



# Red Hat Enterprise Linux 8

## 管理文件系统

在 Red Hat Enterprise Linux 8 中创建、修改和管理文件系统



# Red Hat Enterprise Linux 8 管理文件系统

---

在 Red Hat Enterprise Linux 8 中创建、修改和管理文件系统

## 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 支持各种文件系统。每种类型的文件系统解决不同的问题，它们的用法是特定于应用程序的。使用有关关键差异和注意事项的信息，来根据特定应用程序要求部署合适的文件系统。支持的文件系统包括本地磁盘文件系统 XFS 和 ext4，以及网络和客户端-和-服务器文件系统 NFS 和 SMB。您可以使用文件系统执行多个操作，如创建、挂载、备份、恢复、检查和修复，以及使用配额限制存储空间。

# Table of Contents

对红帽文档提供反馈 .....	7
<b>第 1 章 可用文件系统概述 .....</b>	<b>8</b>
1.1. 文件系统类型 .....	8
1.2. 本地文件系统 .....	8
1.3. XFS 文件系统 .....	9
1.4. EXT4 文件系统 .....	10
1.5. XFS 和 EXT4 的比较 .....	10
1.6. 选择本地文件系统 .....	11
1.7. 网络文件系统 .....	12
1.8. 共享存储文件系统 .....	12
1.9. 在网络和共享存储文件系统间进行选择 .....	13
1.10. 卷管理文件系统 .....	13
<b>第 2 章 使用 RHEL 系统角色管理本地存储 .....</b>	<b>15</b>
2.1. 使用 STORAGE RHEL 系统角色在块设备上创建一个 XFS 文件系统 .....	15
2.2. 使用 STORAGE RHEL 系统角色永久挂载一个文件系统 .....	16
2.3. 使用 STORAGE RHEL 系统角色创建或者调整逻辑卷大小 .....	18
2.4. 使用 STORAGE RHEL 系统角色启用在线块丢弃 .....	20
2.5. 使用 STORAGE RHEL 系统角色创建并挂载文件系统 .....	22
2.6. 使用 STORAGE RHEL 系统角色配置一个 RAID 卷 .....	24
2.7. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池 .....	26
2.8. 使用 STORAGE RHEL 系统角色为 RAID LVM 卷配置条带大小 .....	27
2.9. 使用 STORAGE RHEL 系统角色配置 LVM-VDO 卷 .....	29
2.10. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密卷 .....	31
2.11. 使用 STORAGE RHEL 系统角色创建共享的 LVM 设备 .....	34
<b>第 3 章 使用 WEB 控制台管理分区 .....</b>	<b>37</b>
3.1. 在 WEB 控制台中显示使用文件系统格式化的分区 .....	37
3.2. 在 WEB 控制台中创建分区 .....	38
3.3. 在 WEB 控制台中删除分区 .....	42
<b>第 4 章 挂载 NFS 共享 .....</b>	<b>44</b>
4.1. NFS 客户端所需的服务 .....	44
4.2. 准备 NFSV3 客户端以在防火墙后面运行 .....	45
4.3. 准备 NFSV4 客户端以在防火墙后面运行 .....	46
4.4. 手动挂载 NFS 共享 .....	47
4.5. 系统引导时自动挂载 NFS 共享 .....	47
4.6. 在 WEB 控制台中连接 NFS 挂载 .....	48
4.7. 在 WEB 控制台中自定义 NFS 挂载选项 .....	51
4.8. 在 RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT 域中使用 KERBEROS 设置 NFS 客户端 .....	53
4.9. 配置 GNOME 以将用户设置存储在由 NFS 共享托管的主目录中 .....	55
4.10. 常用的 NFS 挂载选项 .....	56
4.11. 启用 NFS 内容的客户端缓存 .....	58
<b>第 5 章 挂载一个 SMB 共享 .....</b>	<b>65</b>
5.1. 支持的 SMB 协议版本 .....	65
5.2. UNIX 扩展支持 .....	66
5.3. 手动挂载 SMB 共享 .....	67
5.4. 系统启动时自动挂载 SMB 共享 .....	68
5.5. 创建一个凭据文件以对 SMB 共享进行身份验证 .....	69
5.6. 执行多用户 SMB 挂载 .....	70

5.7. 常用的 SMB 挂载选项	72
<b>第 6 章 持久性命名属性概述</b>	<b>74</b>
6.1. 非持久性命名属性的缺陷	74
6.2. 文件系统和设备识别符	75
6.3. 使用 /DEV/DISK/ 中的 UDEV 机制管理的设备名称	76
6.4. 使用 DM 多路径的通用识别符	79
6.5. UDEV 设备命名规则的限制	80
6.6. 列出持久性命名属性	80
6.7. 修改持久性命名属性	82
<b>第 7 章 使用 PARTED 的分区操作</b>	<b>84</b>
7.1. 查看使用 PARTED 的分区表	84
7.2. 使用 PARTED 在磁盘中创建分区表	85
7.3. 使用 PARTED 创建分区	87
7.4. 使用 PARTED 删除分区	90
7.5. 使用 PARTED 重新定义分区大小	91
<b>第 8 章 重新分区磁盘策略</b>	<b>94</b>
8.1. 使用未分区的空闲空间	94
8.2. 使用未使用分区中的空间	95
8.3. 使用活跃分区中的空闲空间	95
<b>第 9 章 XFS 入门</b>	<b>100</b>
9.1. XFS 文件系统	100
9.2. 和 EXT4 和 XFS 一起使用的工具比较	102
<b>第 10 章 创建 XFS 文件系统</b>	<b>103</b>
10.1. 使用 MKFS.XFS 创建 XFS 文件系统	103
<b>第 11 章 备份 XFS 文件系统</b>	<b>105</b>
11.1. XFS 备份特性	105
11.2. 使用 XFSDUMP 备份 XFS 文件系统	106
<b>第 12 章 从备份中恢复 XFS 文件系统</b>	<b>108</b>
12.1. 从备份中恢复 XFS 的特性	108
12.2. 使用 XFSRESTORE 从备份中恢复 XFS 文件系统	108
12.3. 从磁带恢复 XFS 备份时的说明性消息	110
<b>第 13 章 增加 XFS 文件系统的大小</b>	<b>111</b>
13.1. 使用 XFS_GROWFS 增加 XFS 文件系统的大小	111
<b>第 14 章 配置 XFS 错误行为</b>	<b>113</b>
14.1. XFS 中的可配置错误处理	113
14.2. 特定和未定义的 XFS 错误条件的配置文件	114
14.3. 为特定条件设置 XFS 行为	114
14.4. 为未定义条件设置 XFS 行为	115
14.5. 设置 XFS 卸载行为	116
<b>第 15 章 检查和修复文件系统</b>	<b>118</b>
15.1. 需要文件系统检查的场景	118
15.2. 运行 FSCK 的潜在副作用	119
15.3. XFS 中的错误处理机制	119
15.4. 使用 XFS_REPAIR 检查 XFS 文件系统	121
15.5. 使用 XFS_REPAIR 修复 XFS 文件系统	122
15.6. EXT2、EXT3 和 EXT4 中的处理机制出错	123

15.7. 使用 E2FCK 检查 EXT2、EXT3 或者 EXT4 文件系统	124
15.8. 使用 E2FCK 修复 EXT2、EXT3 或者 EXT4 文件系统	124
<b>第 16 章 挂载文件系统</b>	<b>126</b>
16.1. LINUX 挂载机制	126
16.2. 列出当前挂载的文件系统	127
16.3. 使用 MOUNT 挂载文件系统	128
16.4. 移动挂载点	129
16.5. 使用 UMount 卸载文件系统	129
16.6. 在 WEB 控制台中挂载和卸载文件系统	130
16.7. 常用挂载选项	131
<b>第 17 章 在多个挂载点共享挂载</b>	<b>133</b>
17.1. 共享挂载的类型	133
17.2. 创建私有挂载点副本	134
17.3. 创建共享挂载点副本	135
17.4. 创建从挂载点副本	137
17.5. 防止挂载点重复	139
<b>第 18 章 永久挂载文件系统</b>	<b>140</b>
18.1. /ETC/FSTAB 文件	140
18.2. 在 /ETC/FSTAB 中添加文件系统	141
<b>第 19 章 根据需要挂载文件系统</b>	<b>143</b>
19.1. AUTOFS 服务	143
19.2. AUTOFS 配置文件	143
19.3. 配置 AUTOFS 挂载点	146
19.4. 使用 AUTOFS 服务自动挂载 NFS 服务器用户主目录	147
19.5. 覆盖或添加 AUTOFS 站点配置文件	148
19.6. 使用 LDAP 存储自动挂载器映射	150
19.7. 使用 SYSTEMD.AUTOMOUNT 在 /ETC/FSTAB 按需挂载文件系统	151
19.8. 使用 SYSTEMD.AUTOMOUNT 通过挂载单元根据需要挂载文件系统	152
<b>第 20 章 使用 IDM 中的 SSSD 组件来缓存 AUTOFS 映射</b>	<b>155</b>
20.1. 手动配置 AUTOFS，来将 IDM 服务器用作 LDAP 服务器	155
20.2. 配置 SSSD 来缓存 AUTOFS 映射	156
<b>第 21 章 为 ROOT 文件系统设置只读权限</b>	<b>159</b>
21.1. 始终保留写入权限的文件和目录	159
21.2. 将 ROOT 文件系统配置为在引导时使用只读权限挂载	160
<b>第 22 章 对带有配额的 XFS 限制存储空间的使用</b>	<b>162</b>
22.1. 磁盘配额	162
22.2. XFS_QUOTA 工具	162
22.3. XFS 中的文件系统配额管理	163
22.4. 为 XFS 启用磁盘配额	163
22.5. 报告 XFS 使用量	164
22.6. 修改 XFS 配额限制	165
22.7. 为 XFS 设置项目限制	167
<b>第 23 章 对带有配额的 EXT4 限制存储空间使用</b>	<b>168</b>
23.1. 安装配额工具	168
23.2. 在创建文件系统时启用配额功能	168
23.3. 在现有文件系统中启用配额功能	169
23.4. 启用配额强制	170

23.5. 为每个用户分配配额	171
23.6. 为每个组群分配配额	173
23.7. 为每个项目分配配额	174
23.8. 为软限制设置宽限期	175
23.9. 关闭文件系统配额	176
23.10. 报告磁盘配额	176
<b>第 24 章 丢弃未使用块</b>	<b>178</b>
要求	178
24.1. 块丢弃操作的类型	178
24.2. 执行批块丢弃	179
24.3. 启用在线块丢弃	180
24.4. 使用 STORAGE RHEL 系统角色启用在线块丢弃	180
24.5. 启用定期块丢弃	182
<b>第 25 章 设置 STRATIS 文件系统</b>	<b>184</b>
25.1. STRATIS 文件系统的组件	185
25.2. 与 STRATIS 兼容的块设备	186
25.3. 安装 STRATIS	187
25.4. 创建未加密的 STRATIS 池	187
25.5. 使用 WEB 控制台创建未加密的 STRATIS 池	189
25.6. 使用内核密钥环中的密钥创建加密的 STRATIS 池	190
25.7. 使用 WEB 控制台创建加密的 STRATIS 池	192
25.8. 使用 WEB 控制台重命名 STRATIS 池	195
25.9. 在 STRATIS 文件系统中设置过度置备模式	196
25.10. 将 STRATIS 池绑定到 NBDE	197
25.11. 将 STRATIS 池绑定到 TPM	198
25.12. 使用内核密钥环解加密的 STRATIS 池	199
25.13. 解除 STRATIS 池与补充加密的绑定	200
25.14. 启动和停止 STRATIS 池	201
25.15. 创建 STRATIS 文件系统	202
25.16. 使用 WEB 控制台在 STRATIS 池中创建文件系统	203
25.17. 挂载 STRATIS 文件系统	205
25.18. 使用 SYSTEMD 服务在 /ETC/FSTAB 中设置非 ROOT STRATIS 文件系统	206
<b>第 26 章 使用附加块设备扩展 STRATIS 池</b>	<b>207</b>
26.1. 在 STRATIS 池中添加块设备	207
26.2. 使用 WEB 控制台在 STRATIS 池中添加块设备	207
<b>第 27 章 监控 STRATIS 文件系统</b>	<b>210</b>
27.1. 显示 STRATIS 文件系统的信息	210
27.2. 使用 WEB 控制台查看 STRATIS 池	211
<b>第 28 章 在 STRATIS 文件系统中使用快照</b>	<b>213</b>
28.1. STRATIS 快照的特性	213
28.2. 创建 STRATIS 快照	213
28.3. 访问 STRATIS 快照的内容	214
28.4. 将 STRATIS 文件系统恢复到以前的快照	215
28.5. 删除 STRATIS 快照	216
<b>第 29 章 删除 STRATIS 文件系统</b>	<b>217</b>
29.1. 删除 STRATIS 文件系统	217
29.2. 使用 WEB 控制台从 STRATIS 池中删除文件系统	218
29.3. 删除 STRATIS 池	219
29.4. 使用 WEB 控制台删除 STRATIS 池	220



---

<b>第 30 章 开始使用 EXT3 文件系统 .....</b>	<b>223</b>
30.1. EXT3 文件的特性	223
30.2. 创建 EXT3 文件系统	223
30.3. 挂载 EXT3 文件系统	225
30.4. 重新定义 EXT3 文件系统大小	226
 <b>第 31 章 EXT4 文件系统入门 .....</b>	 <b>229</b>
31.1. EXT4 文件的特性	229
31.2. 创建 EXT4 文件系统	230
31.3. 挂载 EXT4 文件系统	232
31.4. 重新定义 EXT4 文件系统大小	233
31.5. 和 EXT4 和 XFS 一起使用的工具比较	234



---

## 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** 字段中输入您的建议以改进。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

# 第 1 章 可用文件系统概述

由于大量的可用选项以及所涉及的权衡，因此选择适合您应用程序的文件系统是一个重要的决定。

以下章节描述了 Red Hat Enterprise Linux 8 默认包括的文件系统，以及对最合适您应用程序的文件系统的建议。

## 1.1. 文件系统类型

Red Hat Enterprise Linux 8 支持各种文件系统(FS)。不同类型的文件系统可以解决不同类型的问题，它们的使用会根据特定应用程序而有所不同。在最一般的级别上，可用的文件系统可以分为以下主要类型：

