



# Red Hat Enterprise Linux 8

## 管理存储设备

配置和管理本地和远程存储设备





## 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

Red Hat Enterprise Linux (RHEL)提供了几个本地和远程存储选项。使用可用的存储选项，您可以执行以下任务：根据您的要求创建磁盘分区。使用磁盘加密保护块设备上的数据。创建一个独立磁盘的冗余阵列(RAID)，来跨多个驱动器存储数据，并避免数据丢失。使用 iSCSI 和 Fabrics 上的 NVMe 通过网络访问存储。

# Table of Contents

对红帽文档提供反馈 .....	7
<b>第 1 章 可用存储选项概述 .....</b>	<b>8</b>
1.1. 本地存储概述 .....	8
1.2. 远程存储概述 .....	9
1.3. GFS2 文件系统概述 .....	10
<b>第 2 章 磁盘分区 .....</b>	<b>11</b>
2.1. 分区概述 .....	11
2.2. 分区表类型比较 .....	11
2.3. MBR 磁盘分区 .....	11
2.4. 扩展 MBR 分区 .....	13
2.5. MBR 分区类型 .....	13
2.6. GUID 分区表 .....	14
2.7. 分区类型 .....	15
2.8. 分区命名方案 .....	17
2.9. 挂载点和磁盘分区 .....	17
<b>第 3 章 分区入门 .....</b>	<b>18</b>
3.1. 使用 PARTED 在磁盘中创建分区表 .....	18
3.2. 查看使用 PARTED 的分区表 .....	19
3.3. 使用 PARTED 创建分区 .....	20
3.4. 使用 FDISK 设置分区类型 .....	21
3.5. 使用 PARTED 重新定义分区大小 .....	22
3.6. 使用 PARTED 删除分区 .....	23
<b>第 4 章 重新分区磁盘策略 .....</b>	<b>25</b>
4.1. 使用未分区的空闲空间 .....	25
4.2. 使用未使用分区中的空间 .....	25
4.3. 使用活跃分区中的空闲空间 .....	26
<b>第 5 章 持久性命名属性概述 .....</b>	<b>30</b>
5.1. 非持久性命名属性的缺陷 .....	30
5.2. 文件系统和设备识别符 .....	30
5.3. 使用 /DEV/DISK/ 中的 UDEV 机制管理的设备名称 .....	31
5.4. 使用 DM 多路径的通用识别符 .....	33
5.5. UDEV 设备命名规则的限制 .....	33
5.6. 列出持久性命名属性 .....	34
5.7. 修改持久性命名属性 .....	35
<b>第 6 章 使用 NVDIMM 持久性内存存储 .....</b>	<b>37</b>
6.1. NVDIMM 持久内存技术 .....	37
6.2. NVDIMM 交集和地区 .....	37
6.3. NVDIMM 命名空间 .....	38
6.4. NVDIMM 访问模式 .....	38
6.5. 安装 NDCTL .....	39
6.6. 在 NVDIMM 上创建扇区命名空间以充当块设备 .....	39
6.7. 在 NVDIMM 上创建设备 DAX 命名空间 .....	42
6.8. 在 NVDIMM 上创建文件系统 DAX 命名空间 .....	47
6.9. 使用 S.M.A.R.T 监控 NVDIMM 健康状况。 .....	52
6.10. 检测和替换断开问题的 NVDIMM 设备 .....	53
<b>第 7 章 丢弃未使用块 .....</b>	<b>57</b>

要求	57
7.1. 块丢弃操作的类型	57
7.2. 执行批块丢弃	57
7.3. 启用在线块丢弃	58
7.4. 使用 STORAGE RHEL 系统角色启用在线块丢弃	58
7.5. 启用定期块丢弃	59
<b>第 8 章 配置 ISCSI 目标</b>	<b>61</b>
8.1. 安装 TARGETCLI	61
8.2. 创建 ISCSI 目标	62
8.3. ISCSI BACKSTORE	63
8.4. 创建 FILEIO 存储对象	63
8.5. 创建块存储对象	64
8.6. 创建 PSCSI 存储对象	64
8.7. 创建内存副本 RAM 磁盘存储对象	65
8.8. 创建 ISCSI 门户	66
8.9. 创建 ISCSI LUN	68
8.10. 创建只读 ISCSI LUN	69
8.11. 创建 ISCSI ACL	71
8.12. 为目标设置 CHALLENGE-HANDSHAKE 验证协议	72
8.13. 使用 TARGETCLI 工具删除 ISCSI 对象	73
<b>第 9 章 配置 ISCSI INITIATOR</b>	<b>76</b>
9.1. 创建 ISCSI 启动程序	76
9.2. 为发起方设置 CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL	78
9.3. 使用 ISCSIADM 工具监控 ISCSI 会话	79
9.4. DM 多路径覆盖设备超时	80
<b>第 10 章 使用光纤通道设备</b>	<b>82</b>
10.1. 在调整 LUN 的大小后重新扫描光纤通道逻辑单元	82
10.2. 使用光纤通道确定设备链路丢失行为	82
10.3. FIBRE CHANNEL 配置文件	83
<b>第 11 章 使用以太网配置光纤</b>	<b>87</b>
11.1. 在 RHEL 中使用硬件 FCOE HBA	87
11.2. 设置 FCOE 设备	87
<b>第 12 章 使用 EH_DEADLINE 配置存储错误恢复的最大时间</b>	<b>91</b>
12.1. EH_DEADLINE 参数	91
12.2. 设置 EH_DEADLINE 参数	92
<b>第 13 章 SWAP 入门</b>	<b>94</b>
13.1. SWAP 空间概述	94
13.2. 推荐的系统 SWAP 空间	94
13.3. 为 SWAP 创建 LVM2 逻辑卷	95
13.4. 创建交换文件	96
13.5. 使用 STORAGE RHEL 系统角色创建交换卷	98
13.6. 在 LVM2 逻辑卷中扩展 SWAP	99
13.7. 在 LVM2 逻辑卷中减少 SWAP	100
13.8. 为 SWAP 删除 LVM2 逻辑卷	101
13.9. 删除交换文件	102
<b>第 14 章 NVME OVER FABRIC 设备概述</b>	<b>104</b>
<b>第 15 章 使用 NVME/RDMA 配置 NVME OVER FABRICS</b>	<b>105</b>

15.1. 使用 CONFIGFS 设置 NVME/RDMA 控制器	105
15.2. 使用 NVMETCLI 设置 NVME/RDMA 控制器	107
15.3. 配置 NVME/RDMA 主机	108
15.4. 后续步骤	110
<b>第 16 章 使用 NVME/FC 配置 NVME OVER FABRICS</b>	<b>111</b>
16.1. 为广播适配器配置 NVME 主机	111
16.2. 为 QLOGIC 适配器配置 NVME 主机	113
16.3. 后续步骤	116
<b>第 17 章 在 NVME 设备中启用多路径</b>	<b>117</b>
17.1. 本地 NVME 多路径和 DM 多路径	117
17.2. 启用原生 NVME 多路径	117
17.3. 在 NVME 设备中启用 DM 多路径	120
<b>第 18 章 设置磁盘调度程序</b>	<b>124</b>
18.1. 可用磁盘调度程序	124
18.2. 不同用例的磁盘调度程序	125
18.3. 默认磁盘调度程序	126
18.4. 确定活跃磁盘调度程序	126
18.5. 使用 TUNED 设置磁盘调度程序	127
18.6. 使用 UDEV 规则设置磁盘调度程序	129
18.7. 为特定磁盘临时设置调度程序	131
<b>第 19 章 设置远程无盘系统</b>	<b>132</b>
19.1. 为远程无盘系统准备环境	132
19.2. 为无盘客户端配置 TFTP 服务	133
19.3. 为无盘客户端配置一个 DHCP 服务器	135
19.4. 为无盘客户端配置导出的文件系统	136
19.5. 重新配置远程无盘系统	139
19.6. 排查与加载远程无盘系统相关的常见问题	140
<b>第 20 章 管理 RAID</b>	<b>143</b>
20.1. RAID 概述	143
20.2. RAID 类型	143
20.3. RAID 级别和线性支持	146
20.4. 支持的 RAID 转换	148
20.5. RAID 子系统	150
20.6. 在安装过程中创建软件 RAID	150
20.7. 在安装的系统中创建软件 RAID	152
20.8. 在 WEB 控制台中创建 RAID	154
20.9. 在 WEB 控制台中格式化 RAID	155
20.10. 使用 WEB 控制台在 RAID 上创建分区表	157
20.11. 使用 WEB 控制台在 RAID 上创建分区	159
20.12. 使用 WEB 控制台在 RAID 上创建卷组	161
20.13. 使用 STORAGE RHEL 系统角色配置一个 RAID 卷	162
20.14. 扩展 RAID	164
20.15. 缩小 RAID	165
20.16. 安装后将根磁盘转换为 RAID1	166
20.17. 创建高级 RAID 设备	167
20.18. 设置用于监控 RAID 的电子邮件通知	168
20.19. 替换 RAID 中失败的磁盘	169
20.20. 修复 RAID 磁盘	171
<b>第 21 章 使用 LUKS 加密块设备</b>	<b>173</b>

21.1. LUKS 磁盘加密	173
21.2. RHEL 中的 LUKS 版本	174
21.3. LUKS2 重新加密过程中数据保护选项	176
21.4. 使用 LUKS2 加密块设备上的现有数据	177
21.5. 使用带有分离标头的 LUKS2 在块设备上加密现有数据	180
21.6. 使用 LUKS2 加密空白块设备	183
21.7. 在 WEB 控制台中配置 LUKS 密码短语	185
21.8. 在 WEB 控制台中更改 LUKS 密码短语	187
21.9. 使用命令行更改 LUKS 密码短语	188
21.10. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密的卷	189
<b>第 22 章 管理磁带设备</b>	<b>193</b>
22.1. 磁带设备的类型	193
22.2. 安装磁带驱动器管理工具	193
22.3. 磁带命令	193
22.4. 写入如回卷磁带设备	194
22.5. 写入非回卷解磁带设备	196
22.6. 在磁带设备中切换磁带头	198
22.7. 从磁带设备中恢复数据	199
22.8. 从磁带设备中删除数据	200
<b>第 23 章 删除存储设备</b>	<b>202</b>
23.1. 安全删除存储设备	202
23.2. 删除块设备和相关元数据	203
<b>第 24 章 设置 STRATIS 文件系统</b>	<b>208</b>
24.1. STRATIS 文件系统的组件	209
24.2. 与 STRATIS 兼容的块设备	210
24.3. 安装 STRATIS	211
24.4. 创建未加密的 STRATIS 池	211
24.5. 使用 WEB 控制台创建未加密的 STRATIS 池	213
24.6. 使用内核密钥环中的密钥创建加密的 STRATIS 池	214
24.7. 使用 WEB 控制台创建加密的 STRATIS 池	216
24.8. 使用 WEB 控制台重命名 STRATIS 池	219
24.9. 在 STRATIS 文件系统中设置过度置备模式	220
24.10. 将 STRATIS 池绑定到 NBDE	221
24.11. 将 STRATIS 池绑定到 TPM	222
24.12. 使用内核密钥环解加密的 STRATIS 池	223
24.13. 解除 STRATIS 池与补充加密的绑定	224
24.14. 启动和停止 STRATIS 池	225
24.15. 创建 STRATIS 文件系统	226
24.16. 使用 WEB 控制台在 STRATIS 池中创建文件系统	227
24.17. 挂载 STRATIS 文件系统	229
24.18. 使用 SYSTEMD 服务在 /ETC/FSTAB 中设置非 ROOT STRATIS 文件系统	230
<b>第 25 章 使用附加块设备扩展 STRATIS 池</b>	<b>231</b>
25.1. 在 STRATIS 池中添加块设备	231
25.2. 使用 WEB 控制台在 STRATIS 池中添加块设备	231
<b>第 26 章 监控 STRATIS 文件系统</b>	<b>234</b>
26.1. 显示 STRATIS 文件系统的信息	234
26.2. 使用 WEB 控制台查看 STRATIS 池	235
<b>第 27 章 在 STRATIS 文件系统中使用快照</b>	<b>237</b>
27.1. STRATIS 快照的特性	237



27.2. 创建 STRATIS 快照	237
27.3. 访问 STRATIS 快照的内容	238
27.4. 将 STRATIS 文件系统恢复到以前的快照	239
27.5. 删除 STRATIS 快照	240
<b>第 28 章 删除 STRATIS 文件系统 .....</b>	<b>241</b>
28.1. 删除 STRATIS 文件系统	241
28.2. 使用 WEB 控制台从 STRATIS 池中删除文件系统	242
28.3. 删除 STRATIS 池	243
28.4. 使用 WEB 控制台删除 STRATIS 池	244



---

## 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**。

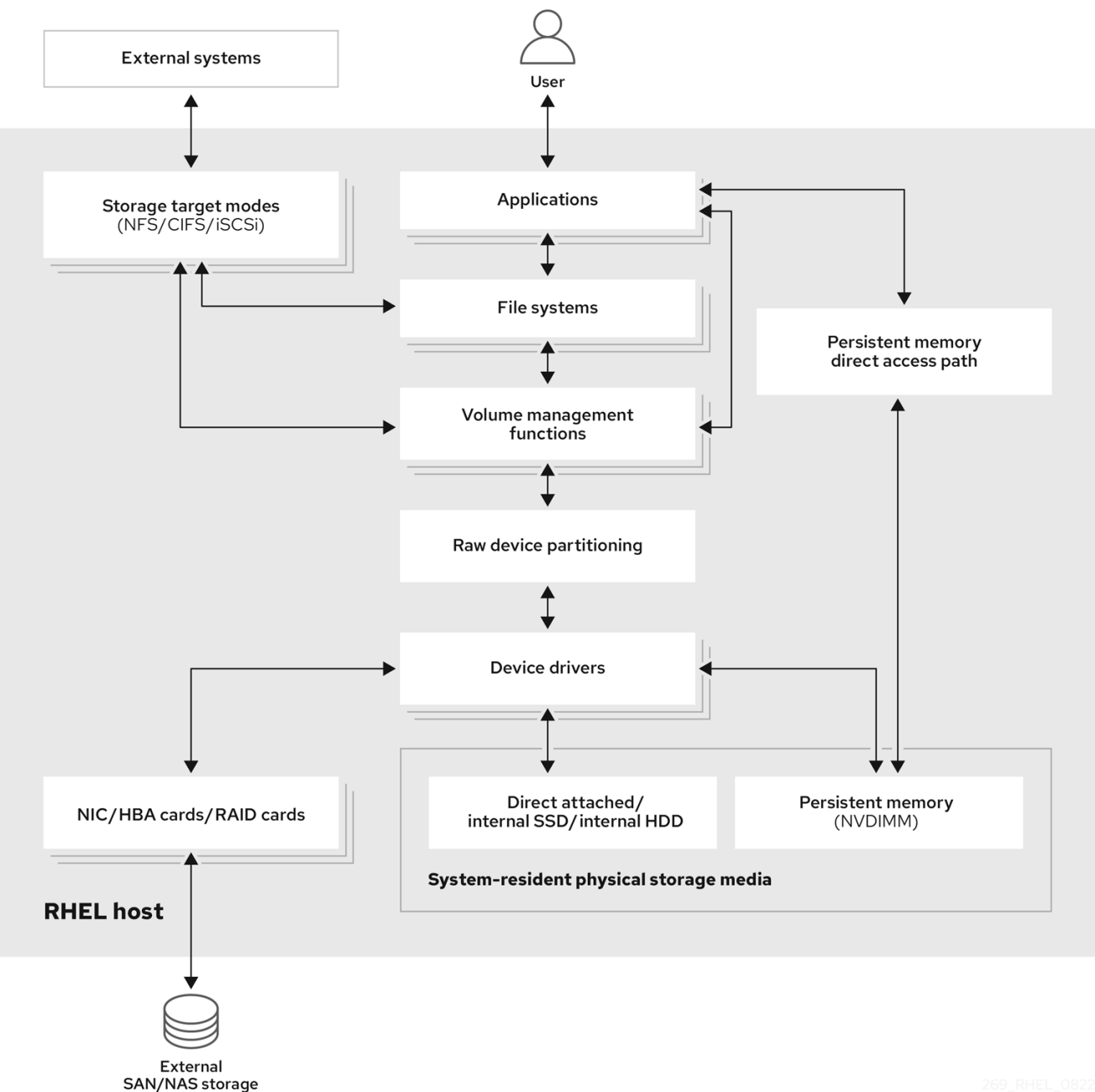
# 第 1 章 可用存储选项概述

RHEL 8 上提供了几个本地、远程和基于集群的存储选项。

本地存储意味着存储设备安装在系统上，或者直接附加到系统。

使用远程存储时，设备可以通过 LAN、互联网或使用光纤通道网络进行访问。以下高级别 Red Hat Enterprise Linux 存储图描述了不同的存储选项。

图 1.1. Red Hat Enterprise Linux 存储图



269\_RHEL\_0822

## 1.1. 本地存储概述

Red Hat Enterprise Linux 8 提供多个本地存储选项。

### 基本磁盘管理

使用 **parted** 和 **fdisk**，您可以创建、修改、删除和查看磁盘分区。以下是分区布局标准：

## 主引导记录 (MBR)

它用于基于 BIOS 的计算机。您可以创建主分区、扩展分区及逻辑分区。

## GUID 分区表 (GPT)

它使用全局唯一标识符(GUID)并提供唯一的磁盘和分区 GUID。

## 存储消耗选项

### Non-Volatile Dual In-line Memory Modules (NVDIMM) 管理

它是内存和存储的组合。您可以在连接到您的系统的 NVDIMM 设备中启用和管理各种存储类型。

### 块存储管理

数据以块形式存储，每个块都有唯一的标识符。

### 文件存储

数据存储在本机系统中的文件级别。这些数据可使用 XFS（默认）或者 ext4 本地访问，并使用 NFS 和 SMB 通过网络访问。

## 逻辑卷

### 逻辑卷管理器 (LVM)

它从物理设备中创建逻辑设备。逻辑卷(LV)是物理卷(PV)和卷组(VG)的组合。

### Virtual Data Optimizer (VDO)

它被用来通过重复数据删除（deduplication）、压缩和精简置备来减少数据。在 LV 下使用 VDO 可以帮助：

- 扩展 VDO 卷
- 通过多个设备生成 VDO 卷

## 本地文件系统

### XFS

默认 RHEL 文件系统。

### ext4

传统的文件系统。

### Stratis

它只是一个技术预览。Stratis 是一个支持高级存储功能的混合用户和内核本地存储管理系统。

## 1.2. 远程存储概述

以下是 RHEL 8 中可用的远程存储选项：

## 存储连接选项

### iSCSI

RHEL 8 使用 targetcli 工具来添加、删除、查看和监控 iSCSI 存储间的连接。

### 光纤频道 (FC)

RHEL 8 提供以下原生 Fibre Channel 驱动程序：

- lpfc

- **qla2xxx**
- **Zfcp**

### Non-volatile Memory Express (NVMe)

允许主机软件实用程序与固态驱动器进行通信的接口。使用以下类型的光纤传输来通过光纤配置 NVMe：

- 使用 Remote Direct Memory Access (NVMe/RDMA)的 NVMe over fabrics
- 使用光纤通道(NVMe/FC)的 NVMe over fabrics

### 设备映射器多路径 (DM 多路径)

允许您将服务器节点和存储阵列间的多个 I/O 路径配置为单一设备。这些 I/O 路径是可包含独立电缆、交换机和控制器的物理 SAN 连接。

### 网络文件系统

- NFS
- SMB

## 1.3. GFS2 文件系统概述

Red Hat Global File System 2 (GFS2) 是一个 64 位对称集群文件系统，它提供了一个共享名称空间，并管理共享一个常见块设备的多个节点间的一致性。GFS2 文件系统旨在提供尽量与本地文件系统类似的功能，同时在节点间强制实施完整集群一致性。为达到此目的，节点在文件系统资源中使用集群范围的锁定方案。这个锁定方案使用 TCP/IP 等通讯协议来交换锁定信息。

在某些情况下，Linux 文件系统 API 不允许具有集群特性的 GFS2 完全透明。例如，在 GFS2 中使用 POSIX 锁定的程序应该避免使用 **GETLK** 功能，因为在集群的环境中，该进程 ID 可能用于集群中的不同节点。然而，多数情况下 GFS2 文件系统的功能和本地文件系统的功能是一样的。

Red Hat Enterprise Linux Resilient Storage Add-On 提供 GFS2，它依赖于 Red Hat Enterprise Linux High Availability 附加组件来提供 GFS2 所需的集群管理。

**gfs2.ko** 内核模块实现 GFS2 文件系统，并加载在 GFS2 集群节点上。

要获得最佳 GFS2 性能，请务必考虑基础设计中给出的性能注意事项。和本地文件系统一样，GFS2 依赖于页面缓存以便通过本地缓存来提高经常使用数据的性能。为了在集群中的节点间保持一致性，缓存控制由 *glock* 状态机器提供。

### 其他资源

- [配置 GFS2 文件系统](#)

## 第 2 章 磁盘分区

要将磁盘分成一个或多个逻辑区域，请使用磁盘分区实用程序。这可以对每个分区进行独立的管理。

### 2.1. 分区概述

硬盘在分区表中保存每个磁盘分区的位置和大小信息。使用分区表中的信息，操作系统会将每个分区视为一个逻辑磁盘。磁盘分区的一些优点包括：

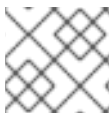
- 减少对物理卷的管理
- 保证有足够的备份
- 提供有效的磁盘管理

#### 其他资源

- [在 LUN 上使用分区（直接或通过 LVM）有哪些优点和缺点？](#)（红帽知识库）

### 2.2. 分区表类型比较

要在设备中启用分区，使用不同类型的分区表格式化块设备。下表比较您可以在块设备中创建的不同类型的分区表的属性。



#### 注意

本节不涵盖 DASD 分区表，它特定于 IBM Z 构架。

表 2.1. 分区表类型

分区表	最多分区数	最大分区大小
主引导记录（MBR）	4 个主分区，或 3 个主分区和 1 个扩展分区（带有 12 个逻辑分区）	2 TiB 如果使用 512 b sector 驱动器 16 TiB （如果使用 4 个 k sector 驱动器）
GUID 分区表（GPT）	128	8 ZiB 如果使用 512 b sector 驱动器 64 ZiB （如果使用 4 个 k sector 驱动器）

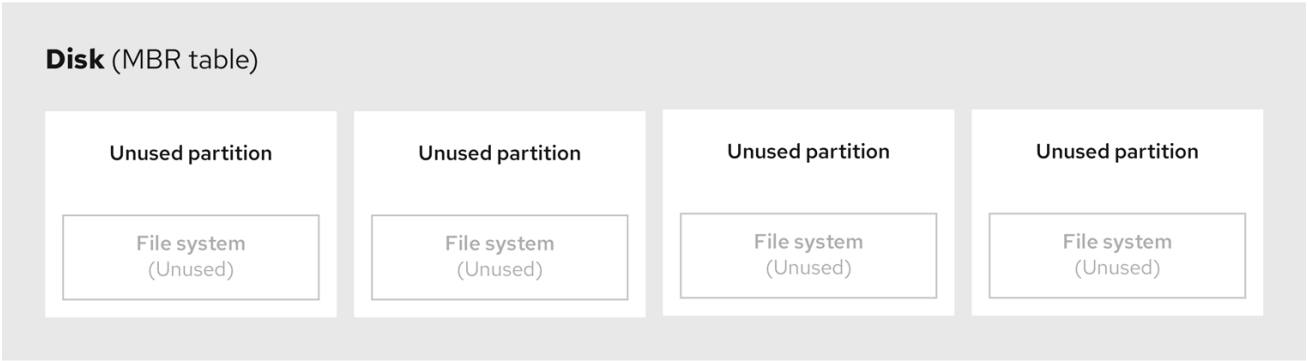
#### 其他资源

- [在 IBM Z 中配置 Linux 实例](#)
- [您需要了解的 DASD 的信息](#)

### 2.3. MBR 磁盘分区

分区表保存在磁盘的最开始，任何文件系统或用户数据都会保存在它的后面。为了更明确的显示，下图中的不同分区表分开显示。

图 2.1. 有 MBR 分区表的磁盘



269\_RHEL\_0822

如上图所示，分区表被分为四个未使用主分区的四个部分。主分区是硬盘上仅包含一个逻辑驱动器（或部分）的分区。每个逻辑驱动器都有定义单个分区所需的信息，这意味着分区表可以定义不超过四个主分区。

每个分区表条目都包含分区的重要特性：

- 磁盘上分区启动和结束的点
- 分区的状态，因为只有一个分区可以被标记为 **活跃分区**
- 分区的类型

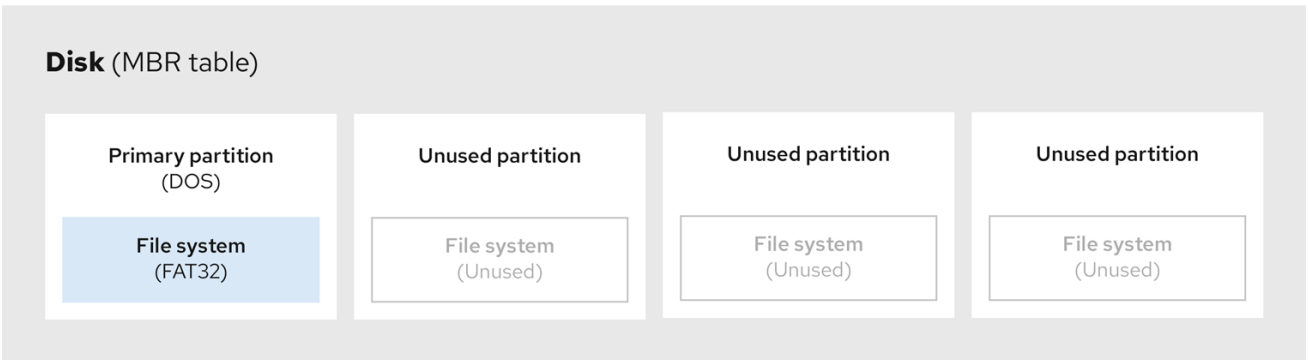
起点和结束点定义了磁盘上分区的大小和位置。有些操作系统引导装载程序使用 **active** 标记。这意味着，在标记为"活跃"的分区中的操作系统被引导。

类型由一个数字代表，用于识别分区预定的使用情况。有些操作系统使用分区类型来：

- 表示特定的文件系统类型
- 将分区标记为与特定操作系统关联的
- 指明分区包含可引导操作系统

下图显示了含有单一分区的驱动器示例：在这个示例中，第一个分区被标记为 **DOS** 分区类型：

图 2.2. 只有一个分区的磁盘



269\_RHEL\_0822



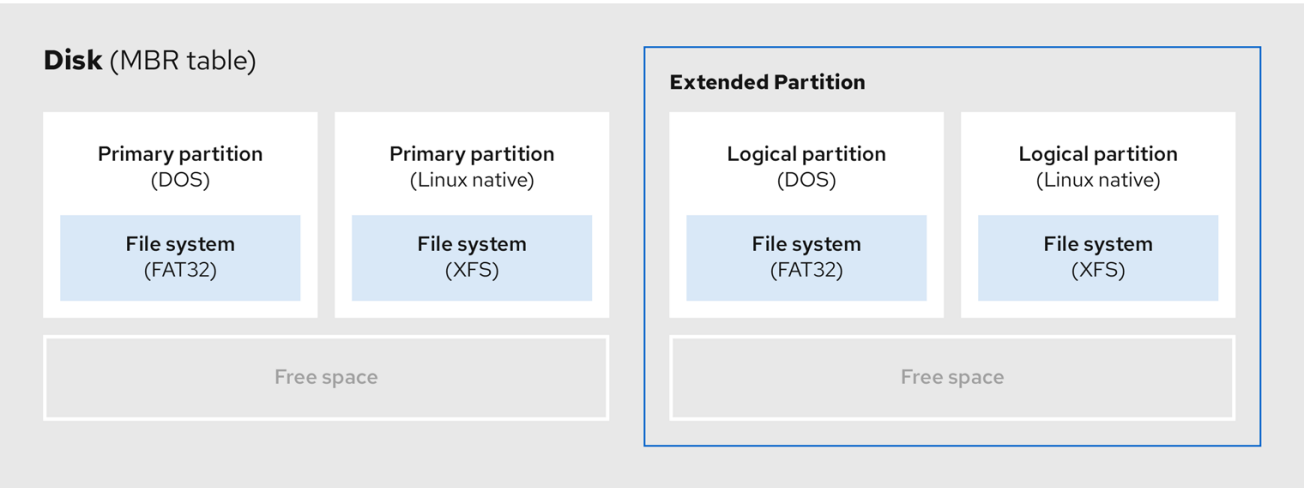
- MBR 分区类型

2.4. 扩展 MBR 分区

要创建额外分区（如果需要），请将类型设置为 **extended**。

扩展分区类似于磁盘驱动器。它有自己的分区表，指向一个或多个逻辑分区，完全包含在扩展分区中。下图显示了一个磁盘驱动器，它有两个主分区和一个包含两个逻辑分区（以及一些未分区的可用空间）的扩展分区：

图 2.3. 带两个主分区和扩展的 MBR 分区的磁盘



269\_RHEL\_0822

最多只能有 4 个主分区和扩展分区，但逻辑分区的数量没有固定限制。作为 Linux 中的限制访问分区，单个磁盘驱动器允许最多 15 个分区。

2.5. MBR 分区类型

下表显示了一些最常用的 MBR 分区类型和用于代表它们的十六进制数字。

表 2.2. MBR 分区类型

MBR 分区类型	值	MBR 分区类型	值
空	00	Novell Netware 386	65
DOS 12-bit FAT	01	PIC/IX	75
XENIX root	02	Old MINIX	80
XENIX usr	03	Linux/MINUX	81
DOS 16-bit ≤32M	04	Linux swap	82
Extended	05	Linux native	83

DOS 16-bit >=32	06	Linux extended	85
OS/2 HPFS	07	Amoeba	93
AIX	08	Amoeba BBT	94
AIX bootable	09	BSD/386	a5
OS/2 Boot Manager	0a	OpenBSD	a6
Win95 FAT32	0b	NEXTSTEP	a7
Win95 FAT32(LBA)	0c	BSDI fs	b7
Win95 FAT16(LBA)	0e	BSDI swap	b8
Win95 Extended (LBA)	0f	Syrinx	c7
Venix 80286	40	CP/M	db
Novell	51	DOS access	e1
PRep Boot	41	DOS R/O	e3
GNU HURD	63	DOS secondary	f2
Novell Netware 286	64	BBT	ff

## 2.6. GUID 分区表

GUID 分区表(GPT)是基于全局唯一标识符(GUID)的分区方案。

GPT 处理 Master Boot Record (MBR)分区表的限制。MBR 分区表无法处理大于 2 TiB 的存储，相当于大约 2.2 TB。相反，GPT 支持容量较大的硬盘。使用 512b 扇区驱动器时，最大可寻址磁盘大小为 8 ZiB，在使用 4096b 扇区驱动器时，为 64 ZiB。另外，默认情况下，GPT 支持创建最多 128 个主分区。通过向分区表分配更多空间来扩展主分区的最大数量。



### 注意

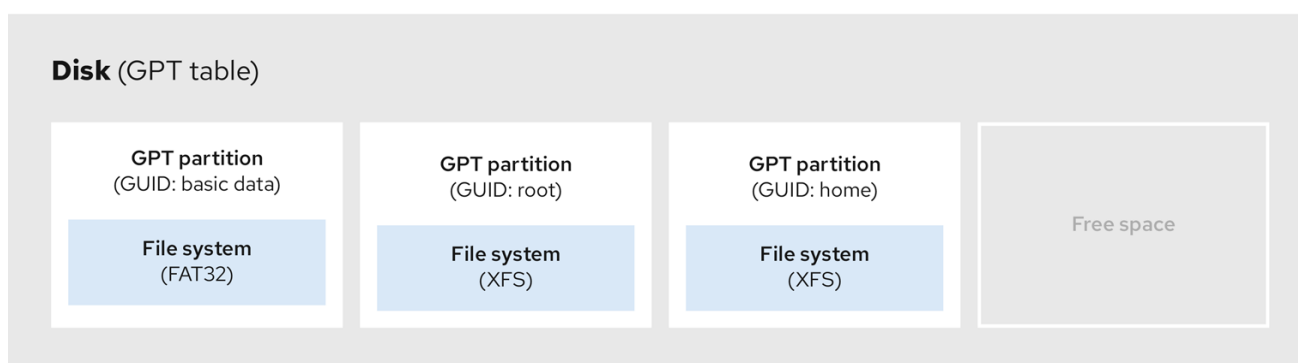
GPT 有基于 GUID 的分区类型。某些分区需要特定的 GUID。例如，可扩展固件接口(EFI)引导装载程序的系统分区需要 GUID **C12A7328-F81F-11D2-BA4B-00A0C93EC93B**。

GPT 磁盘使用逻辑块寻址(LBA)和分区布局，如下所示：

- 为了向后兼容 MBR 磁盘，系统为 MBR 数据保留 GPT 的第一个扇区(LBA 0)，并应用名称"protective MBR"。
- 主 GPT

- 标头从设备的第二个逻辑块(LBA 1)开始。标头中包含磁盘 GUID、主分区表的位置、从属 GPT 标头的位置、自身的 CRC32 checksum 以及主分区表。它还指定表上的分区条目数目。
- 默认情况下，主 GPT 包含 128 个分区条目。每个分区的条目大小为 128 字节、分区类型 GUID 和唯一的分区 GUID。
- 二级 GPT
  - 对于恢复，当主分区表格损坏时，备份表非常有用。
  - 磁盘的最后一个逻辑扇区包含二级 GPT 标头并恢复 GPT 信息（如果主标头损坏）。
  - 它包含：
    - 磁盘 GUID
    - 二级分区表和主 GPT 标头的位置
    - 自身的 CRC32 checksums
    - 二级分区表
    - 可能的分区条目数

图 2.4. 带有 GUID 分区表的磁盘



269\_RHEL\_0822



### 重要

对于在 GPT 磁盘上成功安装引导装载程序，必须存在 BIOS 引导分区。只有当磁盘已包含 BIOS 引导分区时，才可以重复使用。这包括 **Anaconda** 安装程序初始化的磁盘。

## 2.7. 分区类型

管理分区类型的方法有多种：

- **fdisk** 工具通过指定十六进制代码来支持所有的分区类型。
- **systemd-gpt-auto-generator**（单位生成器程序程序）使用分区类型自动识别和挂载设备。
- **parted** 实用程序使用 *flags* 对分区类型进行映射。**parted** 工具只处理某些分区类型，如 LVM、swap 或 RAID。  
**parted** 实用程序支持设置以下标记：

- **boot**
- **root**
- **swap**
- **hidden**
- **raid**
- **lvm**
- **lba**
- **legacy\_boot**
- **irst**
- **esp**
- **palo**

**parted** 工具在创建分区时接受文件系统类型参数。有关所需条件的列表，[请参阅使用 parted 创建分区](#)。使用值来：

- 在 MBR 中设置分区标记。
- 在 GPT 中设定分区 UUID 类型。例如，**swap**、**fat** 或 **hfs** 文件系统类型设置不同的 GUID。默认值为 Linux Data GUID。

参数不会修改分区中的文件系统。它只会区分受支持的标志和 GUID。

支持以下文件系统类型：

- **xf**s
- **ext2**
- **ext3**
- **ext4**
- **fat16**
- **fat32**
- **hfs**
- **hfs+**
- **linux-swap**
- **ntfs**
- **reiserfs**



### 注意

RHEL 8 中唯一支持的本地文件系统是 **ext4** 和 **xfs**。

## 2.8. 分区命名方案

Red Hat Enterprise Linux 使用基于文件的命名方案，其文件名采用 **/dev/xyN** 的形式。

设备和分区名称由以下结构组成：

**/dev/**

包含所有设备文件的目录的名称。硬盘包含分区，因此代表所有可能分区的文件位于 **/dev** 中。

**xx**

分区名的前两个字母标明包含该分区的设备类型。

**y**

这个字母标明包含分区的特定设备。例如：**/dev/sda** 用于第一个硬盘，第二个硬盘为 **/dev/sdb**。您可以在超过 26 个驱动器的系统中使用更多字母，例如 **/dev/sdaa1**。

**N**

最后的字母代表分区的数字。前四个（主或扩展）分区编号为 **1** 到 **4**。逻辑分区从 **5** 开始。例如，**/dev/sda3** 是第一个硬盘上的第三个主分区或扩展分区，**/dev/sdb6** 是第二个硬盘上的第二个逻辑分区。驱动器分区编号只适用于 MBR 分区表。请注意，**N** 并不总是意味着分区。



### 注意

即使 Red Hat Enterprise Linux 可以识别和引用 *所有* 类型的磁盘分区，它可能无法读取文件系统，因此无法访问分区类型中保存的数据。然而，在很多情况下，成功访问专用于另一个操作系统的分区中的数据是可能的。

## 2.9. 挂载点和磁盘分区

在 Red Hat Enterprise Linux 中，每个分区都被用来成为支持一组文件和目录所必需的存储的一部分。挂载分区会导致该分区的存储可用，从指定目录开始，称为 *挂载点*。

例如，如果分区 **/dev/sda5** 挂载在 **/usr/** 上，这意味着 **/usr/** 下的所有文件和目录都在 **/dev/sda5** 上。文件 **/usr/share/doc/FAQ/txt/Linux-FAQ** 位于 **/dev/sda5** 中，而文件 **/etc/gdm/custom.conf** 不在。

