简配置时，存储管理员务必要监控存储池，并在其被完全占用时添加更多容量。有关扩展精简卷大小的详情，请参考创建精简配置的逻辑卷。

- 您还可以为非置备的逻辑卷创建精简配置的快照。因为非置备的逻辑卷不在精简池中，所以它也被称为外部原始卷。外部原始卷可以被很多精简置备的快照卷使用和共享，即使在不同的精简池中也是如此。在创建精简置备快照时，外部原始源必须不活跃且只读。
以下示例会为名为 *origin_volume* 的只读非活动逻辑卷创建一个精简快照卷。精简快照卷名为 *mythinsnap*。逻辑卷 *origin_volume* 然后会成为卷组 *vg001* 中的精简快照卷 *mythinsnap* 的外部源，它会使用现有的精简池 *vg001/pool*。原始卷必须与快照卷位于同一个卷组中。在指定原始逻辑卷时不要指定卷组。

```
# lvcreate -s --thinpool vg001/pool origin_volume --name mythinsnap
```

- 您可以执行以下命令，为第一个快照卷创建第二个精简置备快照卷。

```
# lvcreate -s vg001/mysnapshot1 --name mysnapshot2
Logical volume "mysnapshot2" created.
```

要创建第三个精简置备快照卷，请使用以下命令：

```
# lvcreate -s vg001/mysnapshot2 --name mysnapshot3
Logical volume "mysnapshot3" created.
```

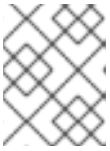
验证

- 显示精简快照逻辑卷的所有祖先和后代列表：

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV          Ancestors                       Descendants
  mysnapshot2 mysnapshot1,thinvolume          mysnapshot3
  mysnapshot1 thinvolume                      mysnapshot2,mysnapshot3
  mysnapshot3 mysnapshot2,mysnapshot1,thinvolume
  mythinpool
  thinvolume                                  mysnapshot1,mysnapshot2,mysnapshot3
```

在这里，

- *thinvolume* 是卷组 *vg001* 中的原始卷。
- *mysnapshot1* 是 *thinvolume* 的快照
- *mysnapshot2* 是 *mysnapshot1* 的快照
- *mysnapshot3* 是 *mysnapshot2* 的快照



注意

lv_ancestors 和 **lv_descendants** 字段会显示现有的依赖项。但是，如果从链的中间删除了条目，则它们无法跟踪删除的条目，这可能会破坏依赖项链。

其他资源

- **lvcreate(8)** man page

第 8 章 LVM 故障排除

您可以使用 LVM 工具排除 LVM 卷和组群中的问题。

8.1. 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。

步骤

- 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

- **lvmdump(8)** man page

8.2. 显示失败的 LVM 设备的信息

您可以显示一个失败的 LVM 卷的信息，以便帮助您确定为什么这个卷失败的原因。

步骤

- 使用 **vgs** 或 **lvs** 实用程序显示失败的卷。

例 8.1. 失败的卷组

在本例中，组成卷组 *myvg* 的设备之一失败。卷组不可用，但您可以看到有关失败设备的信息。

```
# 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 8.2. 逻辑卷失败

在这个示例中，其中一个设备会失败，因为卷组中的逻辑卷会失败。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)
```

8.3. 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。

步骤

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：如果您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。

8.4. 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。

步骤

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
```

8.5. 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。

步骤

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 8.3. 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。

验证步骤

- 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)
```

8.6. 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。

8.7. 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。

步骤

1. 在卷组中找到可用物理扩展数目：

```
# vgdisplay myvg
```

例 8.4. 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 8.5. 通过指定扩展数目来创建逻辑卷

```
# lvcreate --extents 8780 --name mylv myvg
```

例 8.6. 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
```

验证步骤

- 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
```