表 1.1. 文件系统类型及其用例

类型	文件系统	属性和使用案例
磁盘或本地 FS	XFS	XFS 是 RHEL 中的默认文件系统。红帽建议将 XFS 部署为本地文件系统，除非有特定原因部署为其它：例如，兼容性或涉及性能方面的情况。
	ext4	ext4 在 Linux 中有相似性的好处，其从较旧的 ext2 和 ext3 文件系统演变而来。在很多情况下，它在性能上可以与 XFS 相媲美。对 ext4 文件系统和文件大小支持限制比 XFS 上的要低。
网络或客户端-和-服务器 FS	NFS	使用 NFS 在同一个网络中的多个系统间共享文件。
	SMB	使用 SMB 进行与微软 Windows 系统的文件共享。
共享存储或共享磁盘 FS	GFS2	GFS2 为计算集群成员提供共享写入访问。其重点在于稳定性和可靠性，获得与本地文件系统类似的体验。SAS Grid、Tibco MQ、IBM Websphere MQ 和 Red Hat Active MQ 已被成功部署在 GFS2 上。
卷管理 FS	Stratis（技术预览）	Stratis 是基于 XFS 和 LVM 的组合构建的卷管理器。Stratis 的目的是模拟卷管理文件系统（如 Btrfs 和 ZFS）所提供的功能。可以手动构建此堆栈，但 Stratis 可减少配置的复杂度、实施最佳实践并整合错误信息。

## 1.2. 本地文件系统

本地文件系统是在单一本地服务器中运行并直接附加到存储中的文件系统。

例如，本地文件系统是内部 SATA 或 SAS 磁盘的唯一选择，可在当您的服务器具有带有本地驱动器的内部硬件 RAID 控制器时使用。当 SAN 上导出的设备未共享时，本地文件系统也是 SAN 连接的存储上最常用的文件系统。

所有本地文件系统都与 POSIX 兼容，且与所有支持的 Red Hat Enterprise Linux 版本完全兼容。与 POSIX 兼容的文件系统为一组定义良好的系统调用提供支持，如 **read()**、**write()** 和 **seek()**。

在考虑文件系统选择时，根据文件系统需要多大、它必须具有哪些唯一功能，以及如何在工作负载下执行文件系统。

#### 可用的本地文件系统

- XFS
- ext4

### 1.3. XFS 文件系统

XFS 是一个高度可扩展、高性能、健壮且成熟的 64 位日志文件系统，其支持单个主机上非常大的文件和文件系统。它是 Red Hat Enterprise Linux 8 中的默认文件系统。XFS 最初于 1990 年代由 SGI 早期开发，并在非常大型的服务器和存储阵列中运行有很长的历史记录。

XFS 的功能包括：

#### 可靠性

- 元数据日志，其确保系统崩溃后文件系统的完整性，方法是保留系统重启和重新挂载文件系统时可以重新执行的文件系统操作的记录，
- 广泛的运行时元数据一致性检查
- 可扩展且快速修复工具
- 配额日志。这可避免在崩溃后进行冗长的配额一致性检查。

#### 可伸缩性和性能

- 支持最多 1024 TiB 的文件系统大小
- 支持大量并发操作的能力
- B-tree 索引，用于空闲空间的可扩展性管理
- 复杂的元数据读头算法
- 优化流视频工作负载

#### 分配方案

- 基于扩展数据块的分配
- 条带化分配策略
- 延迟分配
- 空间预分配
- 动态分配的 inode

#### 其他功能

- 基于 Reflink 的文件副本

- 严格集成备份和恢复工具
- 在线清理
- 在线文件系统增大
- 全面的诊断功能
- 扩展属性(xattr)。这允许系统能够按文件关联多个额外的名称/值对。
- 项目或目录配额。这允许对目录树的配额限制。
- 小于秒的时间戳

## 性能特性

XFS 在具有企业工作负载的大型系统上具有高性能。大型系统是一个有相对较多的 CPU、多个 HBA 和连接外部磁盘阵列的系统。XFS 在具有多线程、并行 I/O 工作负载的较小系统上也表现良好。

对于单线程、元数据密集型工作负载，XFS 的性能相对较低：例如，在单个线程中创建或删除大量小文件的工作负载。

## 1.4. EXT4 文件系统

ext4 文件系统是 ext 文件系统系列的第四代。它是 Red Hat Enterprise Linux 6 中的默认文件系统。

ext4 驱动程序可以对 ext2 和 ext3 文件系统进行读写，但 ext4 文件系统格式与 ext2 和 ext3 驱动程序不兼容。

ext4 添加了几个新的改进的功能，例如：

- 支持的文件系统大小高达 50 TiB
- 基于扩展的元数据
- 延迟分配
- 日志的 checksum
- 大型存储支持

基于扩展数据块的元数据和延迟分配功能提供了一种更加紧凑和高效的方法来跟踪文件系统上的已用空间。这些功能提高了文件系统性能，并减少了元数据所占用的空间。延迟分配允许文件系统延迟选择新写入用户数据的永久位置，直到数据刷新到磁盘。这可实现更高的性能，因为它允许更大的、连续的分配，允许文件系统根据更佳的信息做出决策。

ext4 中使用 **fsck** 工具的文件系统修复时间比在 ext2 和 ext3 中要快得多。一些文件系统修复的性能会增加最多 6 倍。

## 1.5. XFS 和 EXT4 的比较

XFS 是 RHEL 中的默认文件系统。本节比较 XFS 和 ext4 的用法和功能。

### 元数据错误行为

在 ext4 中，当文件系统遇到元数据错误时您可以配置行为。默认的行为是继续操作。当 XFS 遇到不可恢复的元数据错误时，它会关闭文件系统，并返回 **EFSCORRUPTED** 错误。

## 配额

在 ext4 中，您可以在创建文件系统时启用配额，或稍后在现有文件系统上启用配额。然后您可以使用挂载选项配置配额强制。

XFS 配额不是一个可重新挂载的选项。您必须在初始挂载中激活配额。

在 XFS 文件系统上运行 **quotacheck** 命令没有效果。当您第一次打开配额记帐时，XFS 会自动检查配额。

## 文件系统重新定义大小

XFS 没有工具来缩小文件系统的大小。您只能增大 XFS 文件系统的大小。而 ext4 支持扩展和缩小文件系统大小。

## 内节点 (inode) 号

ext4 文件系统不支持超过  $2^{32}$  内节点。

XFS 支持动态内节点分配。XFS 文件系统上空间索引节点的大小计算为总文件系统空间的百分比。要防止系统耗尽 inode，管理员可以在文件系统创建后调整这个百分比，假设文件系统上剩余的可用空间。

某些应用程序无法正确处理 XFS 文件系统上大于  $2^{32}$  的 inode 数。这些应用程序可能会导致 32 位 stat 调用失败，返回值为 **EOVERFLOW**。在以下情况下，inode 数超过  $2^{32}$ ：

- 文件系统大于 1 TiB，其 inode 为 256 字节。
- 文件系统大于 2 TiB，其 inode 为 512 字节。

如果您的应用程序由于 inode 数太大而失败，请使用 **-o inode32** 选项挂载 XFS 文件系统，来强制 inode 数低于  $2^{32}$ 。请注意，使用 **inode32** 不会影响已分配了 64 位数的 inode。



### 重要

除非特定环境需要，否则 *请勿* 使用 **inode32** 选项。**inode32** 选项可改变分配行为。因此，如果没有可用空间在较低磁盘块中分配 inode，则可能会出现 **ENOSPC** 错误。

## 1.6. 选择本地文件系统

要选择一个满足应用程序要求的文件系统，您必须了解要在其上部署文件系统的目标系统。一般情况下，请使用 XFS，除非您有 ext4 的特定用例。

### XFS

对于大规模部署，请使用 XFS，特别是在处理大型文件（几百兆字节）和高 I/O 并发时。XFS 在具有高带宽（超过 200MB/s）和 1000 IOPS 的环境中执行最佳执行。但是，与 ext4 相比，它会消耗更多 CPU 资源用于元数据操作，且不支持文件系统缩小。

### ext4

对于 I/O 带宽有限的较小的系统或环境，ext4 可能更加适合。它在单线程、较低 I/O 工作负载和具有较低吞吐量要求的环境中表现更好。ext4 也支持离线缩减，这在调整文件系统大小时需要时很有用。

在目标服务器和存储系统上基准测试您的应用程序的性能，以确保所选的文件系统满足您的性能和可扩展性要求。

表 1.2. 本地文件系统建议概述

场景	推荐的文件系统
没有特殊用例	XFS
大服务器	XFS
大存储设备	XFS
大文件	XFS
多线程 I/O	XFS
单线程 I/O	ext4
有限 I/O 功能（在 1000 IOPS 下）	ext4
有限带宽（在 200MB/s 下）	ext4
CPU 绑定工作负载	ext4
支持离线缩小	ext4

## 1.7. 网络文件系统

网络文件系统也称为客户端/服务器文件系统，使客户端系统能够访问存储在共享服务器上的文件。这使得多个系统上的多个用户可以共享文件和存储资源。

此类文件系统构建自一个或多个服务器，它们将一组文件系统导出到一个或多个客户端。客户端节点无法访问底层的块存储，而是使用允许更好的访问控制的协议来与存储进行交互。

### 可用网络文件系统

- RHEL 客户最常用的客户端/服务器文件系统是 NFS 文件系统。RHEL 提供了一个 NFS 服务器组件，来通过网络导出本地文件系统，并提供 NFS 客户端来导入这些文件系统。
- RHEL 还包括 CIFS 客户端，其支持流行的 Microsoft SMB 文件服务器来实现 Windows 互操作性。用户空间 Samba 服务器为 Windows 客户端提供 RHEL 服务器的 Microsoft SMB 服务。

## 1.8. 共享存储文件系统

共享存储文件系统有时称为集群文件系统，允许集群中的每台服务器通过本地存储区域网络(SAN)直接访问共享块设备。

### 和网络文件系统的比较

与客户端/服务器文件系统一样，共享存储文件系统在一组服务器上工作，这些服务器都是群集的成员。但与 NFS 不同，没有一个服务器向其他成员提供对数据或元数据的访问：群集的每个成员都可以直接访问同一存储设备（共享存储），并且所有群集节点都可以访问同一组文件。



## 并发

缓存一致性是集群文件系统中确保数据一致性和完整性的关键。集群中所有文件的单个版本都必须对群集内的所有节点可见。文件系统必须阻止群集成员同时更新同一存储块，以防止数据损坏。为此，共享存储文件系统使用集群范围的锁机制作为并发控制机制来仲裁对存储的访问。例如，在创建新文件或写入在多个服务器上打开的文件之前，服务器上的文件系统组件必须获得正确的锁。

群集文件系统的要求是提供一种像 Apache Web 服务器那样高可用的服务。群集的任何成员都将看到存储在共享磁盘文件系统的数据的完全一致的视图，并且所有更新都会通过锁机制正确仲裁。

## 性能特性

由于锁开销的计算成本，共享磁盘文件系统并不总像运行在同一系统上的本地文件系统那样表现良好。如果每个节点几乎以独占方式写入一组不与其他节点共享的特定文件，或者一组文件在一组节点上以几乎独占的只读方式被共享，那么共享磁盘文件系统可以很好地执行这种工作负载。这将导致最小的跨节点缓存失效，并可最大限度地提高性能。

设置共享磁盘文件系统非常复杂，调优应用以在共享磁盘文件系统中表现良好非常有挑战性。

## 可用的共享存储文件系统

- Red Hat Enterprise Linux 提供 GFS2 文件系统。GFS2 与 Red Hat Enterprise Linux High Availability Add-On 和 Resilient Storage 附加组件紧密整合。

Red Hat Enterprise Linux 支持集群中大小为 2 到 16 个节点的 GFS2。

## 1.9. 在网络和共享存储文件系统间进行选择

在网络和共享存储文件系统间选择时，请考虑以下几点：

- 对于提供 NFS 服务器的环境，基于 NFS 的网络文件系统是非常常见和流行的选择。
- 网络文件系统可以使用非常高性能的网络技术（如 Infiniband 或 10 GB 以太网卡）进行部署。这意味着您不应该仅仅为了获得存储的原始带宽而转向共享存储文件系统。如果访问速度至关重要，则使用 NFS 导出类似 XFS 的本地文件系统。
- 共享存储文件的设置或维护并非易事，因此您应仅在无法用本地或网络文件系统提供所需的可用性时才部署它们。
- 集群环境中的共享存储文件系统通过消除在涉及重新分配高可用性服务的典型故障切换情景中需要执行的卸载和挂载所需的步骤，来帮助减少停机时间。

红帽建议您使用网络文件系统，除非您有共享存储文件系统的特定用例。共享存储文件系统主要用于需要以最少的停机时间提供高可用性服务且具有严格的服务等级要求的部署。

## 1.10. 卷管理文件系统

卷管理文件系统集成整个存储堆栈，以实现简洁和堆栈内优化。

### 可用卷管理文件系统

- Red Hat Enterprise Linux 8 作为技术预览提供 Stratis 卷管理器。Stratis 对文件系统层使用 XFS，并将其与 LVM、设备映射器和其他组件集成。

Stratis 首次在 Red Hat Enterprise Linux 8.0 中发布。它被设计为填补红帽弃用 Btrfs 时出现的空白。Stratis 1.0 是一个直观的、基于命令行的卷管理器，可以在隐藏用户复杂性的同时执行重要的存储管理操作：

- 卷管理
- 创建池
- 精简存储池
- 快照
- 自动读取缓存

Stratis 提供强大的功能，但目前缺乏其他产品（如 Btrfs 或 ZFS）的某些功能。最值得注意的是，它不支持带自我修复的 CRC。

## 第 2 章 使用 RHEL 系统角色管理本地存储

要使用 Ansible 管理 LVM 和本地文件系统(FS)，您可以使用 **存储** 角色，这是 RHEL 8 中可用的 RHEL 系统角色之一。

使用 **存储** 角色可让您自动管理多台机器上的磁盘和逻辑卷上的文件系统，以及从 RHEL 7.7 开始的所有 RHEL 版本。

有关 RHEL 系统角色以及如何应用它们的更多信息，请参阅 [RHEL 系统角色简介](#)。

### 2.1. 使用 **STORAGE** RHEL 系统角色在块设备上创建一个 XFS 文件系统

示例 Ansible playbook 使用 **存储** 角色，通过默认参数在块设备上创建 XFS 文件系统。如果 `/dev/sdb` 设备或者挂载点目录上的文件系统不存在，则 playbook 会创建它们。