继续这个示例，也可以将 **/usr/** 下的一个或多个目录作为其他分区的挂载点。例如：

**/usr/local/man/whatis** 位于 **/dev/sda7** 上，而不是位于 **/dev/sda5** 上，如果 **/usr/local** 包含挂载的 **/dev/sda7** 分区。

## 第 3 章 分区入门

使用磁盘分区将磁盘分成一个或多个逻辑区域，这些区域可以单独在每个分区上工作。硬盘在分区表中保存每个磁盘分区的位置和大小信息。使用表，每个分区然后显示为操作系统的逻辑磁盘。然后您可以在这些独立磁盘中进行读取和写入。

有关在块设备上使用分区的优点和缺点的概述，请查看红帽知识库解决方案在 [LUN 上使用分区（可以在两者间直接或通过 LVM）有哪些优点和缺点？](#)

### 3.1. 使用 PARTED 在磁盘创建分区表

使用 **parted** 实用程序更轻松地使用分区表格式化块设备。



#### 警告

使用分区表格式化块设备会删除该设备中所有存储的数据。

#### 流程

1. 启动交互式 **parted** shell：

```
# parted block-device
```

2. 确定该设备中是否已有一个分区表：

```
(parted) print
```

如果设备已经包含分区，则后续步骤中将删除它们。

3. 创建新分区表：

```
(parted) mklabel table-type
```

- 使用预期的分区表类型替换 *table-type*：
  - 用于 MBR 的 **msdos**
  - 用于 GPT 的 **gpt**

#### 例 3.1. 创建 GUID 分区表(GPT)表

要在磁盘上创建 GPT 表，请使用：

```
(parted) mklabel gpt
```

在输入以下命令后，这些更改将开始应用。

4. 查看分区表以确认其已创建：

```
(parted) print
```

5. 退出 **parted** shell:

```
(parted) quit
```

## 其他资源

- 您系统上的 **parted (8)** 手册页

## 3.2. 查看使用 PARTED 的分区表

显示块设备的分区表，以查看分区布局和单个分区的详情。您可以使用 **parted** 实用程序查看块设备上的分区表。

### 流程

1. 启动 **parted** 工具。例如：以下输出列出了设备 **/dev/sda**：

```
# parted /dev/sda
```

2. 查看分区表：

```
(parted) print
```

```
Model: ATA SAMSUNG MZNLN256 (scsi)
```

```
Disk /dev/sda: 256GB
```

```
Sector size (logical/physical): 512B/512B
```

```
Partition Table: msdos
```

```
Disk Flags:
```

Number	Start	End	Size	Type	File system	Flags
1	1049kB	269MB	268MB	primary	xfs	boot
2	269MB	34.6GB	34.4GB	primary		
3	34.6GB	45.4GB	10.7GB	primary		
4	45.4GB	256GB	211GB	extended		
5	45.4GB	256GB	211GB	logical		

3. 可选：切换到您要检查的设备：

```
(parted) select block-device
```

有关打印命令输出的详细描述，请查看以下信息：

### 模型：ATA SAMSUNG MZNLN256(scsi)

磁盘类型、制造商、型号号和接口。

### 磁盘 **/dev/sda**: 256GB

块设备的文件路径和存储容量。

### 分区表：**msdos**

磁盘标签类型。

### Number

分区号。例如，次号 1 的分区对应于 **/dev/sda1**。

### Start 和 End

在分区启动和结束的设备中的位置。

### Type

有效类型为 metadata、free、primary、extended 或 logical。

### File system

文件系统类型。如果设备的 **File system** 字段未显示值，这意味着其文件系统类型为未知。**parted** 工具无法识别加密设备上的文件系统。

### 标记

列出为分区设置的标记。可用的标志有 **boot**、**root**、**swap**、**hidden**、**raid**、**lvm** 或 **lba**。

### 其他资源

- 您系统上的 **parted (8)** 手册页

## 3.3. 使用 PARTED 创建分区

作为系统管理员，您可以使用 **parted** 实用程序在磁盘上创建新分区。



### 注意

所需分区是 **swap**、**/boot/** 和 **/(root)**。

### 先决条件

- 磁盘上的分区表。
- 如果要创建的分区大于 2TiB，使用 **GUID 分区表(GPT)** 格式化磁盘。

### 流程

1. 启动 **parted** 工具：

```
# parted block-device
```

2. 查看当前的分区表来确定是否有足够空闲空间：

```
(parted) print
```

- 如果分区没有足够的可用空间，则调整分区大小。
- 从分区表中决定：
  - 新分区的开始和结束点。
  - 在 MBR 上，应该是什么分区类型。

3. 创建新分区：



```
(parted) mkpart part-type name fs-type start end
```

- 将 *part-type* 替换为 **primary**, **logical**, 或 **extended**。这只适用于 MBR 分区表。
- 使用任意分区名称替换 *name*。对于 GPT 分区表，这是必需的。
- 将 *fs-type* 替换为 **xfs**, **ext2**, **ext3**, **ext4**, **fat16**, **fat32**, **hfs**, **hfs+**, **linux-swaps**, **ntfs**, 或 **reiserfs**。*fs-type* 参数是可选的。请注意，**parted** 实用程序不会在分区中创建文件系统。
- 使用从磁盘开头计算分区开始和结束点的大小替换 *start* 和 *end*。您可以使用大小后缀，如 **512MiB**、**20GiB** 或 **1.5TiB**。默认的大小是 MB。

### 例 3.2. 创建小的主分区

要从 1024MiB 创建主分区，直到 MBR 表中的 2048MiB，请使用：

```
(parted) mkpart primary 1024MiB 2048MiB
```

在输入以下命令后，这些更改开始应用。

4. 查看分区表以确认创建的分区位于分区表中，并具有正确的分区类型、文件系统类型和大小：

```
(parted) print
```

5. 退出 **parted** shell:

```
(parted) quit
```

6. 注册新设备节点：

```
# udevadm settle
```

7. 验证内核是否识别了新的分区：

```
# cat /proc/partitions
```

### 其他资源

- 您系统上的 **parted (8)** 手册页
- [使用 parted 在磁盘上创建分区表](#)
- [使用 parted 重新定义分区大小](#)

## 3.4. 使用 fdisk 设置分区类型

您可以使用 **fdisk** 实用程序设置分区类型或标志。

### 先决条件

- 磁盘上的一个分区。

## 流程

1. 启动交互式 **fdisk** shell :

```
# fdisk block-device
```

2. 查看当前的分区表以确定副分区号码 :

```
Command (m for help): print
```

您可以在 **Type** 列中看到当前的分区类型，在 **Id** 列中看到相应的类型 ID 。

3. 输入分区类型命令并使用它的副号码选择分区 :

```
Command (m for help): type
Partition number (1,2,3 default 3): 2
```

4. 可选：查看十六进制代码列表 :

```
Hex code (type L to list all codes): L
```

5. 设置分区类型 :

```
Hex code (type L to list all codes): 8e
```

6. 写入更改并退出 **fdisk** shell :

```
Command (m for help): write
The partition table has been altered.
Syncing disks.
```

7. 验证您的更改 :

```
# fdisk --list block-device
```

## 3.5. 使用 PARTED 重新定义分区大小

使用 **parted** 工具扩展分区以使用未使用的磁盘空间，或者缩小分区以将其容量用于不同的目的。

### 先决条件

- 在缩小分区前备份数据。
- 如果要创建的分区大于 2TiB，使用 **GUID 分区表(GPT)** 格式化磁盘。
- 如果您想缩小分区，首先缩小文件系统，使其不大于重新定义大小的分区。



### 注意

XFS 不支持缩小。

## 流程

1. 启动 **parted** 工具：

```
# parted block-device
```

2. 查看当前的分区表：

```
(parted) print
```

从分区表中决定：

- 分区的副号码。
- 调整大小后现有分区的位置和新结束点。

3. 重新定义分区大小：

```
(parted) resizepart 1 2GiB
```

- 使用您要重新定义分区的副号码替换 *1*。
- 将 *2* 替换为确定重新定义重新定义分区大小的新结束点的大小，从磁盘开始计算。您可以使用大小后缀，如 **512MiB**、**20GiB** 或 **1.5TiB**。默认的大小是 MB。

4. 查看分区表以确认调整了大小的分区位于分区表中，且大小正确：

```
(parted) print
```

5. 退出 **parted** shell:

```
(parted) quit
```

6. 验证内核是否注册了新分区：

```
# cat /proc/partitions
```

7. 可选：如果您扩展分区，还要扩展它的文件系统。

## 其他资源

- 您系统上的 **parted (8)** 手册页

## 3.6. 使用 PARTED 删除分区

使用 **parted** 实用程序，您可以删除磁盘分区以释放磁盘空间。

### 流程

1. 启动交互式 **parted** shell：

```
# parted block-device
```

- 使用您要删除分区的设备的路径替换 *block-device*：例如 **/dev/sda**。

2. 查看当前的分区表以确定要删除的分区次号：

```
(parted) print
```

3. 删除分区：

```
(parted) rm minor-number
```

- 使用您要删除的分区副号码替换 *minor-number*。

输入此命令后，这些更改会立即应用。

4. 验证您是否已从分区表中删除了分区：

```
(parted) print
```

5. 退出 **parted** shell:

```
(parted) quit
```

6. 验证内核是否注册分区是否已删除：

```
# cat /proc/partitions
```

7. 如果分区存在，从 **/etc/fstab** 文件中删除分区。找到声明删除的分区行，并将其从文件中删除。

8. 重新生成挂载单元，以便您的系统注册新的 **/etc/fstab** 配置：

```
# systemctl daemon-reload
```

9. 如果您删除了交换分区或删除 LVM 部分，请从内核命令行中删除对分区的所有引用：

- a. 列出活跃内核选项并查看是否有选项引用删除的分区：

```
# grubby --info=ALL
```

- b. 删除引用已删除分区的内核选项：

```
# grubby --update-kernel=ALL --remove-args="option"
```

10. 要在早期引导系统中注册更改，请重建 **initramfs** 文件系统：

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

## 其他资源

- 您系统上的 **parted (8)** 手册页

## 第 4 章 重新分区磁盘策略

重新分区磁盘的方法有多种。包括：

- 有可用的未分区的空闲空间。
- 一个未使用的分区可用。
- 在一个活跃使用的分区中的空闲空间是可用。



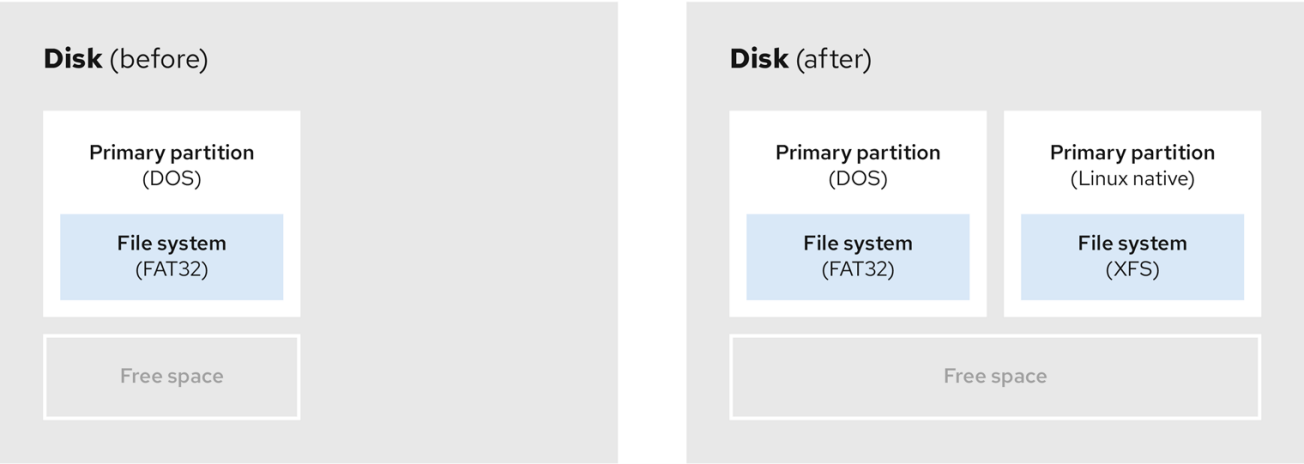
### 注意

为清晰起见，以下示例没有反映在实际安装 Red Hat Enterprise Linux 时的确切分区布局。

### 4.1. 使用未分区的空闲空间

已定义且没有跨越整个硬盘的分区，保留不属于任何定义的分区的未分配空间。下图显示了这种情况。

图 4.1. 有未分区的可用空间的磁盘



269\_RHEL\_0822

第一个图代表一个带有一个主分区的磁盘以及带有未分配空间的未定义分区。第二个图代表有两个已分配空间的分区的磁盘。

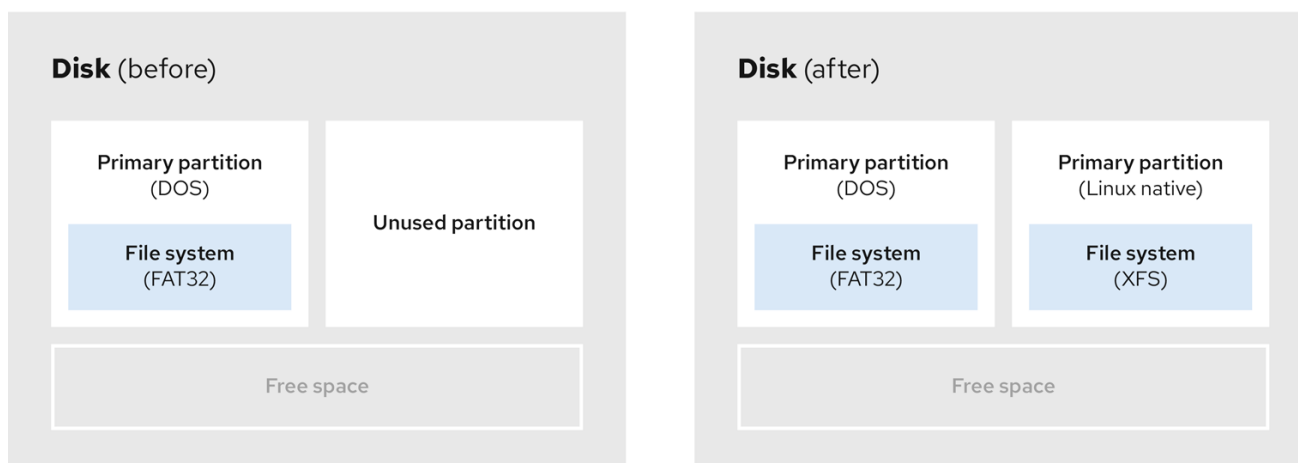
未使用的硬盘也属于这一类别。唯一的区别是，*所有*空间并不是任何定义的分区的一部分。

在新磁盘上，您可以从未使用的空间创建必要的分区。大部分预安装的操作系统都被配置为占据磁盘驱动器上所有可用空间。

### 4.2. 使用未使用分区中的空间

在以下示例中，第一个图代表有未使用分区的磁盘。第二个图代表为 Linux 分配未使用的分区。

图 4.2. 有未使用分区的磁盘



269\_RHEL\_0822

要使用分配给未使用分区的空间，请删除分区，然后创建适当的 Linux 分区。或者，在安装过程中，删除未使用的分区并手动创建新分区。

### 4.3. 使用活跃分区中的空闲空间

因为已经使用的一个活跃分区包含所需的可用空间，所以此过程可能很难管理。在大多数情况下，预安装软件的计算机的硬盘包含一个大型分区，存放操作系统和数据。



#### 警告

如果要在活跃分区中使用操作系统(OS)，您必须重新安装操作系统。请注意，一些计算机可能会包含预安装的软件，不没有提供用于重新安装操作系统的安装介质。在销毁原始分区和操作系统安装前，请检查是否适用于您的操作系统。

要选择使用可用空间，您可以使用破坏性或非破坏性重新分区的方法。

#### 4.3.1. 破坏性重新分区

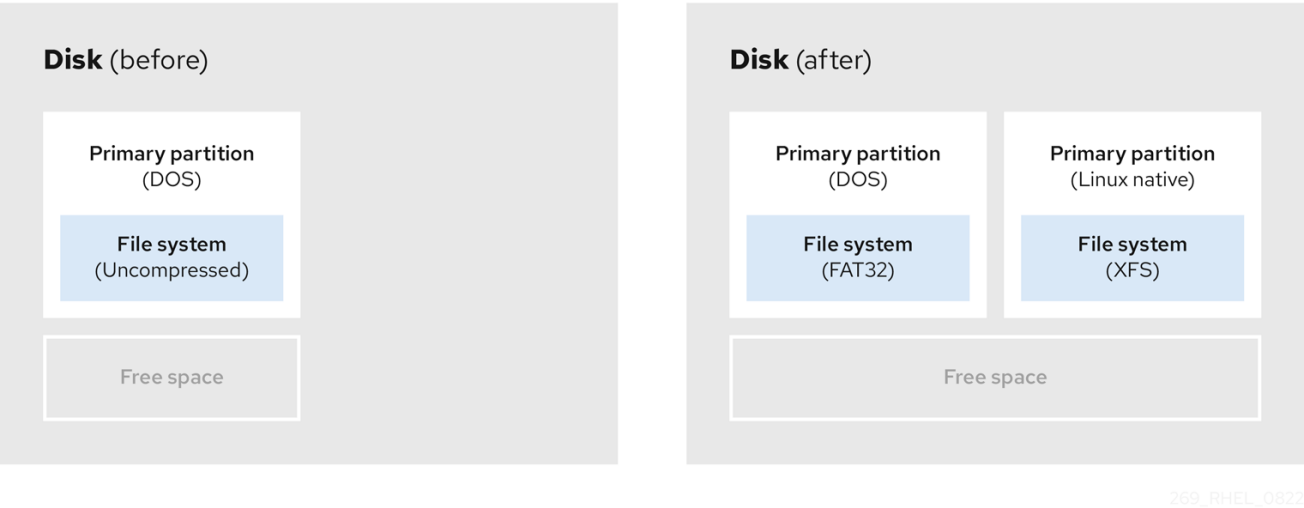
破坏性重新分区破坏硬盘中的分区并创建几个较小的分区。从原始分区备份所有需要的数据，因为此方法会删除完整内容。

为现有操作系统创建一个较小的分区后，您可以：

- 重新安装软件。
- 恢复您的数据。
- 开始您的 Red Hat Enterprise Linux 安装。

下图显示了使用破坏性重新分区方法的简化形式。

图 4.3. 在磁盘上进行破坏性重新分区动作



**警告**

这个方法会删除之前存储在原始分区中的所有数据。

4.3.2. 非破坏性重新分区

非破坏性重新分区分区大小，没有任何数据丢失。这个方法是可靠的，但在大型驱动器上需要更长的处理时间。

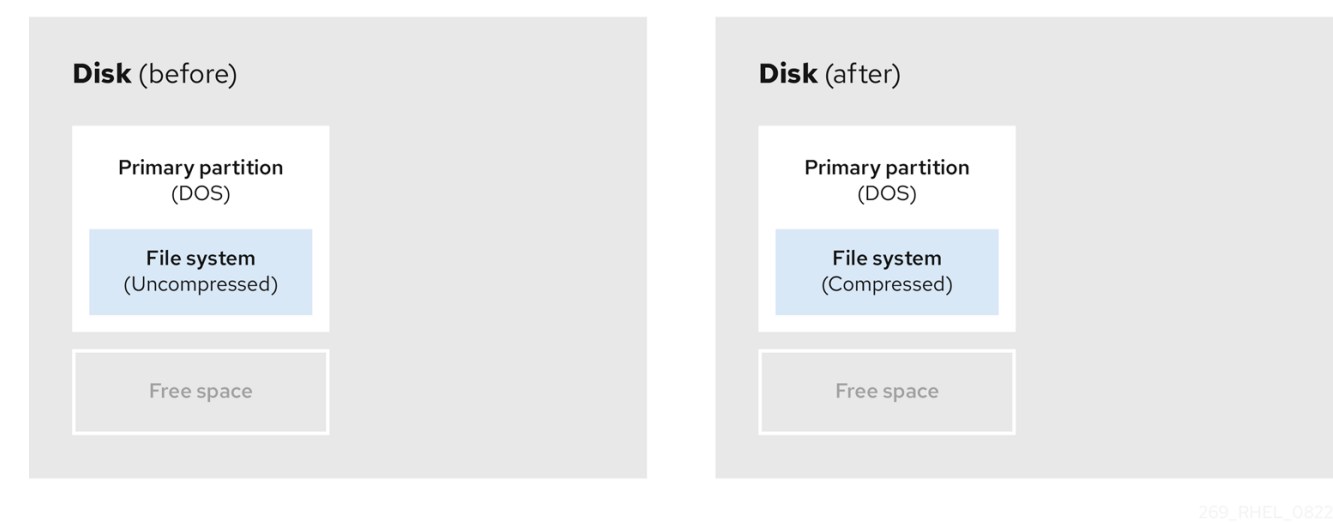
以下是帮助启动非破坏性重新分区的方法列表。

- 压缩现有数据

无法更改部分数据的存储位置。这可以防止分区大小到所需大小，最终导致破坏性重新分区过程。压缩现有分区中的数据可帮助您调整分区的大小。它还有助于最大程度提高可用空间。

下图显示了此过程的简化形式。

图 4.4. 磁盘中的数据压缩



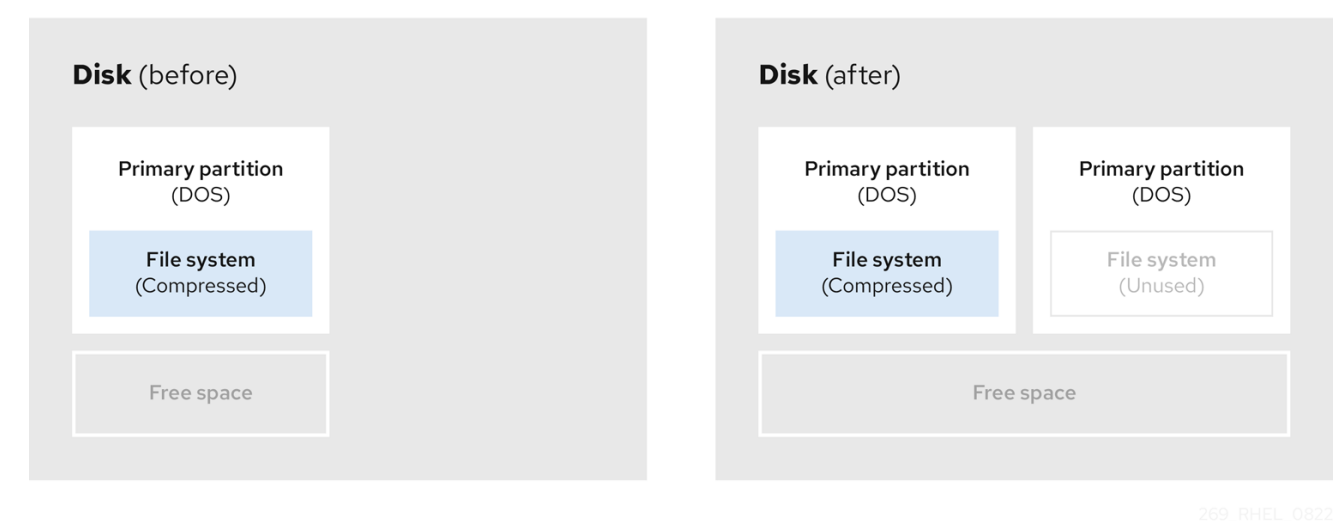
为了避免任何可能的数据丢失，请在继续压缩过程前创建备份。

- 重新划分现存分区的大小

通过重新定义已存在的分区的大小，您可以释放更多空间。根据您的软件重新定义大小，结果可能会有所不同。在大多数情况下，您可以创建同一类型的新未格式化的分区，与原始分区不同。

调整大小后采取的步骤可以取决于您所使用的软件。在以下示例中，最佳实践是删除新的 DOS(Disk Operating System)分区，而是创建一个 Linux 分区。在启动重新定义大小过程前，验证最适合您的磁盘。

图 4.5. 在磁盘上重新定义分区大小



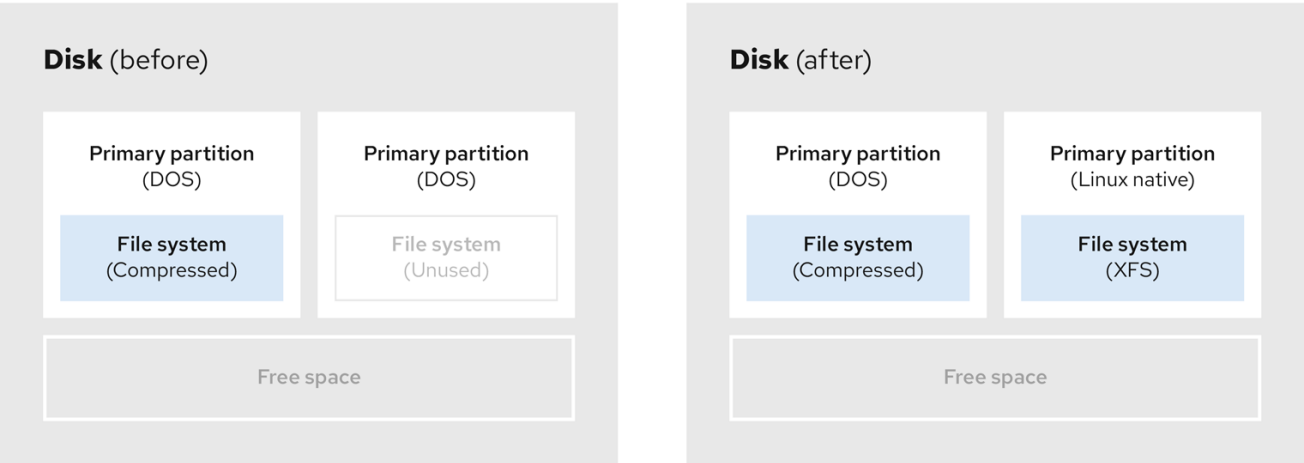
- 可选：创建新分区

一些可以实现重新调整大小的软件会支持基于 Linux 的系统。在这种情况下，在调整大小后不需要删除新创建的分区。之后创建新分区取决于您使用的软件。

下图显示了创建新分区前和之后的磁盘状态。



图 4.6. 带有最终分区配置的磁盘



269\_RHEL\_0822

## 第 5 章 持久性命名属性概述

作为系统管理员，您需要引用使用持久性命名属性的存储卷来构建比多个系统引导更可靠存储设置。

### 5.1. 非持久性命名属性的缺陷

Red Hat Enterprise Linux 提供识别存储设备的多种方法。使用选项时，务必使用正确的选项来识别每个设备，以避免无意中访问错误的设备，特别是在安装到驱动器或重新格式化驱动器时。

通常，在Linux上以 **/dev/sd(主号)(次号)** 形式使用非持久化名称来指向存储设备。当检测设备时，会为每个设备分配主号和次号范围以及相关的 **sd** 名称。这意味着，如果设备的检测顺序发生了变化，主号和次号范围以及相关的 **sd** 名称之间的关联可能也会发生变化。

在以下情况下可能会在以下情况下更改排序：

- 系统引导过程的并行化会根据每个系统引导的顺序检测到存储设备。
- 磁盘无法启动或响应 SCSI 控制器。这会导致通常的设备探测不会检测到它。系统无法访问磁盘，后续设备将下移主号和次号范围，包括相关的 **sd** 名称。例如，如果没有检测到通常称为 **sdb** 的磁盘，则通常称为 **sdc** 的磁盘将显示为 **sdb**。
- SCSI 控制器（主机总线适配器或 HBA）无法初始化，从而导致没有检测到连接到该 HBA 的所有磁盘。任何连接到后续探测到的 HBA 的磁盘都会被分配不同的主号和次号范围，以及不同的相关 **sd** 名称。
- 如果系统中存在不同类型的 HBA，则驱动初始化顺序会改变。这会导致连接到那些 HBA 的磁盘以不同顺序被检测到。当将 HBA 移动到系统的不同 PCI 插槽时也会出现这种情况。
- 例如，由于存储阵列或干预交换机关机，通过光纤通道、iSCSI 或 FCoE 适配器连接到系统的磁盘可能在探测到存储设备时无法访问。如果存储阵列的在线需要比系统启动的时间更长，则系统在电源失败后重启时会出现这种情况。虽然某些光纤通道驱动程序支持一种指定持久性 SCSI 目标 ID 到 WWPN 映射的机制，但这不会导致保留主号和次号范围，以及相关的 **sd** 名称，它只提供一致的 SCSI 目标 ID 号。

这些原因使得在引用设备时，例如在 **/etc/fstab** 文件中，不希望使用主号和次号或相关的 **sd** 名称。可能挂载了错误的设备，并可能导致数据崩溃。

然而，偶尔仍需要引用 **sd** 名称，即使使用了其它机制，比如当设备报告错误时。这是因为 Linux 内核在有关设备的内核消息中使用了 **sd** 名称（以及 SCSI 主机/通道/目标/LUN 元组）。

### 5.2. 文件系统和设备识别符

文件系统标识符与文件系统本身绑定，设备标识符则链接到物理块设备。了解正确的存储管理非常重要。

#### 文件系统识别符

文件系统标识符与在块设备中创建的特定文件系统绑定。标识符也作为文件系统的一部分保存。如果您将文件系统复制到不同的设备中，它仍采用相同的文件系统识别符。但是，如果您重写设备，比如使用 **mkfs** 工具进行格式化，则设备会丢失属性。

文件系统识别符包括：

- 唯一标识符（UUID）
- 标签

## 设备识别符

设备标识符与块设备绑定：例如磁盘或者分区。如果您重写设备，比如使用 **mkfs** 工具对其格式化，则设备会保留属性，因为它不存储在文件系统中。

设备识别符包括：

- World Wide Identifier (WWID)
- 分区 UUID
- 序列号

## 建议

- 有些文件系统（比如逻辑卷）会跨越多个设备。红帽建议您使用文件系统识别符而不是设备标识符访问这些文件系统。

## 5.3. 使用 /DEV/DISK/ 中的 UDEV 机制管理的设备名称

**udev** 机制用于 Linux 中所有设备类型，不仅限于存储设备。它在 **/dev/disk/** 目录中提供不同类型的持久性命名属性。对于存储设备，Red Hat Enterprise Linux 包含 **udev** 规则，该规则在 **/dev/disk/** 目录中创建符号链接。这可让您使用以下方法指向存储设备：

- 其内容
- 唯一标识符
- 它们的序列号。

虽然 **udev** 命名属性是持久的，但它们在系统重启后不会自行更改，有些部分还可配置。

### 5.3.1. 文件系统识别符

#### **/dev/disk/by-uuid/** 中的 UUID 属性

此目录中的条目提供一个符号链接名称，其通过存储在设备上的内容（即数据）中的 **唯一标识符** (UUID) 来指向存储设备。例如：

```
/dev/disk/by-uuid/3e6be9de-8139-11d1-9106-a43f08d823a6
```

您可以使用以下语法，使用 UUID 来指向 **/etc/fstab** 文件中的设备：

```
UUID=3e6be9de-8139-11d1-9106-a43f08d823a6
```

您可以在创建文件系统时配置 UUID 属性，您也可以稍后修改它。

#### **/dev/disk/by-label/** 中的 Label 属性

这个目录中的条目提供了一个符号链接名称，它们使用保存在该设备中的内容（即数据）的一个 **label** 指向存储设备。

例如：

```
/dev/disk/by-label/Boot
```

您可以使用以下语法，使用标签来指向 **/etc/fstab** 文件中的设备：

**LABEL=Boot**

您可以在创建文件系统时配置 Label 属性，您也可以稍后修改它。

5.3.2. 设备识别符

**/dev/disk/by-id/** 中的 WWID 属性  
全球识别符（WWID）是一个持久的、系统独立的标识符，SCSI 标准要求所有 SCSI 设备都使用它。WWID 标识符确保对每个存储设备都是唯一的，并且与用于访问设备的路径无关。标识符是设备的属性，但不存储在设备上的内容（即数据）中。

可通过发出 SCSI 查询来检索设备识别重要产品数据（第 0x83 页）或单元序列号（第 0x80 页）来获取此标识符。

Red Hat Enterprise Linux 自动维护系统上从基于 WWID 的设备名称到当前 **/dev/sd** 名称的正确映射。应用程序可以使用 **/dev/disk/by-id/** 名称来引用磁盘上的数据，即使设备的路径改变了，即使从不同的系统访问设备。

例 5.1. WWID 映射

WWID 符号链接	非持久性设备	备注
<b>/dev/disk/by-id/scsi-3600508b400105e210000900000490000</b>	<b>/dev/sda</b>	具有页 0x83 标识符的设备
<b>/dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6</b>	<b>/dev/sdb</b>	具有页 0x80 标识符的设备
<b>/dev/disk/by-id/ata-SAMSUNG_MZNLN256MHQ-000L7_S2WDNX0J336519-part3</b>	<b>/dev/sdc3</b>	磁盘分区

除了系统提供的这些持久名称外，您也可以使用 **udev** 规则来实现自己的持久名称，映射到存储的 WWID。

**/dev/disk/by-partuuid** 中的分区 UUID 属性  
分区 UUID(PARTUUID)属性标识 GPT 分区表定义的分區。

例 5.2. 分区 UUID 映射

PARTUUID 符号链接	非持久性设备
<b>/dev/disk/by-partuuid/4cd1448a-01</b>	<b>/dev/sda1</b>
<b>/dev/disk/by-partuuid/4cd1448a-02</b>	<b>/dev/sda2</b>
<b>/dev/disk/by-partuuid/4cd1448a-03</b>	<b>/dev/sda3</b>

## **/dev/disk/by-path/ 中的 Path 属性**

此属性提供一个符号链接名称，其通过用于访问设备的 **硬件路径** 指向存储设备。

如果硬件路径的任何部分（如 PCI ID、目标端口或 LUN 号）发生变化，Path 属性会失败。因此 Path 属性是不可靠的。但是 Path 属性在以下情况下可能有用：

- 您需要识别您要替换的磁盘。
- 您计划在特定位置的磁盘安装存储服务。

## **5.4. 使用 DM 多路径的通用识别符**

您可以配置设备映射器(DM)多路径，以在全局识别符(WWID)和非持久性设备名称之间进行映射。

如果系统中有多路径到某个设备，DM 多路径会使用 WWID 探测到这个设备。然后，DM 多路径会在 `/dev/mapper/wwid` 目录中显示一个"pseudo-device"，如 `/dev/mapper/3600508b400105df70000e00000ac0000`。

**multipath -l** 命令显示到非持久性标识符的映射：

- **Host:Channel:Target:LUN**
- **/dev/sd** 名称
- **主：次号**

### **例 5.3. 多路径配置中的 WWID 映射**

**multipath -l** 命令的输出示例：

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
  \_ 5:0:1:1 sdc 8:32 [active][undef]
  \_ 6:0:1:1 sdg 8:96 [active][undef]
\_ round-robin 0 [prio=0][enabled]
  \_ 5:0:0:1 sdb 8:16 [active][undef]
  \_ 6:0:0:1 sdf 8:80 [active][undef]
```

DM 多路径自动维护系统上每个基于 WWID 的设备名称到对应的 `/dev/sd` 名称的正确映射。这些名称可在路径更改之间保留，在从不同系统访问该设备时会保持一致。

当使用 DM 多路径的 **user\_friendly\_names** 功能时，WWID 映射到形式 `/dev/mapper/mpathN` 的名称。默认情况下，此映射在文件 `/etc/multipath/bindings` 中维护。只要该文件得到维护，这些 **mpathN** 名称就会保持不变。



### **重要**

如果使用 **user\_friendly\_names**，则需要额外的步骤来获得集群中的一致名称。

## **5.5. UDEV 设备命名规则的限制**

以下是 **udev** 命名规则的一些限制：

- 执行查询时可能无法访问该设备，因为 **udev** 机制可能依赖于对 **udev** 事件处理 **udev** 规则时查询存储设备的能力。当设备不在服务器机箱中时，这更可能会在光纤频道、iSCSI 或者 FCoE 存储设备中发生。
- 内核可能会随时发送 **udev** 事件，从而导致规则被处理，并可能导致 **/dev/disk/by-\*/** 链接在设备无法访问时被删除。
- 在生成 **udev** 事件和事件被处理时可能会有延迟，如检测到大量设备，以及用户空间 **udev** 服务需要花费一些时间来处理每个事件的规则时。这可能会在内核检测到该设备和在 **/dev/disk/by-\*/** 名称可用之间造成延迟。
- 规则调用的 **blkid** 等外部程序可能会打开设备一段时间，使设备无法用于其他用途。
- **/dev/disk/** 中 **udev** 机制管理的设备名称可能会在主版本之间有所变化，需要您更新链接。

## 5.6. 列出持久性命名属性

您可以找到非持久性存储设备的持久性命名属性。

### 流程

- 要列出 UUID 和 Label 属性，请使用 **lsblk** 工具：

```
$ lsblk --fs storage-device
```

例如：

#### 例 5.4. 查看文件系统的 UUID 和标签

```
$ lsblk --fs /dev/sda1
```

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
sda1	xf	Boot	afa5d5e3-9050-48c3-acc1-bb30095f3dc4	/boot

- 要列出 PARTUUID 属性，请使用 **lsblk** 工具和 **--output +PARTUUID** 选项：

```
$ lsblk --output +PARTUUID
```

例如：

#### 例 5.5. 查看分区的 PARTUUID 属性

```
$ lsblk --output +PARTUUID /dev/sda1
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT	PARTUUID
sda1	8:1	0	512M	0	part	/boot	4cd1448a-01

- 要列出 WWID 属性，请检查 **/dev/disk/by-id/** 目录中符号链接的目标。例如：

**例 5.6. 查看系统中所有存储设备的 WWID**

```
$ file /dev/disk/by-id/*

/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001
symbolic link to ../../sda
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1
symbolic link to ../../sda1
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part2
symbolic link to ../../sda2
/dev/disk/by-id/dm-name-rhel_rhel8-root
symbolic link to ../../dm-0
/dev/disk/by-id/dm-name-rhel_rhel8-swap
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUdivKOz5ofkgFhP0RMFsNyySVihqEI2cWWbR7MjXJoID6g
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUdivKOz5ofkgFhXqH2M45hD2H9nAf2qfWSrIRLhzfMyOKd
symbolic link to ../../dm-0
/dev/disk/by-id/lvm-pv-uuid-atlr2Y-vuMo-ueoH-CpMG-4JuH-AhEF-wu4QQm
symbolic link to ../../sda2
```