#### 注意

**存储** 角色只能在未分区、整个磁盘或逻辑卷(LV)上创建文件系统。它不能在分区中创建文件系统。

#### 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 an XFS file system on a block device
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_volumes:
          - name: barefs
            type: disk
            disks:
              - sdb
            fs_type: xfs
```

示例 playbook 中指定的设置包括：

**名称**：**barefs**

卷名称（示例中的 **barefs**）目前是任意的。**存储**角色 根据 disk 属性下 **列出的磁盘设备** 来识别卷。

**fs\_type**: `<file_system>`

如果要使用默认文件系统 XFS，可以省略 **fs\_type** 参数。

**disks: <list\_of\_disks\_and\_volumes>**

磁盘和 LV 名称的 YAML 列表。要在 LV 上创建文件系统，请在 disk 属性下提供 LVM 设置，包括封闭卷组。详情请参阅 [使用存储 RHEL 系统角色创建或者重新定义逻辑卷大小](#)。

不要提供到 LV 设备的路径。

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

2.

验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook**：

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

#### 其它资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file
- [/usr/share/doc/rhel-system-roles/storage/](#) directory

## 2.2. 使用 STORAGE RHEL 系统角色永久挂载一个文件系统

示例 **Ansible** **playbook** 使用 **存储** 角色来永久挂载现有的文件系统。它通过向 **/etc/fstab** 文件中添加适当的条目来确保文件系统立即可用并永久挂载。这允许文件系统在重启后保持挂载。如果 **/dev/sdb** 设备或者挂载点目录上的文件系统不存在，则 **playbook** 会创建它们。

#### 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: Persistently mount a file system
      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_user: somebody
            mount_group: somegroup
            mount_mode: 0755
```

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

2. 验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook** :

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

#### 其它资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md file](#)
- [/usr/share/doc/rhel-system-roles/storage/ directory](#)

## 2.3. 使用 STORAGE RHEL 系统角色创建或者调整逻辑卷大小

使用 **storage** 角色执行以下任务：

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

如果卷组不存在，则角色会创建它。如果逻辑卷存在于卷组中，如果大小与 **playbook** 中指定的内容不匹配，则会调整大小。

如果您要缩小逻辑卷，为了避免数据丢失，您必须确保该逻辑卷中的文件系统没有使用正在缩小的逻辑卷中的空间。

#### 先决条件

- [您已准备好控制节点和受管节点](#)

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

## 流程

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

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

示例 **playbook** 中指定的设置包括以下内容：

大小：`<size>`

您必须使用单位（如 GiB）或百分比（例如 60%）来指定大小。

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

2. 验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook** :

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

验证

- 验证指定的卷是否已创建或重新定义到请求的大小：

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

其它资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file
- [/usr/share/doc/rhel-system-roles/storage/](#) directory

## 2.4. 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'
```

其它资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` file
- `/usr/share/doc/rhel-system-roles/storage/` directory

## 2.5. 使用 STORAGE RHEL 系统角色创建并挂载文件系统

示例 Ansible playbook 使用 存储 角色来创建和挂载文件系统。它通过向 `/etc/fstab` 文件中添加适当的条目来确保文件系统立即可用并永久挂载。这允许文件系统在重启后保持挂载。如果 `/dev/sdb` 设备或者挂载点目录上的文件系统不存在，则 **playbook** 会创建它们。

### 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 and mount a file system
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
```

示例 `playbook` 中指定的设置包括：

**disks:** *<list\_of\_devices>*

角色在创建卷时使用的设备名称的 YAML 列表。

**fs\_type:** *<file\_system>*

指定角色应在卷上设置的文件系统。您可以选择 `xfs`、`ext3`、`ext4`、`swap` 或未格式化的。

**label-name:** *<file\_system\_label>*

可选：设置文件系统的标签。

**mount\_point:** *<directory>*

可选：如果卷应自动挂载，请将 `mount_point` 变量设置为应挂载卷的目录。

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

2.

验证 `playbook` 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 `playbook`：

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

## 其它资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` file

- `/usr/share/doc/rhel-system-roles/storage/` directory

## 2.6. 使用 STORAGE RHEL 系统角色配置一个 RAID 卷

使用存储系统角色，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'
```

其它资源

- 

`/usr/share/ansible/roles/rhel-system-roles.storage/README.md` file

- 

`/usr/share/doc/rhel-system-roles/storage/` directory

- 

管理 [RAID](#)

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

使用存储系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 上配置带有 RAID 的 LVM 池。您可以使用可用参数建立 Ansible playbook，以配置带有 RAID 的 LVM 池。

### 先决条件

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

### 流程

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

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure LVM pool with RAID
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_safe_mode: false
        storage_pools:
          - name: my_pool
            type: lvm
            disks: [sdh, sdi]
            raid_level: raid1
            volumes:
              - name: my_volume
                size: "1 GiB"
                mount_point: "/mnt/app/shared"
                fs_type: xfs
                state: present
```

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

2.

验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook**：

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

验证

•

验证您的池是否在 **RAID** 中：

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

其它资源

•

[/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file

•

[/usr/share/doc/rhel-system-roles/storage/](#) directory

•

[管理 RAID](#)

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

使用存储系统角色，您可以使用 **Red Hat Ansible Automation Platform** 为 **RHEL** 上的 **RAID LVM** 卷配置条带大小。您可以使用可用参数建立 **Ansible** **playbook**，以配置带有 **RAID** 的 **LVM** 池。

先决条件

•

[您已准备好控制节点和受管节点](#)

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

## 流程

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

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure stripe size for RAID LVM volumes
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_safe_mode: false
        storage_pools:
          - name: my_pool
            type: lvm
            disks: [sdh, sdi]
            volumes:
              - name: my_volume
                size: "1 GiB"
                mount_point: "/mnt/app/shared"
                fs_type: xfs
                raid_level: raid0
                raid_stripe_size: "256 KiB"
                state: present
```

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

2. 验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 **playbook**：



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

## 验证

- 验证条带大小是否已设置为所需的大小：

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

## 其它资源

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

## 2.9. 使用 STORAGE RHEL 系统角色配置 LVM-VDO 卷

您可以使用 **storage** RHEL 系统角色在 LVM (LVM-VDO)上创建一个启用了压缩和重复数据删除的 VDO 卷。



### 注意

由于 存储系统角色 使用 LVM-VDO，每个池只能创建一个卷。

## 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. 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 LVM-VDO volume under volume group 'myvg'
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_pools:
          - name: myvg
            disks:
              - /dev/sdb
            volumes:
              - name: mylv1
                compression: true
                deduplication: true
                vdo_pool_size: 10 GiB
                size: 30 GiB
                mount_point: /mnt/app/shared
```

示例 **playbook** 中指定的设置包括以下内容：

**vdo\_pool\_size:** `<size>`

卷在设备上使用的实际大小。您可以使用人类可读的格式指定大小，如 10 GiB。如果没有指定单位，则默认为字节。

**大小：** `&lt;size>`

VDO 卷的虚拟大小。

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

2. 验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook** :

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

验证

- 

查看压缩和重复数据删除的当前状态：

```
$ ansible managed-node-01.example.com -m command -a 'lvs -
o+vdo_compression,vdo_compression_state,vdo_deduplication,vdo_index_state'
LV   VG   Attr   LSize  Pool  Origin Data%  Meta%  Move Log Cpy%Sync
Convert VDOCompression VDOCompressionState VDODeduplication VDOIndexState
mylv1 myvg vwi-a-v--- 3.00t vpool0                                enabled
online      enabled      online
```

其它资源

- 

[/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file

- 

[/usr/share/doc/rhel-system-roles/storage/](#) directory

## 2.10. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密卷

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]

...

**其它资源**

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md file](#)
- [/usr/share/doc/rhel-system-roles/storage/ directory](#)
- [使用 LUKS 加密块设备](#)
- [Ansible vault](#)

**2.11. 使用 STORAGE RHEL 系统角色创建共享的 LVM 设备**

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

这可带来以下显著优点：

- 资源共享
- 管理存储资源方面的灵活性

- 存储管理任务的简化

#### 先决条件

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

#### 流程

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

```
---
- name: Manage local storage
  hosts: managed-node-01.example.com
  become: true
  tasks:
    - name: Create shared LVM device
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.storage
      vars:
        storage_pools:
          - name: vg1
            disks: /dev/vdb
            type: lvm
            shared: true
            state: present
            volumes:
              - name: lv1
                size: 4g
                mount_point: /opt/test1
            storage_safe_mode: false
            storage_use_partitions: true
```

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

2.

验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 **playbook**：

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

#### 其它资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file
- [/usr/share/doc/rhel-system-roles/storage/](#) directory



## 第 3 章 使用 WEB 控制台管理分区

了解如何使用 web 控制台管理 RHEL 8 上的文件系统。

### 3.1. 在 WEB 控制台中显示使用文件系统格式化的分区

Web 控制台中的 **Storage** 部分会在 **Filesystems** 表中显示所有可用文件系统。

除了使用文件系统格式化的分区的列表外，您还可以使用页来创建新存储。

#### 先决条件

- **cockpit-storaged** 软件包已安装在您的系统上。
- 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](#)。

#### 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. 点 **Storage** 选项卡。

在 **Storage** 表中，您可以看到使用文件系统格式化的所有可用分区、其 ID、类型、位置、大小以及每个分区中有多少可用空间。

Storage

ID	Type	Locati...	Size	
sr0 - QEMU DVD-ROM ...	Media drive			
vda - VirtIO Disk	DOS partitions		16.1 GB	
vda1	xfs filesystem	/boot	0.28 / 1.0 GB	
vda2	LVM2 physical volume (encrypted)	rhel	15 / 15 GB	
rhel	LVM2 logical volumes		15.0 GB	
root	xfs filesystem	/	4.7 / 13 GB	
swap	Swap		1.61 GB	

您还可以使用右上角的下拉菜单来创建新的本地或网络存储。

Storage

ID	Type	Locat	
sr0 - QEMU DVD-ROM ...	Media drive		
vda - VirtIO Disk	DOS partitions		
vda1	xfs filesystem	/boot	
vda2	LVM2 physical volume (encrypted)	rhel	
rhel	LVM2 logical volumes		
root	xfs filesystem	/	
swap	Swap		

Local storage

Create MDRAID device

Create LVM2 volume group

Create Stratis pool

Networked storage

New NFS mount

Change iSCSI initiator name

Add iSCSI portal

3.2. 在 WEB 控制台中创建分区

### 创建新分区：

- 使用现有的分区表
- 创建分区

### 先决条件

- **cockpit-storaged** 软件包已安装在您的系统上。
- 必须安装并可以访问 **Web 控制台**。详情请参阅[安装 Web 控制台](#)。
- 在 **Storage** 选项卡的 **Storage** 表中可以看到一个连接到系统的未格式化的卷。

### 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. 点 **Storage** 选项卡。
3. 在 **Storage** 表中，点您要进行分区的设备，以打开用于该设备的页和选项。
4. 在设备页面中，点菜单按钮 **⋮**，然后选择 **Create partition table**。
5. 在 **Initialize disk** 对话框中选择以下内容：
  - a. **Partitioning**：

- 与所有系统和设备兼容(MBR)
- 与现代系统和硬盘 > 2TB 兼容(GPT)
- 没有分区

b.

**Overwrite :**

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

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

6.

单击 **Initialize**。

7.

点您创建的分区表旁边的菜单按钮 **:**。默认情况下，它被命名为 **Free space**。

8.

单击 **创建分区**。

9.

在 **Create partition** 对话框中输入文件系统 名称。

10.

添加 **Mount point**。

11.

在 **Type** 下拉菜单中选择一个文件系统：

- **XFS** 文件系统支持大的逻辑卷，在不停止工作的情况下在线切换物理驱动器，并可以增大现有的文件系统。如果您没有不同的强大首选项，请保留这个文件系统。

- **ext4 文件系统支持：**
  - 逻辑卷
  - 在不停止工作的情况下在线切换物理驱动器
  - 增大文件系统
  - 缩小文件系统

额外的选项是启用 **LUKS**（Linux 统一密钥设置）完成的分区加密，该加密可让您使用密码短语加密卷。

12. 输入您要创建的卷的大小。

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

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

14. 如果要加密卷，请在 **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**。

15. 在 **At boot** 下拉菜单中选择您要何时挂载卷。

16. 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.

选择 **Custom mount options** 复选框，如果您要更改默认挂载选项，请添加挂载选项。

17.

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

- 

要验证分区是否已成功添加，请切换到 **Storage** 选项卡，并检查 **Storage** 表，并验证是否列出了新分区。

### 3.3. 在 WEB 控制台中删除分区

您可以在 web 控制台界面中删除分区。

先决条件

- 

**cockpit-storaged** 软件包已安装在您的系统上。

- 

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](#)。

## 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.

点 **Storage** 选项卡。

3.

点击您要从中删除分区的设备。

4.

在设备页和 **GPT partitions** 部分中，点您要删除的分区旁的菜单按钮 **⋮**。

5.

从下拉菜单中选择 **Delete**。

RHEL web 控制台终止所有当前使用分区的进程，并在删除分区前卸载分区。

## 验证

•

要验证分区是否已成功删除，请切换到 **Storage** 选项卡，并检查 **Storage** 表。

第 4 章 挂载 NFS 共享

作为系统管理员，您可以在您的系统上挂载远程 NFS 共享来访问共享数据。

4.1. NFS 客户端所需的服务

Red Hat Enterprise Linux 使用内核模块和用户空间进程组合来提供 NFS 文件共享的访问。`nfs-utils` 软件包为用户空间进程提供程序文件。安装 `nfs-utils` 软件包以启用 NFS 客户端功能。NFS 客户端使用的主要服务包括：

表 4.1. NFS 客户端所需的服务

服务名称	NFS 版本	描述
nfsidmap	4	<p>在 NFSv4 名称（以 <code>&lt;user@domain&gt;</code> 的形式为 <code>&lt;user@domain&gt;</code>）和本地用户和组群 ID 的格式，从 NFSv4 客户端映射中启动调用的程序。它提供 <code>rpc.idmapd</code> 代表 NFSv4 服务器提供的类似功能。区别在于，当 <code>rpc.idmapd</code> 是一个守护进程时，<code>nfsidmap</code> 会通过内核请求密钥机制按需调用。<code>nfsidmap</code> 使用两个配置文件：<code>/etc/idmapd.conf</code> 和 <code>/etc/request-key.d/id_resolver.conf</code>。在大多数情况下，默认值已足够，不需要修改其中任何一个配置文件。</p>
rpc.statd	3	<p>实现 Network Status Monitor 协议的守护进程。<code>rpc.statd</code> 的两个主要功能：</p> <ul style="list-style-type: none"><li>• 侦听来自本地锁定进程（实现网络锁定管理器协议的内核守护进程）的请求，以监控网络对等点（如果是 NFS 客户端，<code>rpc.statd</code> 则监控 NFS 服务器）。</li><li>• 侦听来自远程同级（已重启的 NFS 服务器）的重启通知，然后转发到锁定，以便它可以从这些服务器回收所有锁定。</li></ul> <p>使用 <code>/etc/nfs.conf</code> 文件中的 <code>[statd]</code> 部分来配置 <code>rpc.statd</code>。</p>



服务名称	NFS 版本	描述
rpc-statd.service	3	启动 <b>rpc.statd</b> 守护进程的 <b>systemd</b> 单元文件。请注意，不需要手动启用或禁用服务，因为 <b>mount.nfs</b> 程序将在第一次使用 NFSv3 挂载远程文件系统时自动启动 <b>rpc-statd.service</b> （通过 <b>/usr/sbin/start-statd</b> shell 脚本）但是，如果将 NFSv3 客户端配置为在防火墙后面运行，则通常需要重新启动 <b>rpc-statd.service</b> 。
sm-notify	3	当本地系统重启时，向由 <b>rpc.statd</b> 监控的远程对等点发送重启通知的帮助程序。如果是 NFS 客户端， <b>sm-notify</b> 将重启通知发送到 NFS 服务器，以便这些服务器可以丢弃客户端持有的任何锁定。
rpc-statd-notify.service	3	触发 <b>sm-notify</b> 的 <b>systemd</b> 单元。它在系统启动时自动运行，因此不需要手动启用或启动该服务。
rpc.gssd	3, 4	代表内核使用远程对等（通常是从 NFS 客户端到 NFS 服务器启动）的通用安全服务(GSS)上下文的守护进程，但在 NFSv4 回调时从 NFS 服务器启动到 NFS 客户端。此过程对于使用 Kerberos V5 保护 NFS 来说是必要的。 <b>rpc.gssd</b> 程序通过 <b>/etc/nfs.conf</b> 文件中的 <b>[gssd]</b> 部分进行配置。
rpc-gssd.service	3, 4	启动 <b>rpc.gssd</b> 守护进程的 <b>systemd</b> 单元文件。不需要手动启用或禁用该服务，因为如果系统上存在 <b>/etc/krb5.keytab</b> 文件，服务会在系统引导时自动启动。

其它资源

- **nfsidmap (8), rpc.statd (8), sm-notify (8), rpc.gssd (8), 和 nfs.conf (5) man page**

4.2. 准备 NFSV3 客户端以在防火墙后面运行

NFS 服务器通知客户端有关文件锁定和服务状态。要建立回客户端的连接，您必须在客户端的防火墙中打开相关端口。

流程

1. 默认情况下，NFSv3 RPC 服务使用随机端口。要启用防火墙配置，请在 **/etc/nfs.conf** 文件中配置固定端口号：
  - a. 在 **[lockd]** 部分中，为 **nlockmgr** RPC 服务设置固定端口号，例如：

```
port=5555
```

■

使用这个设置时，服务会自动将这个端口号用于 UDP 和 TCP 协议。

b.

在 `[statd]` 部分中，为 `rpc.statd` 服务设置固定端口号，例如：

```
port=6666
```

使用这个设置时，服务会自动将这个端口号用于 UDP 和 TCP 协议。

2.

在 `firewalld` 中打开相关端口：

```
# firewall-cmd --permanent --add-service=rpc-bind
# firewall-cmd --permanent --add-port={5555/tcp,5555/udp,6666/tcp,6666/udp}
# firewall-cmd --reload
```

3.

重启 `rpc-statd` 服务：

```
# systemctl restart rpc-statd nfs-server
```

#### 4.3. 准备 NFSV4 客户端以在防火墙后面运行

NFS 服务器通知客户端有关文件锁定和服务状态。要建立回客户端的连接，您必须在客户端的防火墙中打开相关端口。



#### 注意

NFS v4.1 及之后的版本使用预先存在的客户端端口作为回调，因此无法单独设置回调端口。如需更多信息，请参阅 [如何将 NFS4 客户端回调端口设置为特定端口？](#)

#### 先决条件

●

服务器使用 NFS 4.0 协议。

#### 流程

- 在 `firewalld` 中打开相关端口：

```
# firewall-cmd --permanent --add-port=<callback_port>/tcp
# firewall-cmd --reload
```

#### 4.4. 手动挂载 NFS 共享

如果您不要求在启动时自动挂载 NFS 共享，您可以手动挂载它。



#### 警告

如果您的 NFS 客户端有相同的短主机名，您可能会在 NFSv4 clientid 中遇到冲突，以及其突然过期。为避免 NFSv4 clientid 任何可能的突然过期，您必须为 NFS 客户端使用唯一主机名，或者在每个容器上配置标识符，具体取决于您在使用什么系统。如需更多信息，请参阅红帽知识库解决方案 [NFSv4 clientid 已因为几个 NFS 客户端上使用相同的主机名而突然过期](#)。

#### 流程

- 使用以下命令在客户端上挂载 NFS 共享：

```
# mount <nfs_server_ip_or_hostname>:/<exported_share> <mount point>
```

例如，要将 `/nfs/projects` 共享从 `server.example.com` NFS 服务器挂载到 `/mnt`，请输入：

```
# mount server.example.com:/nfs/projects/ /mnt/
```

#### 验证

- 作为具有访问 NFS 共享权限的用户，显示挂载的共享的内容：

```
$ ls -l /mnt/
```

#### 4.5. 系统引导时自动挂载 NFS 共享

在系统引导过程中自动挂载 **NFS** 共享可确保关键服务依赖于中央数据，如在 **NFS** 服务器上托管的 **/home** 目录，在系统启动时具有无缝且不间断的访问。

## 流程

1.

编辑 **/etc/fstab** 文件，并为您要挂载的共享添加一行：

```
<nfs_server_ip_or_hostname>:/<exported_share> <mount point> nfs default 0 0
```

例如，要将 **/nfs/home** 共享从 **server.example.com** **NFS** 服务器挂载到 **/home**，请输入：

```
server.example.com:/nfs/projects /home nfs defaults 0 0
```

2.

挂载共享：

```
# mount /home
```

## 验证

- 

作为具有访问 **NFS** 共享权限的用户，显示挂载的共享的内容：

```
$ ls -l /mnt/
```

## 其它资源

- 

系统中 **fstab** (5) 手册页

## 4.6. 在 WEB 控制台中连接 NFS 挂载

使用 **NFS** 将远程目录连接到文件系统。

## 先决条件

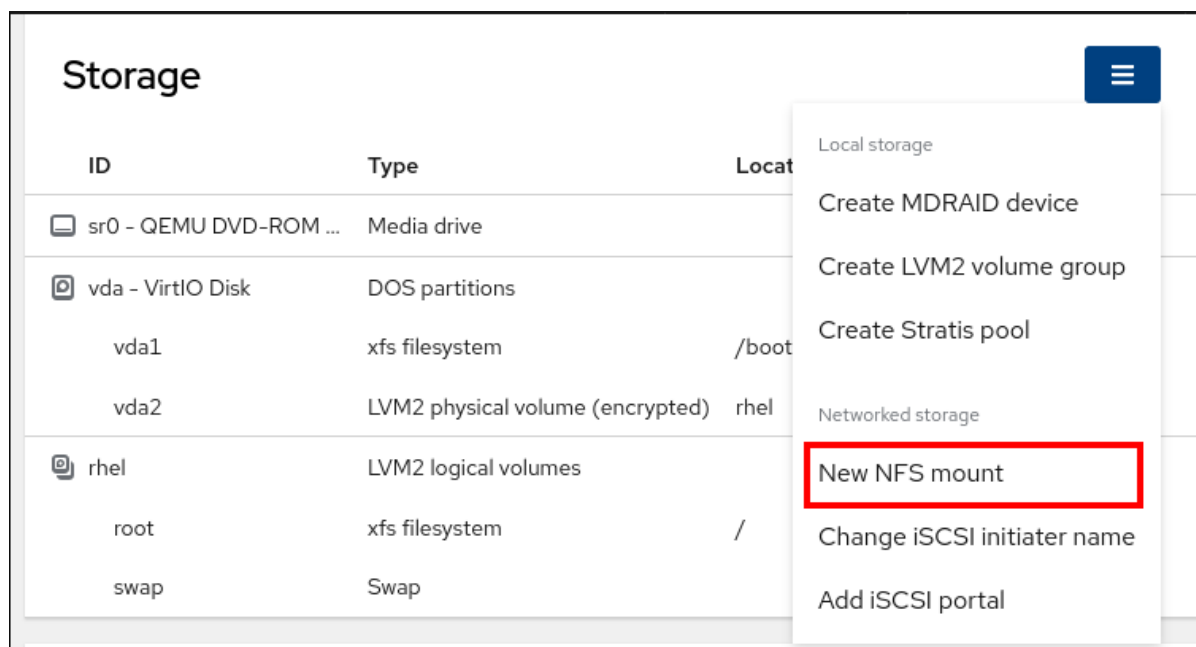
- 

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](#)。
- **cockpit-storaged** 软件包已安装在您的系统上。
- **NFS** 服务器名称或 IP 地址。
- 到远程服务器中的目录的路径。

## 流程

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



5. 在新建 **NFS Mount** 对话框中输入远程服务器的服务器或者 IP 地址。
6. 在 **Path on Server** 字段中，输入您要挂载的目录的路径。
7. 在 **Local Mount Point** 字段中输入您要在本地系统上挂载 **NFS** 的目录的路径。
8. 在 **Mount options** 复选框列表中，选择如何挂载 **NFS**。您可以根据要求选择多个选项。
  - 如果您希望在重启本地系统后仍然可以访问目录，请选中 **Mount at boot** 框。
  - 如果您不想更改 **NFS** 的内容，请选中 **Mount read only** 框。
  - 选中 **Custom mount options** 框，如果您要更改默认挂载选项，请添加挂载选项。

### New NFS mount

Server address	<input type="text" value="fileserver.example.com"/>
Path on server	<input type="text" value="/volume1/videotutorials"/>
Local mount point	<input type="text" value="/mnt/tutorials"/>
Mount options	<div><input checked="" type="checkbox"/> Mount at boot <input checked="" type="checkbox"/> Mount read only <input type="checkbox"/> Custom mount options</div>

9. 点Add。

验证

- 打开挂载的目录，并验证内容可以访问。

#### 4.7. 在 WEB 控制台中自定义 NFS 挂载选项

编辑现有 NFS 挂载并添加自定义挂载选项。

自定义挂载选项可帮助您排除 NFS 挂载的连接或更改参数，如更改超时限制或配置验证。

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。
- **NFS** 挂载被添加到您的系统上。

## 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，点您要调整的 **NFS** 挂载。
4. 如果挂载了远程目录，点 **Unmount**。

您必须在自定义挂载选项配置过程中卸载目录。否则，**Web 控制台**不会保存配置，这会导致错误。

5. 点 **Edit**。
6. 在 **NFS Mount** 对话框中，选择自定义挂载选项。
7. 输入用逗号分开的挂载选项。例如：

- **nfsvers=4** : NFS 协议版本号
- **soft**: NFS 请求超时后恢复的类型



- **sec=krb5** : NFS 服务器上的文件可以通过 Kerberos 身份验证进行保护。NFS 客户端和服务器都必须支持 Kerberos 验证。

如需 NFS 挂载选项的完整列表，请在命令行中输入 `man nfs`。

8. 点 **Apply**。

9. 点 **Mount**。

验证

- 打开挂载的目录，并验证内容可以访问。

#### 4.8. 在 RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT 域中使用 KERBEROS 设置 NFS 客户端

如果 NFS 服务器使用 Kerberos 并注册到 Red Hat Enterprise Linux Identity Management (IdM) 域中，则您的客户端也必须是域的成员才能挂载共享。这可让您集中管理用户和组，并使用 Kerberos 进行身份验证、完整性保护和流量加密。

先决条件

- NFS 客户端已在 Red Hat Enterprise Linux Identity Management (IdM) 域中 [注册](#)。
- 导出的 NFS 共享使用 Kerberos。

流程

1. 以 IdM 管理员身份获取 kerberos 票据：

```
# kinit admin
```

2.

检索主机主体，并将其存储在 `/etc/krb5.keytab` 文件中：

```
# ipa-getkeytab -s idm_server.idm.example.com -p host/nfs_client.idm.example.com -k
/etc/krb5.keytab
```

在将主机加入到 IdM 域时，IdM 会自动创建 host 主体。

3.

可选：显示 `/etc/krb5.keytab` 文件中的主体：

```
# klist -k /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
-----
 6 host/nfs_client.idm.example.com@IDM.EXAMPLE.COM
 6 host/nfs_client.idm.example.com@IDM.EXAMPLE.COM
 6 host/nfs_client.idm.example.com@IDM.EXAMPLE.COM
 6 host/nfs_client.idm.example.com@IDM.EXAMPLE.COM
```

4.

使用 `ipa-client-automount` 工具配置 IdM ID 的映射：

```
# ipa-client-automount
Searching for IPA server...
IPA server: DNS discovery
Location: default
Continue to configure the system with these values? [no]: yes
Configured /etc/idmapd.conf
Restarting sssd, waiting for it to become available.
Started autofs
```

5.

挂载导出的 NFS 共享，例如：

```
# mount -o sec=krb5i server.idm.example.com:/nfs/projects/ /mnt/
```

`-o sec` 选项指定 Kerberos 安全方法。

## 验证

1.

以具有写入挂载的共享的 IdM 用户身份登录。

2.

获取Kerberos ticket：

```
$ kinit
```

3.

在共享中创建文件，例如：

```
$ touch /mnt/test.txt
```

4.