## 5.7. 修改持久性命名属性

您可以更改文件系统的 UUID 或 Label persistent naming 属性。

**注意**

更改 **udev** 属性发生在后台，可能需要很长时间。**udevadm settle** 命令会等待直到更改被完全注册，这样可确保您的下一个命令能够正确使用新属性。

在以下命令中：

- 将 *new-uuid* 替换为您要设置的 UUID；例如，**1cdfbc07-1c90-4984-b5ec-f61943f5ea50**。您可以使用 **uuidgen** 命令生成 UUID。
- 使用标签替换 *new-label*，如 **backup\_data**。

**先决条件**

- 如果您要修改 XFS 文件系统的属性，首先卸载它。

**流程**

- 要更改 XFS 文件系统的 UUID 或 Label 属性，请使用 **xfs\_admin** 工具：

```
# xfs_admin -U new-uuid -L new-label storage-device
# udevadm settle
```

- 要更改 **ext4**、**ext3** 或 **ext2** 文件系统的 UUID 或 Label 属性，请使用 **tune2fs** 工具：

■

```
# tune2fs -U new-uuid -L new-label storage-device  
# udevadm settle
```

- 要更改 swap 卷的 UUID 或 Label 属性，请使用 **swaptlabel** 工具：

```
# swaptlabel --uuid new-uuid --label new-label swap-device  
# udevadm settle
```



## 第 6 章 使用 NVDIMM 持久性内存存储

您可以在连接到您的系统的非线性内存模块 (NVDIMM) 设备上启用和管理各种存储类型。

有关在 NVDIMM 存储上安装 Red Hat Enterprise Linux 8，请参阅安装到 [NVDIMM 设备](#)。

### 6.1. NVDIMM 持久内存技术

非易失性双内存模块 (NVDIMM) 持久内存（也称为存储类内存或 **pmem**）是内存和存储的组合。

NVDIMM 将存储的持久性与低访问延迟和动态 RAM (DRAM) 的高带宽合并。以下是使用 NVDIMM 的其他优点：

- NVDIMM 存储是字节地址的，这意味着可以使用 CPU 负载和存储指令来访问。除了访问传统的基于块的存储所需的 `read()` 和 `write()` 系统调用外，NVDIMM 还支持直接加载和存储编程模型。
- NVDIMM 的性能特征与具有非常低访问延迟的 DRAM 类似，通常以十到百纳秒为单位。
- 当电源关闭时，保存在 NVDIMM 中的数据会被保留，类似于持久性内存。
- 通过直接访问 (DAX) 技术，无需通过系统页面缓存，即可直接到内存映射存储。这样便可为其他目的释放 DRAM。

在以下用例中 NVDIMM 很有用，比如：

#### 数据库

NVDIMM 减少的存储访问延迟可提高数据库性能。

#### 快速重启

快速重启也被称为热缓存效果。例如：文件服务器在启动后没有在内存中的文件内容。客户端连接和读取或者写入数据时，会在页面缓存中缓存这些数据。最后，缓存包括大多数热数据。重启后，该系统必须在传统存储上再次启动该进程。

使用 NVDIMM 时，如果应用程序被正确设计，应用程序可能会在重启后保留 warm 缓存。在本例中，不会涉及页面缓存：应用程序会直接在持久内存中缓存数据。

#### 快速写缓存

在数据被存入持久介质前，文件服务器通常不会确认客户端的写入请求。使用 NVDIMM 作为快速写入缓存可让文件服务器快速确认写入请求，因为会较低延迟。

### 6.2. NVDIMM 交集和地区

非易失性双内存模块 (NVDIMM) 设备支持分组到交集区域。

NVDIMM 设备可以像常规动态 RAM (DRAM) 一样分组为交集。interleave 集与跨多个 DIMM 的 RAID 0 级别（条带）配置类似。Interleave 集也称为区域（region）。

它有以下优点：

- 当 NVDIMM 设备被配置为 interleave 集时，会提高性能。
- 交集可以将多个较小的 NVDIMM 设备组合成一个更大的逻辑设备。

NVDIMM 交集是在系统 BIOS 或 UEFI 固件中配置的。Red Hat Enterprise Linux 为每个交集创建一个区域设备。

## 6.3. NVDIMM 命名空间

根据标签区域的大小，非易失性双内存模块 (NVDIMM) 区域可以划分为一个或多个命名空间。使用命名空间，您可以根据命名空间的访问模式访问设备，如 **sector**、**fsdax**、**devdax** 和 **raw**。更多信息，[NVDIMM 访问模式](#)。

有些 NVDIMM 设备不支持区域中的多个命名空间：

- 如果您的 NVDIMM 设备支持标签，您可以将区域重新划分到命名空间。
- 如果您的 NVDIMM 设备不支持标签，则区域只能包含单个命名空间。在这种情况下，Red Hat Enterprise Linux 会创建一个覆盖整个区域的默认命名空间。

## 6.4. NVDIMM 访问模式

您可以配置 Non-Volatile Dual In-line Memory Modules (NVDIMM) 命名空间，以使用以下模式之一：

### sector

将存储显示为一个快速块设备。这个模式可用于没有修改以使用 NVDIMM 存储的传统应用程序，或者用于使用完整 I/O 堆栈（包括设备映射器）的应用程序。

**sector** 设备可以与系统中的其它块设备相同。您可以在上面创建分区或文件系统，将其配置为软件 RAID 集的一部分，或者将其用作 **dm-cache** 的缓存设备。

此模式中的设备可作为 **/dev/pmemNs** 提供。创建命名空间后，查看列出的 **blockdev** 值。

### devdax 或设备直接访问 (DAX)

使用 **devdax** 时，NVDIMM 设备支持直接访问编程，如存储网络行业关联 (SNIA) 非易失性内存 (NVM) 编程模型规格中所述。在这个模式中，I/O 绕过内核的存储堆栈。因此无法使用设备映射器驱动程序。

设备 DAX 通过使用 DAX 字符设备节点提供对 NVDIMM 存储的原始访问。可以使用 CPU 缓存清除和隔离指令，使 **devdax** 设备中的数据可用。某些数据库和虚拟机虚拟机监控程序可能会受益于此模式。无法在 **devdax** 设备上创建文件系统。

这个模式中的设备可作为 **/dev/daxN.M** 提供。创建命名空间后，请参阅列出的 **chardev** 值。

### fsdax，或者文件系统直接访问 (DAX)

使用 **fsdax** 时，NVDIMM 设备支持直接访问编程，如存储网络行业关联 (SNIA) 非易失性内存 (NVM) 编程模型规格中所述。在这个模式中，I/O 会绕过内核的存储堆栈，因此无法使用很多设备映射器驱动程序。

您可以在文件系统 DAX 设备中创建文件系统。

此模式中的设备可作为 **/dev/pmemN** 提供。创建命名空间后，查看列出的 **blockdev** 值。



### 重要

文件系统 DAX 技术仅作为技术预览提供，不受红帽支持。

### raw

显示不支持 DAX 的内存磁盘。在这个模式中，命名空间有一些限制，不应使用。

此模式中的设备可作为 **/dev/pmemN** 提供。创建命名空间后，查看列出的 **blockdev** 值。

## 6.5. 安装 NDCTL

您可以安装 **ndctl** 工具来配置和监控 Non-Volatile Dual In-line Memory Modules (NVDIMM) 设备。

### 流程

- 安装 **ndctl** 工具：

```
# yum install ndctl
```

## 6.6. 在 NVDIMM 上创建扇区命名空间以充当块设备

您可以在扇区模式（也称为传统模式）中配置非线性内存模块 (NVDIMM) 设备，以支持传统的基于块的存储。

您可以：

- 将现有命名空间重新配置为扇区模式，或者
- 如果有可用空间，创建一个新的 sector 命名空间。

### 先决条件

- 一个 NVDIMM 设备被附加到您的系统。

### 6.6.1. 将现有的 NVDIMM 命名空间重新配置为扇区模式

您可以将非线性内存模块(NVDIMM)命名空间重新配置为扇区模式，将其用作快速块设备。



#### 警告

重新配置命名空间会删除之前在命名空间中存储的数据。

### 先决条件

- 已安装 **ndctl** 工具。如需更多信息，请参阅[安装 ndctl](#)。

### 流程

1. 查看现有命名空间：

```
# ndctl list --namespaces --idle
[
  {
    "dev": "namespace1.0",
    "mode": "raw",
    "size": 34359738368,
    "state": "disabled",
    "numa_node": 1
```

```

    },
    {
      "dev": "namespace0.0",
      "mode": "raw",
      "size": 34359738368,
      "state": "disabled",
      "numa_node": 0
    }
  ]

```

2. 将所选命名空间重新配置为扇区模式：

```
# ndctl create-namespace --force --reconfig=namespace-ID --mode=sector
```

#### 例 6.1. 在扇区模式中重新配置 namespace1.0

```

# ndctl create-namespace --force --reconfig=namespace1.0 --mode=sector
{
  "dev": "namespace1.0",
  "mode": "sector",
  "size": "755.26 GiB (810.95 GB)",
  "uuid": "2509949d-1dc4-4ee0-925a-4542b28aa616",
  "sector_size": 4096,
  "blockdev": "pmem1s"
}

```

重新配置的命名空间现在位于 **/dev** 目录下，存为 **/dev/pmem1s** 文件。

### 验证

- 验证系统中的现有命名空间是否已重新配置：

```

# ndctl list --namespace namespace1.0
[
  {
    "dev": "namespace1.0",
    "mode": "sector",
    "size": 810954706944,
    "uuid": "2509949d-1dc4-4ee0-925a-4542b28aa616",
    "sector_size": 4096,
    "blockdev": "pmem1s"
  }
]

```

### 其他资源

- 系统上的 **ndctl-create-namespace (1)** 手册页

## 6.6.2. 在扇区模式下创建新 NVDIMM 命名空间

您可以在扇区模式中创建一个非线性内存模块 (NVDIMM) 命名空间，以便在区域中存在可用空间时将其用作快速块设备。

## 先决条件

- 已安装 **ndctl** 工具。如需更多信息，请参阅[安装 ndctl](#)。
- NVDIMM 设备支持标签在区域中创建多个命名空间。您可以使用以下命令检查它：

```
# ndctl read-labels nmem0 >/dev/null
read 1 nmem
```

这表示它读取一个 NVDIMM 设备的标签。如果值为 **0**，这表示您的设备不支持标签。

## 流程

1. 列出系统上有可用空间的 **pmem** 区域。在以下示例中，在 *region1* 和 *region0* 区域中有空间：

```
# ndctl list --regions
[
  {
    "dev":"region1",
    "size":2156073582592,
    "align":16777216,
    "available_size":2117418876928,
    "max_available_extent":2117418876928,
    "type":"pmem",
    "iset_id":-9102197055295954944,
    "badblock_count":1,
    "persistence_domain":"memory_controller"
  },
  {
    "dev":"region0",
    "size":2156073582592,
    "align":16777216,
    "available_size":2143188680704,
    "max_available_extent":2143188680704,
    "type":"pmem",
    "iset_id":736272362787276936,
    "badblock_count":3,
    "persistence_domain":"memory_controller"
  }
]
```

2. 在任何可用区域上分配一个或多个命名空间：

```
# ndctl create-namespace --mode=sector --region=regionN --size=namespace-size
```

### 例 6.2. 在 region0 上创建 36-GiB 扇区命名空间

```
# ndctl create-namespace --mode=sector --region=region0 --size=36G
{
  "dev":"namespace0.1",
  "mode":"sector",
  "size":"35.96 GiB (38.62 GB)",
  "uuid":"ff5a0a16-3495-4ce8-b86b-f0e3bd9d1817",
  "sector_size":4096,
  "blockdev":"pmem0.1s"
```



新命名空间现在作为 **/dev/pmem0.1s** 提供。

## 验证

- 验证新命名空间是否在扇区模式中创建：

```
# ndctl list -RN -n namespace0.1
{
  "regions":[
    {
      "dev":"region0",
      "size":2156073582592,
      "align":16777216,
      "available_size":2104533975040,
      "max_available_extent":2104533975040,
      "type":"pmem",
      "iset_id":736272362787276936,
      "badblock_count":3,
      "persistence_domain":"memory_controller",
      "namespaces":[
        {
          "dev":"namespace0.1",
          "mode":"sector",
          "size":38615912448,
          "uuid":"ff5a0a16-3495-4ce8-b86b-f0e3bd9d1817",
          "sector_size":4096,
          "blockdev":"pmem0.1s"
        }
      ]
    }
  ]
}
```

## 其他资源

- 系统上的 **ndctl-create-namespace (1)** 手册页

## 6.7. 在 NVDIMM 上创建设备 DAX 命名空间

在设备 DAX 模式中配置附加到系统的 NVDIMM 设备，以支持具有直接访问功能的字符存储。

考虑以下选项：

- 将现有命名空间重新配置为设备 DAX 模式。
- 如果有可用空间，创建新设备 DAX 命名空间。

### 6.7.1. 设备直接访问模式中的 NVDIMM

设备直接访问（设备 DAX、**devdax**）提供了应用程序直接访问存储的方法，而无需参与文件系统。设备 DAX 的优点是它提供有保证的容错粒度，可以使用 **ndctl** 工具的 **--align** 选项来进行配置。

对于 Intel 64 和 AMD64 构架，支持以下故障颗粒度：

- 4 KiB
- 2 MiB
- 1 GiB

设备 DAX 节点只支持以下系统调用：

- `open()`
- `close()`
- `mmap()`

您可以使用 `ndctl list --human --capabilities` 命令查看 NVDIMM 设备支持的协调。例如，若要为 `region0` 设备查看它，请使用 `ndctl list --human --capabilities -r region0` 命令。



### 注意

不支持 `read()` 和 `write()` 系统调用，因为设备 DAX 用例与 SNIA Non-Volatile Memory Programming Model 关联。

## 6.7.2. 将现有的 NVDIMM 命名空间重新配置为设备 DAX 模式

您可以将现有的 Non-Volatile Dual In-line Memory Modules (NVDIMM) 命名空间重新配置为设备 DAX 模式。



### 警告

重新配置命名空间会删除之前在命名空间中存储的数据。

### 先决条件

- 已安装 `ndctl` 工具。如需更多信息，请参阅[安装 ndctl](#)。

### 流程

1. 列出系统中的所有命名空间：

```
# ndctl list --namespaces --idle

[
  {
    "dev": "namespace1.0",
    "mode": "raw",
    "size": 34359738368,
    "uuid": "ac951312-b312-4e76-9f15-6e00c8f2e6f4"
    "state": "disabled",
```

```

    "numa_node":1
  },
  {
    "dev":"namespace0.0",
    "mode":"raw",
    "size":38615912448,
    "uuid":"ff5a0a16-3495-4ce8-b86b-f0e3bd9d1817",
    "state":"disabled",
    "numa_node":0
  }
]

```

## 2. 重新配置任何命名空间：

```
# ndctl create-namespace --force --mode=devdax --reconfig=namespace-ID
```

### 例 6.3. 将命名空间配置为设备 DAX

以下命令为支持 DAX 的数据存储重新配置 **namespace0.1**。它与 2-MiB 故障粒度一致，以确保操作系统一次在 2-MiB 页面中故障：

```

# ndctl create-namespace --force --mode=devdax --align=2M --reconfig=namespace0.1
{
  "dev":"namespace0.1",
  "mode":"devdax",
  "map":"dev",
  "size":"35.44 GiB (38.05 GB)",
  "uuid":"426d6a52-df92-43d2-8cc7-046241d6d761",
  "daxregion":{
    "id":0,
    "size":"35.44 GiB (38.05 GB)",
    "align":2097152,
    "devices":[
      {
        "chardev":"dax0.1",
        "size":"35.44 GiB (38.05 GB)",
        "target_node":4,
        "mode":"devdax"
      }
    ]
  },
  "align":2097152
}

```

命名空间现在位于 **/dev/dax0.1** 路径。

## 验证

- 验证系统中的现有命名空间是否已重新配置：

```

# ndctl list --namespace namespace0.1
[
  {

```



```

    "dev": "namespace0.1",
    "mode": "devdax",
    "map": "dev",
    "size": 38048628736,
    "uuid": "426d6a52-df92-43d2-8cc7-046241d6d761",
    "chardev": "dax0.1",
    "align": 2097152
  }
]

```

## 其他资源

- 系统上的 **ndctl-create-namespace (1)** 手册页

### 6.7.3. 在设备 DAX 模式下创建新 NVDIMM 命名空间

如果区域中有可用空间，您可以在非线性内存模块(NVDIMM)设备上创建一个新设备 DAX 命名空间。

## 先决条件

- 已安装 **ndctl** 工具。如需更多信息，请参阅[安装 ndctl](#)。
- NVDIMM 设备支持标签在区域中创建多个命名空间。您可以使用以下命令检查它：

```

# ndctl read-labels nmem0 >/dev/null
read 1 nmem

```

这表示它读取一个 NVDIMM 设备的标签。如果值为 **0**，这表示您的设备不支持标签。

## 流程

1. 列出系统上有可用空间的 **pmem** 区域。在以下示例中，在 *region1* 和 *region0* 区域中有空间：

```

# ndctl list --regions
[
  {
    "dev": "region1",
    "size": 2156073582592,
    "align": 16777216,
    "available_size": 2117418876928,
    "max_available_extent": 2117418876928,
    "type": "pmem",
    "iset_id": -9102197055295954944,
    "badblock_count": 1,
    "persistence_domain": "memory_controller"
  },
  {
    "dev": "region0",
    "size": 2156073582592,
    "align": 16777216,
    "available_size": 2143188680704,
    "max_available_extent": 2143188680704,
    "type": "pmem",
    "iset_id": 736272362787276936,

```

```

    "badblock_count":3,
    "persistence_domain":"memory_controller"
  }
]

```

2. 在任何可用区域上分配一个或多个命名空间：

```
# ndctl create-namespace --mode=devdax --region=regionN --size=namespace-size
```

#### 例 6.4. 在区域上创建命名空间

以下命令在 region0 上创建 36-GiB 设备 DAX 命名空间。它与 2-MiB 故障粒度一致，以确保操作系统一次在 2-MiB 页面中故障：

```

# ndctl create-namespace --mode=devdax --region=region0 --align=2M --size=36G
{
  "dev":"namespace0.2",
  "mode":"devdax",
  "map":"dev",
  "size":"35.44 GiB (38.05 GB)",
  "uuid":"89d13f41-be6c-425b-9ec7-1e2a239b5303",
  "daxregion":{
    "id":0,
    "size":"35.44 GiB (38.05 GB)",
    "align":2097152,
    "devices":[
      {
        "chardev":"dax0.2",
        "size":"35.44 GiB (38.05 GB)",
        "target_node":4,
        "mode":"devdax"
      }
    ]
  },
  "align":2097152
}

```

命名空间现在作为 **/dev/dax0.2** 提供。

#### 验证

- 验证新命名空间是否在设备 DAX 模式中创建：

```

# ndctl list -RN -n namespace0.2
{
  "regions":[
    {
      "dev":"region0",
      "size":2156073582592,
      "align":16777216,
      "available_size":2065879269376,
      "max_available_extent":2065879269376,
      "type":"pmem",

```

```

    "iset_id":736272362787276936,
    "badblock_count":3,
    "persistence_domain":"memory_controller",
    "namespaces":[
      {
        "dev":"namespace0.2",
        "mode":"dax",
        "map":"dev",
        "size":38048628736,
        "uuid":"89d13f41-be6c-425b-9ec7-1e2a239b5303",
        "chardev":"dax0.2",
        "align":2097152
      }
    ]
  }
}

```

## 其他资源

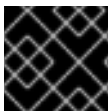
- 系统上的 **ndctl-create-namespace (1)** 手册页

## 6.8. 在 NVDIMM 上创建文件系统 DAX 命名空间

在文件系统 DAX 模式下配置附加到系统的 NVDIMM 设备，以支持具有直接访问功能的文件系统。

考虑以下选项：

- 将现有命名空间重新配置为文件系统 DAX 模式。
- 如果有可用空间，创建新文件系统 DAX 命名空间。



### 重要

文件系统 DAX 技术仅作为技术预览提供，不受红帽支持。

### 6.8.1. 文件系统直接访问模式中的 NVDIMM

当在文件系统直接访问（文件系统 DAX、**fsdax**）模式下配置 NVDIMM 设备时，您可以在上面创建一个文件系统。对此文件系统上的文件执行 **mmap()** 操作的任何应用程序都可以直接访问其存储。这启用了 NVDIMM 的直接访问编程模型。

以下新的 **-o dax** 选项现在可用，如果需要，可以通过文件属性控制直接访问行为：

#### **-o dax=inode**

当您不要在挂载文件系统时指定任何 **dax** 选项时，这个选项是默认选项。使用这个选项，您可以在文件上设置属性标志，以控制是否激活 **dax** 模式。如果需要，您可以在单个文件中设置此标志。

您还可以在一个目录上设置此标志，并且该目录中的任何文件将使用相同的标志创建。您可以使用 **xfs\_io -c 'chattr +x' directory-name** 命令设置此属性标志。

#### **-o dax=never**

使用此选项时，即使 **dax** 标记被设置为 **inode** 模式，也不会启用 **dax** 模式。这意味着，**per-inode dax** 属性标志会被忽略，使用这个标志设置的文件永远不会启用直接访问。

## -o dax=always

这个选项等同于旧的 **-o dax** 行为。使用这个选项，您可以激活文件系统中任何文件的直接访问模式，而不考虑 dax 属性标志。



### 警告

在以后的发行版本中，**-o dax** 可能不被支持，如果需要，您可以使用 **-o dax=always**。在这个模式中，每个文件都可能处于直接访问模式。

## 按页元数据分配

这个模式需要在系统 DRAM 或 NVDIMM 设备本身中分配每个页面元数据。每个 4 KiB 页的这个数据结构的开销是 64 字节：

- 在小设备中，开销非常小，足以满足 DRAM 的要求。例如，16-GiB 命名空间只需要 256 MiB 用于页面结构。因为 NVDIMM 设备通常比较小且昂贵，所以最好将页面跟踪数据结构保存在 DRAM 中。
- 在大小为 TB 级或更大的 NVDIMM 设备中，存储页面跟踪数据结构所需的内存量可能会超过系统中的 DRAM 数量。一个 TiB 的 NVDIMM 需要 16 GiB 用于页面结构。因此，在这种情况下首选将数据结构存储在 NVDIMM 本身中。

您可以在配置命名空间时使用 **--map** 选项配置每个页面元数据存储的位置：

- 要在系统 RAM 中分配，请使用 **--map=mem**。
- 要在 NVDIMM 上分配，请使用 **--map=dev**。

## 6.8.2. 将现有的 NVDIMM 命名空间重新配置为文件系统 DAX 模式

您可以将现有的非易失性双内存模块 (NVDIMM) 命名空间重新配置为文件系统 DAX 模式。



### 警告

重新配置命名空间会删除之前在命名空间中存储的数据。

## 先决条件

- 已安装 **ndctl** 工具。如需更多信息，请参阅[安装 ndctl](#)。

## 流程

1. 列出系统中的所有命名空间：

```
# ndctl list --namespaces --idle
```

```
[
  {
    "dev":"namespace1.0",
    "mode":"raw",
    "size":34359738368,
    "uuid":"ac951312-b312-4e76-9f15-6e00c8f2e6f4"
    "state":"disabled",
    "numa_node":1
  },
  {
    "dev":"namespace0.0",
    "mode":"raw",
    "size":38615912448,
    "uuid":"ff5a0a16-3495-4ce8-b86b-f0e3bd9d1817",
    "state":"disabled",
    "numa_node":0
  }
]
```

## 2. 重新配置任何命名空间：

```
# ndctl create-namespace --force --mode=fsdax --reconfig=namespace-ID
```

### 例 6.5. 将命名空间配置为文件系统 DAX

要将 **namespace0.0** 用于支持 DAX 的文件系统，请使用以下命令：

```
# ndctl create-namespace --force --mode=fsdax --reconfig=namespace0.0
{
  "dev":"namespace0.0",
  "mode":"fsdax",
  "map":"dev",
  "size":"11.81 GiB (12.68 GB)",
  "uuid":"f8153ee3-c52d-4c6e-bc1d-197f5be38483",
  "sector_size":512,
  "align":2097152,
  "blockdev":"pmem0"
}
```

命名空间现在位于 **/dev/pmem0** 路径中。

## 验证

- 验证系统中的现有命名空间是否已重新配置：

```
# ndctl list --namespace namespace0.0
[
  {
    "dev":"namespace0.0",
    "mode":"fsdax",
    "map":"dev",
    "size":12681478144,
    "uuid":"f8153ee3-c52d-4c6e-bc1d-197f5be38483",
```

```

    "sector_size":512,
    "align":2097152,
    "blockdev": "pmem0"
  }
]

```

## 其他资源

- 系统上的 **ndctl-create-namespace (1)** 手册页

### 6.8.3. 在文件系统 DAX 模式下创建新 NVDIMM 命名空间

如果区域中有可用空间，您可以在非线性内存模块(NVDIMM)设备上创建新文件系统 DAX 命名空间。

## 先决条件

- 已安装 **ndctl** 工具。如需更多信息，请参阅[安装 ndctl](#)。
- NVDIMM 设备支持标签在区域中创建多个命名空间。您可以使用以下命令检查它：

```

# ndctl read-labels nmem0 >/dev/null
read 1 nmem

```

这表示它读取一个 NVDIMM 设备的标签。如果值为 **0**，这表示您的设备不支持标签。

## 流程

1. 列出系统上有可用空间的 **pmem** 区域。在以下示例中，在 *region1* 和 *region0* 区域中有空间：

```

# ndctl list --regions
[
  {
    "dev":"region1",
    "size":2156073582592,
    "align":16777216,
    "available_size":2117418876928,
    "max_available_extent":2117418876928,
    "type":"pmem",
    "iset_id":-9102197055295954944,
    "badblock_count":1,
    "persistence_domain":"memory_controller"
  },
  {
    "dev":"region0",
    "size":2156073582592,
    "align":16777216,
    "available_size":2143188680704,
    "max_available_extent":2143188680704,
    "type":"pmem",
    "iset_id":736272362787276936,
    "badblock_count":3,
    "persistence_domain":"memory_controller"
  }
]

```

2. 在任何可用区域上分配一个或多个命名空间：

```
# ndctl create-namespace --mode=fsdax --region=regionN --size=namespace-size
```

### 例 6.6. 在区域上创建命名空间

以下命令在 *region0* 上创建 36-GiB 文件系统 DAX 命名空间：

```
# ndctl create-namespace --mode=fsdax --region=region0 --size=36G
{
  "dev": "namespace0.3",
  "mode": "fsdax",
  "map": "dev",
  "size": "35.44 GiB (38.05 GB)",
  "uuid": "99e77865-42eb-4b82-9db6-c6bc9b3959c2",
  "sector_size": 512,
  "align": 2097152,
  "blockdev": "pmem0.3"
}
```

命名空间现在作为 **/dev/pmem0.3** 提供。

### 验证

- 验证新命名空间是否在文件系统 DAX 模式中创建：

```
# ndctl list -RN -n namespace0.3
{
  "regions": [
    {
      "dev": "region0",
      "size": 2156073582592,
      "align": 16777216,
      "available_size": 2027224563712,
      "max_available_extent": 2027224563712,
      "type": "pmem",
      "iset_id": 736272362787276936,
      "badblock_count": 3,
      "persistence_domain": "memory_controller",
      "namespaces": [
        {
          "dev": "namespace0.3",
          "mode": "fsdax",
          "map": "dev",
          "size": 38048628736,
          "uuid": "99e77865-42eb-4b82-9db6-c6bc9b3959c2",
          "sector_size": 512,
          "align": 2097152,
          "blockdev": "pmem0.3"
        }
      ]
    }
  ]
}
```



## 其他资源

- 系统上的 **ndctl-create-namespace (1)** 手册页

### 6.8.4. 在文件系统 DAX 设备中创建文件系统

您可以在文件系统 DAX 设备中创建文件系统并挂载文件系统。创建文件系统后，应用程序可以使用持久内存并在 *mount-point* 目录中创建文件，打开文件，并使用 **mmap** 操作来映射文件以进行直接访问。

在 Red Hat Enterprise Linux 8 中，作为技术预览可在 NVDIMM 上创建 XFS 和 ext4 文件系统。

## 流程

1. 可选：在文件系统 DAX 设备中创建一个分区。如需更多信息，请参阅[使用 parted 创建分区](#)。



### 注意

当在 **fsdax** 设备中创建分区时，分区必须在页边界上保持一致。在 Intel 64 和 AMD64 构架中，启动和结束分区至少需要 4 KiB 校对。2 MiB 是首选的校对。

默认情况下，**parted** 工具在 1 MiB 边界上对齐分区。对于第一个分区，指定 2 MiB 作为分区的起点。如果分区的大小是 2 MiB 的倍数，则所有其他分区也都一致。

2. 在分区或者 NVDIMM 设备中创建 XFS 或者 ext4 文件系统：

```
# mkfs.xfs -d su=2m,sw=1 fsdax-partition-or-device
```



### 注意

现在，支持 dax 的文件和 reflinked 文件现在可以在文件系统中共存。但是，对于单个文件，dax 和 reflink 是互斥的。

对于 XFS，禁用共享的 copy-on-write 数据扩展，因为它们与 dax 挂载选项不兼容。另外，为了增加大型页面映射的可能性，请设置条带单元和条带宽度。

3. 挂载文件系统：

```
# mount f_sdax-partition-or-device mount-point_
```

不需要使用 dax 选项挂载文件系统来启用直接访问模式。当您在挂载时没有指定 dax 选项时，文件系统处于 **dax=inode** 模式。在激活直接访问模式前，在文件上设置 dax 选项。

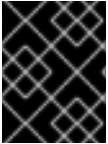
## 其他资源

- 您系统上的 **mkfs.xfs (8)** 手册页
- [文件系统直接访问模式中的 NVDIMM](#)

### 6.9. 使用 S.M.A.R.T 监控 NVDIMM 健康状况。

一些非线性内存模块 (NVDIMM) 设备支持自监控、分析和报告技术 (S.M.A.R.T.) 接口以检索健康信息。





## 重要

定期监控 NVDIMM 健康状况以防止数据丢失。如果 S.M.A.R.T. 报告 NVDIMM 设备健康状态的问题，请替换它，如[弃用并替换有问题的 NVDIMM 设备](#)中所述。

## 先决条件

- 可选：在一些系统中，上传 **acpi\_ipmi** 驱动程序以检索健康信息：

```
# modprobe acpi_ipmi
```

## 流程

- 访问健康信息：

```
# ndctl list --dimms --health
[
  {
    "dev": "nmem1",
    "id": "8089-a2-1834-00001f13",
    "handle": 17,
    "phys_id": 32,
    "security": "disabled",
    "health": {
      "health_state": "ok",
      "temperature_celsius": 36.0,
      "controller_temperature_celsius": 37.0,
      "spares_percentage": 100,
      "alarm_temperature": false,
      "alarm_controller_temperature": false,
      "alarm_spares": false,
      "alarm_enabled_media_temperature": true,
      "temperature_threshold": 82.0,
      "alarm_enabled_ctrl_temperature": true,
      "controller_temperature_threshold": 98.0,
      "alarm_enabled_spares": true,
      "spares_threshold": 50,
      "shutdown_state": "clean",
      "shutdown_count": 4
    }
  },
  [...]
]
```

## 其他资源

- 系统上的 **ndctl-list (1)** 手册页

## 6.10. 检测和替换断开问题的 NVDIMM 设备

如果您发现与系统日志中报告的 Non-Volatile Dual In-line Memory Modules (NVDIMM) 相关的错误消息，或者 S.M.A.R.T.，这可能意味着 NVDIMM 设备失败。在这种情况下，需要：

1. 检测哪个 NVDIMM 设备失败

2. 备份保存的数据
3. 物理替换该设备

## 流程

1. 检测有问题的设备：

```
# ndctl list --dimms --regions --health
{
  "dimms":[
    {
      "dev":"nmem1",
      "id":"8089-a2-1834-00001f13",
      "handle":17,
      "phys_id":32,
      "security":"disabled",
      "health":{"
        "health_state":"ok",
        "temperature_celsius":35.0,
        [...]
      }
    }
  ]
}
```

2. 查找有问题的 NVDIMM 的 **phys\_id** 属性：

```
# ndctl list --dimms --human
```

在上例中，您知道 **nmem0** 是有问题的 NVDIMM。因此，查找 **nmem0** 的 **phys\_id** 属性。

### 例 6.7. NVDIMM 的 **phys\_id** 属性

在以下示例中，**phys\_id** 是 **0x10**:

```
# ndctl list --dimms --human

[
  {
    "dev":"nmem1",
    "id":"XXXX-XX-XXXX-XXXXXXXXXX",
    "handle":"0x120",
    "phys_id":"0x1c"
  },
  {
    "dev":"nmem0",
    "id":"XXXX-XX-XXXX-XXXXXXXXXX",
    "handle":"0x20",
    "phys_id":"0x10",
    "flag_failed_flush":true,
    "flag_smart_event":true
  }
]
```

## 3. 查找有问题的 NVDIMM 的内存插槽：

```
# dmidecode
```

在输出中，找到 **Handle** 标识符与有问题的 NVDIMM 的 **phys\_id** 属性匹配的条目。**Locator** 字段列出了有问题的 NVDIMM 使用的内存插槽。

**例 6.8. NVDIMM 内存插槽列表**

在以下示例中，**nmem0** 设备与 **0x0010** 标识符匹配，并使用 **DIMM-XXX-YYYY** 内存插槽：

```
# dmidecode

...
Handle 0x0010, DMI type 17, 40 bytes
Memory Device
    Array Handle: 0x0004
    Error Information Handle: Not Provided
    Total Width: 72 bits
    Data Width: 64 bits
    Size: 125 GB
    Form Factor: DIMM
    Set: 1
    Locator: DIMM-XXX-YYYY
    Bank Locator: Bank0
    Type: Other
    Type Detail: Non-Volatile Registered (Buffered)
...