列出目录以验证该文件是否已创建：

```
$ ls -l /mnt/test.txt
-rw-r--r--. 1 admin users 0 Feb 15 11:54 /mnt/test.txt
```

其它资源

[AUTH\\_GSS 身份验证方法](#)

#### 4.9. 配置 GNOME 以将用户设置存储在由 NFS 共享托管的主目录中

如果您在托管在 NFS 服务器上的主目录的系统中使用 GNOME，您必须更改 dconf 数据库的 keyfile 后端。否则，dconf 可能无法正常工作。

这个更改会影响主机上的所有用户，因为它更改了 dconf 管理存储在主目录中的用户设置和配置。

请注意，只有安装了 glib2-fam 软件包时，dconf keyfile 后端 才能工作。如果没有此软件包，则在远程机器上配置更改的通知无法正确显示。

在 Red Hat Enterprise Linux 8 中，glib2-fam 软件包在 BaseOS 存储库中提供。

先决条件

- 

已安装 glib2-fam 软件包：

```
# yum install glib2-fam
```

■

## 流程

1. 将以下行添加到 `/etc/dconf/profile/user` 文件的开头。如果文件不存在，则进行创建。

```
service-db:keyfile/user
```

使用这个设置时，**dconf** 会轮询 **keyfile** 后端 来确定是否进行了更新，因此可能无法立即更新设置。

2. 当用户退出和登录时，这些更改将生效。

## 4.10. 常用的 NFS 挂载选项

以下是挂载 **NFS** 共享时常用的选项。您可以在 `/etc/fstab` 设置和 **autofs automapper** 中对 **mount** 命令使用这些选项。

### **lookupcache=mode**

指定内核应该如何管理给定挂载点的目录条目缓存。**mode** 的有效参数为 **、none** 或 **positive**。

### **nfsvers=version**

指定要使用的 **NFS** 协议版本，其中 **version** 为 **3、4、4.0、4.1** 或 **4.2**。这对于运行多个 **NFS** 服务器的主机很有用，或者禁止使用较低版本重试挂载。如果没有指定版本，客户端首先会尝试版本 **4.2**，然后进行协商，直到找到服务器支持的版本。

选项 **vers** 等同于 **nfsvers**，出于兼容性的原因包含在此发行版本中。

### **noacl**

关闭所有 **ACL** 处理。当与最新 **ACL** 技术不兼容的旧 **Red Hat Enterprise Linux** 版本交互时，可能需要这样做。

### **nolock**

禁用文件锁定。当您连接到非常旧的 **NFS** 服务器时，可能需要此设置。

### **noexec**

防止在挂载的文件系统中执行二进制文件。这在系统挂载不兼容二进制文件的非 Linux 文件系统时有用。

#### **nosuid**

禁用 **set-user-identifier** 和 **set-group-identifier** 位。这可防止远程用户通过运行 **setuid** 程序获得更高的特权。

#### **retrans=num**

NFS 客户端在尝试进一步恢复操作前重试请求的次数。如果没有指定 **retrans** 选项，NFS 客户端会尝试每个 UDP 请求三次，每个 TCP 请求两次。

#### **timeo=num**

NFS 客户端在重试 NFS 请求前等待响应的第二个时间（以十秒为单位）。对于 TCP 上的 NFS，默认的 **timeo** 值为 600 (60 秒)。NFS 客户端执行线性 **backoff**：在每次重新传输超时后，会延长到最多 600 秒。

#### **port=num**

指定 NFS 服务器端口的数字值。对于 NFSv3，如果 **num** 为 0（默认值），或未指定，则 **mount** 会查询远程主机上的 **rpcbind** 服务，以获取要使用的端口号。对于 NFSv4，如果 **num** 为 0，则 **mount** 会查询 **rpcbind** 服务，但如果未指定，则使用标准 NFS 端口号 TCP 2049，并且不再检查远程 **rpcbind**。

#### **rsize=num** 和 **wsiz=num**

这些选项设定单一 NFS 读写操作传输的最大字节数。

**rsiz** 和 **wsiz** 没有固定的默认值。默认情况下，NFS 使用服务器和客户端都支持的最大的可能值。在 Red Hat Enterprise Linux 9 中，客户端和服务器的最大值为 1,048,576 字节。如需更多信息，请参阅红帽知识库解决方案 [使用 NFS 挂载的默认和最大值？](#)

#### **sec=options**

用于访问挂载导出上文件的安全选项。**options** 值是一个或多个安全选项的冒号分隔列表。

默认情况下，客户端会尝试查找客户端和服务器都支持的安全选项。如果服务器不支持任何所选选项，挂载操作会失败。

可用选项：

- **sec=sys** 使用本地 UNIX UID 和 GID。它们使用 AUTH\_SYS 验证 NFS 操作。
- **sec=krb5** 使用 Kerberos V5 ,而不是本地 UNIX UID 和 GID 来验证用户。
- **sec=krb5i** 使用 Kerberos V5 进行用户身份验证, 并使用安全校验和执行 NFS 操作的完整性检查, 以防止数据被篡改。
- **sec=krb5p** 使用 Kerberos V5 进行用户身份验证、完整性检查, 并加密 NFS 流量以防止流量嗅探。这是最安全的设置, 但它也会涉及最大的性能开销。

#### 其它资源

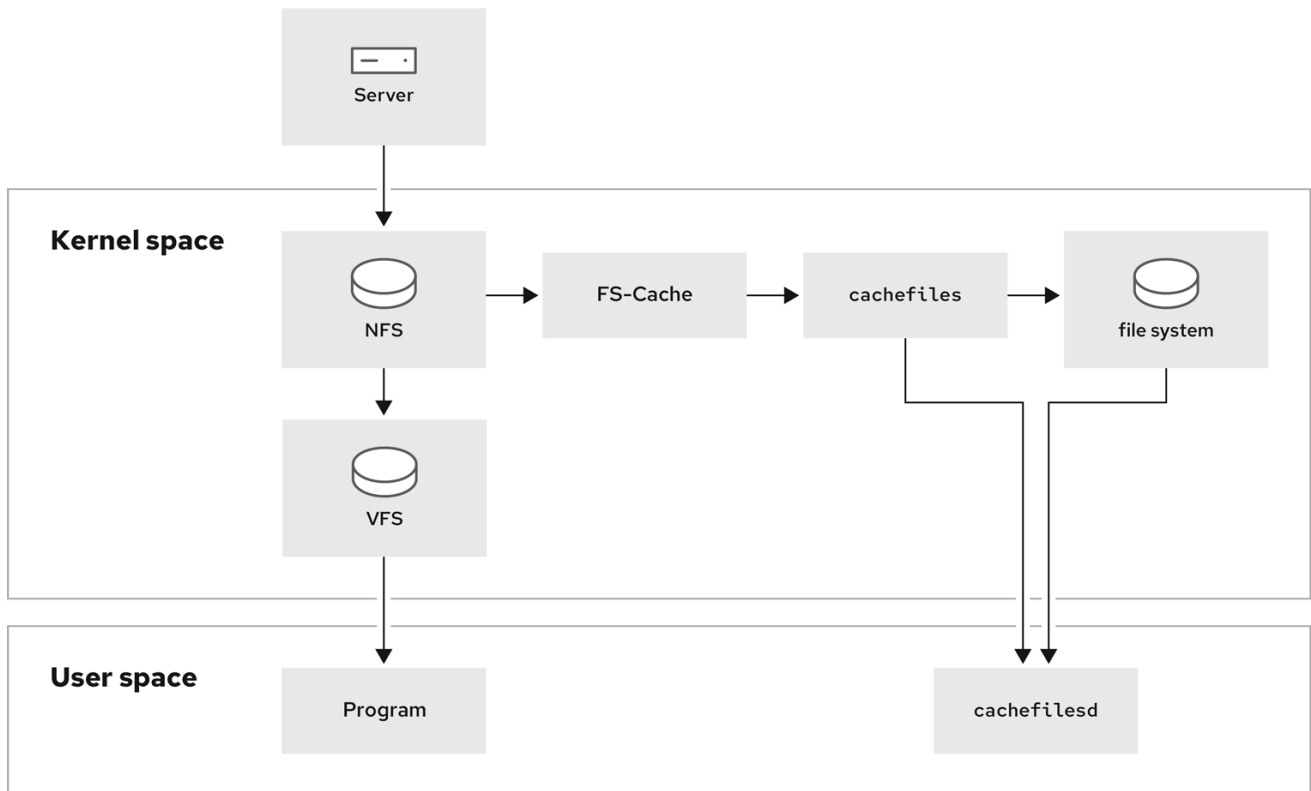
- 在您的系统上 mount (8) 和 'nfs (5)'man page

### 4.11. 启用 NFS 内容的客户端缓存

**FS-Cache** 是客户端上的持久本地缓存, 文件系统可以使用它通过网络检索数据并将其缓存在本地磁盘上。这有助于最小化网络流量。

#### 4.11.1. NFS 缓存如何工作

下图显示了 FS-Cache 的工作原理：



478\_RHEL\_0824

**FS-Cache** 旨在对系统的用户和管理员尽可能透明。**FS-Cache** 允许服务器上的文件系统直接与客户端的本地缓存交互，而无需创建过度挂载的文件系统。使用 **NFS** 时，挂载选项指示客户端挂载启用了 **FS-cache** 的 **NFS** 共享。挂载点将导致两个内核模块的自动上传：**fscache** 和 **cachefile**。**cachefilesd** 守护进程与内核模块进行通信来实施缓存。

**FS-Cache** 不会改变通过网络工作的文件系统的基本操作。它只是为文件系统提供一个永久位置，以便可以缓存数据。例如，客户端仍然可以挂载 **NFS** 共享，无论是否启用了 **FS-Cache**。此外，缓存的 **NFS** 可以处理不能全部放入缓存的文件（无论是单独的还是总体的），因为文件可以部分缓存，且不必预先完全读取。**FS-Cache** 还会隐藏发生在客户端文件系统驱动程序的缓存中的所有 I/O 错误。

要提供缓存服务，**FS-Cache** 需要一个缓存后端，即 **cachefiles** 服务。**FS-Cache** 需要挂载的基于块的文件系统，它支持块映射(**bmap**)和扩展属性作为其缓存后端：

- **XFS**
- **ext3**

- **ext4**

**FS-Cache** 无法任意缓存任何文件系统，无论是通过网络还是通过其他方式：必须更改共享文件系统的驱动程序，以允许与 **FS-Cache**、数据存储或检索以及元数据设置和验证进行交互。**FS-Cache** 需要来自缓存文件系统的 **索引密钥** 和 **一致性数据** 来支持持久性：使用索引密钥匹配文件系统对象来缓存对象，使用一致性数据来确定缓存对象是否仍然有效。

使用 **FS-Cache** 是各种因素之间的折衷。如果使用 **FS-Cache** 缓存 **NFS** 流量，则客户端可能会减慢，但可以通过满足本地的读取请求来大规模减少网络和服务器的加载，而无需消耗网络带宽。

#### 4.11.2. 安装和配置 **cachefilesd** 服务

Red Hat Enterprise Linux 只提供 **cachefiles** 缓存后端。**cachefilesd** 服务启动和管理 **cachefiles**。**/etc/cachefilesd.conf** 文件控制 **cachefile** 如何提供缓存服务。

##### 先决条件

- 挂载到 **/var/cache/fscache/** 目录下的文件系统是 **ext3**、**ext4** 或 **xfs**。
- 挂载到 **/var/cache/fscache/** 下的文件系统使用扩展属性，如果您在 **RHEL 8** 或更高版本上创建文件系统，这是默认设置。

##### 流程

1.  
安装 **cachefilesd** 软件包：  

```
# dnf install cachefilesd
```
2.  
启用并启动 **cachefilesd** 服务：  

```
# systemctl enable --now cachefilesd
```

##### 验证



1.

使用 **fsc** 选项挂载 NFS 共享以使用缓存：

a.

要临时挂载共享，请输入：

```
# mount -o fsc server.example.com:/nfs/projects/ /mnt/
```

b.

要永久挂载共享，请在 `/etc/fstab` 文件中的条目中添加 **fsc** 选项：

```
<nfs_server_ip_or_hostname>:/<exported_share> <mount point> nfs fsc 0 0
```

2.

显示 **FS-cache** 统计信息：

```
# cat /proc/fs/fscache/stats
```

#### 其它资源

- `/usr/share/doc/cachefilesd/README` 文件
- `/usr/share/doc/kernel-doc-<kernel_version>/Documentation/filesystems/caching/fscache.txt` 由 `kernel-doc` 软件包提供

#### 4.11.3. 共享 NFS 缓存

因为缓存是持久的，所以缓存中的数据块会根据由四个键组成的序列来索引的：

- 第 1 级：服务器详情
- 第 2 级：有些挂载选项；安全类型；FSID；uniquifier 字符串
- 第 3 级：文件处理

## 第 4 级：文件中的页号

为避免超级块之间一致性管理问题，需要缓存数据的所有 NFS 超级块都有唯一的 2 级密钥。通常，使用同一源卷和选项的两个 NFS 挂载共享超级块，因此共享缓存，即使它们在该卷中挂载不同的目录。

### 例 4.1. NFS 缓存共享：

以下两个挂载可能会共享超级块，因为它们具有相同的挂载选项，特别是因为它们来自 NFS 服务器上的同一分区：

```
# mount -o fsc home0:/nfs/projects /projects
# mount -o fsc home0:/nfs/home /home/
```

如果挂载选项不同，它们不共享超级块：

```
# mount -o fsc,rsiz=8192 home0:/nfs/projects /projects
# mount -o fsc,rsiz=65536 home0:/nfs/home /home/
```

### 注意

用户不能在具有不同通信或协议参数的超级块之间共享缓存。例如，无法在 NFSv4.0 和 NFSv3 之间或在 NFSv4.1 和 NFSv4.2 之间共享缓存，因为它们强制执行不同的超级块。另外，设置读取大小(rsiz)等参数可防止缓存共享，因为它会强制使用不同的超级块。

#### 4.11.4. NFS 缓存限制

NFS 有一些缓存限制：

- 为直接 I/O 打开共享文件系统的文件将自动绕过缓存。这是因为这种访问类型必须与服务器直接进行。
- 从共享文件系统打开一个文件直接 I/O 或写入清除文件缓存的副本。FS-Cache 不会再次缓存文件，直到它不再为直接 I/O 或写操作而打开。

- 另外，FS-Cache 的这个发行版本只缓存常规 NFS 文件。FS-Cache 不会缓存目录、符号链接、设备文件、FIFO 和套接字。

#### 4.11.5. 缓存剔除如何工作

**cachefilesd** 服务的工作原理为：缓存来自共享文件系统的远程数据，以释放本地磁盘上的空间。这可能会消耗所有可用空间，如果磁盘还包含根分区，这可能会导致问题。为控制这一点，**cachefilesd** 会尝试通过丢弃不集中访问的对象（如从缓存中访问）来维护一定数量的可用空间。这个行为被称为 **cache culling**。

缓存筛选是根据底层文件系统中可用块的百分比以及可用文件的百分比来实现的。**/etc/cachefilesd.conf** 中有控制六个限制的设置：

**brun *N*%**(块百分比)、**frun *N*%**(文件百分比)

如果缓存中空闲空间的数量和可用文件的数量超过这两个限制，则关闭筛选。

**bcull *N*%**(块百分比)、**fcull *N*%**(文件百分比)

如果缓存中可用空间的数量或文件的数量低于其中任何一个限制，则启动筛选。

**bstop *N*%**(块百分比)、**fstop *N*%**(文件百分比)

如果缓存中可用空间的数量或可用文件的数量低于其中任何一个限制，则不允许进一步分配磁盘空间或文件，直到筛选再次引发超过这些限制的情况。

每个设置的 **N** 的默认值如下：

- **brun/frun: 10%**
- **bcull/fcull: 7%**
- **bstop/fstop: 3%**

在配置这些设置时，必须满足以下条件：

- $0 \text{ wagon bstop} < \text{bcull} < \text{brun} < 100$
- $0 \text{ wagon fstop} < \text{fcull} < \text{frun} < 100$