```

4. 备份 NVDIMM 命名空间中的所有数据。如果您在替换 NVDIMM 前没有备份数据，当您从系统中删除 NVDIMM 时数据将会丢失。

**警告**

在某些情况下，比如 NVDIMM 完全无法正常工作，备份可能会失败。

要防止这种情况，请使用 S.M.A.R.T.T 定期监控 NVDIMM 设备，如 [使用 S.M.A.R.T.T 监控 NVDIMM 健康状况](#) 中所述，并在它们中断前替换失败的 NVDIMM。

5. 列出 NVDIMM 上的命名空间：

```
# ndctl list --namespaces --dimm=DIMM-ID-number
```

**例 6.9. NVDIMM 命名空间列表**

在以下示例中，**nmem0** 设备包含 **namespace0.0** 和 **namespace0.2** 命名空间，您需要备份：

```
# ndctl list --namespaces --dimm=0

[
  {
    "dev":"namespace0.2",
    "mode":"sector",
    "size":67042312192,
    "uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "raw_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "sector_size":4096,
    "blockdev":"pmem0.2s",
    "numa_node":0
  },
  {
    "dev":"namespace0.0",
    "mode":"sector",
    "size":67042312192,
    "uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "raw_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "sector_size":4096,
    "blockdev":"pmem0s",
    "numa_node":0
  }
]
```

6. 以物理方式替换失效的 NVDIMM。

## 其他资源

- **ndctl-list (1)** 和 **dmidecode (8)** man page

## 第 7 章 丢弃未使用块

您可以在支持它们的块设备中执行或调度丢弃操作。块丢弃操作与挂载的文件系统不再使用的文件系统块进行通信。块丢弃操作允许 SSD 优化垃圾回收例程，它们可以通知精简置备存储重新调整未使用的物理块的用途。

### 要求

- 基本文件系统的块设备必须支持物理的丢弃（discard）操作。  
如果 `/sys/block/<device>/queue/discard_max_bytes` 文件中的值不为零，则支持物理丢弃操作。

### 7.1. 块丢弃操作的类型

您可以使用不同方法运行 discard 操作：

#### 批量丢弃

由用户明确触发，并丢弃所选文件系统中所有未使用的块。

#### 在线丢弃

在挂载时指定，并在无需用户干预的情况下实时触发。在线丢弃操作只丢弃从 **used** 转换到 **free** 状态的块。

#### 定期丢弃

是 **systemd** 服务定期运行的批处理操作。

XFS 和 ext4 文件系统支持所有类型。

#### 建议

红帽建议您使用批处理或周期性丢弃。

仅在以下情况下使用在线丢弃：

- 系统负载不允许使用批量丢弃，或者
- 为了保持性能，需要在线丢弃操作。

### 7.2. 执行批块丢弃

您可以执行批量块丢弃操作，以丢弃挂载的文件系统上未使用的块。

#### 先决条件

- 挂载文件系统。
- 文件系统底层的块设备支持物理忽略操作。

#### 流程

- 使用 **fstrim** 工具：
  - 要只在所选文件系统中执行丢弃，请使用：

```
# fstrim mount-point
```

- 要在所有挂载的文件系统中执行丢弃，请使用：

```
# fstrim --all
```

如果您在以下设备上执行 **fstrim** 命令：

- 不支持丢弃操作的设备，或者
- 由多个设备组成的逻辑设备（LVM 或者 MD），其中任意设备不支持丢弃操作：

下面的信息将显示：

```
# fstrim /mnt/non_discard
```

```
fstrim: /mnt/non_discard: the discard operation is not supported
```

#### 其他资源

- 您系统上的 **fstrim (8)** 手册页

## 7.3. 启用在线块丢弃

您可以执行在线块丢弃操作，以自动丢弃所有支持的文件系统上未使用的块。

#### 流程

- 在挂载时启用在线丢弃：
  - 手动挂载文件系统时，请添加 **-o discard** 挂载选项：

```
# mount -o discard device mount-point
```
  - 永久挂载文件系统时，请将 **discard** 选项添加到 **/etc/fstab** 文件中的挂载条目中。

#### 其他资源

- 在您的系统中 **mount (8)** 和 **fstab (5)** 手册页

## 7.4. 使用 STORAGE RHEL 系统角色启用在线块丢弃

您可以使用在线块丢弃选项挂载 XFS 文件系统，以自动丢弃未使用的块。

#### 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: Enable online block discard
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
```

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

- 验证在线块丢弃选项是否已启用：

```
# ansible managed-node-01.example.com -m command -a 'findmnt /mnt/data'
```

## 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

## 7.5. 启用定期块丢弃

您可以启用 **systemd** 计时器来定期丢弃所有支持的文件系统上未使用的块。

### 流程

- 启用并启动 **systemd** 计时器：

```
# systemctl enable --now fstrim.timer
Created symlink /etc/systemd/system/timers.target.wants/fstrim.timer →
/usr/lib/systemd/system/fstrim.timer.
```

## 验证

- 验证计时器的状态：

```
# systemctl status fstrim.timer
fstrim.timer - Discard unused blocks once a week
   Loaded: loaded (/usr/lib/systemd/system/fstrim.timer; enabled; vendor preset: disabled)
   Active: active (waiting) since Wed 2023-05-17 13:24:41 CEST; 3min 15s ago
     Trigger: Mon 2023-05-22 01:20:46 CEST; 4 days left
       Docs: man:fstrim
```

```
May 17 13:24:41 localhost.localdomain systemd[1]: Started Discard unused blocks once a week.
```



## 第 8 章 配置 iSCSI 目标

Red Hat Enterprise Linux 使用 **targetcli** shell 作为命令行界面执行以下操作：

- 添加、删除、查看和监控 iSCSI 存储间的连接来利用 iSCSI 硬件。
- 将由文件、卷、本地 SCSI 设备或者 RAM 磁盘支持的本地存储资源导出到远程系统。

**targetcli** 工具带有一个基于树形的布局，包括内置的 tab 完成、自动完成支持和在线文档。

### 8.1. 安装 TARGETCLI

安装 **targetcli** 工具来添加、监控和删除 iSCSI 存储间的连接。

#### 步骤

1. 安装 **targetcli** 工具：

```
# yum install targetcli
```

2. 启动目标服务：

```
# systemctl start target
```

3. 将目标配置为在引导时启动：

```
# systemctl enable target
```

4. 在防火墙中打开端口 **3260**，并重新载入防火墙配置：

```
# firewall-cmd --permanent --add-port=3260/tcp
Success
```

```
# firewall-cmd --reload
Success
```

#### 验证

- 查看 **targetcli** 布局：

```
# targetcli
/> ls
o- /.....[...]
  o- backstores.....[...]
    | o- block.....[Storage Objects: 0]
    | o- fileio.....[Storage Objects: 0]
    | o- pscsi.....[Storage Objects: 0]
    | o- ramdisk.....[Storage Objects: 0]
    o- iscsi.....[Targets: 0]
    o- loopback.....[Targets: 0]
```

## 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.2. 创建 iSCSI 目标

您可以创建一个 iSCSI 目标，使客户端的 iSCSI 启动器能够访问服务器上的存储设备。目标和发起方都有唯一的标识名称。

### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。

### 步骤

1. 导航到 iSCSI 目录。您还可以使用 **cd** 命令来导航到 iSCSI 目录。

```
/> iscsi/
```

2. 使用以下选项之一创建 iSCSI 对象：

- a. 使用默认目标名称创建 iSCSI 对象：

```
/iscsi> create

Created target
iqn.2003-01.org.linux-iscsi.hostname.x8664:sn.78b473f296ff
Created TPG1
```

- b. 使用特定名称创建 iSCSI 对象：

```
/iscsi> create iqn.2006-04.com.example:444

Created target iqn.2006-04.com.example:444
Created TPG1
Here iqn.2006-04.com.example:444 is target_iqn_name
```

将 *iqn.2006-04.com.example:444* 替换为特定目标名称。

3. 验证新创建的目标：

```
/iscsi> ls

o- iscsi.....[1 Target]
  o- iqn.2006-04.com.example:444.....[1 TPG]
    o- tpg1.....[enabled, auth]
      o- acls.....[0 ACL]
      o- luns.....[0 LUN]
      o- portals.....[0 Portal]
```

## 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.3. iSCSI BACKSTORE

iSCSI 后端存储支持在本地计算机上存储导出的 LUN 数据的不同方法。创建存储对象定义了后端存储使用的资源。

管理员可以选择 Linux-IO(LIO)支持的任何后端存储设备：

### fileio 后端存储

如果您将本地文件系统中的常规文件用作磁盘镜像，请创建一个 **fileio** 存储对象。有关创建 **fileio** 后端存储的信息，请参阅[创建 fileio 存储对象](#)。

### Block backstore

如果您使用任何本地块设备和逻辑设备，请创建一个 **块存储** 对象。有关创建块后备存储的信息，请参阅[创建块存储对象](#)。

### pscsi 后端存储

如果您的存储对象支持直接通过 SCSI 命令，请创建一个 **pscsi** 存储对象。有关创建 **pscsi** 后端存储的信息，请参阅[创建 pscsi 存储对象](#)。

### ramdisk 后端存储

如果要创建临时 RAM 支持的设备，请创建一个 **ramdisk** 存储对象。有关创建 **ramdisk** 后端存储，请参阅[创建内存复制 RAM 磁盘存储对象](#)。

### 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.4. 创建 FILEIO 存储对象

**fileio** 存储对象可以支持 **write\_back** 或 **write\_thru** 操作。**write\_back** 操作启用本地文件系统缓存。这提高了性能，但会增加数据丢失的风险。

建议使用 **write\_back=false** 禁用 **write\_back** 操作来使用 **write\_thru** 操作。

### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。

### 步骤

1. 从 **backstores/** 目录中进入 **fileio/**:

```
/ > backstores/fileio
```

2. 创建 **fileio** 存储对象：

```
/backstores/fileio> create file1 /tmp/disk1.img 200M write_back=false
Created fileio file1 with size 209715200
```

### 验证

- 验证创建的 **fileio** 存储对象：

```
/backstores/fileio> ls
```

## 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.5. 创建块存储对象

块驱动程序允许使用 **/sys/block/** 目录中出现的任何块设备来与 Linux-IO(LIO)一起使用。这包括物理设备，如 HDD、SSD、CD 和 DVD，以及逻辑设备，如软件或硬件 RAID 卷或 LVM 卷。

### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。

### 步骤

1. 从 **backstores/** 目录中进入 **block/**:

```
/> backstores/block/
```

2. 创建块后备存储：

```
/backstores/block> create name=block_backend dev=/dev/sdb  
  
Generating a wwn serial.  
Created block storage object block_backend using /dev/sdb.
```

### 验证

- 验证创建的块存储对象：

```
/backstores/block> ls
```

## 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.6. 创建 PSCSI 存储对象

您可以将配置为支持直接通过 SCSI 命令而无需 SCSI 模拟的后端存储，并且使用 **/proc/scsi/scsi** 中显示的底层 SCSI 设备（如 SAS 硬盘）。这个子系统支持 **SCSI-3 及更高系统**。



### 警告

**pscsi** 应该仅由高级用户使用。高级 SCSI 命令（如 Asymmetric Logical Unit Assignment (ALUAs) 或 Persistent Reservations（例如被 VMware ESX, 和 vSphere 使用））通常不会在设备固件中实施，并可能导致出现故障或崩溃。如果有疑问，请在生产环境中改为使用 **block** 后端存储。

### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。

### 步骤

1. 从 **backstores/** 目录中进入到 **pscsi/**：

```
/> backstores/pscsi/
```

2. 为物理 SCSI 设备创建 **pscsi** 后端存储，本示例中是使用 **/dev/sr0** 的 **TYPE\_ROM** 设备：

```
/backstores/pscsi> create name=pscsi_backend dev=/dev/sr0

Generating a wwn serial.
Created pscsi storage object pscsi_backend using /dev/sr0
```

### 验证

- 验证创建的 **pscsi** 存储对象：

```
/backstores/pscsi> ls
```

### 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.7. 创建内存副本 RAM 磁盘存储对象

**Memory Copy RAM 磁盘(ramdisk)**为 RAM 磁盘提供完整的 SCSI 模拟，并使用启动器的内存副本来分隔内存映射。这为多会话提供了功能，在用于生产目的的快速和可变量存储中特别有用。

#### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。

#### 步骤

1. 从 **backstores/** 目录进入到 **ramdisk/** ：

```
/> backstores/ramdisk/
```

2. 创建 **1GB RAM 磁盘**后备存储：

```
/backstores/ramdisk> create name=rd_backend size=1GB
```

```
Generating a wwn serial.
```

```
Created rd_mcp ramdisk rd_backend with size 1GB.
```

#### 验证

- 验证创建的 **ramdisk** 存储对象：

```
/backstores/ramdisk> ls
```

#### 其他资源

- 您系统上的 **targetcli (8)** 手册页

## 8.8. 创建 iSCSI 门户

您可以创建 **iSCSI** 门户。这会在目标中添加启用目标的 **IP 地址**和**端口**。

#### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。
- 与目标门户组(TPG)关联的 iSCSI 目标。如需更多信息，请参阅[创建 iSCSI 目标](#)。

## 步骤

1.

进入 **TPG** 目录：

```
/iscsi> iqn.2006-04.com.example:444/tpg1/
```

2.

使用以下选项之一创建 **iSCSI** 门户：

a.

创建默认门户使用默认 **iSCSI** 端口 **3260**，并允许目标侦听该端口上的所有 **IP** 地址：

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create
```

```
Using default IP port 3260
Binding to INADDR_Any (0.0.0.0)
Created network portal 0.0.0.0:3260
```

b.

使用特定 **IP** 地址创建门户：

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create 192.168.122.137
```

```
Using default IP port 3260
Created network portal 192.168.122.137:3260
```

## 验证

- 验证新创建的门户网站：

```
/iscsi/iqn.20...mple:444/tpg1> ls
```

```
o- tpg..... [enabled, auth]
  o- acls .....[0 ACL]
  o- luns .....[0 LUN]
  o- portals .....[1 Portal]
    o- 192.168.122.137:3260.....[OK]
```

## 其他资源

- [您系统上的 targetcli \(8\) 手册页](#)

## 8.9. 创建 iSCSI LUN

逻辑单元号(LUN)是一个由 iSCSI 后端存储支持的物理设备。每个 LUN 都有唯一的数字。

### 先决条件

- 已安装并运行 **targetcli**。如需更多信息，请参阅[安装 targetcli](#)。
- 与目标门户组(TPG)关联的 iSCSI 目标。如需更多信息，请参阅[创建 iSCSI 目标](#)。
- 已创建存储对象。如需更多信息，请参阅 [iSCSI Backstore](#)。

### 步骤

1. 创建已创建的存储对象的 LUN :

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/ramdisk/rd_backend
Created LUN 0.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/block/block_backend
Created LUN 1.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/fileio/file1
Created LUN 2.
```

2. 验证创建的 LUN :

```
/iscsi/iqn.20...mple:444/tpg1> ls

o- tpg..... [enabled, auth]
  o- acls .....[0 ACL]
  o- luns .....[3 LUNs]
    | o- lun0.....[ramdisk/ramdisk1]
    | o- lun1.....[block/block1 (/dev/vdb1)]
```



```
| o- lun2.....[fileio/file1 (/foo.img)]
o- portals .....[1 Portal]
o- 192.168.122.137:3260.....[OK]
```

默认 LUN 名称以 0 开头。



#### 重要

默认情况下，使用读写权限创建 LUN。如果在创建 ACL 后添加了新的 LUN，LUN 会自动映射到所有可用 ACL，并可能导致安全风险。要创建具有只读权限的 LUN，请参阅[创建只读 iSCSI LUN](#)。

3.

配置 ACL。如需更多信息，请参阅[创建 iSCSI ACL](#)。

#### 其他资源

- [您系统上的 targetcli \(8\) 手册页](#)

### 8.10. 创建只读 iSCSI LUN

默认情况下，使用读写权限创建 LUN。您可以创建一个只读 LUN。

#### 先决条件

- 已安装并运行 targetcli。如需更多信息，请参阅[安装 targetcli](#)。
- 与目标门户组(TPG)关联的 iSCSI 目标。如需更多信息，请参阅[创建 iSCSI 目标](#)。
- 已创建存储对象。如需更多信息，请参阅[iSCSI Backstore](#)。

#### 步骤

1. 设置只读权限：



```
/> set global auto_add_mapped_luns=false
```

Parameter `auto_add_mapped_luns` is now 'false'.

这样可防止 LUN 自动映射到现有 ACL，从而允许手动映射 LUN。

2.

进入 *initiator\_iqn\_name* 目录：

```
/> iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/
```

3.

创建 LUN：

```
/iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name> create  
mapped_lun=next_sequential_LUN_number tpg_lun_or_backstore=backstore  
write_protect=1
```

例如：

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example:888> create mapped_lun=1  
tpg_lun_or_backstore=/backstores/block/block2 write_protect=1
```

```
Created LUN 1.  
Created Mapped LUN 1.
```

4.

验证所创建的 LUN：

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example:888> ls  
o- 2006-04.com.example:888 .. [Mapped LUNs: 2]  
| o- mapped_lun0 ..... [lun0 block/disk1 (rw)]  
| o- mapped_lun1 ..... [lun1 block/disk2 (ro)]
```

`mapping_lun1` 行现在在结尾带有(ro)（不像 `mapping_lun0` 为 (rw)），这表示它是只读的。

5.

配置 ACL。如需更多信息，请参阅[创建 iSCSI ACL](#)。

其他资源

- 您系统上的 **targetcli (8)** 手册页

### 8.11. 创建 iSCSI ACL

**targetcli** 服务使用访问控制列表(ACL)来定义访问规则，并授予每个启动器访问逻辑单元号(LUN)的权限。

目标和发起方都有唯一的标识名称。您必须知道配置 ACL 的发起方的唯一名称。由 **iscsi-initiator-utils** 软件包提供的 `/etc/iscsi/initiatorname.iscsi` 文件包含 iSCSI 启动器名称。

#### 先决条件

- **targetcli** 服务 **已安装** 并在运行。
- 与目标门户组(TPG)关联的 **iSCSI 目标**。

#### 步骤

1. 可选：要禁用 LUN 到 ACL 的自动映射，请参阅 [创建只读 iSCSI LUN](#)。

2. 进到 **acls** 目录：

```
/> iscsi/target_iqn_name/tpg_name/acls/
```

3. 使用以下选项之一创建一个 ACL：

- 使用启动器上的 `/etc/iscsi/initiatorname.iscsi` 文件中的 *initiator\_iqn\_name*：

```
iscsi/target_iqn_name/tpg_name/acls> create initiator_iqn_name

Created Node ACL for initiator_iqn_name
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```

- 使用 ***custom\_name***，并更新启动器以匹配它：

```
iscsi/target_iqn_name/tpg_name/acls> create custom_name

Created Node ACL for custom_name
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```

有关更新启动器名称的详情，请参考 [创建一个 iSCSI 启动器](#)。

## 验证

- 验证创建的 ACL：

```
iscsi/target_iqn_name/tpg_name/acls> ls

o- acls .....[1 ACL]
o- target_iqn_name ....[3 Mapped LUNs, auth]
  o- mapped_lun0 .....[lun0 ramdisk/ramdisk1 (rw)]
  o- mapped_lun1 .....[lun1 block/block1 (rw)]
  o- mapped_lun2 .....[lun2 fileio/file1 (rw)]
```

## 其他资源

- [您系统上的 targetcli \(8\) 手册页](#)

## 8.12. 为目标设置 CHALLENGE-HANDSHAKE 验证协议

通过使用 **Challenge-Handshake Authentication Protocol(CHAP)**，用户可以使用密码保护目标。发起方必须了解这个密码才能连接到目标。

## 先决条件

- [创建 iSCSI ACL](#)。如需更多信息，请参阅[创建 iSCSI ACL](#)。

## 步骤

1.

设置属性身份验证：

```
/iscsi/iqn.20...mple:444/tpg1> set attribute authentication=1
```

```
Parameter authentication is now '1'.
```

2.

设置 **userid** 和密码：

```
/tpg1> set auth userid=redhat
```

```
Parameter userid is now 'redhat'.
```

```
/iscsi/iqn.20...689dcbb3/tpg1> set auth password=redhat_passwd
```

```
Parameter password is now 'redhat_passwd'.
```

3.

进入 **acls** 目录：

```
/> iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/
```

4.

设置属性身份验证：

```
/iscsi/iqn.20...:605fcc6a48be> set attribute authentication=1
```

```
Parameter authentication is now '1'.
```

5.

设置 **userid** 和密码：

```
/iscsi/iqn.20...:605fcc6a48be> set auth userid=redhat
```

```
Parameter userid is now 'redhat'.
```

```
/iscsi/iqn.20...:605fcc6a48be> set auth password=redhat_passwd
```

```
Parameter password is now 'redhat_passwd'.
```

## 其他资源



您系统上的 **targetcli (8)** 手册页

## 8.13. 使用 TARGETCLI 工具删除 iSCSI 对象

您可以使用 **targetcli** 工具删除 iSCSI 对象。

## 步骤

1.

从目标登出：

```
# iscsiadm -m node -T iqn.2006-04.com.example:444 -u
```

有关如何登录到目标的更多信息，请参阅创建 [iSCSI 启动器](#)。

2.

删除整个目标，包括所有 **ACL**、**LUN** 和门户：

```
/> iscsi/ delete iqn.2006-04.com.example:444
```

将 *iqn.2006-04.com.example:444* 替换为 **target\_iqn\_name**。

- 

删除 iSCSI 后端存储：

```
/> backstores/backstore-type/ delete block_backend
```

使用 **fileio**、**block**、**pscsi** 或 **ramdisk** 替换 *backstore-type*。

使用您要删除的 *backstore-name* 替换 *block\_backend*。

- 

要删除 iSCSI 目标的部分，如 **ACL**：

```
/> /iscsi/iqn-name/tpg/acls/ delete iqn.2006-04.com.example:444
```

## 验证

- 

查看更改：

```
/> iscsi/ ls
```

## 其他资源

- [您系统上的 targetcli \(8\) 手册页](#)

## 第 9 章 配置 iSCSI INITIATOR

iSCSI 启动程序形成连接 iSCSI 目标的会话。默认情况下，iSCSI 服务是“懒启动”，服务只在运行 `iscsiadm` 命令后才启动。如果 `root` 没有位于 iSCSI 设备，或者没有标记为 `node.startup = automatic` 的节点，则 iSCSI 服务将不会启动，直到执行 `iscsiadm` 命令后，需要 `iscsid` 或 `iscsi` 内核模块启动。

以 `root` 用户身份执行 `systemctl start iscsid` 命令，以强制 `iscsid` 服务运行和 iSCSI 内核模块加载。

### 9.1. 创建 iSCSI 启动程序

创建 iSCSI 启动器以连接到 iSCSI 目标，以访问服务器上的存储设备。

#### 先决条件

- 您有一个 iSCSI 目标的主机名和 IP 地址：
  - 如果您要连接到外部软件创建的存储目标，请从存储管理员查找目标主机名和 IP 地址。
  - 如果您要创建 iSCSI 目标，请参阅[创建 iSCSI 目标](#)。

#### 流程

1. 在客户端机器上安装 `iscsi-initiator-utils`:

```
# yum install iscsi-initiator-utils
```

2. 启动 `iscsid` 服务：

```
# systemctl start iscsid
```

3. 检查 `initiator` 名称：

```
# cat /etc/iscsi/initiatorname.iscsi
```



```
InitiatorName=iqn.2006-04.com.example:888
```

4.

如果在创建 **iSCSI ACL** 时 **ACL** 被指定了一个自定义名称，请更新启动器名称以匹配 **ACL**：

a.

打开 `/etc/iscsi/initiatorname.iscsi` 文件，并修改启动器名称：

```
# vi /etc/iscsi/initiatorname.iscsi

InitiatorName=custom-name
```

b.

重启 **iscsid** 服务：

```
# systemctl restart iscsid
```

5.

发现目标并使用显示的目标 **IQN** 登录到目标：

```
# iscsiadm -m discovery -t st -p 10.64.24.179
10.64.24.179:3260,1 iqn.2006-04.com.example:444

# iscsiadm -m node -T iqn.2006-04.com.example:444 -l
Logging in to [iface: default, target: iqn.2006-04.com.example:444, portal:
10.64.24.179,3260] (multiple)
Login to [iface: default, target: iqn.2006-04.com.example:444, portal: 10.64.24.179,3260]
successful.
```

将 **10.64.24.179** 替换为 **target-ip-address**。

如果将相应的启动器名称添加到 **ACL** 中，您可以将这个步骤用于连接到同一目标的任意启动器名称，如创建 **iSCSI ACL** 所述。

6.

找到 **iSCSI** 磁盘名称并在这个 **iSCSI** 磁盘创建文件系统：

```
# grep "Attached SCSI" /var/log/messages

# mkfs.ext4 /dev/disk_name
```

使用 `/var/log/messages` 文件中显示的 **iSCSI** 磁盘名称替换 **disk\_name**。

7.

挂载文件系统：

```
# mkdir /mount/point  
  
# mount /dev/disk_name /mount/point
```

使用分区的挂载点替换 */mount/point*。

8.

编辑 */etc/fstab* 文件，以便在系统引导时自动挂载文件系统：

```
# vi /etc/fstab  
  
/dev/disk_name /mount/point ext4 _netdev 0 0
```

使用 iSCSI 磁盘名称替换 *disk\_name*，使用分区的挂载点替换 */mount/point*。

#### 其他资源

- 您系统上的 [targetcli \(8\)](#) 和 [iscsiadm \(8\)](#) 手册页

## 9.2. 为发起方设置 CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL

通过使用 Challenge-Handshake Authentication Protocol(CHAP)，用户可以使用密码保护目标。发起方必须了解这个密码才能连接到目标。

#### 先决条件

- 创建 iSCSI initiator。如需更多信息，请参阅[创建 iSCSI 启动器](#)。
- 为目标设置 CHAP。如需更多信息，请参阅[为目标设置 Challenge-Handshake Authentication Protocol](#)。

#### 步骤

1.

在 **iscsid.conf** 文件中启用 **CHAP** 验证：

```
# vi /etc/iscsi/iscsid.conf

node.session.auth.authmethod = CHAP
```

默认情况下，**node.session.auth.authmethod** 设置为 **None**

2.

在 **iscsid.conf** 文件中添加目标用户名和密码：

```
node.session.auth.username = redhat
node.session.auth.password = redhat_passwd
```

3.

重启 **iscsid** 服务：

```
# systemctl restart iscsid
```

#### 其他资源

•

您系统上的 **iscsiadm** (8) 手册页

### 9.3. 使用 ISCSIADM 工具监控 ISCSI 会话

您可以使用 **iscsiadm** 工具监控 **iscsi** 会话。

默认情况下，iSCSI 服务是“懒启动”，服务只在运行 **iscsiadm** 命令后才启动。如果 **root** 没有位于 iSCSI 设备，或者没有标记为 **node.startup = automatic** 的节点，则 iSCSI 服务将不会启动，直到执行 **iscsiadm** 命令后，需要 **iscsid** 或 **iscsi** 内核模块启动。

以 **root** 用户身份使用 **systemctl start iscsid** 命令，以强制 **iscsid** 服务运行和 iSCSI 内核模块加载。

#### 步骤

1.

在客户端机器上安装 **iscsi-initiator-utils**：

```
# yum install iscsi-initiator-utils
```

2.

查找正在运行的会话的信息：

```
# iscsiadm -m session -P 3
```

这个命令显示会话或设备状态、会话 ID(sid)、一些协商的参数以及可通过会话访问的 SCSI 设备。

- 如果只需要简短的输出，例如：只显示 **sid-to-node** 映射，请运行：

```
# iscsiadm -m session -P 0
or
# iscsiadm -m session

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

这些命令以以下格式显示了正在运行的会话列表：**driver [sid]**  
**target\_ip:port,target\_portal\_group\_tag proper\_target\_name。**

#### 其他资源

- [/usr/share/doc/iscsi-initiator-utils-version/README 文件](#)
- [您系统上的 iscsiadm \(8\) 手册页](#)

#### 9.4. DM 多路径覆盖设备超时

**restore\_tmo sysfs** 选项控制一个特定 iSCSI 设备的超时时间。以下选项全局覆盖 **recovery\_tmo** 值：

- **replacement\_timeout** 配置选项会全局覆盖所有 iSCSI 设备的 **recovery\_tmo** 值。
- 对于由 DM 多路径管理的所有 iSCSI 设备，DM 多路径中的 **fast\_io\_fail\_tmo** 选项会全局覆盖 **recovery\_tmo** 值。

DM 多路径中的 `fast_io_fail_tmo` 选项会覆盖光纤通道设备的 `fast_io_fail_tmo` 选项。

DM 多路径 `fast_io_fail_tmo` 选项优先于 `replacement_timeout`。每次重新载入 `multipathd` 服务时，它会将 `recovery_tmo` 重置为 `fast_io_fail_tmo` 配置选项的值。使用 DM 多路径 `fast_io_fail_tmo` 配置选项覆盖由 DM 多路径管理的设备中的 `recovery_tmo`。

## 第 10 章 使用光纤通道设备

Red Hat Enterprise Linux 8 提供以下原生光纤频道驱动程序：

- **lpfc**
- **qla2xxx**
- **zfcp**

### 10.1. 在调整 LUN 的大小后重新扫描光纤通道逻辑单元

如果您更改了外部存储上的逻辑单元号(LUN)大小，请使用 **echo** 命令更新大小的内核视图。

#### 步骤

1. 确定哪些设备是 多路径 逻辑单元的路径：

```
# multipath -ll
```

2. 在使用多路径的系统中重新扫描光纤通道逻辑单元：

```
$ echo 1 > /sys/block/<device_ID>/device/rescan
```

将 **<device\_ID>** 替换为您的设备的 ID，如 **sda**。

#### 其他资源

- [系统上的 multipath \(8\) 手册页](#)

### 10.2. 使用光纤通道确定设备链路丢失行为

如果驱动程序实现了传输 `dev_loss_tmo` 回调，当检测到传输问题时，通过链接访问设备的尝试将被阻止。

## 步骤

- 确定远程端口的状态：

```
$ cat /sys/class/fc_remote_ports/rport-host:bus:remote-port/port_state
```

这个命令返回以下输出之一：

- 当远程端口以及通过它访问的设备被阻止时为 **Blocked**。
- 如果远程端口正常运行，则为 **Online**

如果在 `dev_loss_tmo` 秒内没有解决这个问题，则 `rport` 和设备将被取消阻塞。所有在该设备上运行的 I/O 以及发送到该设备的新 I/O 将失败。

当链路丢失超过 `dev_loss_tmo` 时，会删除 `scsi_device` 和 `sd_N` 设备。通常，光纤通道类不会更改设备，例如 `/dev/sda` 保留 `/dev/sda`。这是因为目标绑定由 Fibre Channel 驱动程序保存，当目标端口返回时，SCSI 地址会被重新创建。但是，这无法保证，只有在没有 LUN 的存储框中配置时，才会恢复该设备。

## 其他资源

- 您系统上的 `multipath.conf` (5) 手册页
- [在配置 Oracle RAC 集群时，建议在 scsi、multipath 和应用程序层进行调优](#)（红帽知识库）

## 10.3. FIBRE CHANNEL 配置文件

以下是 `/sys/class/` 目录中为 Fibre Channel 提供用户空间 API 的配置文件列表。

项目使用以下变量：

**H**  
主机号

**B**  
总线号

**T**  
目标

**L**  
逻辑单元(LUN)

**R**  
远程端口号



重要

如果您的系统正在使用多路径软件，在更改本节中描述的任何值前参考您的硬件供应商。

传输配置 `/sys/class/fc_transport/targetH:B:T/`

`port_id`  
24 位端口 ID/地址

`node_name`  
64 位节点名称

`port_name`  
64 位端口名称

远程端口配置 `/sys/class/fc_remote_ports/rport-H:B-R/`



- `port_id`
- `node_name`
- `port_name`
- `dev_loss_tmo`

控制 `scsi` 设备从系统中删除的时间。在 `dev_loss_tmo` 触发后，`scsi` 设备被删除。在 `multipath.conf` 文件中，您可以将 `dev_loss_tmo` 设置为 `infinity`。

在 Red Hat Enterprise Linux 8 中，如果您没有设置 `fast_io_fail_tmo` 选项，`dev_loss_tmo` 的上限将为 600 秒。默认情况下，如果 `multipathd` 服务正在运行，`fast_io_fail_tmo` 会在 Red Hat Enterprise Linux 8 中被设置为 5 秒；否则，它被设置为 `off`。

- `fast_io_fail_tmo`

指定在将链接标记为 "bad" 之前要等待的秒数。链接被标记为坏的后，现有正在运行的 I/O 或相应路径上的任何新 I/O 都将失败。

如果 I/O 处于阻塞队列中，则在 `dev_loss_tmo` 到期前和队列未阻塞前，它不会失败。

如果 `fast_io_fail_tmo` 被设为不是 `off` 的任何值时，则会取消对 `dev_loss_tmo` 的上限。如果 `fast_io_fail_tmo` 设为 `off`，则在设备从系统中删除之前不会出现 I/O 失败。如果 `fast_io_fail_tmo` 设置为一个数字，则在达到 `fast_io_fail_tmo` 设置的超时会立即触发 I/O 失败。

主机设置 `/sys/class/fc_host/hostH/`

- `port_id`

- **node\_name**
- **port\_name**
- **issue\_lip**

指示驱动重新发现远程端口。

## 第 11 章 使用以太网配置光纤

根据 IEEE T11 FC-BB-5 标准，使用以太网（FCoE）的光纤通道是通过以太网传输光纤通道帧的协议。通常数据中心有一个专用的 LAN 和 Storage Area Network（SAN），它和它们自己的配置是相互分开的。FCoE 将这些网络合并为一个整合的网络结构。例如 FCoE 的优点是降低硬件和能源成本。

### 11.1. 在 RHEL 中使用硬件 FCOE HBA

在 RHEL 中，您可以使用以太网的硬件光纤通道(FCoE)主机总线适配器(HBA)，这些驱动程序支持以下驱动程序：

- `qedf`
- `bnx2fc`
- `fnic`

如果您使用这样的 HBA，在 HBA 设置中配置 FCoE 设置。如需更多信息，请参阅适配器文档。

配置 HBA 后，从 Storage Area Network(SAN)中导出的逻辑单元号(LUN)将自动用于 RHEL 作为 `/dev/sd*` 设备。您可以使用类似本地存储设备的设备。

### 11.2. 设置 FCOE 设备

使用软件 FCoE 设备，使用部分支持 FCoE 卸载的以太网适配器通过 FCoE 访问逻辑单元号(LUN)。



#### 重要

RHEL 不支持需要 `fcoe.ko` 内核模块的软件 FCoE 设备。

完成此步骤后，RHEL 会自动从 Storage Area Network(SAN)导出的 LUN 作为 `/dev/sd*` 设备。您可以使用类似本地存储设备的方法使用这些设备。

## 先决条件

- 您已将网络交换机配置为支持 VLAN。
- SAN 使用 VLAN 来将存储流量与普通以太网流量分开。
- 您已在其 BIOS 中配置了服务器的 HBA。
- HBA 连接至网络，连接是在线的。如需更多信息，请参阅您的 HBA 文档。

## 步骤

1.

安装 **fcoe-utils** 软件包：

```
# yum install fcoe-utils
```

2.

将 **/etc/fcoe/cfg-ethx** 模板文件复制到 **/etc/fcoe/cfg-interface\_name**。例如，如果要将 **enp1s0** 接口配置为使用 FCoE，请输入以下命令：

```
# cp /etc/fcoe/cfg-ethx /etc/fcoe/cfg-enp1s0
```

3.

启用并启动 **fcoe** 服务：

```
# systemctl enable --now fcoe
```

4.

在接口 **enp1s0** 中发现 FCoE VLAN，为发现的 VLAN 创建网络设备，并启动启动器：

```
# fipvlan -s -c enp1s0
Created VLAN device enp1s0.200
Starting FCoE on interface enp1s0.200
Fibre Channel Forwarders Discovered
interface      | VLAN | FCF MAC
-----
enp1s0         | 200  | 00:53:00:a7:e7:1b
```

5.

可选：显示发现的目标、LUN 和与 LUN 关联的设备详情：

```
# fcoeadm -t
Interface:      enp1s0.200
Roles:         FCP Target
Node Name:      0x500a0980824acd15
Port Name:      0x500a0982824acd15
Target ID:      0
MaxFrameSize:   2048 bytes
OS Device Name: rport-11:0-1
FC-ID (Port ID): 0xba00a0
State:          Online

LUN ID Device Name Capacity Block Size Description
-----
0 sdb    28.38 GiB    512    NETAPP LUN (rev 820a)
...
```

本例显示 SAN 中的 LUN 0 已作为 /dev/sdb 设备附加到主机中。

验证

•

显示所有活跃 FCoE 接口的信息：

```
# fcoeadm -i
Description:     BCM57840 NetXtreme II 10 Gigabit Ethernet
Revision:        11
Manufacturer:    Broadcom Inc. and subsidiaries
Serial Number:   000AG703A9B7

Driver:          bnx2x Unknown
Number of Ports: 1

Symbolic Name:   bnx2fc (QLogic BCM57840) v2.12.13 over enp1s0.200
OS Device Name:  host11
Node Name:       0x2000000af70ae935
Port Name:       0x2001000af70ae935
Fabric Name:     0x20c8002a6aa7e701
Speed:           10 Gbit
Supported Speed: 1 Gbit, 10 Gbit
MaxFrameSize:    2048 bytes
FC-ID (Port ID): 0xba02c0
State:           Online
```

其他资源

- [系统中 fcoeadm \(8\) 手册页](#)
- [/usr/share/doc/fcoe-utils/README](#)
- [使用光纤通道设备](#)

## 第 12 章 使用 EH\_DEADLINE 配置存储错误恢复的最大时间

您可以配置最大允许的时间来恢复失败的 SCSI 设备。这个配置保证了 I/O 响应时间，即使存储硬件因为失败而变得无响应。

### 12.1. EH\_DEADLINE 参数

SCSI 错误处理(EH)机制尝试在失败的 SCSI 设备上执行错误恢复。SCSI 主机对象 `eh_deadline` 参数允许您配置恢复的最大时间。配置的时间过期后，SCSI EH 会停止并重置整个主机总线适配器(HBA)。

使用 `eh_deadline` 可以缩短时间：

- 关闭失败的路径，
- 切换路径，或者
- 禁用 RAID 分片。



#### 警告

当 `eh_deadline` 过期时，SCSI EH 会重置 HBA，这会影响那个 HBA 中的所有目标路径，而不仅仅是故障。如果由于其他原因无法使用冗余路径，则可能会出现 I/O 错误。如果在所有目标上都配置了多路径，请只启用 `eh_deadline`。另外，如果您的多路径设备没有完全冗余，您应该验证 `no_path_retry` 是否已设置为足够大，以允许路径恢复。

`eh_deadline` 参数的值以秒为单位指定。默认设置为 `off`，它会禁用时间限制并允许进行所有错误恢复。

`eh_deadline` 很有用的情况

在大多数情况下，您不需要启用 `eh_deadline`。在某些特定场景中，使用 `eh_deadline` 非常有用。例如，如果在光纤通道(FC)交换机和目标端口之间发生链接丢失，且 HBA 没有收到 **Registered State Change Notifications(RSCN)**。在这种情况下，I/O 请求和错误恢复命令会超时，而不是遇到错误。在这个环境中设置 `eh_deadline` 会针对恢复时间设置上限。这可让失败的 I/O 在由 DM 多路径的另一个可用路径中检索。

在以下条件下，`eh_deadline` 参数不提供额外的好处，因为 I/O 和错误恢复命令会立即失败，这会导致 DM 多路径重试：

- 如果启用了 RSCN
- 如果 HBA 没有注册链接不可用

## 12.2. 设置 EH\_DEADLINE 参数

您可以配置 `eh_deadline` 参数的值来限制最大 SCSI 恢复时间。

### 流程

- 您可以使用以下方法之一配置 `eh_deadline` ：
  - `multipath.conf` 文件的 `defaults` 部分

在 `multipath.conf` 文件的 `defaults` 部分，将 `eh_deadline` 参数设置为所需的秒数：

```
# eh_deadline 300
```



### 注意

在 RHEL 8.4 中，使用 `multipath.conf` 文件的 `defaults` 部分设置 `eh_deadline` 参数是首选的方法。

要使用此方法关闭 `eh_deadline` 参数，请将 `eh_deadline` 设置为 `off`。



- 

### sysfs

将秒数写入 `/sys/class/scsi_host/host<host-number>/eh_deadline` 文件中。例如，要在 SCSI 主机 6 上通过 `sysfs` 设置 `eh_deadline` 参数：

```
# echo 300 > /sys/class/scsi_host/host6/eh_deadline
```

要使用此方法关闭 `eh_deadline` 参数，请使用 `echo off`。

- 

### 内核参数

使用 `scsi_mod.eh_deadline` 内核参数为所有 SCSI HBA 设置默认值。

```
# echo 300 > /sys/module/scsi_mod/parameters/eh_deadline
```

要使用此方法关闭 `eh_deadline` 参数，请使用 `echo -1`。

## 其他资源

- 

[如何使用 udev 规则（红帽知识库）永久设置 eh\\_deadline 和 eh\\_timeout](#)

第 13 章 SWAP 入门

使用交换空间为不活跃的进程和数据提供临时存储，并防止物理内存满时内存不足。交换空间充当物理内存的扩展，并允许系统在物理内存耗尽时继续平稳运行。请注意，使用 `swap` 空间可能会降低系统性能，因此在依赖 `swap` 空间前，优化物理内存的使用可能会更有利。

13.1. SWAP 空间概述

当物理内存(RAM)已满时，将使用 Linux 中的 *交换空间*。如果系统需要更多的内存资源并且 RAM 已满，内存中的不活动页面将移到交换空间。虽然交换空间可以帮助具有少量 RAM 的计算机，但不应将其视为更多 RAM 的替代品。

交换空间位于硬盘驱动器上，其访问时间比物理内存要慢。交换空间可以是专用的交换分区（推荐）、交换文件，或者交换分区和交换文件的组合。

过去数年，推荐的 `swap` 空间会随系统中的 RAM 量增加而线性增大。然而，现代系统通常包含了成百 GB 内存。因此，推荐的交换空间被视为系统内存工作负载的功能，而不是系统内存的功能。

13.2. 推荐的系统 SWAP 空间

推荐的 `swap` 分区的大小取决于系统中的 RAM 量，以及是否需要足够的内存供系统休眠。推荐的 `swap` 分区大小在安装过程中自动设置。但是，为了允许休眠，您需要在自定义分区阶段编辑交换空间。

以下建议对于内存不足的系统（如 1 GB 或更小）特别重要。无法在这些系统上分配足够的 `swap` 空间可能会导致问题，如不稳定，甚至导致安装的系统无法引导。

表 13.1. 推荐的 swap 空间

系统中的 RAM 量	推荐的 swap 空间	如果允许休眠则推荐使用 swap 空间
≤ 2 GB	RAM 量的 2 倍	RAM 量的 3 倍
> 2 GB - 8 GB	与 RAM 量相等	RAM 量的 2 倍
> 8 GB - 64 GB	至少 4 GB	RAM 量的 1.5 倍
> 64 GB	至少 4 GB	不推荐休眠

对于边界值，如 2 GB、8 GB 或 64 GB 系统 RAM，请根据您的需要或偏好选择 swap 大小。如果您的系统资源允许此操作，增加 swap 空间可提高性能。

请注意，将交换空间分布到多个存储设备也可以提高交换空间的性能，特别是在具有快速驱动器、控制器和接口的系统上。

### 重要

在修改时，不应使用分配被为交换空间的文件系统和 LVM2 卷。如果系统进程或内核正在使用交换空间，则任何修改交换的尝试都会失败。使用 `free` 和 `cat /proc/swaps` 命令验证交换的使用量以及位置。

调整 swap 空间大小需要临时将其从系统中删除。如果运行的应用程序依赖于额外的 swap 空间，且可能会遇到低内存情况，这可能会出现问題。最好是，在救援模式中执行 swap 大小调整，请参阅 [Debug 引导选项](#)。当提示挂载文件系统时，请选择 **Skip**。

## 13.3. 为 SWAP 创建 LVM2 逻辑卷

您可以为 swap 创建一个 LVM2 逻辑卷。假设 `/dev/VolGroup00/LogVol02` 是您要添加的交换卷。

### 先决条件

- 您有足够的磁盘空间。

### 步骤

1. 创建大小为 2 GB 的 LVM2 逻辑卷：

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 格式化新 swap 空间：

```
# mkswap /dev/VolGroup00/LogVol02
```

3.

在 `/etc/fstab` 文件中添加以下条目：

```
/dev/VolGroup00/LogVol02 none swap defaults 0 0
```

4.

重新生成挂载单元以便您的系统注册新配置：

```
# systemctl daemon-reload
```

5.

在逻辑卷中激活 **swap**：

```
# swapon -v /dev/VolGroup00/LogVol02
```

## 验证

•

要测试是否成功创建并激活 **swap** 逻辑卷，请使用以下命令检查活跃 **swap** 空间：

```
# cat /proc/swaps
      total        used        free      shared buff/cache available
Mem:      30Gi      1.2Gi      28Gi         12Mi       994Mi       28Gi
Swap:      22Gi         0B       22Gi
```

```
# free -h
      total        used        free      shared buff/cache available
Mem:      30Gi      1.2Gi      28Gi         12Mi       995Mi       28Gi
Swap:      17Gi         0B       17Gi
```

## 13.4. 创建交换文件

当系统内存不足时，您可以创建一个 **swap** 文件来在固态驱动器或硬盘上创建一个临时存储空间。

### 先决条件

•

您有足够的磁盘空间。

### 步骤

1.

以 **MB** 为单位确定新交换文件的大小，再乘以 **1024** 来确定块的数量。例如：**64MB swap** 文件的块大小为 **65536**。

2.

创建一个空文件：

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

将 **65536** 替换为等于所需块大小的值。

3.

使用以下命令设定 **swap** 文件：

```
# mkswap /swapfile
```

4.

更改交换文件的安全性，使其不可读。

```
# chmod 0600 /swapfile
```

5.

使用以下条目编辑 **/etc/fstab** 文件，以在引导时启用交换文件：

```
/swapfile none swap defaults 0 0
```

下次系统引导时，它会激活新的 **swap** 文件。

6.

重新生成挂载单元，以便您的系统注册新的 **/etc/fstab** 配置：

```
# systemctl daemon-reload
```

7.

立即激活 **swap** 文件：

```
# swapon /swapfile
```

## 验证

- 

要测试新 **swap** 文件是否已成功创建并激活，请使用以下命令检查活跃 **swap** 空间：

■

```
$ cat /proc/swaps
```

```
$ free -h
```

### 13.5. 使用 STORAGE RHEL 系统角色创建交换卷

本节提供了一个 Ansible playbook 示例。此 playbook 应用 存储 角色来创建交换卷（如果不存在），或者使用默认参数修改块设备上的交换卷（如果已存在）。

#### 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: Create a disk device with swap
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: swap_fs
        type: disk
        disks:
          - /dev/sdb
        size: 15 GiB
        fs_type: swap
```

卷名称（示例中的 *swap\_fs*）目前是任意的。存储 角色根据 **disks:** 属性下列出的磁盘设备来识别卷。

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

## 13.6. 在 LVM2 逻辑卷中扩展 SWAP

您可以在现有 LVM2 逻辑卷上扩展 **swap** 空间。假设 */dev/VolGroup00/LogVol01* 是您要将其扩展为 **2 GB** 的卷。

### 先决条件

- 您有足够的磁盘空间。

### 步骤

1.

为关联的逻辑卷禁用交换：

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2.

将 LVM2 逻辑卷调整为 **2 GB**：

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3.

**格式化新 swap 空间：**

```
# mkswap /dev/VolGroup00/LogVol01
```

4.

**启用扩展的逻辑卷：**

```
# swapon -v /dev/VolGroup00/LogVol01
```

**验证**

•

**要测试是否成功扩展并激活 swap 逻辑卷，请检查活跃 swap 空间：**

```
# cat /proc/swaps
Filename      Type      Size      Used      Priority
/dev/dm-1     partition 16322556   0         -2
/dev/dm-4     partition 7340028    0         -3
```

```
# free -h
              total        used        free      shared  buff/cache   available
Mem:           30Gi        1.2Gi        28Gi        12Mi       994Mi        28Gi
Swap:          22Gi          0B         22Gi
```

**13.7. 在 LVM2 逻辑卷中减少 SWAP**

您可以减少 LVM2 逻辑卷上的 swap。假设 `/dev/VolGroup00/LogVol01` 是您要缩小的卷。

**步骤**

1.

**为关联的逻辑卷禁用交换：**

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2.

**清理 swap 签名：**

```
# wipefs -a /dev/VolGroup00/LogVol01
```



3.

将 LVM2 逻辑卷减少 512 MB :

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

4.

格式化新 swap 空间 :

```
# mkswap /dev/VolGroup00/LogVol01
```

5.

在逻辑卷中激活 swap :

```
# swapon -v /dev/VolGroup00/LogVol01
```

## 验证

•

要测试是否成功缩小 swap 逻辑卷, 请使用以下命令检查活跃 swap 空间 :

```
$ cat /proc/swaps
```

```
$ free -h
```

## 13.8. 为 SWAP 删除 LVM2 逻辑卷

您可以删除 swap 的一个 LVM2 逻辑卷。假设 `/dev/VolGroup00/LogVol02` 是您要删除的交换卷。

## 步骤

1.

为关联的逻辑卷禁用交换 :

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2.

删除 LVM2 逻辑卷 :

```
# lvremove /dev/VolGroup00/LogVol02
```

3.

从 **/etc/fstab** 文件中删除以下关联的条目：

```
/dev/VolGroup00/LogVol02 none swap defaults 0 0
```

4.

重新生成挂载单元以注册新配置：

```
# systemctl daemon-reload
```

#### 验证

- 

测试逻辑卷是否被成功删除，使用以下命令检查活跃的 **swap** 空间：

```
$ cat /proc/swaps
```

```
$ free -h
```

### 13.9. 删除交换文件

您可以删除一个 **swap** 文件。

#### 步骤

1.

禁用 **/swapfile swap** 文件：

```
# swapoff -v /swapfile
```

2.

相应地从 **/etc/fstab** 文件中删除其条目。

3.

重新生成挂载单元以便您的系统注册新配置：

```
# systemctl daemon-reload
```

4.

删除实际文件：

---

```
# rm /swapfile
```

## 第 14 章 NVME OVER FABRIC 设备概述

**Non-volatile Memory Express™(NVMe™)**是一个接口，它允许主机软件工具与固态硬盘进行通信。

使用以下类型的光纤传输来通过光纤设备配置 NVMe：

### **NVMe over Remote Direct Memory Access(NVMe/RDMA)**

有关如何配置 NVMe™/RDMA 的详情，请参考 [使用 NVMe/RDMA 配置光纤上的 NVMe](#)。

### **NVMe over Fibre Channel (NVMe/FC)**

有关如何配置 NVMe™/FC 的详情，请参考 [使用 NVMe/FC 配置光纤上的 NVMe](#)。

当使用 **NVMe over fabrics** 时，固态驱动器不必位于您的系统本地，它可以通过 **NVMe over fabrics** 设备远程配置。

## 第 15 章 使用 NVME/RDMA 配置 NVME OVER FABRICS

在通过 RDMA (NVMe™/RDMA)设置中的 Non-volatile Memory Express™(NVMe™)设置中，您要配置 NVMe 控制器和 NVMe initiator。

### 15.1. 使用 CONFIGFS 设置 NVME/RDMA 控制器

您可以使用 `configfs` 通过 RDMA (NVMe™/RDMA)控制器配置 Non-volatile Memory Express™(NVMe™) 控制器。

#### 先决条件

- 验证您有一个要分配给 `nvmet` 子系统的块设备。

#### 步骤

1.

创建 `nvmet-rdma` 子系统：

```
# modprobe nvmet-rdma
# mkdir /sys/kernel/config/nvmet/subsystems/testnqn
# cd /sys/kernel/config/nvmet/subsystems/testnqn
```

使用子系统名称替换 `testnqn`。

2.

允许任何主机连接到此控制器：

```
# echo 1 > attr_allow_any_host
```

3.

配置命名空间：

```
# mkdir namespaces/10
# cd namespaces/10
```

**使用命名空间号替换 10**

4.

**设置 NVMe 设备的路径：**

```
# echo -n /dev/nvme0n1 > device_path
```

5.

**启用命名空间：**

```
# echo 1 > enable
```

6.

**创建带有 NVMe 端口的目录：**

```
# mkdir /sys/kernel/config/nvmet/ports/1
# cd /sys/kernel/config/nvmet/ports/1
```

7.

**显示 *mlx5\_ib0* 的 IP 地址：**

```
# ip addr show mlx5_ib0

8: mlx5_ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 4092 qdisc mq state UP
group default qlen 256
    link/infiniband 00:00:06:2f:fe:80:00:00:00:00:00:00:e4:1d:2d:03:00:e7:0f:f6 brd
    00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 172.31.0.202/24 brd 172.31.0.255 scope global noprefixroute mlx5_ib0
        valid_lft forever preferred_lft forever
    inet6 fe80::e61d:2d03:e7:ff6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

8.

**为控制器设置传输地址：**

```
# echo -n 172.31.0.202 > addr_traddr
```

9.

**将 RDMA 设置为传输类型：**

```
# echo rdma > addr_trtype
# echo 4420 > addr_trsvcid
```

10.

为端口设置地址系列：

```
# echo ipv4 > addr_adrfam
```

11.

创建软链接：

```
# ln -s /sys/kernel/config/nvmet/subsystems/testnqn  
/sys/kernel/config/nvmet/ports/1/subsystems/testnqn
```

## 验证

•

验证 NVMe 控制器是否在指定端口上侦听并准备好连接请求：

```
# dmesg | grep "enabling port"  
[ 1091.413648] nvmet_rdma: enabling port 1 (172.31.0.202:4420)
```

## 其他资源

•

您系统上的 `nvme (1)` 手册页

## 15.2. 使用 NVMETCLI 设置 NVME/RDMA 控制器

您可以使用 `nvmetcli` 程序通过 RDMA (NVMe™/RDMA) 控制器配置 Non-volatile Memory Express™ (NVMe™) 控制器。 `nvmetcli` 实用程序提供命令行和交互式 `shell` 选项。

## 先决条件

•

验证您有一个要分配给 `nvmet` 子系统的块设备。

•

以 `root` 用户身份执行以下 `nvmetcli` 操作。

## 步骤

1.

安装 `nvmetcli` 软件包：

```
# yum install nvmetcli
```

2.

下载 **rdma.json** 文件：

```
# wget
http://git.infradead.org/users/hch/nvmetcli.git/blob_plain/0a6b088db2dc2e5de11e6f23f
1e890e4b54fee64:/rdma.json
```

3.

编辑 **rdma.json** 文件，并将 **traddr** 值更改为 **172.31.0.202**。

4.

通过载入 **NVMe** 控制器配置文件来设置控制器：

```
# nvmetcli restore rdma.json
```



### 注意

如果没有指定 **NVMe** 控制器配置文件名称，则 **nvmetcli** 使用 **/etc/nvmet/config.json** 文件。

### 验证

•

验证 **NVMe** 控制器是否在指定端口上侦听并准备好连接请求：

```
# dmesg | tail -1
[ 4797.132647] nvmet_rdma: enabling port 2 (172.31.0.202:4420)
```

•

可选：清除当前 **NVMe** 控制器：

```
# nvmetcli clear
```

### 其他资源

•

您系统上的 **nvmetcli** 和 **nvme** (1) man page

## 15.3. 配置 NVMe/RDMA 主机



您可以使用 NVMe 管理命令行界面(nvme-cli)工具通过 RDMA (NVMe™/RDMA)主机配置 Non-volatile Memory Express™(NVMe™)主机。

## 流程

1.

安装 **nvme-cli** 工具：

```
# yum install nvme-cli
```

2.

如果没有加载，则加载 **nvme-rdma** 模块：

```
# modprobe nvme-rdma
```

3.

在 NVMe 控制器中发现可用的子系统：

```
# nvme discover -t rdma -a 172.31.0.202 -s 4420
```

```
Discovery Log Number of Records 1, Generation counter 2
```

```
=====Discovery Log Entry 0=====
```

```
trtype: rdma
```

```
adrfam: ipv4
```

```
subtype: nvme subsystem
```

```
treq: not specified, sq flow control disable supported
```

```
portid: 1
```

```
trsvcid: 4420
```

```
subnqn: testnqn
```

```
traddr: 172.31.0.202
```

```
rdma_prtype: not specified
```

```
rdma_qptype: connected
```

```
rdma_cms: rdma-cm
```

```
rdma_pkey: 0x0000
```

4.

连接到发现的子系统：

```
# nvme connect -t rdma -n testnqn -a 172.31.0.202 -s 4420
```

```
# lsblk
```

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
```

```
sda 8:0 0 465.8G 0 disk
```

```
├─sda1 8:1 0 1G 0 part /boot
```

```
└─sda2 8:2 0 464.8G 0 part
```

```
├─rhel_rdma--virt--03-root 253:0 0 50G 0 lvm /
```

```
└─rhel_rdma--virt--03-swap 253:1 0 4G 0 lvm [SWAP]
```

```
└─rhel_rdma--virt--03-home 253:2 0 410.8G 0 lvm /home
nvme0n1

# cat /sys/class/nvme/nvme0/transport
rdma
```

使用 NVMe 子系统名称替换 *testnqn*。

将 *172.31.0.202* 替换为控制器 IP 地址。

使用端口号替换 *4420*。

## 验证

- 列出当前连接的 NVMe 设备：

```
# nvme list
```

- 可选：从控制器断开连接：

```
# nvme disconnect -n testnqn
NQN:testnqn disconnected 1 controller(s)

# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                8:0    0 465.8G  0 disk
├─sda1                            8:1    0    1G  0 part /boot
└─sda2                            8:2    0 464.8G  0 part
   ├─rhel_rdma--virt--03-root 253:0    0   50G  0 lvm /
   ├─rhel_rdma--virt--03-swap 253:1    0    4G  0 lvm [SWAP]
   └─rhel_rdma--virt--03-home 253:2    0 410.8G  0 lvm /home
```

## 其他资源

- [您系统上的 nvme \(1\) 手册页](#)

## 15.4. 后续步骤

- [在 NVMe 设备中启用多路径](#)

## 第 16 章 使用 NVME/FC 配置 NVME OVER FABRICS

当与某些 Broadcom Emulex 和 Marvell Qlogic Fibre Channel 适配器一起使用时，在主机模式下完全支持光纤通道 (NVMe™/FC) 传输上的 Non-volatile Memory Express™(NVMe™)。

## 16.1. 为广播适配器配置 NVME 主机

您可以使用 NVMe 管理命令行界面(nvme-cli)实用程序使用 Broadcom 适配器配置 Non-volatile Memory Express™(NVMe™)主机。

## 流程

1.

安装 nvme-cli 工具：

```
# yum install nvme-cli
```

这会在 /etc/nvme/ 目录中创建 hostnqn 文件。hostn 文件标识 NVMe 主机。

2.

查找本地和远程端口的全球节点名称(WWNN)和全球端口名称(WWPN)标识符：

```
# cat /sys/class/scsi_host/host*/nvme_info
```

NVME Host Enabled

XRI Dist lpfc0 Total 6144 IO 5894 ELS 250

NVME LPORT lpfc0 WWPN x10000090fae0b5f5 WWNN x20000090fae0b5f5 DID x010f00

ONLINE

NVME RPORT WWPN x204700a098cbcac6 WWNN x204600a098cbcac6 DID

x01050e TARGET DISCSRV ONLINE

NVME Statistics

LS: Xmt 000000000e Cmpl 000000000e Abort 00000000

LS XMIT: Err 00000000 CMPL: xb 00000000 Err 00000000

Total FCP Cmpl 00000000000008ea Issue 00000000000008ec OutIO 0000000000000002

abort 00000000 noxri 00000000 nondlp 00000000 qdepth 00000000 wqerr 00000000  
err 00000000

FCP CMPL: xb 00000000 Err 00000000

使用这些 host-traddr 和 traddr 值，查找子系统 NVMe 限定名称(NQN)：

```
# nvme discover --transport fc \
```

```
--traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6\  
--host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5
```

Discovery Log Number of Records 2, Generation counter 49530

=====Discovery Log Entry 0=====

trtype: fc

adrfam: fibre-channel

subtype: nvme subsystem

treql: not specified

portid: 0

trsvcid: none

subnqn: nqn.1992-

08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14\_nvme\_ss\_1\_1

traddr: nn-0x204600a098cbcac6:pn-0x204700a098cbcac6

将 `nn-0x204600a098cbcac6:pn-0x204700a098cbcac6` 替换为 `traddr`。

将 `nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5` 替换为 `host-traddr`。

3.

使用 `nvme-cli` 连接到 NVMe 控制器：

```
# nvme connect --transport fc \  
--traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6\  
--host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5\  
-n nqn.1992-  
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_  
1\  
-k 5
```



注意

如果在连接时间超过默认的 keep-alive 超时值时看到 `keep-alive timer (5 seconds) expired!` 错误，请使用 `-k` 选项来增加它。例如，您可以使用 `-k 7`。

在这里，

将 `nn-0x204600a098cbcac6:pn-0x204700a098cbcac6` 替换为 `traddr`。

将 `nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5` 替换为 `host-traddr`。

将 `nqn.1992-08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1` 替换为 `subnqn`。

将 `5` 替换为 `keep-alive` 超时值（以秒为单位）。

验证

- 列出当前连接的 NVMe 设备：

```
# nvme list
Node          SN          Model          Namespace Usage
Format        FW Rev
-----
- -----
/dev/nvme0n1  80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller      1
107.37 GB / 107.37 GB    4 KiB + 0 B  FFFFFFFF

# lsblk |grep nvme
nvme0n1          259:0    0 100G 0 disk
```

其他资源

- 您系统上的 `nvme` (1) 手册页

16.2. 为 QLOGIC 适配器配置 NVME 主机

您可以使用 NVMe 管理命令行界面(`nvme-cli`)工具配置带有 Qlogic 适配器的 Non-volatile Memory Express™(NVMe™)主机。

流程

- 安装 `nvme-cli` 工具：

```
# yum install nvme-cli
```

这会在 `/etc/nvme/` 目录中创建 `hostnqn` 文件。`hostn` 文件标识 NVMe 主机。

2.

重新载入 `qla2xxx` 模块：

```
# modprobe -r qla2xxx
# modprobe qla2xxx
```

3.

查找本地和远程端口的全球节点名称(WWNN)和全球端口名称(WWPN)标识符：

```
# dmesg |grep traddr

[ 6.139862] qla2xxx [0000:04:00.0]-ffff:0: register_localport: host-traddr=nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 on portID:10700
[ 6.241762] qla2xxx [0000:04:00.0]-2102:0: qla_nvme_register_remote: traddr=nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 PortID:01050d
```

使用这些 `host-traddr` 和 `traddr` 值，查找子系统 NVMe 限定名称(NQN)：

```
# nvme discover --transport fc \
    --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
    --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62

Discovery Log Number of Records 2, Generation counter 49530
=====Discovery Log Entry 0=====
trtype: fc
adrfam: fibre-channel
subtype: nvme subsystem
treq: not specified
portid: 0
trsvcid: none
subnqn: nqn.1992-08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468
traddr: nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6
```

将 `nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6` 替换为 `traddr`。

将 `nn-0x20000024ff19bb62:pn-0x21000024ff19bb62` 替换为 `host-traddr`。

4.

使用 `nvme-cli` 工具连接到 NVMe 控制器：

```
# nvme connect --transport fc \
    --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
    --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 \
    -n nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipat
h_1_subsystem_468\
    -k 5
```



### 注意

如果在连接时间超过默认的 keep-alive 超时值时看到 keep-alive timer (5 seconds) expired! 错误, 请使用 -k 选项来增加它。例如, 您可以使用 -k 7。

在这里,

将 `nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6` 替换为 `traddr`。

将 `nn-0x20000024ff19bb62:pn-0x21000024ff19bb62` 替换为 `host-traddr`。

将 `nqn.1992-08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468` 替换为 `subnqn`。

将 5 替换为 keep-live 超时值 (以秒为单位)。

### 验证



列出当前连接的 NVMe 设备：

```
# nvme list
Node          SN          Model          Namespace Usage
Format        FW Rev
-----
/dev/nvme0n1  80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller 1
107.37 GB / 107.37 GB  4 KiB + 0 B  FFFFFFFF

# lsblk |grep nvme
nvme0n1          259:0  0 100G 0 disk
```

■

## 其他资源

- [您系统上的 nvme \(1\) 手册页](#)

## 16.3. 后续步骤

- [在 NVMe 设备中启用多路径](#)



## 第 17 章 在 NVME 设备中启用多路径

您可以对通过光纤传输连接到您系统的 **Non-volatile Memory Express™(NVMe™)** 设备（如光纤通道 (FC)）进行多路径。您可以在多个多路径解决方案之间进行选择。

### 17.1. 本地 NVME 多路径和 DM 多路径

**Non-volatile Memory Express™ (NVMe™)** 设备支持原生多路径功能。当在 NVMe 中配置多路径时，您可以在标准 DM 多路径和原生 NVMe 多路径之间进行选择。

DM 多路径和原生 NVMe 多路径都支持 NVMe 设备的 **Asymmetric Namespace Access(ANA)**多路径方案。ANA 识别控制器和主机之间的优化路径，并提高性能。

当启用原生 NVMe 多路径时，它会全局地应用于所有 NVMe 设备。它可以提供更高的性能，但不包含 DM 多路径提供的所有功能。例如，原生 NVMe 多路径只支持 **numa** 和 **round-robin** 路径选择方法。

红帽建议您在 Red Hat Enterprise Linux 8 中使用 DM 多路径作为默认多路径解决方案。

### 17.2. 启用原生 NVME 多路径

**nvme\_core.multipath** 选项的默认内核设置被设置为 **N**，这意味着原生 **Non-volatile Memory Express™(NVMe™)**多路径被禁用。您可以使用原生 NVMe 多路径解决方案启用原生 NVMe 多路径。

#### 先决条件

- NVMe 设备连接到您的系统。如需更多信息，请参阅 [通过光纤设备概述 NVMe](#)。

#### 步骤

1. 检查内核中是否启用了原生 NVMe 多路径：

```
# cat /sys/module/nvme_core/parameters/multipath
```

这个命令显示以下之一：

**N**

禁用原生 NVMe 多路径。

**Y**

启用原生 NVMe 多路径。

2.

如果原生 NVMe 多路径被禁用，使用以下方法之一启用它：

- 

使用内核选项：

a.

在命令行中添加 `nvme_core.multipath=Y` 选项：

```
# grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
```

b.

在 64 位 IBM Z 构架中更新引导菜单：

```
# zipl
```

c.

重启系统：

- 

使用内核模块配置文件：

a.

使用以下内容创建 `/etc/modprobe.d/nvme_core.conf` 配置文件：

```
options nvme_core multipath=Y
```

b.

备份 `initramfs` 文件：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date
+%m-%d-%H%M%S).img
```

c.

重建 initramfs :

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

d.

重启系统 :

3.

可选 : 在运行的系统中, 更改 NVMe 设备中的 I/O 策略, 以便在所有可用路径中分发 I/O :

```
# echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

4.

可选 : 使用 udev 规则永久设置 I/O 策略。使用以下内容创建 `/etc/udev/rules.d/71-nvme-iopolicy.rules` 文件 :

```
ACTION=="add|change", SUBSYSTEM=="nvme-subsystem", ATTR{iopolicy}="round-
robin"
```

## 验证

1.

验证您的系统是否识别 NVMe 设备。以下示例假设您有一个通过光纤连接的 NVMe 存储子系统, 它有两个 NVMe 命名空间 :

```
# nvme list
```

Node Format	SN FW Rev	Model	Namespace Usage
/dev/nvme0n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme0n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	

2.

列出所有连接的 NVMe 子系统 :

```
# nvme list-subsys
```

```
nvme-subsys0 - NQN=testnqn
```

```
\
+- nvme0 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme1 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1de live
+- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

检查活动传输类型。例如，`nvme0 fc` 表示设备通过光纤通道传输连接，`nvme tcp` 则表示设备通过 TCP 连接。

3.

如果您编辑了内核选项，请验证内核命令行上是否启用了原生 NVMe 多路径：

```
# cat /proc/cmdline

BOOT_IMAGE=[...] nvme_core.multipath=Y
```

4.

如果您更改了 I/O 策略，请验证 `round-robin` 是否在 NVMe 设备上活跃的 I/O 策略：

```
# cat /sys/class/nvme-subsystem/nvme-subsys0/iopolicy

round-robin
```

## 其他资源



[配置内核命令行参数](#)

## 17.3. 在 NVME 设备中启用 DM 多路径

您可以通过禁用原生 NVMe 多路径，在连接的 NVMe 设备上启用 DM 多路径。

## 先决条件



NVMe 设备连接到您的系统。如需更多信息，请参阅 [通过光纤设备概述 NVMe](#)。

## 步骤

1.

检查是否原生 NVMe 多路径被禁用：

```
# cat /sys/module/nvme_core/parameters/multipath
```

这个命令显示以下之一：

**N**

禁用原生 NVMe 多路径。

**Y**

启用原生 NVMe 多路径。

2.

如果启用了原生 NVMe 多路径，请使用以下方法之一禁用它：

- 

使用内核选项：

- a.

在内核命令行中删除 `nvme_core.multipath=Y` 选项：

```
# grubby --update-kernel=ALL --remove-args="nvme_core.multipath=Y"
```

- b.

在 64 位 IBM Z 构架中更新引导菜单：

```
# zipl
```

- c.

重启系统。

- 

使用内核模块配置文件：

- a.

如果其存在，请从 `/etc/modprobe.d/nvme_core.conf` 文件中删除 `nvme_core multipath=Y` 选项行。

b.

备份 **initramfs** 文件：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m%d-%H%M%S).img
```

c.

重建 **initramfs**：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
# dracut --force --verbose
```

d.

重启系统。

3.

启用 **DM** 多路径：

```
# systemctl enable --now multipathd.service
```

4.

在所有可用路径中分发 I/O。在 **/etc/multipath.conf** 文件中添加以下内容：

```
devices {
    device {
        vendor "NVME"
        product ".*"
        path_grouping_policy group_by_prio
    }
}
```



注意

当 **DM** 多路径管理 **NVMe** 设备时，**/sys/class/nvme-subsys0/iopolicy** 配置文件不会影响 I/O 分发。

5.

重新载入 **multipathd** 服务以应用配置更改：

```
# multipath -r
```

验证

- 验证原生 NVMe 多路径是否已禁用：

```
# cat /sys/module/nvme_core/parameters/multipath
N
```

- 验证 DM 多路径是否可以识别 nvme 设备：

```
# multipath -l

eui.00007a8962ab241100a0980000d851c8 dm-6 NVME,NetApp E-Series
size=20G features='0' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=0 status=active
  |- 0:10:2:2 nvme0n2 259:3 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 4:11:2:2 nvme4n2 259:28 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 5:32778:2:2 nvme5n2 259:38 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 6:32779:2:2 nvme6n2 259:44 active undef running
```

## 其他资源

- [配置内核命令行参数](#)
- [配置 DM 多路径](#)

## 第 18 章 设置磁盘调度程序

磁盘调度程序负责对提交至存储设备的 I/O 请求进行排序。

您可以通过几种不同方式配置调度程序：

- 使用 TuneD 设置调度程序，如[使用 TuneD 设置磁盘调度程序](#)中所述
- 使用 udev 规则设置调度程序，如[使用 udev 规则设置磁盘调度程序](#)中所述
- 在运行中的系统上临时更改调度程序，如临时 [为特定磁盘设置调度程序](#)



### 注意

在 Red Hat Enterprise Linux 8 中，块设备只支持多队列调度。这可让块层性能针对使用快速固态驱动器（SSD）和多核系统进行正常扩展。

Red Hat Enterprise Linux 7 和更早的版本中的传统、单一队列调度程序已被删除。

### 18.1. 可用磁盘调度程序

Red Hat Enterprise Linux 8 支持以下多队列磁盘调度程序：

#### **none**

实施第一出 (FIFO) 调度算法。它将请求合并到通用块层，并通过一个简单的最近缓存来合并。

#### **mq-deadline**

尝试为请求到达调度程序的时间点提供有保证的延迟。

**mq-deadline** 调度程序将排队的 I/O 请求分为读取或写入批处理，然后调度它们以增加逻辑块寻址 (LBA) 顺序执行。默认情况下，读取批处理的优先级高于写入批处理，因为应用程序更有可能阻止



读 I/O 操作。在 `mq-deadline` 批处理后，它会检查写操作在处理器时间耗尽的时间，并根据情况调度下一个读取或写入批处理。

这个调度程序适用于大多数用例，特别是那些写入操作是异步的。

**bfq**

以桌面系统和互动任务为目标。

**bfq** 调度程序可确保任何单个应用程序都不会使用所有带宽。实际上，存储设备总是像它们处于闲置时一样进行响应。在其默认配置中，**bfq** 注重提供最低延迟，而不是达到最大吞吐量。

**BFQ** 基于 **cfq** 代码。它不会为每个进程授予固定时间片段，而是为进程分配一个以扇区数衡量的预算。

在复制大型文件时，这个调度程序不适用。

**kyber**

调度程序调整自身，以通过计算提交到块 I/O 层的每个 I/O 请求的延迟来实现延迟目标。您可以为读取配置目标延迟，如 `cache-misses` 和同步写入请求。

此调度程序适用于快速设备，如 NVMe、SSD 或其他低延迟设备。

18.2. 不同用例的磁盘调度程序

根据系统执行的任务，建议在进行任何分析和调优任务前，将以下磁盘调度程序作为基准：

表 18.1. 适用于不同用例的磁盘调度程序

使用案例	磁盘调度程序
传统的使用 SCSI 接口的 HDD	使用 <b>mq-deadline</b> 或 <b>bfq</b> 。
高性能 SSD 或具有快速存储的 CPU 绑定系统	使用 <b>none</b> ，特别是在运行企业级应用程序时。或者，使用 <b>kyber</b> 。

使用案例	磁盘调度程序
桌面或互动任务	使用 <b>bfq</b> 。
虚拟客户端	使用 <b>mq-deadline</b> 。使用可以多队列的主机总线适配器 (HBA) 驱动程序，使用 <b>none</b> 。

18.3. 默认磁盘调度程序

块设备使用默认的磁盘调度程序，除非您指定了另一个调度程序。



注意

具体来说，对于 非易失性内存 Express (NVMe) 块设备，默认调度程序为 **none**，红帽建议不更改此设备。

内核会根据设备类型选择默认磁盘调度程序。自动选择调度程序通常是最佳设置。如果您需要不同的调度程序，红帽建议使用 **udev** 规则或 **TuneD** 应用程序来配置它。匹配所选设备并只为那些设备切换调度程序。

18.4. 确定活跃磁盘调度程序

此流程决定了哪个磁盘调度程序目前在给定块设备中活跃。

步骤

- 读取 `/sys/block/设备/queue/scheduler` 文件的内容：

```
# cat /sys/block/device/queue/scheduler  
[mq-deadline] kyber bfq none
```

在文件名中，将 **device** 替换为块设备名称，如 **sdc**。

活跃的调度程序列在方括号中 ([ ])。

## 18.5. 使用 TUNED 设置磁盘调度程序

此流程创建并启用 **TuneD** 配置集，该配置集为所选块设备设置给定磁盘调度程序。这个设置会在系统重启后保留。

在以下命令和配置中替换：

- 带有块设备名称的 *device*，如 **sdf**
- 带有您要为该设备设置的磁盘调度程序的 *selected-scheduler*，例如 **bfq**

### 先决条件

- **TuneD** 服务已安装并启用。详情请参阅[安装和启用 TuneD](#)。

### 流程

1. 可选：选择一个您的配置集将要基于的现有 **Tuned** 配置集。有关可用配置集列表，请参阅 [RHEL 提供的 TuneD 配置集](#)。

要查看哪个配置集当前处于活跃状态，请使用：

```
$ tuned-adm active
```

2. 创建一个新目录来保存 **TuneD** 配置集：

```
# mkdir /etc/tuned/my-profile
```

3. 查找所选块设备系统唯一标识符：

```
$ udevadm info --query=property --name=/dev/device | grep -E '(WWN|SERIAL)'

ID_WWN=0x5002538d00000000_
ID_SERIAL=Generic-_SD_MMC_20120501030900000-0:0
ID_SERIAL_SHORT=20120501030900000
```



## 注意

本例中的命令将返回以 **World Wide Name (WWN)** 或与指定块设备关联的序列号的所有值。虽然最好使用 **WWN**，但给定设备始终不能使用 **WWN**，但 **example** 命令返回的任何值都可以接受用作 **device system unique ID**。

4.

创建 `/etc/tuned/my-profile/tuned.conf` 配置文件。在该文件中设置以下选项：

a.

可选：包含现有配置集：

```
[main]
include=existing-profile
```

b.

为与 **WWN** 标识符匹配的设备设置所选磁盘调度程序：

```
[disk]
devices_udev_regex=IDNAME=device system unique id
elevator=selected-scheduler
```

在这里：

- 使用要使用的标识符的名称替换 **IDNAME**（如 **ID\_WWN**）。
- 将 **device system unique id** 替换为所选标识符的值（如 **0x5002538d00000000**）。

要匹配 **devices\_udev\_regex** 选项中的多个设备，将标识符放在括号中，并使用垂直栏来分离它们：

```
devices_udev_regex=(ID_WWN=0x5002538d00000000)|
(ID_WWN=0x1234567800000000)
```

5.

启用您的配置集：

```
# tuned-adm profile my-profile
```

## 验证

1.

验证 **TuneD** 配置集是否活跃并应用：

```
$ tuned-adm active
```

```
Current active profile: my-profile
```

```
$ tuned-adm verify
```

```
Verification succeeded, current system settings match the preset profile.  
See TuneD log file ('/var/log/tuned/tuned.log') for details.
```

2.

读取 `/sys/block/设备/queue/scheduler` 文件的内容：

```
# cat /sys/block/device/queue/scheduler
```

```
[mq-deadline] kyber bfq none
```

在文件名中，将 **device** 替换为块设备名称，如 **sdc**。

活跃的调度程序列在方括号中 ([])。

## 其他资源

- [自定义 TuneD 配置集。](#)

## 18.6. 使用 UDEV 规则设置磁盘调度程序

此流程使用 **udev** 规则为特定块设备设置给定磁盘调度程序。这个设置会在系统重启后保留。

在以下命令和配置中替换：

- 带有块设备名称的 **device**，如 **sdf**

- 带有您要为该设备设置的磁盘调度程序的 ***selected-scheduler***，例如 ***bfq***

## 步骤

1. 查找块设备系统唯一标识符：