这些是可用空间和可用文件的百分比，不会显示为 100 减去 `df` 程序所显示的百分比。

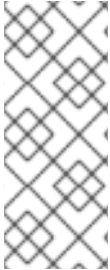


#### 重要

筛选同时依赖于 `bxxx` 和 `fxxx` 对；用户不能单独处理它们。

## 第 5 章 挂载一个 SMB 共享

服务器消息块(SMB)协议实现用于访问服务器上资源的应用层网络协议，如文件共享和共享打印机。



### 注意

在 SMB 的上下文中，您可以发现提到了通用 Internet 文件系统(CIFS)协议，该协议是 SMB 的一种方言。SMB 和 CIFS 协议都支持，并且挂载 SMB 和 CIFS 共享时所涉及的内核模块和工具均使用名称 `cifs`。

`cifs-utils` 软件包为以下情况提供工具：

- 挂载 SMB 和 CIFS 共享
- 管理内核 keyring 中的 NT LAN Manager (NTLM)凭据
- 在 SMB 和 CIFS 共享上的安全描述符中设置和显示访问控制列表(ACL)

### 5.1. 支持的 SMB 协议版本

`cifs.ko` 内核模块支持以下 SMB 协议版本：

- SMB 1

**警告**

因为已知的安全问题，**SMB1** 协议已弃用，仅 在私有网络上可以安全使用。**SMB1** 仍然作为受支持的选项提供，其主要原因是，当前它是唯一支持 **UNIX** 扩展的 **SMB** 协议版本。如果您不需要在 **SMB** 上使用 **UNIX** 扩展，红帽强烈建议您使用 **SMB2** 或更高版本。

- **SMB 2.0**
- **SMB 2.1**
- **SMB 3.0**
- **SMB 3.1.1**

**注意**

根据协议版本，并非所有 **SMB** 功能都已实施。

## 5.2. UNIX 扩展支持

**Samba** 在 **SMB** 协议中使用 **CAP\_UNIX** 功能位来提供 **UNIX** 扩展功能。**cifs.ko** 内核模块也支持这些扩展。但是，**Samba** 和内核模块仅支持 **SMB 1** 协议中的 **UNIX** 扩展。

### 先决条件

- **cifs-utils** 软件包已安装。

### 流程

1. 将 `/etc/samba/smb.conf` 文件 [global] 部分中的 `server min protocol` 参数设为 NT1。
2. 通过向 `mount` 命令提供 `-o vers=1.0` 选项，使用 SMB 1 协议来挂载共享。例如：

```
# mount -t cifs -o vers=1.0,username=<user_name> //<server_name>/<share_name>
/mnt/
```

默认情况下，内核模块使用 SMB 2 或服务器支持的最高协议版本。将 `-o vers=1.0` 选项传给 `mount` 命令会强制内核模块使用 SMB 1 协议，该协议在使用 UNIX 扩展时是必需的。

## 验证

- 显示挂载的共享的选项：

```
# mount
...
//<server_name>/<share_name> on /mnt type cifs (...,unix,...)
```

如果在挂载选项列表中显示了 `unix` 条目，则启用了 UNIX 扩展。

## 5.3. 手动挂载 SMB 共享

如果您只需要临时挂载 SMB 共享，您可以使用 `mount` 工具手动挂载它。



### 注意

重启系统时，手动挂载的共享不会再次自动挂载。要在系统引导时将 Red Hat Enterprise Linux 配置为自动挂载共享，请参阅 [在系统引导时自动挂载 SMB 共享](#)。

## 先决条件

- `cifs-utils` 软件包已安装。

## 流程

- 

使用带有 **-t cifs** 参数的 **mount** 工具来挂载 **SMB** 共享：

```
# mount -t cifs -o username=<user_name> //<server_name>/<share_name> /mnt/
Password for <user_name>@//<server_name>/<share_name>: password
```

在 **-o** 参数中，您可以指定用于挂载共享的选项。详情请查看 **mount.cifs(8)** 手册页中的 **OPTIONS** 部分，以及 [常用的挂载选项](#)。

#### 例 5.1. 使用加密的 SMB 3.0 连接挂载共享

要以 **DOMAINAdministrator** 用户的身份，将 **\\server\example\** 共享通过加密的 **SMB 3.0** 连接挂载到 **/mnt/** 目录：

```
# mount -t cifs -o username=DOMAIN\Administrator,seal,vers=3.0 //server/example
/mnt/
Password for DOMAIN\Administrator@//server_name/share_name: password
```

#### 验证

- 

列出挂载的共享的内容：

```
# ls -l /mnt/
total 4
drwxr-xr-x. 2 root root 8748 Dec  4 16:27 test.txt
drwxr-xr-x. 17 root root 4096 Dec  4 07:43 Demo-Directory
```

#### 5.4. 系统启动时自动挂载 **SMB** 共享

如果服务器上需要永久访问挂载的 **SMB** 共享，请在启动时自动挂载共享。

#### 先决条件

- 

**cifs-utils** 软件包已安装。

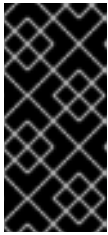
#### 流程



1.

向 `/etc/fstab` 文件中添加一个共享条目。例如：

```
//<server_name>/<share_name> /mnt cifs credentials=/root/smb.cred 0 0
```



#### 重要

要让系统自动挂载共享，您必须将用户名、密码和域名存储在凭据文件中。详情请参阅 [创建一个凭据文件以向 SMB 共享进行身份验证](#)

在 `/etc/fstab` 行的第四个字段中，指定挂载选项，如凭据文件的路径。详情请查看 `mount.cifs(8)` 手册页中的 **OPTIONS** 部分，以及 [常用的挂载选项](#)。

### 验证

•

通过指定挂载点来挂载共享：

```
# mount /mnt/
```

## 5.5. 创建一个凭据文件以对 SMB 共享进行身份验证

在某些情况下，比如在启动时自动挂载共享，应当在不输入用户名和密码的情况下挂载共享。要实施此操作，请创建一个凭据文件。

### 先决条件

•

`cifs-utils` 软件包已安装。

### 流程

1.

创建一个文件，如 `/root/smb.cred`，并指定用户名、密码和域名：

```
username=user_name
password=password
domain=domain_name
```

2.

将权限设置为只允许所有者可以访问该文件：

```
# chown user_name /root/smb.cred
# chmod 600 /root/smb.cred
```

现在，您可以将 `credentials=file_name` 挂载选项传给 `mount` 工具，或者在 `/etc/fstab` 文件中使用它来挂载共享，而无需提示输入用户名和密码。

## 5.6. 执行多用户 SMB 挂载

您为挂载共享提供的凭据默认确定对挂载点的访问权限。例如，如果您在挂载共享时使用 `DOMAINexample` 用户，则共享上的所有操作都将以该用户的身份执行，而不管哪个本地用户执行此操作。

然而，在某些情况下，管理员希望在系统启动时自动挂载共享，但用户应使用他们自己的凭据对共享的内容执行操作。`multiuser` 挂载选项允许您配置此场景。

### 重要

要使用 `multiuser` 挂载选项，还必须将 `sec` 挂载选项设置为支持以非交互方式提供凭据的安全类型，如 `krb5`，或带有凭据文件的 `ntlmssp` 选项。详情请参阅 [以用户身份访问共享](#)。

`root` 用户使用 `multiuser` 选项以及对共享内容具有最少访问权限的帐户挂载共享。然后，常规用户可以使用 `cifscreds` 工具将其用户名和密码提供给当前会话的内核密钥环。如果用户访问挂载的共享的内容，则内核将使用内核密钥环中的凭据，而不是最初用来挂载共享的凭据。

使用此功能包含以下步骤：

- [使用 `multiuser` 选项挂载共享](#)。
- [（可选）验证是否使用 `multiuser` 选项成功挂载了共享](#)。

- [以用户身份访问共享。](#)

#### 先决条件

- **cifs-utils** 软件包已安装。

#### 5.6.1. 使用 **multiuser** 选项挂载共享

在用户可以使用他们自己的凭据访问共享之前，使用权限有限的帐户，以 **root** 用户身份挂载共享。

#### 流程

在系统启动时，使用 **multiuser** 选项自动挂载共享：

1. 在 **/etc/fstab** 文件中为共享创建条目。例如：

```
//server_name/share_name /mnt cifs
multiuser,sec=ntlmssp,credentials=/root/smb.cred 0 0
```

2. 挂载共享：

```
# mount /mnt/
```

如果您不想在系统启动时自动挂载共享，请通过将 **-o multiuser,sec=security\_type** 传给 **mount** 命令来手动挂载它。有关手动挂载 **SMB** 共享的详情，请参考 [手动挂载 \*\*SMB\*\* 共享](#)。

#### 5.6.2. 验证是否使用 **multiuser** 选项挂载 **SMB** 共享

若要验证共享是否是通过 **multiuser** 选项挂载的，可显示挂载选项。

#### 流程

```
# mount
```

```
...
```

```
//server_name/share_name on /mnt type cifs (sec=ntlmssp,multiuser,...)
```

如果挂载选项列表中显示了 **multiuser** 条目，则启用了该功能。

### 5.6.3. 以用户身份访问共享

如果 **SMB** 共享是使用 **multiuser** 选项挂载的，则用户可以向内核密钥环提供其服务器凭据：

```
# cifscreds add -u SMB_user_name server_name
Password: password
```

当用户在包含挂载的 **SMB** 共享的目录中执行操作时，服务器将为此用户应用文件系统权限，而不是挂载共享时最初使用的权限。



#### 注意

多个用户可以同时对挂载的共享使用自己的凭据来执行操作。

## 5.7. 常用的 **SMB** 挂载选项

当您挂载 **SMB** 共享时，挂载选项将决定：

- 如何与服务器建立连接。例如：连接到服务器时使用 **SMB** 协议版本。
- 如何将共享挂载到本地文件系统。例如，如果系统覆盖了远程文件和目录的权限，使多个本地用户能够访问服务器上的内容。

要在 **/etc/fstab** 文件的第四个字段或在 **mount** 命令的 **-o** 参数中设置多个选项，请将它们用逗号分开。例如，请参阅 [使用 \*\*multiuser\*\* 选项挂载共享](#)。

以下列表给出了常用的挂载选项：

选项	描述
<code>credentials=file_name</code>	设置凭证文件的路径。请参阅 <a href="#">使用凭据文件认证到 SMB 共享</a> 。
<code>dir_mode=mode</code>	如果服务器不支持 CIFS UNIX 扩展，则设置目录模式。
<code>file_mode=mode</code>	如果服务器不支持 CIFS UNIX 扩展，则设置文件模式。
<code>password=password</code>	设置在 SMB 服务器中验证的密码。另外，也可使用 <b>credentials</b> 选项指定凭据文件。
<code>seal</code>	使用 SMB 3.0 或更高的协议版本启用对连接的加密支持。因此，使用 <b>seal</b> 和 <b>vers</b> 挂载选项来设置成 <b>3.0</b> 或更高版本。请参阅 <a href="#">手动挂载 SMB 共享</a> 中的示例。
<code>sec=security_mode</code>	<p>设置安全模式，如 <b>ntlmsspi</b>，来启用 NTLMv2 密码哈希和已启用的数据包签名。有关支持值的列表，请查看您系统的 <b>mount.cifs (8)</b> 手册页中的选项描述。</p> <p>如果服务器不支持 <b>ntlmv2</b> 安全模式，则使用 <b>sec=ntlmssp</b>，这是默认值。</p> <p>出于安全考虑，请不要使用不安全的 <b>ntlm</b> 安全模式。</p>
<code>username=user_name</code>	设置在 SMB 服务器中验证的用户名。另外，也可使用 <b>credentials</b> 选项指定凭据文件。
<code>vers=SMB_protocol_version</code>	设定用于与服务器通信的 SMB 协议版本。

有关完整列表，请查看您系统上的 **mount.cifs (8)** 手册页中的 **OPTIONS** 部分。

## 第 6 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。

### 6.1. 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。在使用时，务必要使用正确的选项来识别每个设备，以避免意外访问错误的设备，特别是在安装到或重新格式化驱动器时。

通常，Linux 中使用非持久性名称来表示存储设备，格式为 `/dev/sd (主号) (次号)`。当检测到设备时，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，如果设备检测顺序发生了变化，主号和次号范围之间的关联以及相关 `sd` 名称可能会发生变化。

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。磁盘不能被系统访问，后续的设备将有其主号和次号范围，包括相关的 `sd` 名称会下移。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，从而导致不能检测到与该 **HBA** 连接的所有磁盘。任何连接到随后探测到的 **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。
- 连接到带有光纤通道、**iSCSI** 或 **FCoE** 适配器的系统的磁盘可能在检测存储设备时无法访问，例如，由于存储阵列或中间交换机断电。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机制来将持久性 **SCSI** 目标 ID 指定到 **WWPN** 映射，但这不会保留主号和次号范围，以及相关 `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(例如在 `/etc/fstab` 文件中的)时不希望使用主号和次号范围或相关的 `sd` 名称。可能挂载了错误的设备，并可能导致数据崩溃。

然而，仍然有必要引用 `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 `sd` 名称（以及 SCSI 主机/通道/目标/LUN 元组）。

## 6.2. 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` 工具进行格式化，设备会保留属性，因为它没有存储在文件系统中。

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。

### 6.3. 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，但有一部分也是可以配置的。

#### 6.3.1. 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，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** 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。

### 6.3.2. 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。保证 **WWID** 标识符对于每个存储设备都是唯一的，并且独立于用于访问该设备的路径。标识符是设备的属性，但不存储在设备上的内容（也就是数据）中。

可通过发出 **SCSI** 询问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** 自动维护从基于 **WWID** 的设备名称到该系统上当前 **/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，即使设备的路径有变化，即使从不同的系统访问该设备也一样。

#### 例 6.1. **WWID** 映射

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

除了系统提供的这些持久名称外，您还可以使用 `udev` 规则来实现映射到存储的 WWID 的持久名称。

### `/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 6.2. 分区 UUID 映射

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

### `/dev/disk/by-path/` 中的 Path 属性

此属性通过用于访问该设备的 **硬件路径** 来提供一个指向存储设备的符号链接。

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中安装存储服务。

#### 6.4. 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
- *major:minor* 号

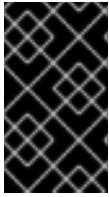
#### 例 6.3. 多路径配置中的 WWID 映射

`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 多路径自动维护每个基于 WWID 的设备名称到系统上相应的 `/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。只要该文件被维护，这些 `mpathN` 名称就会持久存在。