```
$ udevadm info --name=/dev/device | grep -E '(WWN|SERIAL)'
E: ID_WWN=0x5002538d00000000
E: ID_SERIAL=Generic_SD_MMC_20120501030900000-0:0
E: ID_SERIAL_SHORT=20120501030900000
```



### 注意

本例中的命令将返回以 **World Wide Name (WWN)** 或与指定块设备关联的序列号的所有值。虽然最好使用 **WWN**，但给定设备始终不能使用 **WWN**，但 **example** 命令返回的任何值都可以接受用作 **device system unique ID**。

2. 配置 **udev** 规则。使用以下内容创建 **/etc/udev/rules.d/99-scheduler.rules** 文件：

```
ACTION=="add|change", SUBSYSTEM=="block", ENV{IDNAME}=="device system unique id", ATTR{queue/scheduler}="selected-scheduler"
```

在这里：

- 使用要使用的标识符的名称替换 **IDNAME**（如 **ID\_WWN**）。
- 将 **device system unique id** 替换为所选标识符的值（如 **0x5002538d00000000**）。

3. 重新载入 **udev** 规则：

```
# udevadm control --reload-rules
```

4. 应用调度程序配置：

```
# udevadm trigger --type=devices --action=change
```

## 验证

- 验证活跃的调度程序：

```
# cat /sys/block/device/queue/scheduler
```

## 18.7. 为特定磁盘临时设置调度程序

此流程为特定块设备设置给定磁盘调度程序。系统重启后该设置不会保留。

## 步骤

- 将所选调度程序的名称写入 `/sys/block/device/queue/scheduler` 文件：

```
# echo selected-scheduler > /sys/block/device/queue/scheduler
```

在文件名中，将 `device` 替换为块设备名称，如 `sdc`。

## 验证

- 验证调度程序是否在该设备中活跃：

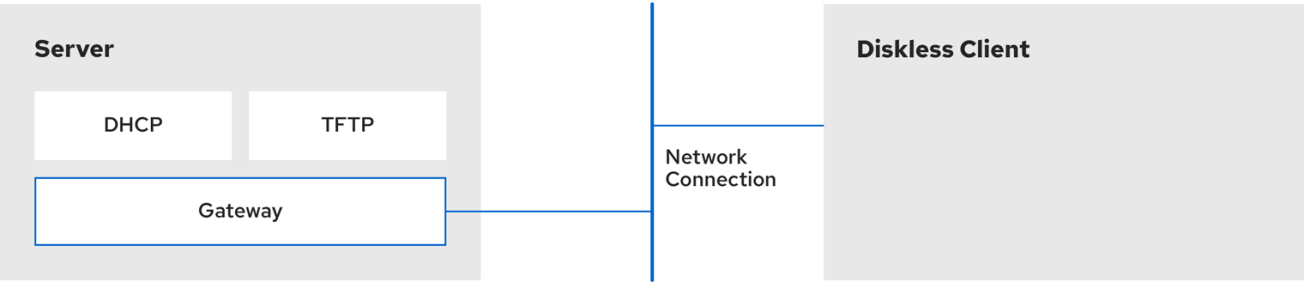
```
# cat /sys/block/device/queue/scheduler
```

第 19 章 设置远程无盘系统

在网络环境中，您可以通过部署远程无盘系统来设置多个具有相同配置的客户端。通过使用当前的 Red Hat Enterprise Linux 服务器版本，您可以为这些客户端保存硬盘驱动器成本，并在单独的服务器上配置网关。

下图描述了通过动态主机配置协议(DHCP)和普通文件传输协议(TFTP)服务的无盘客户端与服务器的连接。

图 19.1. 远程无盘系统设置图



269\_RHEL\_0822

19.1. 为远程无盘系统准备环境

准备您的环境以继续远程无盘系统实施。远程无盘系统引导需要以下服务：

- 普通文件传输协议(TFTP)服务，由 `tftp-server` 提供。系统通过 **Preboot Execution Environment (PXE)**加载程序，通过网络检索内核镜像以及初始 RAM 磁盘 `initrd`。
- DHCP 服务，由 `dhcp` 提供。

先决条件

- 已安装 `xinetd` 软件包。
- 您已设置了网络连接。

流程



1.

安装 dracut-network 软件包：

```
# yum install dracut-network
```

2.

在 /etc/dracut.conf.d/network.conf 文件中添加以下行：

```
add_dracutmodules+=" nfs "
```

3.

通过按照以下顺序配置服务，确保环境中远程无盘系统的功能正确：

a.

配置 TFTP 服务。如需更多信息，[请参阅为无盘客户端配置 TFTP 服务](#)。

b.

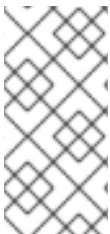
配置 DHCP 服务器。如需更多信息，[请参阅为无盘客户端配置 DHCP 服务器](#)。

c.

配置网络文件系统(NFS)和导出的文件系统。如需更多信息，[请参阅为无盘客户端配置导出的文件系统](#)。

## 19.2. 为无盘客户端配置 TFTP 服务

要使远程无盘系统在您的环境中正常工作，您需要首先为无盘客户端配置普通文件传输协议(TFTP)服务。



### 注意

此配置不会通过统一可扩展固件接口(UEFI)引导。对于基于 UEFI 的安装，[请参阅为基于 UEFI 的客户端配置 TFTP 服务器](#)。

### 先决条件

•

您已安装了以下软件包：

○

tftp-server

- **syslinux**
- **xinetd**

## 流程

1. 启用 **tftp** 服务：

```
# systemctl enable --now tftp
```

2. 在 **tftp** 根目录中创建 **pxelinux** 目录：

```
# mkdir -p /var/lib/tftpboot/pxelinux/
```

3. 将 **/usr/share/syslinux/pxelinux.0** 文件复制到 **/var/lib/tftpboot/pxelinux/** 目录中：

```
# cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/pxelinux/
```

4. 将 **/usr/share/syslinux/ldlinux.c32** 复制到 **/var/lib/tftpboot/pxelinux/** 中：

```
# cp /usr/share/syslinux/ldlinux.c32 /var/lib/tftpboot/pxelinux/
```

5. 在 **tftp** 根目录中创建 **pxelinux.cfg** 目录：

```
# mkdir -p /var/lib/tftpboot/pxelinux/pxelinux.cfg/
```

## 验证

- 检查服务 **tftp** 的状态：

```
# systemctl status tftp
...
Active: active (running)
...
```

### 19.3. 为无盘客户端配置一个 DHCP 服务器

远程无盘系统需要几个预安装的服务才能启用正确的功能。

#### 先决条件

- 安装 Trivial 文件传输协议(TFTP)服务。
- 您已安装了以下软件包：
  - `dhcp-server`
  - `xinetd`
- 您已为无盘客户端配置了 tftp 服务。如需更多信息，[请参阅为无盘客户端配置 TFTP 服务](#)。

#### 流程

1. 在 `/etc/dhcp/dhcpd.conf` 文件中添加以下配置，以设置 DHCP 服务器并启用 Preboot Execution Environment (PXE)以进行引导：

```
option space pxelinux;
option pxelinux.magic code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.reboottime code 211 = unsigned integer 32;
option architecture-type code 93 = unsigned integer 16;

subnet 192.168.205.0 netmask 255.255.255.0 {
    option routers 192.168.205.1;
    range 192.168.205.10 192.168.205.25;

    class "pxeclients" {
        match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
        next-server 192.168.205.1;

        if option architecture-type = 00:07 {
            filename "BOOTX64.efi";
        } else {
            filename "pxelinux/pxelinux.0";
        }
    }
}
```

```
}
}
}
```

您的 DHCP 配置可能根据您的环境而异，如设置租期或固定地址。详情请参阅 [提供 DHCP 服务](#)。



#### 注意

虽然可以使用 libvirt 虚拟机作为无盘客户端，但 libvirt 守护进程提供 DHCP 服务，不使用单独的 DHCP 服务器。在这种情况下，必须使用 libvirt 网络配置中的 `bootp file=<filename>` 选项（`virsh net-edit`）启用网络引导。

2.

启用 `dhcpd.service`：

```
# systemctl enable --now dhcpd.service
```

验证

- 

检查服务 `dhcpd.service` 的状态：

```
# systemctl status dhcpd.service
...
Active: active (running)
...
```

## 19.4. 为无盘客户端配置导出的文件系统

作为在您的环境中配置远程无盘系统的一部分，您必须为无盘客户端配置一个导出的文件系统。

先决条件

- 

您已为无盘客户端配置了 `tftp` 服务。请参阅 [为无盘客户端配置一个 TFTP 服务](#) 部分。

- 

您已配置了动态主机配置协议(DHCP)服务器。请参阅 [为无盘客户端配置一个 DHCP 服务器](#) 部分。

## 流程

1. 通过将根目录添加到 `/etc/exports` 目录中，将网络文件系统(NFS)服务器配置为导出根目录。有关完整的说明集合，请参阅[部署 NFS 服务器](#)
2. 将 Red Hat Enterprise Linux 的完整版本安装到根目录中，以容纳整个无盘客户端。为此，您可以安装新的基础系统或克隆现有的安装。
  - 通过使用导出的文件系统的路径替换 `exported-root-directory`，将 Red Hat Enterprise Linux 安装到导出的位置：

```
# yum install @Base kernel dracut-network nfs-utils --installroot=exported-root-directory --releasever=/
```

通过将 `releasever` 选项设置为 `/`，可以从主机(`/`)系统中检测到 `releasever`。

- 使用 `rsync` 工具与正在运行的系统同步：

```
# rsync -a -e ssh --exclude='/proc/' --exclude='/sys/' example.com:/ exported-root-directory
```

- 使用要通过 `rsync` 工具同步的正在运行的系统的主机名替换 `example.com`。
- 使用导出的文件系统的路径替换 `exported-root-directory`。

请注意，对于这个选项，您必须有一个单独的现有运行系统，通过以上命令将其克隆到服务器。

3. 配置文件系统，可用于导出，然后才能将其用于无盘客户端：

- a. 将无盘客户端支持的内核(`vmlinuz-_kernel-version_pass:attributes`)复制到 `tftp` 引导目录中：

```
# cp exported-root-directory/boot/vmlinuz-kernel-version /var/lib/tftpboot/pxelinux/
```

b.

在本地创建 `initramfs-kernel-version.img` 文件，并将其移到支持 NFS 的根目录中：

```
# dracut --add nfs initramfs-kernel-version.img kernel-version
```

例如：

```
# dracut --add nfs /exports/root/boot/initramfs-5.14.0-202.el9.x86_64.img 5.14.0-202.el9.x86_64
```

创建 `initrd` 的示例，使用当前运行的内核版本，并覆盖现有镜像：

```
# dracut -f --add nfs "boot/initramfs-$(uname -r).img" "$(uname -r)"
```

c.

将 `initrd` 的文件权限改为 `0644`：

```
# chmod 0644 /exported-root-directory/boot/initramfs-kernel-version.img
```



#### 警告

如果您不更改 `initrd` 文件权限，则 `pxelinux.0` 引导装载程序会失败，并显示 `"file not found"` 错误。

d.

将生成的 `initramfs-kernel-version.img` 文件复制到 `tftp` 引导目录中：

```
# cp /exported-root-directory/boot/initramfs-kernel-version.img  
/var/lib/tftpboot/pxelinux/
```

e.

在 `/var/lib/tftpboot/pxelinux/pxelinux.cfg/default` 文件中添加以下配置，来编辑使用 `initrd` 和内核的默认引导配置：

```
default menu.c32  
prompt 0  
menu title PXE Boot Menu
```

```

ontimeout rhel8-over-nfsv4.2
timeout 120
label rhel8-over-nfsv4.2
  menu label Install diskless rhel8{} nfsv4.2{}
  kernel $vmlinuz
  append initrd=$initramfs root=nfs4:$nfsserv:/:vers=4.2,rw rw panic=60
  ipv6.disable=1 console=tty0 console=ttyS0,115200n8
label rhel8-over-nfsv3
  menu label Install diskless rhel8{} nfsv3{}
  kernel $vmlinuz
  append initrd=$initramfs root=nfs:$nfsserv:$nfsroot:vers=3,rw rw panic=60
  ipv6.disable=1 console=tty0 console=ttyS0,115200n8

```

此配置指示无盘客户端 `root` 以读/写格式挂载 `/exported-root-directory` 导出的文件系统。

a.

可选：使用以下配置编辑 `/var/lib/tftpboot/pxelinux/pxelinux.cfg/default` 文件来以只读格式挂载文件系统：

```

default rhel8

label rhel8
  kernel vmlinuz-kernel-version
  append initrd=initramfs-kernel-version.img root=nfs:server-ip:/exported-root-directory ro

```

b.

重启 NFS 服务器：

```
# systemctl restart nfs-server.service
```

现在，您可以将 NFS 共享导出到无盘客户端。这些客户端可以通过预引导执行环境(PXE)，通过网络引导。

## 19.5. 重新配置远程无盘系统

如果要安装软件包、重启服务或调试问题，您可以重新配置系统。

### 先决条件

- 您已在导出的文件系统中启用了 `no_root_squash` 选项。

## 流程

- 更改用户密码：
  - 将命令行改为 */exported/root/directory*:
 

```
# chroot /exported/root/directory /bin/bash
```
  - 更改您想要的用户的密码：
 

```
# passwd <username>
```

将 *<username>* 替换为您要更改密码的实际用户。
  - 退出命令行。
- 在远程无盘系统上安装软件：
 

```
# yum install <package> --installroot=/exported/root/directory --releasever=/ --config /etc/dnf/dnf.conf --setopt=reposdir=/etc/yum.repos.d/
```

用要安装的实际软件包替换 *<package>*。
- 配置两个单独的导出，来将远程无盘系统分成 */usr* 和 */var*。如需更多信息，请参阅 [部署 NFS 服务器](#)。

## 19.6. 排查与加载远程无盘系统相关的常见问题

根据之前的配置，在加载远程无盘系统时可能会出现一些问题。以下是在 Red Hat Enterprise Linux 服务器上对最常见的问题进行故障排除的一些示例。

### 例 19.1. 客户端无法获得 IP 地址



1.  
检查是否服务器上启用了动态主机配置协议(DHCP)服务。
  - a.  
检查 `dhcp.service` 是否在运行：

```
# systemctl status dhcpd.service
```
  - b.  
如果 `dhcp.service` 不活跃，请启用并启动它：

```
# systemctl enable dhcpd.service  
# systemctl start dhcpd.service
```
  - c.  
重启无磁盘客户端。
  - d.  
检查 DHCP 配置文件 `/etc/dhcp/dhcpd.conf`。详情请参阅 [为无盘客户端配置一个 DHCP 服务器](#)。
2.  
检查防火墙端口是否已打开。
  - a.  
检查 `dhcp.service` 是否在活跃服务中列出：

```
# firewall-cmd --get-active-zones  
# firewall-cmd --info-zone=public
```
  - b.  
如果 `dhcp.service` 没有列在活跃的服务中，请将其添加到列表中：

```
# firewall-cmd --add-service=dhcp --permanent
```
  - c.  
检查 `nfs.service` 是否在活跃服务中列出：

```
# firewall-cmd --get-active-zones  
# firewall-cmd --info-zone=public
```
  - d.  
如果 `nfs.service` 没有列在活跃的服务中，将其添加到列表中：

```
# firewall-cmd --add-service=nfs --permanent
```

### 例 19.2. 在引导远程无盘系统的过程中，文件不可用

1.  
检查该文件是否在 `/var/lib/tftpboot/` 目录中。
2.  
如果该文件位于目录中，请确保其是否具有以下权限：

```
# chmod 644 pxelinux.0
```

3.  
检查防火墙端口是否已打开。

### 例 19.3. 载入 `kernel/initrd`后，系统引导失败

1.  
检查是否在服务器上启用了 **NFS** 服务。

- a.  
检查 `nfs.service` 是否正在运行：

```
# systemctl status nfs.service
```

- b.  
如果 `nfs.service` 停用，您必须启动并启用它：

```
# systemctl start nfs.service  
# systemctl enable nfs.service
```

2.  
检查 `/var/lib/tftpboot/pxelinux.cfg/` 目录中的参数是否正确。详情请参阅 [为无盘客户端配置一个导出的文件系统](#)。
3.  
检查防火墙端口是否已打开。

## 第 20 章 管理 RAID

您可以使用独立磁盘的冗余阵列 (RAID) 在多个驱动器间存储数据。它可以帮助避免驱动器失败时数据丢失。

### 20.1. RAID 概述

在 RAID 中，多个设备（如 HDD、SSD 或 NVMe）组合成一个阵列来实现性能或冗余目标，而不是使用一个大而昂贵的驱动器。这组设备以单一逻辑存储单元或驱动器在计算机中出现。

RAID 支持各种配置，包括 0、1、4、5、6、10 和 linear。RAID 使用磁盘条带 (RAID 0)、磁盘镜像 (RAID 1) 和带有奇偶校验 (RAID 4、5 和 6) 的磁盘条带等技术来实现冗余、较低延迟、增大带宽以及最大程度从硬盘崩溃中恢复的能力。

RAID 通过将数据分解为一致的块（通常为 256 KB 或 512 KB）来跨阵列中的每个设备分发数据，但其他值可以接受。根据所使用的 RAID 级别，它会将这些块写入 RAID 阵列中的硬盘中。在读取数据时，进程是相反的顺序，实现阵列中的多个设备实际上是一个大驱动器的效果。

对于需要管理大量数据的人员，RAID 技术非常有用。以下是部署 RAID 的主要原因：

- 它可以提高速度
- 它使用单个虚拟磁盘增加存储容量
- 尽可能减少磁盘出现问题时导致数据丢失的问题
- RAID 布局和级别在线转换

### 20.2. RAID 类型

以下是可能的 RAID 类型：

## 固件 RAID

固件 RAID（也称为 ATARAID）是一种软件 RAID，可以使用基于固件的菜单配置 RAID 集。此类 RAID 使用的固件也会挂接到 BIOS 中，允许您从 RAID 集启动。不同的供应商使用不同的磁盘元数据格式来标记 RAID 集成员。Intel Matrix RAID 是固件 RAID 系统的示例。

## 硬件 RAID

基于硬件的阵列独立于主机管理 RAID 子系统。它可能会在每个 RAID 阵列中存在多个设备。

硬件 RAID 设备可能是系统内部或外部的。内部设备通常由一个专用的控制器卡组成，它处理对操作系统透明的 RAID 任务。外部设备通常通过 SCSI、光纤、iSCSI、InfiniBand 或者其它高速网络互联连接的系统,并显示卷如逻辑单元到系统。

RAID 控制器卡的功能类似于操作系统的 SCSI 控制器，并处理所有实际驱动器通信。您可以将驱动器插入与普通 SCSI 控制器类似的 RAID 控制器，然后将其添加到 RAID 控制器配置中。操作系统将无法辨别它们的不同。

## 软件 RAID

软件 RAID 在内核块设备代码中实现各种 RAID 级别。它提供最便宜的解决方案，因为不需要昂贵的磁盘控制器卡或热插拔机箱。使用热插拔机箱时，您可以在不关闭系统的情况下删除硬盘驱动器。软件 RAID 也可用于任何由 Linux 内核支持的块存储，如 SATA、SCSI 和 NVMe。随着新 CPU 的速度加快，除非您使用高端存储设备，软件 RAID 通常会优于硬件 RAID。

由于 Linux 内核包含多个设备 (MD) 驱动程序，因此 RAID 解决方案将完全独立于硬件。基于软件的阵列的性能取决于服务器 CPU 的性能和负载。

以下是 Linux 软件 RAID 堆栈的主要功能：

- 多线程设计
- 在不同的 Linux 机器间移动磁盘阵列不需要重新构建数据
- 使用空闲系统资源进行后台阵列重构

- 支持热插拔驱动器
- 自动 CPU 检测以利用某些 CPU 功能，如流传输单一指令多个数据 (SIMD) 支持。
- 自动更正阵列磁盘中的错误扇区。
- 定期检查 RAID 数据，以确保阵列的健康状态。
- 主动监控阵列，在发生重要事件时将电子邮件报警发送到指定的电子邮件地址。
- Write-intent 位图通过允许内核准确了解磁盘的哪些部分需要重新同步，而不必在系统崩溃后重新同步整个阵列，可以大大提高了重新同步事件的速度。



注意

resync 是一个通过现有 RAID 设备同步数据以达到冗余的过程

- 重新同步检查点，以便如果您在重新同步期间重新启动计算机，则在启动时重新同步会从其停止的地方开始，而不是从头开始
- 安装后更改阵列参数的功能，称为重塑 (reshaping)。例如：当有新设备需要添加时，您可以将 4 磁盘 RAID5 阵列增加成 5 磁盘 RAID5 阵列。这种增加操作是实时的，不需要您在新阵列上重新安装。
- 重塑支持更改设备数量、RAID 算法或 RAID 阵列类型的大小，如 RAID4、RAID5、RAID6 或 RAID10。
- 接管支持 RAID 级别转换，比如 RAID0 到 RAID6。
- 集群 MD 是集群的存储解决方案，可为集群提供 RAID1 镜像的冗余。目前，只支持

## RAID1。

### 20.3. 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，RAID0 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。

## 20.4. 支持的 RAID 转换

可以从一个 RAID 级别转换到另一个 RAID 级别。例如：您可以从 RAID5 转换到 RAID10，但不能从 RAID10 转换为 RAID5。下表描述了支持的 RAID 转换：

RAID 转换级别	转换步骤	注
RAID 级别 0 到 RAID 4	<pre># mdadm --grow /dev/md0 --level=4 -n3 --add /dev/vdd</pre>	您需要在 MD 阵列中添加磁盘，因为它至少需要 3 个磁盘。
RAID 级别 0 到 RAID 5	<pre># mdadm --grow /dev/md0 --level=5 -n3 --add /dev/vdd</pre>	您需要在 MD 阵列中添加磁盘，因为它至少需要 3 个磁盘。
RAID 0 到 RAID 10	<pre># mdadm --grow /dev/md0 --level 10 -n 4 --add /dev/vd[ef]</pre>	您需要在 MD 阵列中添加两个额外磁盘。
RAID 1 到 RAID 级别 0	<pre># mdadm --grow /dev/md0 -l0</pre>	
RAID 1 到 RAID 级别 5	<pre># mdadm --grow /dev/md0 --level=5</pre>	
RAID 4 到 RAID 0	<pre># mdadm --grow /dev/md0 --level=0</pre>	
RAID 4 到 RAID 5	<pre># mdadm --grow /dev/md0 --level=5</pre>	
RAID 5 到 RAID 级别 0	<pre># mdadm --grow /dev/md0 --level=0</pre>	



RAID 转换级别	转换步骤	注
RAID 5 到 RAID 级别 1	<pre># mdadm -CR /dev/md0 -l5 -n3 /dev/sd[abc] --assume-clean --size 1G  # mdadm -D /dev/md0   grep Level  # mdadm --grow /dev/md0 --array-size 1048576  # mdadm --grow -n 2 /dev/md0 -- backup=internal  # mdadm --grow -l1 /dev/md0  # mdadm -D /dev/md0   grep Level</pre>	
RAID 5 到 RAID 4	<pre># mdadm --grow /dev/md0 --level=4</pre>	
RAID 5 到 RAID 6	<pre># mdadm --grow /dev/md0 --level=6 --add /dev/vde</pre>	
RAID 5 到 RAID 10	<pre># mdadm --grow /dev/md0 --level=0 # mdadm --grow /dev/md0 --level=10 --add /dev/vde /dev/vdf</pre>	<p>将 RAID 5 转换为 RAID 10 是两个步骤转换：</p> <ol style="list-style-type: none"> <li>1. 转换为 RAID 0</li> <li>2. 在转换为 RAID10 时添加两个附加磁盘。</li> </ol>
RAID 6 到 RAID 5	<pre># mdadm --grow /dev/md0 --level=5</pre>	
RAID 10 到 RAID 级别 0	<pre># mdadm --grow /dev/md0 --level=0</pre>	

**注意**

只有在使用 **ALGORITHM\_PARITY\_N** 布局时，才可以将 **RAID 5** 转换为 **RAID0** 和 **RAID4**。

转换 RAID 级别后，使用 **mdadm --detail /dev/md0** 或 **cat /proc/mdstat** 命令验证转换。

**其他资源**

- [系统中 mdadm \(8\) 手册页](#)

**20.5. RAID 子系统**

以下子系统组成 RAID：

**硬件 RAID 控制器驱动程序**

硬件 RAID 控制器没有特定的 RAID 子系统。由于它们使用特殊的 RAID 芯片组，因此硬件 RAID 控制器有自己的驱动程序。使用这些驱动程序时，系统会检测 RAID 集作为常规磁盘。

**mdraid**

**mdraid** 子系统设计为软件 RAID 解决方案。这也是 Red Hat Enterprise Linux 中软件 RAID 的首选解决方案。此子系统使用自己的元数据格式，称为原生 MD 元数据。

它还支持其他元数据格式，称为外部元数据。Red Hat Enterprise Linux 8 使用带有外部元数据的 **mdraid** 来访问 Intel Rapid Storage (ISW) 或 Intel Matrix Storage Manager (IMSM) 设置和存储网络行业关联 (SNIA) 磁盘驱动器格式 (DDF)。mdraid 子系统集通过 **mdadm** 程序进行配置和控制。

**20.6. 在安装过程中创建软件 RAID**

独立磁盘冗余阵列 (RAID) 设备由被安排的多个存储设备组成，以便在一些配置中提供更高的性能和容错能力。创建 RAID 设备只需要一步，并可根据需要添加或者删除磁盘。您可以为系统中的每个物理磁盘配置一个 RAID 分区，因此安装程序可使用的磁盘数决定可用 RAID 设备的级别。例如：如果您的系统有两个磁盘，则无法创建 RAID 10 设备，因为它至少需要三个独立的磁盘。要优化您的系统的存储性能和可靠性，RHEL 支持软件 RAID 0、RAID 1、RAID 4、RAID 5、RAID 6、RAID 6 和 RAID 10 类型，以便在安装的系统中设置存储。



## 注意

在 64 位 IBM Z 上，存储子系统以透明方式使用 RAID。您不必手动配置软件 RAID。

## 先决条件

- 您已经选择了两个或者多个磁盘，然后才能看到 RAID 配置选项。根据您要创建的 RAID 类型，至少需要两个磁盘。
- 您创建了挂载点。通过配置挂载点，您可以配置 RAID 设备。
- 您已在 安装目的 窗口中选择了 自定义 单选按钮。

## 流程

1. 在 **Manual Partitioning** 窗口左面地框中，选所需的分区。
2. 在 **Device(s)** 部分点 **修改**。此时会打开 **Configure Mount Point** 对话框。
3. 选择您要包含在 RAID 设备中的磁盘并点击 **选择**。
4. 点击设备类型下拉菜单并选择 **RAID**。
5. 点击文件系统下拉菜单并选择您首选的文件系统类型。
6. 点击 RAID 级别下拉菜单并选择您需要的 RAID 级别。
7. 点击 **更新设置** 保存您的更改。
8. 点 **Done** 应用设置并返回到按照概述窗口。

## 其他资源

- [创建带有 DM 完整性的 RAID LV](#)
- [管理 RAID](#)

## 20.7. 在安装的系统中创建软件 RAID

您可以使用 `mdadm` 程序在现有系统上创建一个软件独立磁盘阵列 (RAID)。

### 先决条件

- 已安装 `mdadm` 软件包。
- 您已在系统上创建了两个或多个分区。有关具体说明，请参阅 [使用 parted 创建分区](#)。

### 流程

1. 创建两个块设备的 RAID，如 `/dev/sda1` 和 `/dev/sdc1`：

```
# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sda1 /dev/sdc1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

`level_value` 选项定义 RAID 级别。

2. 可选：检查 RAID 的状态：

```
# mdadm --detail /dev/md0
/dev/md0:
  Version : 1.2
  Creation Time : Thu Oct 13 15:17:39 2022
  Raid Level : raid0
  Array Size : 18649600 (17.79 GiB 19.10 GB)
  Raid Devices : 2
  Total Devices : 2
  Persistence : Superblock is persistent
```

```

Update Time : Thu Oct 13 15:17:39 2022
State : clean
Active Devices : 2
Working Devices : 2
Failed Devices : 0
Spare Devices : 0
[...]

```

3.

可选：观察 RAID 中每个设备的详细信息：

```

# mdadm --examine /dev/sda1 /dev/sdc1
/dev/sda1:
    Magic : a92b4efc
    Version : 1.2
    Feature Map : 0x1000
    Array UUID : 77ddfb0a:41529b0e:f2c5cde1:1d72ce2c
    Name : 0
    Creation Time : Thu Oct 13 15:17:39 2022
    Raid Level : raid0
    Raid Devices : 2
[...]

```

4.

在 RAID 驱动器中创建文件系统：

```
# mkfs -t xfs /dev/md0
```

使用您选择格式化驱动器的文件系统替换 *xfs*。

5.

为 RAID 驱动器创建挂载点并挂载它：

```

# mkdir /mnt/raid1
# mount /dev/md0 /mnt/raid1

```

使用挂载点替换 */mnt/raid1*。

如果您希望 RHEL 在系统引导时自动挂载 md0 RAID 设备，请将设备的条目添加到 */etc/fstab* 文件中：

```
/dev/md0 /mnt/raid1 xfs defaults 0 0
```

## 20.8. 在 WEB 控制台中创建 RAID

在 RHEL 8 web 控制台中配置 RAID。

### 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](#)。

- 您已在系统中安装了 cockpit-storaged 软件包。
- 您已连接了物理磁盘，它们可以被系统可见。

### 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. 在面板中，单击 **Storage**。
3. 在存储 表中，单击菜单按钮，然后选择 创建 MDRAID 设备。
4. 在 Create RAID Device 字段中，为新 RAID 输入一个名称。

5. 在 RAID 级别下拉列表中，选择您要使用的 RAID 级别。

6. 在 Chunk Size 下拉列表中，从可用选项列表中选择大小。

Chunk Size 值指定用于写数据的每个块有多大。例如，如果块大小为 512 KiB，系统将把第一个 512 KiB 写入第一个磁盘中，第二个 512 KiB 将被写入第二个磁盘中，第三个块将被写入第三个磁盘中。如果您的 RAID 中有三个磁盘，则第四个 512 KiB 被再次写入第一个磁盘中。

7. 选择您要用于 RAID 的磁盘。

8. 点 Create。

#### 验证

- 进到 Storage 部分，并在 RAID devices 框中选中您可以看到的新 RAID。

### 20.9. 在 WEB 控制台中格式化 RAID

您可以在 RHEL 8 web 控制台中格式化并挂载软件 RAID 设备。

根据卷大小以及选择格式化选项，格式化可能需要几分钟。

#### 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](#)。

- 已安装 **cockpit-storage** 软件包。
- 您已连接了物理磁盘，它们可以被系统可见。
- 您已创建了 **RAID**。

## 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. 在面板中，单击 **Storage**。
3. 在 **Storage** 表中，点您要格式化的 **RAID** 设备的菜单按钮 **swig**。
4. 从下拉菜单中选择 **Format**。
5. 在 **Format** 字段中输入名称。
6. 在 **Mount Point** 字段中添加挂载路径。
7. 从 **Type** 下拉列表中选择文件系统的类型。
8. 可选：使用零选项检查 **Overwrite** 现有数据，如果磁盘包含任何敏感数据，并且您要覆盖它们。否则 **RHEL web 控制台** 只重写磁盘标头。
9. 在 **Encryption** 下拉菜单中选择加密类型。如果您不想加密卷，请选择 **No encryption**。



10. 在 **At boot** 下拉菜单中选择您要何时挂载卷。
11. 在 **Mount options** 部分中 :
  - a. 如果您希望将卷挂载为只读逻辑卷，请选择 **Mount read only** 复选框。
  - b. 如果要更改默认挂载选项，请选中 **自定义挂载选项** 复选框并添加挂载选项。
12. 格式化 RAID 分区 :
  - 如果要格式化并挂载分区，请点 **Format and mount** 按钮。
  - 如果您只想格式化分区，请点 **Format only** 按钮。

#### 验证

- 格式化成功完成后，您可以在 **Storage** 页的 **Storage** 表中看到格式化逻辑卷的详情。

### 20.10. 使用 WEB 控制台在 RAID 上创建分区表

在 **RHEL 8** 接口中创建的新软件 **RAID** 设备中使用分区表格式化 **RAID**。

#### 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](#)。

- 已安装 `cockpit-storaged` 软件包。
- 您已连接了物理磁盘，它们可以被系统可见。
- 您已创建了 RAID。

## 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. 在面板中，单击 **Storage**。
3. 在 **Storage** 表中，点您要在其上创建分区表的 RAID 设备。
4. 点 **MDRAID device** 部分中的菜单按钮。
5. 从下拉菜单中选择 **Create partition table**。
6. 在 **Initialize disk** 对话框中选择以下内容：
  - a. **Partitioning** :
    - 如果分区应该与所有系统和设备兼容，请选择 **MBR**。

- 如果分区应该与现代系统兼容，且硬盘必须大于 2 TB，请选择 **GPT**。

- 如果您不需要分区，请选择 **No partitioning**。

b.

**Overwrite :**

- 如果磁盘包含任何敏感数据并希望覆盖它们，请检查 **Overwrite existing data with zeros** 选项。否则 RHEL web 控制台只重写磁盘标头。

7.

单击 **Initialize**。

## 20.11. 使用 WEB 控制台在 RAID 上创建分区

在现有分区表中创建一个分区。您可以在创建分区后创建更多的分区。

### 先决条件

- RHEL 8 web 控制台已安装并可以访问。详情请参阅[安装 Web 控制台](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- 创建 RAID 上的分区表。

### 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. 在面板中，单击 **Storage**。

3.  
    点击您要在其上创建分区的 **RAID** 设备。
4.  
    在 **RAID** 设备页面中，滚动到 **GPT 分区** 部分，然后点击菜单按钮 [ &&]。
5.  
    点 **Create partition**，然后在 **Create partition** 字段中输入文件系统的名称。不要在名称中使用空格。
6.  
    在 **Mount Point** 字段中输入挂载路径。
7.  
    在 **Type** 下拉列表中选择文件系统的类型。
8.  
    在 **Size slider** 中，设置分区的大小。
9.  
    可选：如果磁盘包含任何敏感数据且您要覆盖它们，请选择 **Overwrite existing data with zeros**。否则 **RHEL web** 控制台只重写磁盘标头。
10.  
    在 **Encryption** 下拉菜单中选择加密类型。如果您不想加密卷，请选择 **No encryption**。
11.  
    在 **At boot** 下拉菜单中选择您要何时挂载卷。
12.  
    在 **Mount options** 部分中：
  - a.  
        如果要将在卷挂载到只读逻辑卷，请选择 **Mount read only** 复选框。
  - b.  
        如果要更改默认挂载选项，请选中 **自定义挂载选项** 复选框并添加挂载选项。
13.  
    创建分区：

- 如果要创建并挂载分区，请点 **Create and mount** 按钮。
- 如果您只想创建分区，点 **Create only** 按钮。

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

- 您可以在主存储页的 **Storage** 表中看到格式化的逻辑卷的详情。

### 20.12. 使用 WEB 控制台在 RAID 上创建卷组

从软件 RAID 构建卷组。

#### 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](#)。

- 已安装 **cockpit-storaged** 软件包。
- 您有一个没有格式化的 **RAID** 设备，且没有挂载。

#### 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.

在面板中，单击 **Storage**。

3.

在 **Storage** 表中，点菜单按钮 [ swig] 并选择 **Create LVM2 卷组**。

4.

在 **Create LVM2 卷组** 字段中，为新卷组输入一个名称。

5.

在 **Disks** 列表选择一个 **RAID** 设备。

如果您在列表中没有看到 **RAID**，从系统中卸载 **RAID**。RHEL 8 系统不能使用 **RAID** 设备。

6.

点 **Create**。

### 20.13. 使用 STORAGE RHEL 系统角色配置一个 RAID 卷

使用 **storage** 系统角色，您可以使用 Red Hat Ansible Automation Platform 和 Ansible-Core 在 RHEL 上配置 RAID 卷。使用参数创建一个 Ansible playbook，以配置 RAID 卷以满足您的要求。



#### 警告

设备名称在某些情况下可能会改变，例如：当您在系统中添加新磁盘时。因此，为了避免数据丢失，请在 **playbook** 中使用持久性命名属性。如需有关持久性命名属性的更多信息，请参阅 [持久性命名属性概述](#)。

#### 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 a RAID on sdd, sde, sdf, and sdg
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_safe_mode: false
        storage_volumes:
          - name: data
            type: raid
            disks: [sdd, sde, sdf, sdg]
            raid_level: raid0
            raid_chunk_size: 32 KiB
            mount_point: /mnt/data
            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

- 验证阵列是否已正确创建：

```
# ansible managed-node-01.example.com -m command -a 'mdadm --detail /dev/md/data'
```

## 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

## 20.14. 扩展 RAID

您可以使用 `mdadm` 工具的 `--grow` 选项扩展 RAID。

## 先决条件

- 有足够的磁盘空间。
- `parted` 软件包已经安装。

## 流程

1. 扩展 RAID 分区。如需更多信息，请参阅[使用 parted 重新定义分区](#)。
2. 将 RAID 扩展到最大分区容量：

```
# mdadm --grow --size=max /dev/md0
```



要设置特定大小，设置 `--size` 参数的值（单位为 kB），如 `--size=524228`。

3.

增加文件系统的大小。例如，如果卷使用 XFS 并挂载到 `/mnt/`，请输入：

```
# xfs_growfs /mnt/
```

#### 其他资源

- [系统中 mdadm \(8\) 手册页](#)
- [管理文件系统](#)

### 20.15. 缩小 RAID

您可以使用 `mdadm` 工具的 `--grow` 选项缩小 RAID。



#### 重要

XFS 文件系统不支持缩小。

#### 先决条件

- `parted` 软件包已经安装。

#### 流程

1. 缩小文件系统。如需更多信息，请参阅[管理文件系统](#)。
2. 减少 RAID 的大小，例如 `512 MB`：

```
# mdadm --grow --size=524228 /dev/md0
```

写 `--size` 参数（单位为 kB）。

3. 将分区缩小到您需要的大小。

## 其他资源

- [系统中 mdadm \(8\) 手册页](#)
- [使用 parted 重新定义分区大小](#)

## 20.16. 安装后将根磁盘转换为 RAID1

您可在安装 Red Hat Enterprise Linux 8 后将非 RAID 根磁盘转换为 RAID1 镜像。

在 PowerPC (PPC) 构架中，执行以下步骤：

## 先决条件

- 完成红帽知识库解决方案中的步骤，[在安装 Red Hat Enterprise Linux 7 后如何将根磁盘转换为 RAID1？](#)



### 注意

执行 `grub2-install /dev/sda` 命令无法在 PowerPC 机器上工作并返回错误，但系统会按预期引导。

## 流程

1. 将 PowerPC Reference Platform (PReP) 引导分区从 `/dev/sda1` 复制到 `/dev/sdb1`：

```
# dd if=/dev/sda1 of=/dev/sdb1
```

2.

在两个磁盘的第一个分区中更新 **prep** 和 **boot** 标志：

```
$ parted /dev/sda set 1 prep on
$ parted /dev/sda set 1 boot on

$ parted /dev/sdb set 1 prep on
$ parted /dev/sdb set 1 boot on
```

## 20.17. 创建高级 RAID 设备

在某些情况下，您可能想要在安装完成前创建的阵列上安装操作系统。通常，这意味着在复杂的 RAID 设备中设置 **/boot** 或 **root** 文件系统阵列。在这种情况下，您可能需要使用 **Anaconda** 安装程序不支持的数组选项。要临时解决这个问题，请执行以下步骤。



### 注意

安装程序的有限 **Rescue Mode** 不包括 **man page**。**mdadm** 和 **md man page** 都包含创建自定义 RAID 阵列的有用信息，在整个临时解决方案中可能需要。

### 流程

1.

插入安装磁盘。

2.

在初始启动过程中，选择 **Rescue Mode** 而不是 **Install** 或 **Upgrade**。当系统完全引导至 **Rescue mode** 时，您可以看到命令行终端。

3.

在这个终端中执行以下命令：

a.

使用 **parted** 命令在目标硬盘上创建 RAID 分区。

b.

使用这些分区中的 **mdadm** 命令，使用任何以及所有可用的设置和选项来手动创建 **raid** 阵列。

4.

可选：创建阵列后，在阵列上创建文件系统。

5. 重新启动计算机，再选择要安装的 **Install** 或 **Upgrade**。当 **Anaconda** 安装程序搜索系统中的磁盘时，它会找到预先存在的 **RAID** 设备。
6. 当问到如何使用系统中的磁盘时，请选择 **Custom Layout**，并点 **Next**。在设备列表中，会列出预先存在的 **MD RAID** 设备。
7. 选择一个 **RAID** 设备并点 **Edit**。
8. 如果之前没有创建挂载点，则配置其挂载点以及应使用的文件系统类型，然后点 **Done**。  
**Anaconda** 安装至此已存在的 **RAID** 设备，保留在 **Rescue Mode** 中创建时选择的自定义选项。

## 20.18. 设置用于监控 RAID 的电子邮件通知

您可以使用 **mdadm** 工具设置电子邮件警报来监控 **RAID**。当 **MAILADDR** 变量设置为所需的电子邮件地址后，监控系统会将警报发送到添加的电子邮件地址。

### 先决条件

- 已安装 **mdadm** 软件包。
- 设定邮件服务。

### 流程

1. 通过扫描 **RAID** 详情来创建 **/etc/mdadm.conf** 配置文件来监控阵列：

```
# mdadm --detail --scan >> /etc/mdadm.conf
```

请注意，**ARRAY** 和 **MAILADDR** 是强制的变量。

2. 使用您选择的文本编辑器打开 **/etc/mdadm.conf** 配置文件，并使用通知的电子邮件地址添加 **MAILADDR** 变量。例如，添加新行：

—

**MAILADDR** *example@example.com*

在这里，*example@example.com* 是从阵列监控接收警报的电子邮件地址。

3. 保存 `/etc/mdadm.conf` 文件中的更改并关闭它。

#### 其他资源

- [系统中 mdadm.conf \(5\) 手册页](#)

### 20.19. 替换 RAID 中失败的磁盘

您可以使用剩余的磁盘从失败磁盘重建数据。RAID 级别和磁盘总数决定了成功重建数据所需的最小剩余磁盘量。

在此过程中，`/dev/md0` RAID 包含四个磁盘。`/dev/sdd` 磁盘失败，您需要将它替换为 `/dev/sdf` 磁盘。

#### 先决条件

- 用于替换的备用磁盘。
- 已安装 `mdadm` 软件包。

#### 流程

1. 检查失败的磁盘：
  - a. 查看内核日志：
 

```
# journalctl -k -f
```
  - b. 搜索类似如下的消息：

```
md/raid:md0: Disk failure on sdd, disabling device.
```

```
md/raid:md0: Operation continuing on 3 devices.
```

c.

按 **Ctrl+C** 键退出 **journalctl** 程序。

2.

将失败的磁盘标记为故障：

```
# mdadm --manage /dev/md0 --fail /dev/sdd
```

3.

可选：检查失败的磁盘是否已正确标记：

```
# mdadm --detail /dev/md0
```

输出末尾是 **/dev/md0** RAID 中的磁盘列表，其中磁盘 **/dev/sdd** 具有 **faulty** 状态：

```
Number Major Minor RaidDevice State
 0     8    16     0   active sync  /dev/sdb
 1     8    32     1   active sync  /dev/sdc
-     0     0     2   removed
 3     8    64     3   active sync  /dev/sde

 2     8    48    -   faulty   /dev/sdd
```

4.

从 RAID 中删除失败的磁盘：

```
# mdadm --manage /dev/md0 --remove /dev/sdd
```



### 警告

如果您的 RAID 无法与另一个磁盘失败，在新磁盘有 **active sync** 状态前不会删除任何磁盘。您可以使用 **watch cat /proc/mdstat** 命令监控进度。

5.

在 RAID 中添加新磁盘：

```
# mdadm --manage /dev/md0 --add /dev/sdf
```

**/dev/md0 RAID** 现在包括新磁盘 **/dev/sdf**，**mdadm** 服务将自动从其他磁盘将数据复制到其中。

验证

•

检查阵列的详情：

```
# mdadm --detail /dev/md0
```

如果这个命令显示 **/dev/md0 RAID** 中的磁盘列表，其中新磁盘在输出末尾具有 **spare rebuilding** 状态，则数据仍会从其他磁盘复制到其中：

Number	Major	Minor	RaidDevice	State
0	8	16	0	active sync /dev/sdb
1	8	32	1	active sync /dev/sdc
4	8	80	2	spare rebuilding /dev/sdf
3	8	64	3	active sync /dev/sde

数据复制完成后，新磁盘会处于 **active sync** 状态。

其他资源

•

[设置用于监控 RAID 的电子邮件通知](#)

## 20.20. 修复 RAID 磁盘

您可以使用 **repair** 选项修复 RAID 阵列中的磁盘。

先决条件

•

已安装 **mdadm** 软件包。

## 流程

1.

检查阵列失败的磁盘行为：

```
# echo check > /sys/block/md0/md/sync_action
```

这会检查数组和 `/sys/block/md0/md/sync_action` 文件显示 `sync` 操作。

2.

使用您选择的文本编辑器打开 `/sys/block/md0/md/sync_action` 文件，并查看是否有任何有关磁盘同步失败的消息。

3.

查看 `/sys/block/md0/md/mismatch_cnt` 文件。如果 `mismatch_cnt` 参数不是 0，这意味着 RAID 磁盘需要修复。

4.

修复阵列中的磁盘：

```
# echo repair > /sys/block/md0/md/sync_action
```

这会修复阵列中的磁盘，并将结果写入 `/sys/block/md0/md/sync_action` 文件。

5.

查看同步进度：

```
# cat /sys/block/md0/md/sync_action
repair

# cat /proc/mdstat
Personalities : [raid0] [raid6] [raid5] [raid4] [raid1]
md0 : active raid1 sdg[1] dm-3[0]
      511040 blocks super 1.2 [2/2] [UU]
unused devices: <none>
```



## 第 21 章 使用 LUKS 加密块设备

通过使用磁盘加密，您可以通过对其进行加密来保护块设备上的数据。要访问设备的解密内容，请输入密码短语或密钥作为验证。这对移动计算机和可移动介质非常重要，因为它有助于保护设备的内容，即使它在物理已从系统中移除。LUKS 格式是 Red Hat Enterprise Linux 中块设备加密的默认实现。

### 21.1. LUKS 磁盘加密

Linux Unified Key Setup-on-disk-format (LUKS)提供了一组简化管理加密设备的工具。使用 LUKS，您可以加密块设备，并使多个用户密钥能够解密主密钥。要批量加密分区，请使用这个主密钥。

Red Hat Enterprise Linux 使用 LUKS 执行块设备加密。默认情况下，在安装过程中不选中加密块设备的选项。如果您选择加密磁盘的选项，则系统会在每次引导计算机时都提示您输入密码短语。这个密码短语解锁解密分区的批量加密密钥。如果要修改默认分区表，您可以选择要加密的分区。这是在分区表设置中设定的。

#### 加密系统

LUKS 使用的默认密码是 `aes-xts-plain64`。LUKS 的默认密钥大小为 512 字节。Anaconda XTS 模式的 LUKS 的默认密钥大小为 512 位。以下是可用的密码：

- 高级加密标准(AES)
- Twofish
- Serpent

#### LUKS 执行的操作

- LUKS 对整个块设备进行加密，因此非常适合保护移动设备的内容，如可移动存储介质或笔记本电脑磁盘驱动器。
- 加密块设备的底层内容是任意的，这有助于加密交换设备。对于将特殊格式化块设备用于数据存储的某些数据库，这也很有用。

- LUKS 使用现有的设备映射器内核子系统。
- LUKS 增强了密码短语，防止字典攻击。
- LUKS 设备包含多个密钥插槽，这意味着您可以添加备份密钥或密码短语。



### 重要

在以下情况下不建议使用 LUKS：

- LUKS 等磁盘加密解决方案仅在您的系统关闭时保护数据。在系统启动且 LUKS 解密磁盘后，该磁盘上的文件可被任何有权访问它们的用户使用。
- 需要多个用户对同一设备具有不同的访问密钥的情况。LUKS1 格式提供八个密钥插槽，LUKS2 提供最多 32 个密钥插槽。
- 需要文件级加密的应用程序。

### 其它资源

- [LUKS 项目主页](#)
- [LUKS 磁盘格式规范](#)
- [FIPS 197: 高级加密标准\(AES\)](#)

## 21.2. RHEL 中的 LUKS 版本

在 Red Hat Enterprise Linux 中，LUKS 加密的默认格式为 LUKS2。旧的 LUKS1 格式仍被完全支持，并作为与早期 Red Hat Enterprise Linux 版本兼容的格式提供。与 LUKS1 重新加密相比，LUKS2 重新加密被视为更强大且更安全。

LUKS2 格式允许将来对各种部分的更新，而无需修改二进制结构。它在内部对元数据使用 JSON 文本格式，提供元数据的冗余，检测元数据损坏，并从元数据副本自动修复。



重要

不要在只支持 LUKS1 的系统中使用 LUKS2，因为 LUKS2 和 LUKS1 使用不同的命令来加密磁盘。对 LUKS 版本使用错误的命令可能会导致数据丢失。

表 21.1. 加密命令取决于 LUKS 版本

LUKS 版本	加密命令
LUKS2	<code>cryptsetup reencrypt</code>
LUKS1	<code>cryptsetup-reencrypt</code>

在线重新加密

LUKS2 格式支持在设备正在使用时重新加密加密设备。例如：您不必卸载该设备中的文件系统来执行以下任务：

- 更改卷密钥
- 更改加密算法

加密未加密的设备时，您仍然必须卸载文件系统。您可以在简短初始化加密后重新挂载文件系统。

LUKS1 格式不支持在线重新加密。

转换

在某些情况下，您可以将 LUKS1 转换为 LUKS2。在以下情况下无法进行转换：

- LUKS1 设备被标记为被基于策略的解密(PBD) Clevis 解决方案使用。当检测到某些 luksmeta 元数据时，cryptsetup 工具不会转换设备。

- 设备正在活跃。在任何可能的转换前，设备必须处于不活跃状态。

### 21.3. LUKS2 重新加密过程中数据保护选项

在重新加密过程中，LUKS2 提供了几个优先考虑性能或数据保护的选项。它为弹性选项提供以下模式，您可以使用 `cryptsetup reencrypt -- resilience -mode /dev/ <device_ID>` 命令选择其中的任何模式，您可以在其中将 `< device_ID >` 替换为您的设备的 ID。

#### checksum

默认模式。它在数据保护和性能之间保持平衡。

这个模式将扇区的各个校验和存储在重新加密的区域中，恢复过程会检测由 LUKS2 重新加密的扇区。模式要求块设备扇区写入具有“原子”性。

#### journal

最安全的模式，但也是最慢的模式。由于此模式将重新加密区域记录在二进制区域中，因此 LUKS2 将数据写入两次。

#### none

`none` 模式优先考虑性能，不提供数据保护。它只保护数据免受安全进程终止的影响，如 `SIGTERM` 信号或用户按了 `Ctrl+C` 键。任何意外的系统故障或应用程序失败都可能会导致数据损坏。

如果 LUKS2 重新加密进程意外被强行终止，LUKS2 可通过以下方法执行恢复：

#### 自动

在下一个 LUKS2 设备打开操作过程中，执行以下操作之一会触发自动恢复操作：

- 执行 `cryptsetup open` 命令。
- 使用 `systemd-cryptsetup` 命令附加设备。

## 手动

通过在 LUKS2 设备中使用 `cryptsetup repair /dev/ <device_ID>` 命令。

## 其他资源

- 您系统上的 `cryptsetup-reencrypt (8)` 和 `cryptsetup-repair (8)` 手册页

## 21.4. 使用 LUKS2 加密块设备上的现有数据

您可以使用 LUKS2 格式，在尚未加密设备上加密现有的数据。新的 LUKS 标头保存在设备的标头中。

## 先决条件

- 块设备有一个文件系统。
- 已备份了数据。



### 警告

由于硬件、内核或人为故障，您可能在加密过程中丢失数据。在开始加密数据之前，请确保您有可靠的备份。

## 步骤

1. 卸载您要加密的设备上的所有文件系统，例如：

```
# umount /dev/mapper/vg00-lv00
```

2. 为存储 LUKS 标头腾出空间。使用以下适合您场景的选项之一：

- 如果是加密逻辑卷，您可以扩展逻辑卷而无需调整文件系统的大小。例如：
- ```
# lvextend -L+32M /dev/mapper/vg00-lv00
```
- 使用分区管理工具（如 **parted**）扩展分区。
  - 缩小该设备的文件系统。您可以对 **ext2**、**ext3** 或 **ext4** 文件系统使用 **resize2fs** 工具。请注意，您无法缩小 **XFS** 文件系统。

3.

初始化加密：

```
# cryptsetup reencrypt --encrypt --init-only --reduce-device-size 32M /dev/mapper/vg00-lv00
lv00_encrypted

/dev/mapper/lv00_encrypted is now active and ready for online encryption.
```

4.

挂载该设备：

```
# mount /dev/mapper/lv00_encrypted /mnt/lv00_encrypted
```

5.

为持久映射向 **/etc/crypttab** 文件中添加一个条目：

a.

查找 **luksUUID**：

```
# cryptsetup luksUUID /dev/mapper/vg00-lv00

a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
```

b.

在您选择的文本编辑器中打开 **/etc/crypttab**，并在此文件中添加一个设备：

```
$ vi /etc/crypttab

lv00_encrypted UUID=a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325 none
```

将 **a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325** 替换为您设备的 **luksUUID**。

c.

**使用 dracut 刷新 initramfs :**

```
$ dracut -f --regenerate-all
```

6.

**在 `/etc/fstab` 文件中为永久挂载添加一个条目 :**

a.

**查找活跃的 LUKS 块设备的文件系统 UUID :**

```
$ blkid -p /dev/mapper/lv00_encrypted

/dev/mapper/lv00-encrypted: UUID="37bc2492-d8fa-4969-9d9b-bb64d3685aa9"
BLOCK_SIZE="4096" TYPE="xfs" USAGE="filesystem"
```

b.

**在您选择的文本编辑器中打开 `/etc/fstab`，并在此文件中添加一个设备，例如 :**

```
$ vi /etc/fstab

UUID=37bc2492-d8fa-4969-9d9b-bb64d3685aa9 /home auto rw,user,auto 0
```

**将 `37bc2492-d8fa-4969-9d9b-bb64d3685aa9` 替换为您文件系统的 UUID。**

7.

**恢复在线加密 :**

```
# cryptsetup reencrypt --resume-only /dev/mapper/vg00-lv00

Enter passphrase for /dev/mapper/vg00-lv00:
Auto-detected active dm device 'lv00_encrypted' for data device /dev/mapper/vg00-lv00.
Finished, time 00:31.130, 10272 MiB written, speed 330.0 MiB/s
```

## 验证

1.

**验证现有数据是否已加密 :**

```
# cryptsetup luksDump /dev/mapper/vg00-lv00

LUKS header information
Version: 2
Epoch: 4
Metadata area: 16384 [bytes]
```

```

Keyslots area: 16744448 [bytes]
UUID: a52e2cc9-a5be-47b8-a95d-6bdf4f2d9325
Label: (no label)
Subsystem: (no subsystem)
Flags: (no flags)

Data segments:
  0: crypt
  offset: 33554432 [bytes]
  length: (whole device)
  cipher: aes-xts-plain64
  [...]

```

2.

查看加密的空白块设备的状态：

```

# cryptsetup status lv00_encrypted

/dev/mapper/lv00_encrypted is active and is in use.
type:    LUKS2
cipher:  aes-xts-plain64
keysize: 512 bits
key location: keyring
device:  /dev/mapper/vg00-lv00

```

## 其他资源

- [cryptsetup \(8\), cryptsetup-reencrypt \(8\), lvextend \(8\), resize2fs \(8\), 和 parted \(8\) man page](#)

## 21.5. 使用带有分离标头的 LUKS2 在块设备上加密现有数据

您可以加密块设备上的现有数据，而无需为存储 LUKS 标头创建可用空间。标头存储在分离的位置，它也充当额外的安全层。该流程使用 LUKS2 加密格式。

### 先决条件

- 块设备有一个文件系统。
- 您的数据已备份。



**警告**

由于硬件、内核或人为故障，您可能会在加密过程中丢失数据。在开始加密数据之前，请确保您有可靠的备份。

**步骤**

1.

卸载设备上的所有文件系统，例如：

```
# umount /dev/<nvme0n1p1>
```

将 **<nvme0n1p 1>** 替换为与您要卸载的分区对应的设备标识符。

2.

初始化加密：

```
# cryptsetup reencrypt --encrypt --init-only --header </home/header> /dev/<nvme0n1p1>
<nvme_encrypted>
```

WARNING!

=====

Header file does not exist, do you want to create it?

Are you sure? (Type 'yes' in capital letters): YES

Enter passphrase for </home/header>:

Verify passphrase:

/dev/mapper/<nvme\_encrypted> is now active and ready for online encryption.

替换：

•

**</home/header>**，带有分离 LUKS 标头的文件的路径。分离的 LUKS 标头必须可以访问，以便稍后解锁加密的设备。

•

**<nvme\_encrypted>** 替换为加密后创建的设备映射器的名称。

3.

挂载该设备：

```
# mount /dev/mapper/<nvme_encrypted> /mnt/<nvme_encrypted>
```

4.

为持久映射向 `/etc/crypttab` 文件中添加一个条目：

```
# <nvme_encrypted> /dev/disk/by-id/<nvme-partition-id> none header=</home/header>
```

将 `<nvme-partition-id>` 替换为 NVMe 分区的标识符。

5.

使用 `dracut` 重新生成 `initramfs`：

```
# dracut -f --regenerate-all -v
```

6.

在 `/etc/fstab` 文件中为永久挂载添加一个条目：

a.

查找活跃的 LUKS 块设备的文件系统 UUID：

```
$ blkid -p /dev/mapper/<nvme_encrypted>

/dev/mapper/<nvme_encrypted>: UUID="37bc2492-d8fa-4969-9d9b-bb64d3685aa9"
BLOCK_SIZE="4096" TYPE="xfs" USAGE="filesystem"
```

b.

在文本编辑器中打开 `/etc/fstab`，并在此文件中添加一个设备，例如：

```
UUID=<file_system_UUID> /home auto rw,user,auto 0
```

将 `<file_system_UUID>` 替换为上一步中找到的文件系统的 UUID。

7.

恢复在线加密：

```
# cryptsetup reencrypt --resume-only --header </home/header> /dev/<nvme0n1p1>

Enter passphrase for /dev/<nvme0n1p1>:
```

```
Auto-detected active dm device '<nvme_encrypted>' for data device /dev/<nvme0n1p1>.
Finished, time 00m51s, 10 GiB written, speed 198.2 MiB/s
```

## 验证

1.

验证使用带有分离标头的 **LUKS2** 块设备上的现有数据是否已加密：

```
# cryptsetup luksDump </home/header>

LUKS header information
Version:      2
Epoch:       88
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID:        c4f5d274-f4c0-41e3-ac36-22a917ab0386
Label:        (no label)
Subsystem:    (no subsystem)
Flags:        (no flags)

Data segments:
 0: crypt
offset: 0 [bytes]
length: (whole device)
cipher: aes-xts-plain64
sector: 512 [bytes]
[...]
```

2.

查看加密的空白块设备的状态：

```
# cryptsetup status <nvme_encrypted>

/dev/mapper/<nvme_encrypted> is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/<nvme0n1p1>
```

## 其他资源

- 

您系统上的 **cryptsetup (8)** 和 **cryptsetup-reencrypt (8)** 手册页

## 21.6. 使用 LUKS2 加密空白块设备

您可以加密空白块设备，您可以使用 **LUKS2** 格式将其用于加密的存储。

## 先决条件

- 空白块设备。您可以使用 `lsblk` 等命令来查找该设备上是否没有实际数据，例如文件系统。

## 步骤

1. 将分区设置为加密的 LUKS 分区：

```
# cryptsetup luksFormat /dev/nvme0n1p1

WARNING!
=====
This will overwrite data on /dev/nvme0n1p1 irrevocably.
Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/nvme0n1p1:
Verify passphrase:
```

2. 打开加密的 LUKS 分区：

```
# cryptsetup open /dev/nvme0n1p1 nvme0n1p1_encrypted

Enter passphrase for /dev/nvme0n1p1:
```

这会解锁分区，并使用设备映射器将其映射到新设备。如果不覆盖加密的数据，这个命令会警告内核，设备是一个加密的设备，可使用 `/dev/mapper/device_mapped_name` 路径，通过 LUKS 处理。

3. 创建一个文件系统来将加密的数据写入分区，该分区必须通过设备映射名称访问：

```
# mkfs -t ext4 /dev/mapper/nvme0n1p1_encrypted
```

4. 挂载该设备：

```
# mount /dev/mapper/nvme0n1p1_encrypted mount-point
```

## 验证

1. 验证空白块设备是否已加密：

■

```
# cryptsetup luksDump /dev/nvme0n1p1

LUKS header information
Version:      2
Epoch:       3
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID:         34ce4870-ffdf-467c-9a9e-345a53ed8a25
Label:        (no label)
Subsystem:    (no subsystem)
Flags:        (no flags)

Data segments:
 0: crypt
offset: 16777216 [bytes]
length: (whole device)
cipher: aes-xts-plain64
sector: 512 [bytes]
[...]
```

2.

查看加密的空白块设备的状态：

```
# cryptsetup status nvme0n1p1_encrypted

/dev/mapper/nvme0n1p1_encrypted is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/nvme0n1p1
sector size: 512
offset: 32768 sectors
size: 20938752 sectors
mode: read/write
```

## 其他资源

•

**cryptsetup (8), cryptsetup-open (8), 和 cryptsetup-luksFormat (8) man page**

## 21.7. 在 WEB 控制台中配置 LUKS 密码短语

如果要在系统中的现有逻辑卷中添加加密，则只能通过格式化卷进行。

## 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。
- 在没有加密的情况下可用的现有逻辑卷。

## 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. 在面板中，单击 **Storage**。
3. 在 **Storage** 表中，点您要加密的存储设备的菜单按钮，然后单击 **Format**。
4. 在 **Encryption field** 中，选择加密规格 **LUKS1** 或 **LUKS2**。
5. 设置并确认您的新密码短语。
6. 可选：修改进一步加密选项。

7.  
完成格式化设置。
8.  
点 **Format**。

## 21.8. 在 WEB 控制台中更改 LUKS 密码短语

在 web 控制台中的加密磁盘或分区上更改 LUKS 密码短语。

### 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。

### 流程

1.  
登录到 RHEL 8 web 控制台。
- 详情请参阅 [登录到 web 控制台](#)。
2.  
在面板中，单击 **Storage**。
  3.  
在 **Storage** 表中，选择带有加密数据的磁盘。

4.  
在磁盘页面中，滚动到 **Keys** 部分，然后点 **edit** 按钮。
5.  
在更改密码短语对话框中：
  - a.  
输入您当前的密码短语。
  - b.  
输入您的新密码短语。
  - c.  
确认您的新密码短语。
6.  
点击 **Save**。

## 21.9. 使用命令行更改 LUKS 密码短语

使用命令行更改加密磁盘或分区上的 LUKS 密码短语。使用 `cryptsetup` 实用程序，您可以使用各种配置选项和功能控制加密过程，并在现有自动化工作流中集成。

### 先决条件

- 您有 **root** 权限或使用 **sudo** 输入管理命令的权限。

### 流程

1.  
更改 LUKS 加密设备中现有的密码短语：

```
# cryptsetup luksChangeKey /dev/<device_ID>
```

将 `<device_ID>` 替换为设备设计器，例如 `sda`。

如果您配置了多个密钥插槽，您可以指定要使用的插槽：

■



```
# cryptsetup luksChangeKey /dev/<device_ID> --key-slot <slot_number>
```

将 **<slot\_number>** 替换为您要修改的密钥插槽的数量。

2.

插入当前的密码短语和新密码短语：

```
Enter passphrase to be changed:
Enter new passphrase:
Verify passphrase:
```

3.

验证新密码短语：

```
# cryptsetup --verbose open --test-passphrase /dev/<device_ID>
```

## 验证

1.

验证新密码短语是否可以解锁设备：

```
Enter passphrase for /dev/<device_ID>:
Key slot <slot_number> unlocked.
Command successful.
```

## 21.10. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密的卷

您可以使用 **存储** 角色来通过运行 **Ansible playbook** 创建和配置使用 LUKS 加密的卷。

### 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.

将您的敏感变量存储在一个加密文件中：

a.

创建 vault：

```
$ ansible-vault create ~/vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

b.

在 `ansible-vault create` 命令打开编辑器后，以 `<key>: <value>` 格式输入敏感数据：

```
luks_password: <password>
```

c.

保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2.

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
  vars_files:
    - ~/vault.yml
  tasks:
    - name: Create and configure a volume encrypted with LUKS
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_volumes:
          - name: barefs
            type: disk
            disks:
              - sdb
            fs_type: xfs
            fs_label: <label>
            mount_point: /mnt/data
            encryption: true
            encryption_password: "{{ luks_password }}"
```

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。

3.

验证 **playbook** 语法 :

```
$ ansible-playbook --ask-vault-pass --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。

4.

运行 **playbook** :

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

## 验证

1.

查找 LUKS 加密卷的 **luksUUID** 值 :

```
# ansible managed-node-01.example.com -m command -a 'cryptsetup luksUUID /dev/sdb'

4e4e7970-1822-470e-b55a-e91efe5d0f5c
```

2.

查看卷的加密状态 :

```
# ansible managed-node-01.example.com -m command -a 'cryptsetup status luks-4e4e7970-1822-470e-b55a-e91efe5d0f5c'

/dev/mapper/luks-4e4e7970-1822-470e-b55a-e91efe5d0f5c is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdb
...
```

3.

验证创建的 LUKS 加密的卷 :

```
# ansible managed-node-01.example.com -m command -a 'cryptsetup luksDump /dev/sdb'

LUKS header information
Version:    2
Epoch:     3
Metadata area: 16384 [bytes]
```

```
Keyslots area: 16744448 [bytes]
UUID:         4e4e7970-1822-470e-b55a-e91efe5d0f5c
Label:        (no label)
Subsystem:    (no subsystem)
Flags:        (no flags)
```

**Data segments:**

0: crypt

offset: 16777216 [bytes]

length: (whole device)

cipher: aes-xts-plain64

sector: 512 [bytes]

...

## 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](#)
- [使用 LUKS 加密块设备](#)
- [Ansible vault](#)

## 第 22 章 管理磁带设备

磁带设备是保存数据并按顺序访问的磁带。使用磁带驱动器将数据写入此磁带设备。不需要创建文件系统来存储磁带设备中的数据。可以使用各种接口（如 SCSI、FC、USB、SATA 和其他接口）连接到主机计算机。

### 22.1. 磁带设备的类型

以下是不同类型的磁带设备列表：

- `/dev/st0` 是一个回卷磁带设备。
- `/dev/nst0` 是一个非回卷磁带设备。使用非缓解设备进行日常备份。

使用磁带设备有几个优点。它们成本效益高且稳定。磁带设备也对数据崩溃具有弹性，并适合长久保存数据。

### 22.2. 安装磁带驱动器管理工具

为磁带驱动器操作安装 `mt-st` 软件包。使用 `mt` 工具控制 magnetic 磁带驱动器操作，以及 SCSI 磁带驱动程序的 `st` 工具。

#### 流程

- 安装 `mt-st` 软件包：

```
# yum install mt-st
```

#### 其他资源

- 您系统上的 `mt` (1) 和 `st` (4) man page

### 22.3. 磁带命令

以下是常见的 `mt` 命令：

表 22.1. `mt` 命令

| 命令                                  | 描述                                                                 |
|-------------------------------------|--------------------------------------------------------------------|
| <code>mt -f /dev/st0 status</code>  | 显示磁带设备的状态。                                                         |
| <code>mt -f /dev/st0 erase</code>   | 擦除整个磁带。                                                            |
| <code>mt -f /dev/nst0 rewind</code> | 插入磁带设备。                                                            |
| <code>mt -f /dev/nst0 fsf n</code>  | 将磁带头切换到转发记录。在这里， <i>n</i> 是一个可选的文件数。如果指定了文件计数，磁带头将跳过 <i>n</i> 个记录。 |
| <code>mt -f /dev/nst0 bsfm n</code> | 将磁头切换到之前的记录。                                                       |
| <code>mt -f /dev/nst0 eod</code>    | 将磁带头切换到数据的末尾。                                                      |

22.4. 写入如回卷磁带设备

在每次操作后，回卷磁带设备会进行回卷。要备份数据，您可以使用 `tar` 命令。默认情况下，在磁带设备中，块大小为 10KB(`bs=10k`)。您可以使用 `export TAPE=/dev/st0` 属性设置 `TAPE` 环境变量。使用 `-f` 设备选项指定磁带设备文件。当您使用多个磁带设备时，这个选项很有用。

先决条件

1. 已安装 `mt-st` 软件包。如需更多信息，请参阅[安装磁带驱动器管理工具](#)。
2. 加载磁带驱动器：

```
# mt -f /dev/st0 load
```

流程

1. 检查磁带头：

```
# mt -f /dev/st0 status
```

```
SCSI 2 tape drive:
File number=-1, block number=-1, partition=0.
Tape block size 0 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (50000):
DR_OPEN IM_REP_EN
```

在这里：

- 当前文件号为 -1。
- 块编号 定义磁带头。默认情况下，它被设置为 -1。
- 块大小 0 表示磁带设备没有固定的块大小。
- 软错误数表示在执行 **mt status** 命令后遇到的错误数量。
- **General status** 位解释了磁带设备的状态。
- **DR\_OPEN** 表示公开，磁带设备为空。**IM\_REP\_EN** 是即时报告模式。

2.

如果磁带设备不是空的，覆盖它：

```
# tar -czf /dev/st0 _/source/directory
```

该命令使用 **/source/directory** 目录的内容覆盖 **tape** 设备中的数据

3.

将 **/source/directory** 目录备份到磁带设备中：

```
# tar -czf /dev/st0 _/source/directory
tar: Removing leading '/' from member names
/source/directory
/source/directory/man_db.conf
```

```
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

4.

查看磁带设备的状态：

```
# mt -f /dev/st0 status
```

## 验证

•

查看磁带设备上的所有文件列表：

```
# tar -tzf /dev/st0
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

## 其他资源

•

**mt (1), st (4), 和 tar (1) man page**

•

[磁带驱动器介质被识别为写保护](#)（红帽知识库）

•

[如何检查系统中是否检测到磁带驱动器](#)（红帽知识库）

## 22.5. 写入非回卷解磁带设备

在完成特定命令后，非回卷磁带设备会将磁带保持在当前位置。例如，备份后，您可以将更多数据附加到非回卷的磁带设备中。您还可以使用它来避免任何意外的回卷。

## 先决条件

1.

已安装 **mt-st** 软件包。如需更多信息，请参阅[安装磁带驱动器管理工具](#)。



2.

**加载磁带驱动器：**

```
# mt -f /dev/nst0 load
```

**流程**

1.

**检查非回卷磁带设备 /dev/nst0 的磁带头：**

```
# mt -f /dev/nst0 status
```

2.

**指定位于头或磁带末尾的指针：**

```
# mt -f /dev/nst0 rewind
```

3.

**附加磁带设备中的数据：**

```
# mt -f /dev/nst0 eod
# tar -czf /dev/nst0 /source/directory/
```

4.

**将 /source/directory/ 备份到磁带设备中：**

```
# tar -czf /dev/nst0 /source/directory/
tar: Removing leading '/' from member names
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

5.