#### 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。

## 6.5. 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设备，因为当为 `udev` 事件处理 `udev` 规则时，`udev` 机制可能依赖于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，并可能导致设备无法访问时，`/dev/disk/by-*/` 链接被删除。
- 在 `udev` 事件产生和处理时，如检测到大量设备，用户空间 `udev` 服务花费一些时间来处理每个事件的规则时，可能会有延迟。这可能会在内核检测到该设备和在 `/dev/disk/by-*/` 名称可用之间出现延迟。
- 规则调用的 `blkid` 等外部程序可能会打开设备一小段时间，从而使设备无法被其他用途访问。
- `/dev/disk/` 中由 `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。

## 6.6. 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 6.4. 查看文件系统的 **UUID** 和标签

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

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

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

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

例如：

#### 例 6.5. 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 6.6. 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-
QIWtEHtXGobe5bewIIUDivKOz5ofkgFhP0RMFsNyySVihqEI2cWWbR7MjXJoID6g
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewIIUDivKOz5ofkgFhXqH2M45hD2H9nAf2qfWSrIRLhzfMyOKd
symbolic link to ../../dm-0
/dev/disk/by-id/lvm-pv-uuid-atlr2Y-vuMo-ueoH-CpMG-4JuH-AhEF-wu4QQm
symbolic link to ../../sda2

```

## 6.7. 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** 属性在后台进行，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** 命令生成一个 **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 7 章 使用 PARTED 的分区操作

**parted** 是操作磁盘分区的程序。它支持多种分区表格式，包括 MS-DOS 和 GPT。创建用于新操作系统的空间、重新组织磁盘使用情况并将数据复制到新的硬盘中非常有用。

### 7.1. 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** 工具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：切换到您要检查下一个设备：

```
(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`。

其它资源

- 您系统上的 `parted` (8) 手册页

## 7.2. 使用 PARTED 在磁盘中创建分区表

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* :

- 

用于的 MBR 的 **msdos**

- 

用于 GPT 的 **gpt**

### 例 7.1. 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
```

#### 其它资源

- [您系统上的 parted \(8\) 手册页](#)

### 7.3. 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 交换、**/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*** 替换为主、逻辑或扩展。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。

- 

使用确定分区的开始和结束点的大小（从磁盘开始计算）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 7.2. 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
```

#### 其它资源

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

## 7.4. 使用 PARTED 删除分区

使用 **parted** 工具，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的次号：

```
(parted) print
```

3.

删除分区：

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

- 

使用您要删除的分区的副号码替换 ***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的行，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
```

## 其它资源

•

您系统上的 `parted` (8) 手册页

## 7.5. 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，或者缩小分区以将容量用于不同目的。

## 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：如果您扩展了分区，也扩展了它上面的文件系统。

## 其它资源

- 您系统上的 **parted (8)** 手册页

第 8 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。
- 活跃使用的分区中的空闲空间可用。



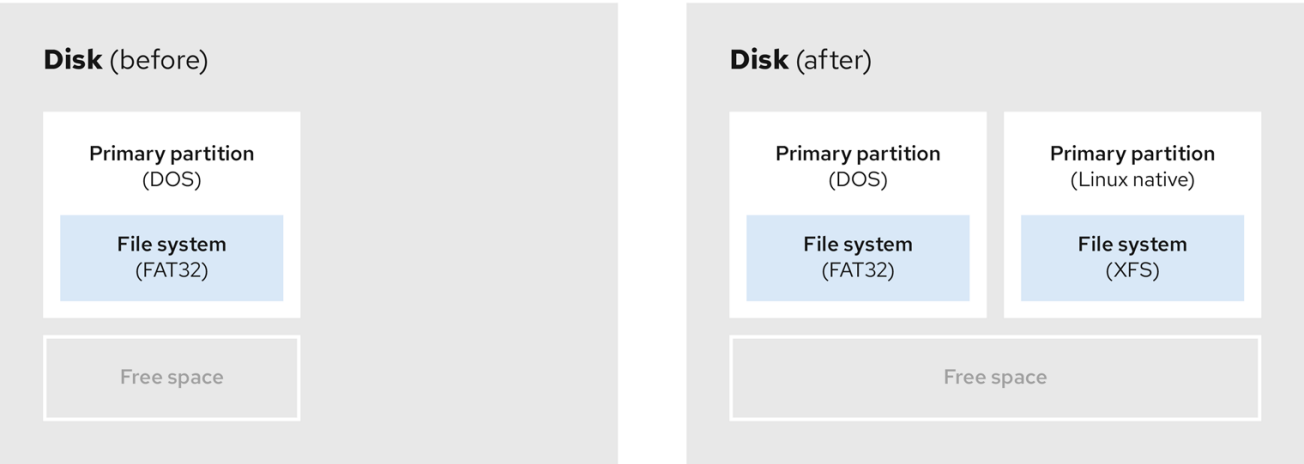
注意

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。

8.1. 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的样子。

图 8.1. 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。

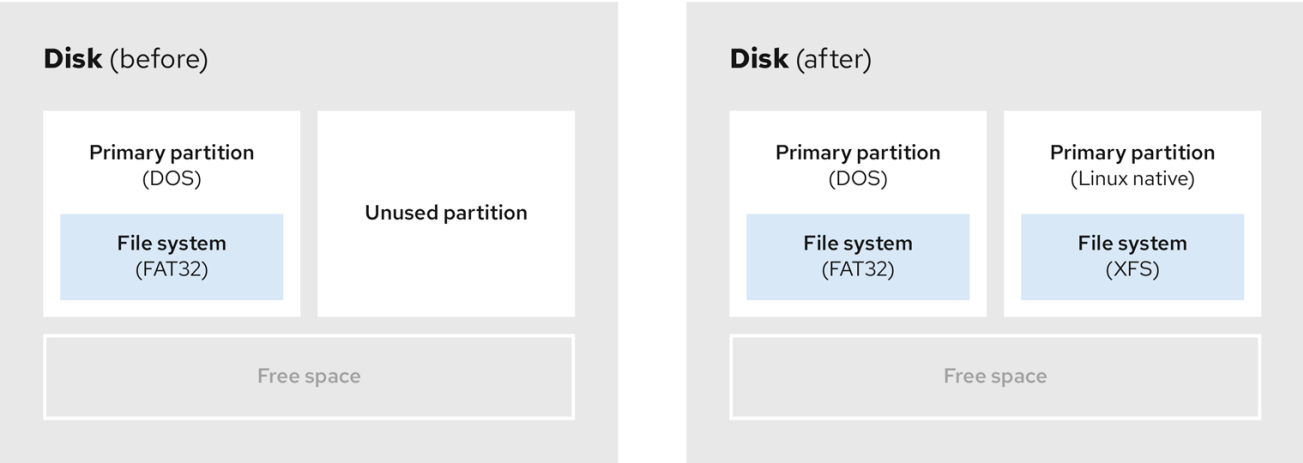
未使用的硬盘也属于这一类别。唯一的区别是，*所有* 空间并不是任何定义的分区的部分。

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。

8.2. 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 8.2. 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。

8.3. 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，一些计算机包含预安装的软件，不包括安装介质以重新安装原始操作系统。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。

### 8.3.1. 破坏性重新分区

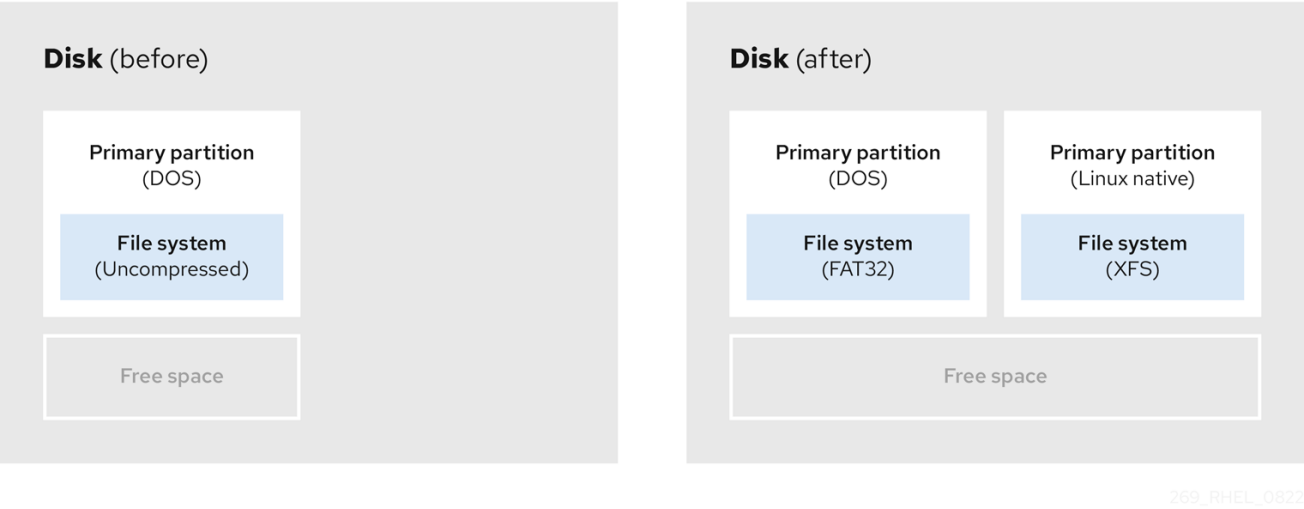
破坏性重新分区破坏硬盘中的分区并创建几个较小的分区。从原始分区备份所有需要的数据，因为此方法会删除完整内容。

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 8.3. 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。

**8.3.2. 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。

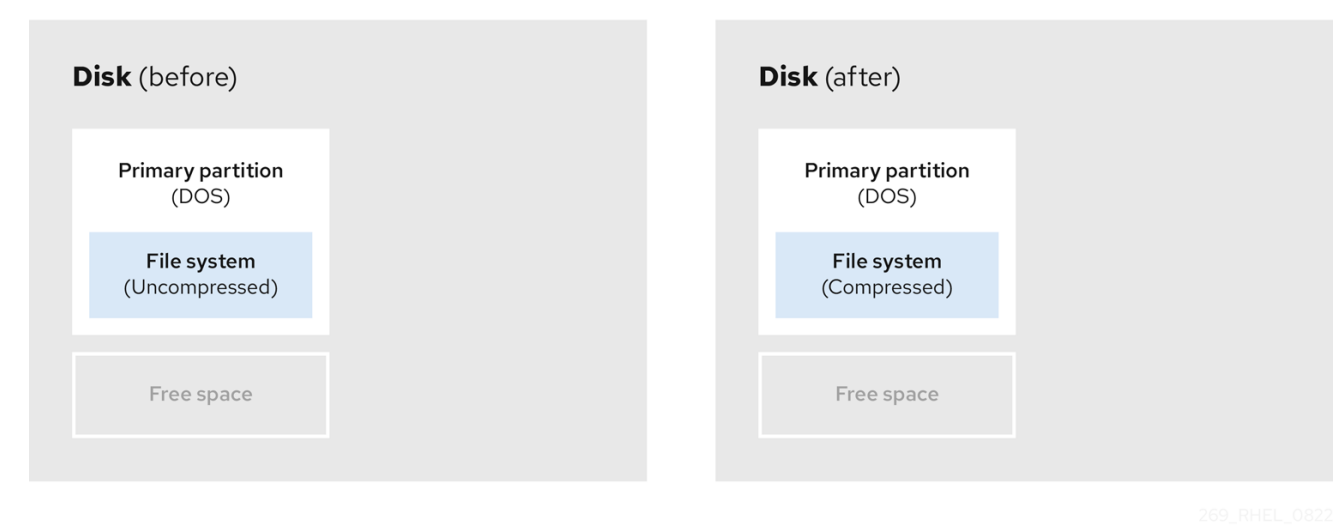
- 

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 8.4. 磁盘中的数据压缩



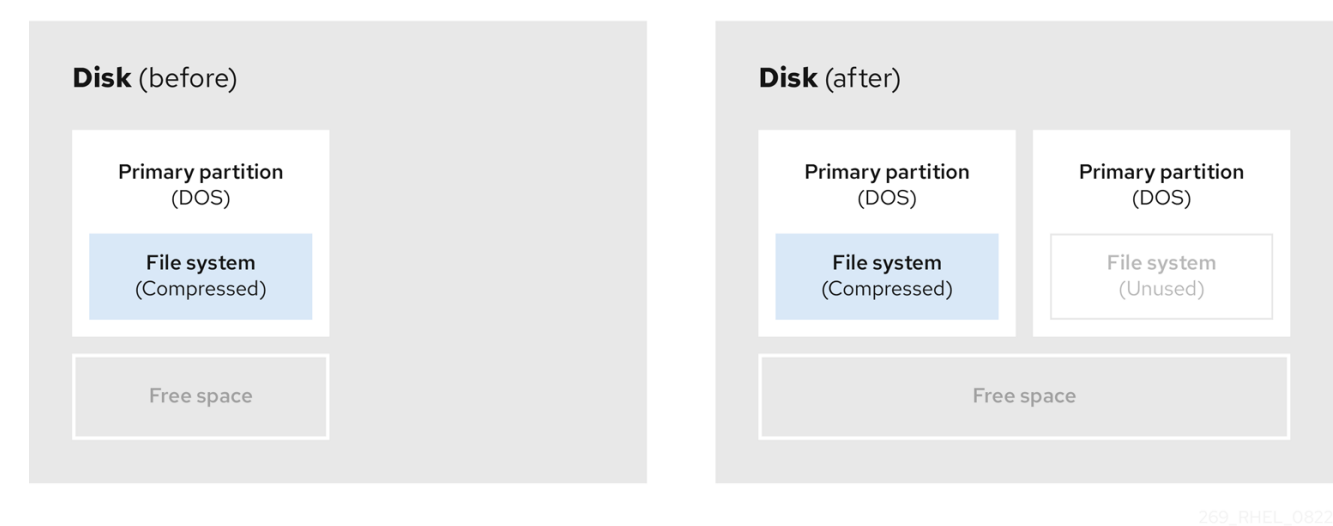
为了避免任何可能的数据丢失，请在继续压缩过程前创建备份。

- 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 8.5. 在磁盘上重新定义分区大小

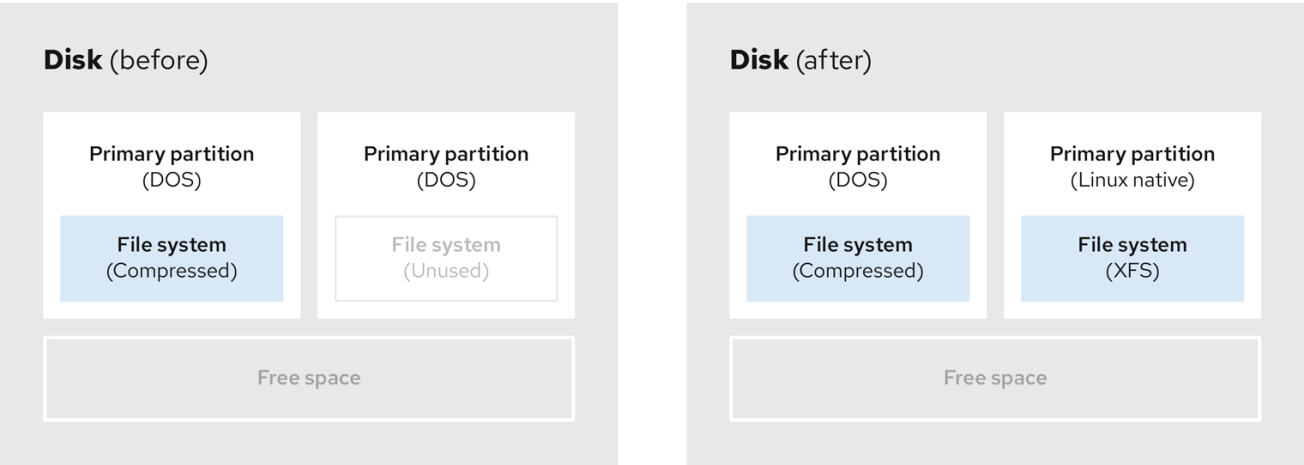


- 可选：创建新分区

一些调整大小的软件支持基于 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 8.6. 带有最终分区配置的磁盘



269\_RHEL\_0822

## 第 9 章 XFS 入门

这是如何创建和维护 XFS 文件系统的概述。

### 9.1. XFS 文件系统

XFS 是一个高度可扩展、高性能、健壮且成熟的 64 位日志文件系统，其支持单个主机上非常大的文件和文件系统。它是 Red Hat Enterprise Linux 8 中的默认文件系统。XFS 最初于 1990 年代由 SGI 早期开发，并在非常大型的服务器和存储阵列中运行有很长的历史记录。

XFS 的功能包括：

#### 可靠性

- 元数据日志，其确保系统崩溃后文件系统的完整性，方法是保留系统重启和重新挂载文件系统时可以重新执行的文件系统操作的记录，
- 广泛的运行时元数据一致性检查
- 可扩展且快速修复工具
- 配额日志。这可避免在崩溃后进行冗长的配额一致性检查。

#### 可伸缩性和性能

- 支持最多 1024 TiB 的文件系统大小
- 支持大量并发操作的能力
- B-tree 索引，用于空闲空间的可扩展性管理



- 复杂的元数据读头算法

- 优化流视频工作负载

## 分配方案

- 基于扩展数据块的分配

- 条带化分配策略

- 延迟分配

- 空间预分配

- 动态分配的 inode

## 其他功能

- 基于 Reflink 的文件副本

- 严格集成备份和恢复工具

- 在线清理

- 在线文件系统增大

- 全面的诊断功能

- 扩展属性(xattr)。这允许系统能够按文件关联多个额外的名称/值对。
- 项目或目录配额。这允许对目录树的配额限制。
- 小于秒的时间戳

性能特性

XFS 在具有企业工作负载的大型系统上具有高性能。大型系统是一个有相对较多的 CPU、多个 HBA 和连接外部磁盘阵列的系统。XFS 在具有多线程、并行 I/O 工作负载的较小系统上也表现良好。

对于单线程、元数据密集型工作负载，XFS 的性能相对较低：例如，在单个线程中创建或删除大量小文件的工作负载。

9.2. 和 EXT4 和 XFS 一起使用的工具比较

这部分比较用于完成 ext4 和 XFS 文件系统中常用任务的工具。

任务	ext4	XFS
创建文件系统	mkfs.ext4	mkfs.xfs
文件系统检查	e2fsck	xfs_repair
重新定义文件系统大小	resize2fs	xfs_growfs
保存文件系统的镜像	e2image	xfs_metadump 和 xfs_mdrestore
标签或者调整文件系统	tune2fs	xfs_admin
备份文件系统	dump 和 restore	xfsdump 和 xfsrestore
配额管理	quota	xfs_quota
文件映射	filefrag	xfs_bmap

## 第 10 章 创建 XFS 文件系统

作为系统管理员，您可以在块设备上创建 XFS 文件系统，使其可以存储文件和目录。

### 10.1. 使用 MKFS.XFS 创建 XFS 文件系统

这个流程描述了如何在块设备上创建 XFS 文件系统。

#### 流程

1.

要创建文件系统，请执行以下操作：

- 

如果设备是常规分区、LVM 卷、MD 卷、磁盘或者类似的设备，请使用以下命令：