**查看磁带设备的状态：**

```
# mt -f /dev/nst0 status
```

**验证**

•

**查看磁带设备上的所有文件列表：**

```
# tar -tzf /dev/nst0
```

```
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

## 其他资源

- **mt (1), st (4), 和 tar (1) man page**
- [磁带驱动器介质被识别为写保护](#)（红帽知识库）
- [如何检查系统中是否检测到磁带驱动器](#)（红帽知识库）

## 22.6. 在磁带设备中切换磁带头

您可以使用 **eod** 选项切换磁带设备中的磁带头。

## 先决条件

1. 已安装 **mt-st** 软件包。如需更多信息，请参阅[安装磁带驱动器管理工具](#)。
2. 数据被写入磁带设备。如需更多信息，请参阅[写入回卷磁带设备](#)，或[写入非回卷磁带设备](#)。

## 流程

- 查看磁带指针的当前位置：

```
# mt -f /dev/nst0 tell
```

- 在将数据附加到磁带设备时切换磁带头：

```
# mt -f /dev/nst0 eod
```

- 使用之前的记录：

```
# mt -f /dev/nst0 bsfm 1
```

- 转至正向记录：

```
# mt -f /dev/nst0 fsf 1
```

## 其他资源

- 系统中的 `mt (1)` 手册页

## 22.7. 从磁带设备中恢复数据

您可以使用 `tar` 命令从磁带设备中恢复数据。

## 先决条件

1. 已安装 `mt-st` 软件包。如需更多信息，请参阅[安装磁带驱动器管理工具](#)。
2. 数据被写入磁带设备。如需更多信息，请参阅[写入回卷磁带设备](#)，或[写入非回卷磁带设备](#)。

## 流程

- 对于回卷磁带设备 `/dev/st0`：
  - 恢复 `/source/directory/`：

```
# tar -xzf /dev/st0 /source/directory/
```

- 对于非回卷磁带设备 `/dev/nst0`：

- **重新构建磁带设备：**

```
# mt -f /dev/nst0 rewind
```

- **恢复 etc 目录：**

```
# tar -xzf /dev/nst0 /source/directory/
```

## 其他资源

- **系统中的 `mt` (1) 和 `tar` (1) 手册页**

## 22.8. 从磁带设备中删除数据

您可以使用 `erase` 选项从磁带设备中删除数据。

## 先决条件

1. **已安装 `mt-st` 软件包。**如需更多信息，请参阅[安装磁带驱动器管理工具](#)。
2. **数据被写入磁带设备。**如需更多信息，请参阅[写入回卷磁带设备](#)，或[写入非回卷磁带设备](#)。

## 流程

1. **从磁带设备中删除数据：**

```
# mt -f /dev/st0 erase
```

2. **卸载磁带设备：**

```
# mt -f /dev/st0 offline
```

## 其他资源

- [系统中的 mt \(1\) 手册页](#)

## 第 23 章 删除存储设备

您可以从正在运行的系统中安全地删除存储设备，这有助于防止系统内存过载和数据丢失。不要删除系统中的存储设备：

- 空闲内存低于内存总量的 5%，每 100 个超过 10 个样本。
- 交换是活跃的（在 `vmstat` 命令的输出中非零的 `si` 和 `so` 列）。

### 先决条件

- 在删除存储设备前，请确保在 I/O 刷新过程中由于系统内存负载增加而有足够的可用内存。使用以下命令查看系统的当前内存负载和可用内存：

```
# vmstat 1 100
# free
```

### 23.1. 安全删除存储设备

从正在运行的系统中安全地删除存储设备需要顶级的方法。从顶层（通常是应用程序或文件系统）开始，并在底层（即物理设备）上工作。

您可以通过多种方式使用存储设备，它们可以在物理设备之上有不同的虚拟配置。例如：您可以将设备的多个实例分组到多路径设备中，使其成为 **RAID** 的一部分，或者您可以将其成为 **LVM** 组的一部分。此外，设备可以通过文件系统访问，或者可以直接访问设备，如“原始”设备。

使用 **top-to-bottom** 方法时，您必须确保：

- 要删除的设备没有被使用
- 对该设备的所有待处理的 I/O 都会被清除

- 操作系统无法引用存储设备

## 23.2. 删除块设备和相关元数据

要从正在运行的系统中安全地删除块设备，以帮助防止系统内存过载和数据丢失，您需要首先从它们中删除元数据。从文件系统开始，处理堆栈中的每个层，然后继续到磁盘。这些操作可防止将您的系统处于不一致的状态。

根据您要删除的设备类型，使用可能不同的特定命令：

- `lvremove`、`vgremove` 和 `pvremove` 特定于 LVM。
- 对于软件 RAID，请运行 `mdadm` 以删除该阵列。如需更多信息，[请参阅管理 RAID](#)。
- 对于使用 LUKS 加密的块设备，有特定的额外步骤。以下流程不适用于使用 LUKS 加密的块设备。如需更多信息，[请参阅使用 LUKS 加密块设备](#)。



### 警告

重新扫描 SCSI 总线或执行更改操作系统状态的其他操作，而无需遵循这个流程，因为 I/O 超时、设备被意外删除或数据丢失。

### 先决条件

- 您有一个现有的块设备堆栈，其中包含文件系统、逻辑卷和卷组。
- 您确定没有其他应用程序或服务正在使用您要删除的设备。

- 您从您要删除的设备备份了数据。
- 可选：如果要删除多路径设备，且您无法访问其路径设备，请运行以下命令来禁用多路径设备的队列：

```
# multipathd disablequeueing map multipath-device
```

这可让设备的 I/O 失败，允许使用该设备的应用程序关闭。



#### 注意

一次删除其元数据层的设备，确保不会在磁盘上保留过时的签名。

#### 流程

1.

卸载文件系统：

```
# umount /mnt/mount-point
```

2.

删除文件系统：

```
# wipefs -a /dev/vg0/myvol
```

如果您已在 `/etc/fstab` 文件中添加了一个条目，以便在文件系统和挂载点之间建立持久关联，请在此时编辑 `/etc/fstab` 以删除该条目。

根据您要删除的设备类型，继续执行以下步骤：

3.

删除包含文件系统的逻辑卷(LV)：

```
# lvremove vg0/myvol
```



4.

如果卷组中没有剩余的其他逻辑卷(VG)，您可以安全地删除包含该设备的 VG：

```
# vgremove vg0
```

5.

从 PV 设备中删除物理卷(PV)元数据：

```
# pvremove /dev/sdc1
```

```
# wipefs -a /dev/sdc1
```

6.

删除包含 PV 的分区：

```
# parted /dev/sdc rm 1
```

7.

如果要完全擦除该设备，请删除分区表：

```
# wipefs -a /dev/sdc
```

8.

只有在您要物理删除该设备时才执行以下步骤：

- 

如果您要删除多路径设备，请执行以下命令：

- a.

查看该设备的所有路径：

```
# multipath -l
```

稍后需要这个命令的输出。

- b.

清除 I/O 并删除多路径设备：

```
# multipath -f multipath-device
```

- 

如果该设备没有配置为多路径设备，或者设备配置为多路径设备，并且您之前将 I/O 传

递给单个路径，请将任何未完成的 I/O 刷新到所有使用的设备路径：

```
# blockdev --flushbufs device
```

对于直接访问的设备非常重要，`umount` 或 `vgreduce` 命令不会清除 I/O。

- 

如果您要删除 SCSI 设备，请执行以下命令：

- a.

删除对基于路径的设备名称的任何引用，如 `/dev/sd`、`/dev/disk/by-path` 或 `major:minor number`（在系统上的应用程序、脚本或工具中）。这样可保证以后添加的不同设备不会为当前的设备错误。

- b.

从 SCSI 子系统中删除该设备的每个路径：

```
# echo 1 > /sys/block/device-name/device/delete
```

此处，如果设备之前用作多路径设备，则 `device-name` 从 `multipath -l` 命令的输出中检索。

- 9.

从正在运行的系统中删除物理设备。请注意，当您删除此设备时，I/O 到其它设备不会停止。

## 验证

- 

验证您要删除的设备是否没有显示 `lsblk` 命令的输出。以下是一个输出示例：

```
# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0    0  5G  0 disk
sr0   11:0    1 1024M 0 rom
vda   252:0    0  10G  0 disk
|-vda1 252:1    0   1M  0 part
|-vda2 252:2    0 100M  0 part /boot/efi
`-vda3 252:3    0  9.9G  0 part /
```

## 其他资源

-

**multipath (8), pvremove (8), vgremove (8), lvremove (8), wipefs (8), parted (8), blockdev (8), 和 umount (8) man page**

## 第 24 章 设置 STRATIS 文件系统

**Stratis** 是 Red Hat Enterprise Linux 的本地存储管理解决方案。它着重介绍简单、易用性，并可让您访问高级存储功能。

**Stratis** 作为服务运行，来管理物理存储设备池，简化本地存储管理，易于使用，同时帮助您设置和管理复杂的存储配置。



### 重要

**Stratis** 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

**Stratis** 可帮助您进行以下操作：

- 存储的初始配置
- 稍后进行修改
- 使用高级存储功能

**Stratis** 的核心概念是一个存储池。这个池是从一个或多个本地磁盘或分区创建的，文件系统是从池中创建的。该池启用了如下功能：

- 文件系统快照
- 精简置备

- **Caching**
- **加密**

## 24.1. STRATIS 文件系统的组件

在外部，**Stratis** 在命令行和 **API** 中显示以下文件系统组件：

### **blockdev**

块设备，如磁盘或者磁盘分区。

### **pool**

由一个或多个块设备组成。

池有固定大小，与块设备的大小相等。

池包含大多数 **Stratis** 层，如使用 **dm-cache** 目标的非易失性数据缓存。

**Stratis** 为每个池创建一个 **/dev/stratis/my-pool/** 目录。这个目录包含了到代表池里 **Stratis** 文件系统的设备的链接。

### **filesystem**

每个池可以包含零个或多个文件系统。包含文件系统的池可以存储任意数量的文件。

文件系统会被精简置备，且没有固定的总大小。文件系统的实际大小随着保存着文件系统中的数据而增长。如果数据的大小接近文件系统的虚拟大小，**Stratis** 将自动增大精简卷和文件系统。

文件系统使用 **XFS** 文件系统格式化。



## 重要

**Stratis 跟踪它不知道 XFS 的文件系统的信息，并且使用 XFS 所做的更改不会在 Stratis 中自动创建更新。用户不得重新格式化或重新配置由 Stratis 管理的 XFS 文件系统。**

**Stratis 在 `/dev/stratis/my-pool/my-fs` 路径创建到文件系统的链接。**

Stratis 使用很多设备映射器设备，它们出现在 `dmsetup` 列表中和 `/proc/partitions` 文件中。类似地，`lsblk` 命令输出反映了 Stratis 的内部工作方式和层。

## 24.2. 与 STRATIS 兼容的块设备

可与 Stratis 一起使用的存储设备。

### 支持的设备

Stratis 池已被测试以可用于这些块设备：

- **LUKS**
- **LVM 逻辑卷**
- **MD RAID**
- **DM Multipath**
- **iSCSI**
- **HDD 和 SSD**

- **NVMe 设备**

### 不支持的设备

因为 **Stratis** 包含精简置备层，因此红帽不推荐将 **Stratis** 池放在已经精简置备的块设备中。

## 24.3. 安装 STRATIS

安装 **Stratis** 所需的软件包。

### 流程

1. 安装提供 **Stratis** 服务和命令行工具的软件包：

```
# dnf install stratisd stratis-cli
```

2. 启动 **stratisd** 服务并启用它在引导时启动：

```
# systemctl enable --now stratisd
```

### 验证

- 验证 **stratisd** 服务是否已启用并正在运行：

```
# systemctl status stratisd
stratisd.service - Stratis daemon
Loaded: loaded (/usr/lib/systemd/system/stratisd.service; enabled; preset:>
Active: active (running) since Tue 2025-03-25 14:04:42 CET; 30min ago
Docs: man:stratisd(8)
Main PID: 24141 (stratisd)
Tasks: 22 (limit: 99365)
Memory: 10.4M
CPU: 1.436s
CGroup: /system.slice/stratisd.service
└─24141 /usr/libexec/stratisd --log-level debug
```

## 24.4. 创建未加密的 STRATIS 池

您可以从一个或多个块设备创建未加密的 **Stratis** 池。

#### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 **1 GB**。
- 在 **IBM Z** 构架中，必须对 **/dev/dasd\*** 块设备进行分区。使用分区设备来创建 **Stratis** 池。

有关分区 **DASD** 设备的详情，请参考在 [IBM Z 中配置 Linux 实例](#)。



#### 注意

您只能在创建过程中加密 **Stratis** 池，而不稍后加密。

#### 流程

1. 删除您要在 **Stratis** 池中使用的每个块设备上存在的任何文件系统、分区表或 **RAID** 签名：

```
# wipefs --all block-device
```

**block-device** 值是块设备的路径，例如 **/dev/sdb**。

2. 在所选的块设备上创建新的未加密的 **Stratis** 池：

```
# stratis pool create my-pool block-device
```

**block-device** 值是到空或者擦除的块设备的路径。

您还可以使用以下命令在一行中指定多个块设备：

-



```
# stratis pool create my-pool block-device-1 block-device-2
```

## 验证

- 确认创建了新的 **Stratis** 池：

```
# stratis pool list
```

## 24.5. 使用 WEB 控制台创建未加密的 STRATIS 池

您可以使用 Web 控制台从一个或多个块设备创建一个未加密的 **Stratis** 池。

### 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](#)。

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 **1 GB**。



### 注意

您不能在其创建后加密一个未加密的 **Stratis** 池。

## 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，点菜单按钮并选择 **Create Stratis pool**。

4.

在 **Name** 字段中输入 **Stratis** 池的名称。

5.

选择您要从中创建 **Stratis** 池的 **Block devices**。

6.

可选：如果要为池中创建的每个文件系统指定最大大小，请选择 **Manage filesystem size**。

7.

点 **Create**。

#### 验证

- 

进到 **Storage** 部分，并确认您可以在 **Devices** 表中看到新的 **Stratis** 池。

## 24.6. 使用内核密钥环中的密钥创建加密的 STRATIS 池

要保护数据，您可以使用内核密钥环从一个或多个块设备创建加密的 **Stratis** 池。

当您以这种方式创建加密的 **Stratis** 池时，内核密钥环将用作主加密机制。后续系统重启此内核密钥环后，用来解锁加密的 **Stratis** 池。

当从一个或多个块设备创建加密的 **Stratis** 池时，请注意以下几点：

- 每个块设备都使用 **cryptsetup** 库进行加密，并实施 **LUKS2** 格式。
- 每个 **Stratis** 池都可以有一个唯一的密钥，或者与其他池共享相同的密钥。这些密钥保存在内核密钥环中。
- 组成 **Stratis** 池的块设备必须全部加密或者全部未加密。不可能同时在同一个 **Stratis** 池中加密和未加密块设备。
- 添加到加密 **Stratis** 池的数据缓存中的块设备会自动加密。

### 先决条件

- **Stratis v2.1.0** 或更高版本已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 **1 GB**。
- 在 **IBM Z** 构架中，必须对 **/dev/dasd\*** 块设备进行分区。使用 **Stratis** 池中的分区。  
  
有关分区 **DASD** 设备的详情，请参考在 [IBM Z 中配置 Linux 实例](#)。

### 流程

1. 删除您要在 **Stratis** 池中使用的每个块设备上存在的任何文件系统、分区表或 **RAID** 签名：

```
# wipefs --all block-device
```

**block-device** 值是块设备的路径，例如 **/dev/sdb**。

2. 如果您还没有设置密钥，请运行以下命令，并按照提示创建用于加密的密钥集：

```
# stratis key set --capture-key key-description
```

***key-description*** 是对在内核密钥环中创建的密钥的引用。系统将提示您在命令行中输入键值。您还可以将 **key** 值放在文件中，并使用 **--keyfile-path** 选项而不是 **-capture-key** 选项。

3.

创建加密的 **Stratis** 池并指定用于加密的密钥描述：

```
# stratis pool create --key-desc key-description my-pool block-device
```

***key-description***

引用您在上一步中创建的内核密钥环中存在的密钥。

***my-pool***

指定新的 **Stratis** 池的名称。

***block-device***

指定到空或者有线块设备的路径。

您还可以使用以下命令在一行中指定多个块设备：

```
# stratis pool create --key-desc key-description my-pool block-device-1 block-device-2
```

验证

- 

确认创建了新的 **Stratis** 池：

```
# stratis pool list
```

## 24.7. 使用 WEB 控制台创建加密的 STRATIS 池

要保护您的数据，您可以使用 **Web** 控制台从一个或多个块设备创建一个加密的 **Stratis** 池。

当从一个或多个块设备创建加密的 **Stratis** 池时，请注意以下几点：

- 每个块设备都使用 **cryptsetup** 库进行加密，并实施 **LUKS2** 格式。
- 每个 **Stratis** 池都可以有一个唯一的密钥，或者与其他池共享相同的密钥。这些密钥保存在内核密钥环中。
- 组成 **Stratis** 池的块设备必须全部加密或者全部未加密。不可能同时在同一个 **Stratis** 池中加密和未加密块设备。
- 添加到加密 **Stratis** 池的数据层中的块设备会自动加密。

#### 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](#)。
- **Stratis v2.1.0** 或更高版本已安装，**stratisd** 服务正在运行。
  - 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 **1 GB**。

#### 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，点菜单按钮并选择 **Create Stratis pool**。
4.  
在 **Name** 字段中输入 **Stratis** 池的名称。
5.  
选择您要从中创建 **Stratis** 池的 **Block devices**。
6.  
选择加密类型，您可以使用密码短语、**Tang keyserver** 或两者：

- **passphrase:**
  - i.  
输入密码短语。
  - ii.  
确认密码短语。
- **Tang keyserver :**
  - i.  
输入 **keyserver** 地址。如需更多信息，请参阅 [部署 SELinux 为 enforcing 模式的 Tang 服务器](#)。

7.  
可选：如果要为池中创建的每个文件系统指定最大大小，请选择 **Manage filesystem size**。
8.  
点 **Create**。

#### 验证

- 进到 **Storage** 部分，并确认您可以在 **Devices** 表中看到新的 **Stratis** 池。

## 24.8. 使用 WEB 控制台重命名 STRATIS 池

您可以使用 Web 控制台重命名现有的 Stratis 池。

### 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](#)。

- Stratis 已安装，stratisd 服务正在运行。

默认情况下，web 控制台会检测并安装 Stratis。但是，要手动安装 Stratis，请参阅[安装 Stratis](#)。

- 一个 Stratis 池已创建。

### 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，点您要重命名的 Stratis 池。

4. 在 **Stratis pool** 页面中，点 **Name** 字段旁边的 **edit**。
5. 在 **Rename Stratis pool** 对话框中输入新名称。
6. 点击 **Rename**。

## 24.9. 在 STRATIS 文件系统中设置过度置备模式

默认情况下，每个 **Stratis** 池都过度置备，这意味着逻辑文件系统大小可能会超过物理分配的空间。**Stratis** 监控文件系统使用情况，并在需要时自动使用可用空间增加分配量。但是，如果已经分配了所有可用空间，并且池已满，则不会为文件系统分配额外的空间。



### 注意

如果文件系统耗尽空间，用户可能会丢失数据。对于数据丢失的风险超过过度置备的好处应用程序，可以禁用此功能。

**Stratis** 持续监控池使用情况，并使用 **D-Bus API** 报告值。存储管理员必须监控这些值，并根据需要将设备添加到池中，以防止它达到容量。

### 先决条件

- 已安装 **Stratis**。如需更多信息，请参阅 [安装 Stratis](#)。

### 流程

要正确设置池，您可以有两个可能：

1. 从一个或多个块设备创建池：

```
# stratis pool create pool-name /dev/sdb
```



2.

在现有池中设置过度置备模式：

```
# stratis pool overprovision pool-name <yes|no>
```

•

如果设置为 "yes", 则启用过度置备到池。这意味着池支持的 **Stratis** 文件系统的逻辑大小总和可能会超过可用空间量。如果池被过度置备, 且所有文件系统的逻辑大小总和都超过池中可用的空间, 则系统无法关闭过度置备, 并返回错误。

## 验证

1.

查看 **Stratis** 池的完整列表：

```
# stratis pool list
```

| Name             | Total Physical                  | Properties  | UUID                                     | Alerts |
|------------------|---------------------------------|-------------|------------------------------------------|--------|
| <i>pool-name</i> | 1.42 TiB / 23.96 MiB / 1.42 TiB | ~Ca,~Cr,~Op | cb7cb4d8-9322-4ac4-a6fd-<br>eb7ae9e1e540 |        |

2.

检查 **stratis pool list** 输出中是否有池 **overprovisioning** 模式标记。" ~ " 是 "NOT" 的数学符号, 因此 ~Op 表示不进行过度配置。

3.

可选：在特定池中检查过度置备：

```
# stratis pool overprovision pool-name yes
```

```
# stratis pool list
```

| Name             | Total Physical                  | Properties  | UUID                                     | Alerts |
|------------------|---------------------------------|-------------|------------------------------------------|--------|
| <i>pool-name</i> | 1.42 TiB / 23.96 MiB / 1.42 TiB | ~Ca,~Cr,~Op | cb7cb4d8-9322-4ac4-a6fd-<br>eb7ae9e1e540 |        |

## 24.10. 将 STRATIS 池绑定到 NBDE

将加密的 **Stratis** 池绑定到网络绑定磁盘加密(NBDE)需要 **Tang** 服务器。当包含 **Stratis** 池的系统重启时, 它与 **Tang** 服务器进行连接, 以自动解锁加密的池, 而无需提供内核密钥环描述。



### 注意

将 Stratis 池绑定到补充的 Clevis 加密机制不会删除主内核密钥环加密。

### 先决条件

- Stratis v2.3.0 或更高版本已安装，`stratisd` 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 已创建一个加密的 Stratis 池，并且您有用于加密的密钥的密钥描述。如需更多信息，请参阅 [使用内核密钥环 中的 使用密钥创建加密的 Stratis 池](#)。
- 您可以连接到 Tang 服务器。如需更多信息，请参阅 [部署 SELinux 为 enforcing 模式的 Tang 服务器](#)。

### 流程

- 将加密的 Stratis 池绑定到 NBDE :

```
# stratis pool bind nbde --trust-url my-pool tang-server
```

*my-pool*

指定加密的 Stratis 池的名称。

*tang-server*

指定 Tang 服务器的 IP 地址或 URL。

### 其他资源

- [使用基于策略的解密配置加密卷的自动解锁](#)

## 24.11. 将 STRATIS 池绑定到 TPM

当您将加密的 Stratis 池绑定到受信任的平台模块(TPM) 2.0 时，包含池的系统会重启，并且池会自动解锁，而无需提供内核 keyring 描述。

## 先决条件

- **Stratis v2.3.0 或更高版本已安装，stratisd 服务正在运行。**如需更多信息，请参阅 [安装 Stratis](#)。
- **已创建一个加密的 Stratis 池，并且您有用于加密的密钥的密钥描述。**如需更多信息，请参阅 [使用内核密钥环 中的 使用密钥创建加密的 Stratis 池](#)。
- **您的系统支持 TPM 2.0。**

## 流程

- **将加密的 Stratis 池绑定到 TPM:**

```
# stratis pool bind tpm my-pool
```

*my-pool*

指定加密的 Stratis 池的名称。

*key-description*

引用内核密钥环中存在的密钥，该密钥是在您创建加密的 Stratis 池时生成的。

## 24.12. 使用内核密钥环解加密的 STRATIS 池

系统重启后，您的加密 Stratis 池或组成它的块设备可能不可见。您可以使用用来加密池的内核密钥环来解锁池。

## 先决条件

- **Stratis v2.1.0 已安装，stratisd 服务正在运行。**如需更多信息，请参阅 [安装 Stratis](#)。
- **已创建一个加密的 Stratis 池。**如需更多信息，请参阅 [使用内核密钥环 中的 使用密钥创建加密的 Stratis 池](#)。

## 流程

1. 使用之前使用的相同密钥描述重新创建密钥集：

```
# stratis key set --capture-key key-description
```

**key-description** 引用内核密钥环中存在的密钥，该密钥是在您创建加密的 **Stratis** 池时生成的。

2. 验证 **Stratis** 池是可见的：

```
# stratis pool list
```

## 24.13. 解除 STRATIS 池与补充加密的绑定

当您解除加密的 **Stratis** 池与支持的附加加密机制的绑定时，主内核密钥环加密将保持不变。对于一开始就使用 **Clevis** 加密创建的池，这并不适用。

### 先决条件

- **Stratis v2.3.0** 或更高版本已安装在您的系统上。如需更多信息，请参阅 [安装 Stratis](#)。
- 已创建一个加密的 **Stratis** 池。如需更多信息，请参阅 [使用内核密钥环 中的 使用密钥创建加密的 Stratis 池](#)。
- 加密的 **Stratis** 池绑定到受支持的补充加密机制。

## 流程

- 解除加密的 **Stratis** 池与补充加密机制的绑定：

```
# stratis pool unbind clevis my-pool
```

*my-pool* 指定您要解绑的 Stratis 池的名称。

#### 其他资源

- [将加密的 Stratis 池绑定到 NBDE](#)
- [将加密的 Stratis 池绑定到 TPM](#)

### 24.14. 启动和停止 STRATIS 池

您可以启动和停止 Stratis 池。这可让您选择忽略或关闭用于构建池的所有对象，如文件系统、缓存设备、精简池和加密设备。请注意，如果池主动使用任何设备或文件系统，则可能会发出警告且无法停止。

**stopped** 状态记录在池的元数据中。这些池不会在以下引导上启动，直到池收到 **start** 命令。

#### 先决条件

- Stratis 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 创建未加密的或加密的 Stratis 池。如需更多信息，请参阅 [创建未加密的 Stratis 池](#)，或使用 [内核密钥环中的密钥创建加密的 Stratis 池](#)。

#### 流程

- 使用以下命令停止 Stratis 池。这会关闭存储堆栈，但保留所有元数据不变：
 

```
# stratis pool stop --name pool-name
```
- 使用以下命令启动 Stratis 池。**--unlock-method** 选项指定池被加密的解锁方法：
 

```
# stratis pool start --unlock-method <keyring|clevis> --name pool-name
```

**注意**

您可以使用池名称或池 **UUID** 启动池。

**验证**

- 使用以下命令列出系统中的所有活跃池：
- ```
# stratis pool list
```
- 使用以下命令列出所有已停止的池：
- ```
# stratis pool list --stopped
```
- 使用以下命令查看已停止池的详细信息。如果指定了 **UUID**，该命令会打印与 **UUID** 对应的池的详细信息：

```
# stratis pool list --stopped --uuid UUID
```

**24.15. 创建 STRATIS 文件系统**

在现有 **Stratis** 池上创建 **Stratis** 文件系统。

**先决条件**

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 一个 **Stratis** 池已创建。如需更多信息，请参阅 [创建未加密的 Stratis 池](#) 或使用内核密钥环中的密钥。

**流程**

1. 在池中创建 **Stratis** 文件系统：

```
# stratis filesystem create --size number-and-unit my-pool my-fs
```

***number-and-unit***

指定文件系统的大小。规格格式必须遵循标准大小规格格式进行输入，即 B、KiB、MiB、GiB、TiB 或 PiB。

***my-pool***

指定 Stratis 池的名称。

***my-fs***

为文件系统指定一个任意名称。

例如：

**例 24.1. 创建 Stratis 文件系统**

```
# stratis filesystem create --size 10GiB pool1 filesystem1
```

**验证**

- 列出池中的文件系统，以检查是否已创建 Stratis 文件系统：

```
# stratis fs list my-pool
```

**其他资源**

- [挂载 Stratis 文件系统](#)

**24.16. 使用 WEB 控制台在 STRATIS 池中创建文件系统**

您可以使用 Web 控制台在现有 Stratis 池中创建一个文件系统。

**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](#)。
- **stratisd** 服务在运行。
- 一个 **Stratis** 池已创建。

## 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. 点击您要在其上创建文件系统的 **Stratis** 池。
4. 在 **Stratis pool** 页面中，滚动到 **Stratis filesystems** 部分，然后单击 **Create new filesystem**。
5. 输入文件系统的名称。
6. 输入文件系统的挂载点。
7. 选择 挂载选项。



8. 在 **At boot** 下拉菜单中选择您要何时挂载文件系统。

9. 创建文件系统：

- 如果要创建并挂载文件系统，点 **Create and mount**。
- 如果您只想创建文件系统，请单击 **Create only**。

#### 验证

- 新文件系统在 **Stratis filesystems** 选项卡下的 **Stratis pool** 页面中可见。

### 24.17. 挂载 STRATIS 文件系统

挂载现有的 **Stratis** 文件系统以访问其内容。

#### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 已创建一个 **Stratis** 文件系统。如需更多信息，请参阅 [创建 Stratis 文件系统](#)。

#### 流程

- 要挂载文件系统，请使用 **Stratis** 在 **/dev/stratis/** 目录中维护的条目：

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

现在该文件系统被挂载到 **mount-point** 目录中并可使用。



### 注意

在停止池之前，卸载属于池的所有文件系统。如果仍然挂载了任何文件系统，则池不会停止。

## 24.18. 使用 SYSTEMD 服务在 /ETC/FSTAB 中设置非 ROOT STRATIS 文件系统

您可以使用 **systemd** 服务管理 **/etc/fstab** 中的非 **root** 文件系统。

### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 已创建一个 **Stratis** 文件系统。如需更多信息，请参阅 [创建 Stratis 文件系统](#)。

### 流程

- 以 **root** 用户身份，编辑 **/etc/fstab** 文件并添加一行来设置非 **root** 文件系统：

```
/dev/stratis/my-pool/my-fs mount-point xfs defaults,x-systemd.requires=stratis-fstab-
setup@pool-uuid.service,x-systemd.after=stratis-fstab-setup@pool-uuid.service dump-value
fsck_value
```



### 重要

从加密的 **Stratis** 池永久挂载 **Stratis** 文件系统可能会导致引导过程停止，直到提供密码为止。如果池使用任何无人值守机制进行加密，例如 **NBDE** 或 **TPM2**，则 **Stratis** 池将自动解锁。如果没有，用户需要在控制台中输入密码。

### 其他资源

- [永久挂载文件系统](#)

## 第 25 章 使用附加块设备扩展 STRATIS 池

您可以在 **Stratis** 池中添加附加块设备以便为 **Stratis** 文件系统提供更多存储容量。您可以手动或使用 **Web 控制台** 进行此操作。



### 重要

**Stratis** 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议 (SLA) 支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

### 25.1. 在 STRATIS 池中添加块设备

您可以在 **Stratis** 池中添加一个或多个块设备。

#### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 1 GB。

#### 流程

- 要在池中添加一个或多个块设备，请使用：

```
# stratis pool add-data my-pool device-1 device-2 device-n
```

#### 其他资源

- [系统中 Stratis \(8\) 手册页](#)

### 25.2. 使用 WEB 控制台在 STRATIS 池中添加块设备

您可以使用 **Web 控制台** 向现有 **Stratis** 池中添加块设备。您还可以将缓存添加为块设备。

#### 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](#)。

- **stratisd** 服务在运行。
- 一个 **Stratis** 池已创建。
- 创建 **Stratis** 池的块设备没有被使用、卸载，且空间中至少为 **1 GB**。

#### 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，点您要向其添加块设备的 **Stratis** 池。
4. 在 **Stratis** 池 页面中，点 **Add block devices** 并选择 **Tier where to add a block device as data 或 cache**。

5.  
如果您要将块设备添加到使用密码短语加密的 **Stratis** 池中，请输入密码短语。
6.  
在 **Block devices** 下，选择要添加到池中的设备。
7.  
点**Add**。

## 第 26 章 监控 STRATIS 文件系统

作为 **Stratis** 用户，您可以查看系统中 **Stratis** 文件系统的信息，以监控其状态和可用空间。



## 重要

**Stratis** 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

## 26.1. 显示 STRATIS 文件系统的信息

您可以使用 **stratis** 工具列出 **Stratis** 文件系统的统计信息，如总数、使用以及可用大小或文件系统以及属于池的块设备。

**XFS** 文件系统的大小是它可以管理的用户数据总量。在精简配置的 **Stratis** 池中，**Stratis** 文件系统可能会显示为大于分配给它的空间的大小。**XFS** 文件系统的大小与这个明显大小匹配，这意味着它通常大于分配的空间。标准 **Linux** 实用程序（如 **df**）报告 **XFS** 文件系统的大小。这个值通常代表 **XFS** 文件系统所需的空间，因此由 **Stratis** 分配的空间。



## 重要

定期监控过度置备的 **Stratis** 池的使用情况。如果文件系统使用量接近分配的空间，**Stratis** 会自动使用池中的可用空间增加分配。但是，如果已经分配了所有可用空间，并且池已满，则不会分配额外的空间，从而导致文件系统耗尽空间。这可能导致使用 **Stratis** 文件系统的应用程序中数据丢失的风险。

## 先决条件



**Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅[安装 \*\*Stratis\*\*](#)。

## 流程



要显示系统中用于 **Stratis** 的所有块设备的信息：

```
# stratis blockdev
```

```
Pool Name Device Node Physical Size State Tier
my-pool /dev/sdb 9.10 TiB In-use Data
```

- 显示系统中所有 **Stratis** 池的信息：

```
# stratis pool

Name Total Physical Size Total Physical Used
my-pool 9.10 TiB 598 MiB
```

- 显示系统中所有 **Stratis** 文件系统的信息：

```
# stratis filesystem

Pool Name Name Used Created Device
my-pool my-fs 546 MiB Nov 08 2018 08:03 /dev/stratis/my-pool/my-fs
```

## 其他资源

- 系统中 **Stratis (8)** 手册页

## 26.2. 使用 WEB 控制台查看 STRATIS 池

您可以使用 Web 控制台查看现有的 **Stratis** 池，以及其包含的文件系统。

### 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](#)。

- **stratisd 服务在运行。**
- **您有一个现有的 Stratis 池。**

## 流程

1. **登录到 RHEL 8 web 控制台。**
2. **点 Storage。**
3. **在 Storage 表中，点您要查看的 Stratis 池。**

**Stratis 池**页面显示有关池以及您在池中创建的文件系统的所有信息。



## 第 27 章 在 STRATIS 文件系统中使用快照

您可以使用 **Stratis** 文件系统的快照任意时间捕获文件系统状态，并在以后恢复它。



### 重要

**Stratis** 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

### 27.1. STRATIS 快照的特性

在 **Stratis** 中，快照是作为另一个 **Stratis** 文件系统的副本创建的常规 **Stratis** 文件系统。

**Stratis** 中的当前快照实现的特征如下：

- 文件系统快照是另一个文件系统。
- 快照及其原始卷在生命周期中不会被链接。快照的文件系统可以比它从中创建的文件系统更长。
- 文件系统不一定被挂载来生成快照。
- 每个快照使用大约一半的实际后备存储，这是 XFS 日志所需要的。

### 27.2. 创建 STRATIS 快照

您可以创建 **Stratis** 文件系统作为现有 **Stratis** 文件系统的快照。

先决条件

- **Stratis 已安装，stratisd 服务正在运行。**如需更多信息，请参阅 [安装 Stratis](#)。
- **您已创建了 Stratis 文件系统。**如需更多信息，请参阅 [创建 Stratis 文件系统](#)。

## 流程

- **创建 Stratis 快照：**

```
# stratis fs snapshot my-pool my-fs my-fs-snapshot
```

快照是第一个类 **Stratis** 文件系统。您可以创建多个 **Stratis** 快照。这包括单个原始文件系统或者另一个快照文件系统的快照。如果文件系统是快照，则其 **origin** 字段将在详细的文件系统列表中显示其原始文件系统的 **UUID**。

## 其他资源

- **系统中 Stratis (8) 手册页**

## 27.3. 访问 STRATIS 快照的内容

您可以挂载 **Stratis** 文件系统的快照，使其可以被读写操作访问。

## 先决条件

- **Stratis 已安装，stratisd 服务正在运行。**如需更多信息，请参阅 [安装 Stratis](#)。
- **您已创建了 Stratis 快照。**如需更多信息，请参阅 [创建 Stratis 快照](#)。

## 流程

- **要访问快照，请将其作为常规文件系统挂载到 `/dev/stratis/my-pool/` 目录：**

```
# mount /dev/stratis/my-pool/my-fs-snapshot mount-point
```

## 其他资源

- [挂载 Stratis 文件系统](#)
- [在您的系统中的 mount \(8\) 手册页](#)

## 27.4. 将 STRATIS 文件系统恢复到以前的快照

您可以将 Stratis 文件系统的内容恢复到 Stratis 快照中捕获的状态。

### 先决条件

- **Stratis 已安装，stratisd 服务正在运行。**如需更多信息，请参阅 [安装 Stratis](#)。
- **您已创建了 Stratis 快照。**如需更多信息，请参阅 [创建 Stratis 快照](#)。

### 流程

1. 可选：备份文件系统的当前状态，以便以后可以访问它：

```
# stratis filesystem snapshot my-pool my-fs my-fs-backup
```

2. 卸载并删除原始文件系统：

```
# umount /dev/stratis/my-pool/my-fs
# stratis filesystem destroy my-pool my-fs
```

3. 在原始文件系统名称下创建快照副本：

```
# stratis filesystem snapshot my-pool my-fs-snapshot my-fs
```

4. 挂载快照，它现在可以和原始文件系统的名称相同：

```
# mount /dev/stratis/my-pool/my-fs mount-point
```

-

名为 *my-fs* 的文件系统的内容与快照 *my-fs-snapshot* 一致。

#### 其他资源

- [系统中 Stratis \(8\) 手册页](#)

## 27.5. 删除 STRATIS 快照

您可以从池中删除 Stratis 快照。快照中的数据会丢失。

#### 先决条件

- Stratis 已安装，`stratisd` 服务正在运行。如需更多信息，[请参阅安装 Stratis](#)。
- 您已创建了 Stratis 快照。如需更多信息，[请参阅创建 Stratis 快照](#)。

#### 流程

1. 卸载快照：

```
# umount /dev/stratis/my-pool/my-fs-snapshot
```

2. 销毁快照：

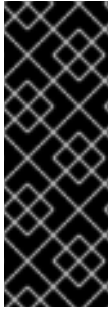
```
# stratis filesystem destroy my-pool my-fs-snapshot
```

#### 其他资源

- [系统中 Stratis \(8\) 手册页](#)

## 第 28 章 删除 STRATIS 文件系统

您可以删除现有 **Stratis** 文件系统或池。删除 **Stratis** 文件系统或池后，它无法被恢复。



### 重要

**Stratis** 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

### 28.1. 删除 STRATIS 文件系统

您可以删除现有 **Stratis** 文件系统。保存的数据会丢失。

#### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 您已创建了 **Stratis** 文件系统。如需更多信息，请参阅 [创建 Stratis 文件系统](#)。

#### 流程

1.

卸载文件系统：

```
# umount /dev/stratis/my-pool/my-fs
```

2.

销毁文件系统：

```
# stratis filesystem destroy my-pool my-fs
```

#### 验证

- 验证文件系统不再存在：

■

```
# stratis filesystem list my-pool
```

## 其他资源

- [系统中 Stratis \(8\) 手册页](#)

## 28.2. 使用 WEB 控制台从 STRATIS 池中删除文件系统

您可以使用 Web 控制台从现有 Stratis 池中删除一个文件系统。



### 注意

删除 Stratis 池文件系统会删除其包含的所有数据。

## 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](#)。

- Stratis 已安装，stratisd 服务正在运行。

默认情况下，web 控制台会检测并安装 Stratis。但是，要手动安装 Stratis，请参阅[安装 Stratis](#)。

- 您有一个现有的 Stratis 池，并在 Stratis 池中创建一个文件系统。

## 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，点击您要从中删除文件系统的 **Stratis** 池。
4.  
在 **Stratis** 池 页面中，滚动到 **Stratis** 文件系统 部分，然后点击您要删除的文件系统的菜单按钮 **also**。
5.  
从下拉菜单中选择 **Delete**。
6.  
在 **Confirm deletion** 对话框中，单击 **Delete**。

### 28.3. 删除 STRATIS 池

您可以删除现有 **Stratis** 池。保存的数据会丢失。

#### 先决条件

- **Stratis** 已安装，**stratisd** 服务正在运行。如需更多信息，请参阅 [安装 Stratis](#)。
- 您已创建了 **Stratis** 池：
  - 要创建未加密的池，请参阅 [创建未加密的 Stratis 池](#)。
  - 要创建加密的池，请参阅使用 [内核密钥环 中的密钥创建加密的 Stratis 池](#)。

#### 流程

1.

列出池中的文件系统：

```
# stratis filesystem list my-pool
```

2.

卸载池中的所有文件系统：

```
# umount /dev/stratis/my-pool/my-fs-1 \  
         /dev/stratis/my-pool/my-fs-2 \  
         /dev/stratis/my-pool/my-fs-n
```

3.

销毁文件系统：

```
# stratis filesystem destroy my-pool my-fs-1 my-fs-2
```

4.

销毁池：

```
# stratis pool destroy my-pool
```

验证

•

验证池不再存在：

```
# stratis pool list
```

其他资源

•

系统中 **Stratis (8)** 手册页

## 28.4. 使用 WEB 控制台删除 STRATIS 池

您可以使用 Web 控制台删除现有的 Stratis 池。





### 注意

删除 Stratis 池会删除其包含的所有数据。

### 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](#)。

- stratisd 服务在运行。
- 您有一个现有的 Stratis 池。

### 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，点您要删除的 Stratis 池的菜单按钮。
4. 从下拉菜单中选择 Delete pool。

5.

在 **Permanently delete pool** 对话框中，单击 **Delete**。