```
# mkfs.xfs block-device
```

- 

使用到块设备的路径替换 *block-device*。例如：`/dev/sdb1`、`/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a` 或 `/dev/my-volgroup/my-lv`。

- 

一般情况下，默认选项是常见用途的最佳选择。

- 

在包含现有文件系统的块设备上使用 `mkfs.xfs` 时，添加 `-f` 选项来覆盖该文件系统。

- 

要在硬件 RAID 设备上创建文件系统，检查系统是否正确检测到该设备的条带几何结构：

- 

如果条带几何结构信息正确，则不需要额外的选项。创建文件系统：

```
# mkfs.xfs block-device
```

- 

如果信息不正确，请使用 `-d` 选项的 `su` 和 `sw` 参数来手动指定条带几何结构。`su` 参数指定 RAID 块大小，`sw` 参数指定 RAID 设备中数据磁盘的数量。

例如：

```
# mkfs.xfs -d su=64k,sw=4 /dev/sda3
```

2.

使用以下命令等待系统注册新设备节点：

```
# udevadm settle
```

### 其它资源

- [您系统上的 mkfs.xfs \(8\) 手册页](#)

## 第 11 章 备份 XFS 文件系统

作为系统管理员，您可以使用 `xfsdump` 将 XFS 文件系统备份到文件或磁带。这提供了一个简单的备份机制。

### 11.1. XFS 备份特性

这部分描述了使用 `xfsdump` 工具备份 XFS 文件系统的主要概念和功能。

您可以使用 `xfsdump` 工具来：

- 对常规文件镜像执行备份。

只能将一个备份写入常规文件。

- 在磁带驱动器中执行备份。

`xfsdump` 工具还允许您将多个备份写入同一磁带。备份可跨越多个标题。

要将多个文件系统备份到单个磁带设备，只需将备份写入已包含 XFS 备份的磁带。这会将新备份附加到上一个备份。默认情况下，`xfsdump` 永远不会覆盖现有的备份。

- 创建增量备份。

`xfsdump` 工具使用转储级来确定其他备份所相对的基本备份。从 0 到 9 的数字表示增加的转储级。增量备份只备份自上一次较低级别转储以来发生变化的文件：

- 要执行全备份，请对文件系统中执行 0 级转储。

- 1 级转储是全备份后的第一个增量备份。下一个增量备份为 2 级，它仅备份自 1 级转储以来更改的文件，以此类推，最高到 9 级。

- 使用大小、子树或 **inode** 标志从备份中排除文件，以过滤它们。

#### 其它资源

- 您系统上的 **xfsdump** (8) 手册页

## 11.2. 使用 XFS DUMP 备份 XFS 文件系统

这个步骤描述了如何将 XFS 文件系统的内容备份到文件或者磁带中。

#### 先决条件

- 您可以备份的 XFS 文件系统。
- 可以保存备份的其它文件系统或者磁带驱动器。

#### 流程

- 使用以下命令备份 XFS 文件系统：

```
# xfsdump -l level [-L label] \ -f backup-destination path-to-xfs-filesystem
```

- 使用备份的转储级别替换 *level*。使用 0 执行全备份，或使用 1 到 9 执行后续增量备份。
- 使用您要存储备份的路径替换 *backup-destination*。目的地可以是常规文件、磁带驱动器或远程磁带设备。例如：用于文件的 `/backup-files/Data.xfsdump` 或者用于磁带驱动器的 `/dev/st0`。
- 使用您要备份的 XFS 文件系统的挂载点替换 *path-to-xfs-filesystem*。例如：`/mnt/data/`。文件系统必须挂载。
- 当备份多个文件系统，并将它们保存在单个磁带设备上时，请使用 **-L *label*** 选项来为每

个备份添加一个会话标签，以便在恢复时更轻松地识别它们。使用备份的任何名称替换 *label*：例如 `backup_data`。

### 例 11.1. 备份多个 XFS 文件系统

- 要备份挂载在 `/boot/` 和 `/data/` 目录中的 XFS 文件系统内容，并将它们保存为 `/backup-files/` 目录中的文件：

```
# xfsdump -l 0 -f /backup-files/boot.xfsdump /boot
# xfsdump -l 0 -f /backup-files/data.xfsdump /data
```

- 要备份单个磁带设备中的多个文件系统，请使用 `-L label` 选项来为每个备份添加一个会话标签：

```
# xfsdump -l 0 -L "backup_boot" -f /dev/st0 /boot
# xfsdump -l 0 -L "backup_data" -f /dev/st0 /data
```

### 其它资源

- 您系统上的 `xfsdump` (8) 手册页

## 第 12 章 从备份中恢复 XFS 文件系统

作为系统管理员，您可以使用 `xfsrestore` 工具来恢复用 `xfsdump` 工具创建的，并存储在文件或磁带中的 XFS 备份。

### 12.1. 从备份中恢复 XFS 的特性

`xfsrestore` 工具从 `xfsdump` 生成的备份中恢复文件系统。`xfsrestore` 工具有两个模式：

- 简单 模式允许用户从 0 级转储恢复整个文件系统。这是默认的模式。
- 累计 模式启用从增量备份恢复文件系统：即，1 级到 9 级。

唯一 会话 ID 或 会话标签 标识每个备份。从包含多个备份的磁带恢复备份需要相应的会话 ID 或标签。

要从备份中提取、添加或删除特定文件，请进入 `xfsrestore` 交互模式。交互模式提供了一组命令来操作备份文件。

#### 其它资源

- [系统中的 xfsrestore \(8\) 手册页](#)

### 12.2. 使用 XFSRESTORE 从备份中恢复 XFS 文件系统

这个步骤描述了如何从文件或者磁带备份中恢复 XFS 文件系统的内容。

#### 先决条件

- XFS 文件的文件或磁带备份，如 [备份 XFS 文件系统](#) 中所述。
- 您可以恢复备份的存储设备。



## 流程

- 恢复备份的命令因您是从全备份或增量备份中恢复，还是从单个磁带设备恢复多个备份而有所不同：

```
# xfsrestore [-r] [-S session-id] [-L session-label] [-i] -f backup-location restoration-path
```

- 使用备份位置替换 **backup-location**。这可以是常规文件、磁带驱动器或远程磁带设备。例如：用于文件的 `/backup-files/Data.xfsdump` 或者用于磁带驱动器的 `/dev/st0`。

- 使用要恢复文件系统的目录的路径替换 **restore-path**。例如：`/mnt/data/`。

- 要从增量（1 级到 9 级）备份恢复文件系统，请添加 **-r** 选项。

- 要从包含多个备份的磁带设备恢复备份，请使用 **-S** 或 **-L** 选项指定备份。

**-S** 选项允许您按会话 ID 选择备份，而 **-L** 选项则允许您按会话标签进行选择。要获取会话 ID 和会话标签，请使用 **xfsrestore -l** 命令。

使用备份的会话 ID 替换 **session-id**。例如，`b74a3586-e52e-4a4a-8775-c3334fa8ea2c`。使用备份的会话标签替换 **session-label**。例如，`my_backup_session_label`。

- 要以交互方式使用 **xfsrestore**，请使用 **-i** 选项。

在 **xfsrestore** 完成读取指定设备后，交互对话框开始。交互式 **xfsrestore shell** 中的可用命令包括 `cd`、`ls`、`add`、`delete` 和 `extract`；如需命令的完整列表，请使用 `help` 命令。

### 例 12.1. 恢复多个 XFS 文件系统

- 要恢复 XFS 备份文件，并将其内容保存到 `/mnt/` 下的目录中：

```
# xfsrestore -f /backup-files/boot.xfsdump /mnt/boot/
# xfsrestore -f /backup-files/data.xfsdump /mnt/data/
```

- 要从包含多个备份的磁带设备恢复，请使用会话标签或会话 ID 指定每个备份：

```
# xfsrestore -L "backup_boot" -f /dev/st0 /mnt/boot/  
# xfsrestore -S "45e9af35-efd2-4244-87bc-4762e476cbab" -f /dev/st0 /mnt/data/
```

## 其它资源

- [系统中的 xfsrestore \(8\) 手册页](#)

## 12.3. 从磁带恢复 XFS 备份时的说明性消息

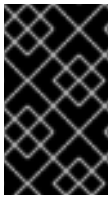
当从存有多多个文件系统备份的磁带恢复备份时，**xfsrestore** 工具可能会发出消息。当 **xfsrestore** 按顺序检查磁带上的每个备份时，消息会通知您是否找到了与请求的备份相匹配的备份。例如：

```
xfsrestore: preparing drive  
xfsrestore: examining media file 0  
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the  
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)  
xfsrestore: examining media file 1  
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the  
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)  
[...]
```

说明性消息会一直显示，直到找到匹配的备份。

## 第 13 章 增加 XFS 文件系统的大小

作为系统管理员，您可以增大 XFS 文件系统的大小，以充分利用更大的存储容量。



### 重要

目前不能缩小 XFS 文件系统的大小。

### 13.1. 使用 XFS\_GROWFS 增加 XFS 文件系统的大小

这个流程描述了如何使用 `xfs_growfs` 工具增大 XFS 文件系统。

#### 先决条件

- 确保底层块设备的大小适当，以便以后保留调整了大小的文件系统。为受影响的块设备使用合适的调整大小的方法。
- 挂载 XFS 文件系统。

#### 流程

- 在挂载 XFS 文件系统时，使用 `xfs_growfs` 工具来增加其大小：

```
# xfs_growfs file-system -D new-size
```

- 使用 XFS 文件系统的挂载点替换 `file-system`。
- 使用 `-D` 选项，将 `new-size` 替换为在文件系统块数中指定的文件系统所需的新大小。

要找出给定 XFS 文件系统的块大小（kB），请使用 `xfs_info` 工具：

```
# xfs_info block-device
```

```
...
```

```
data = bsize=4096
...
```

- 如果没有 **-D** 选项，**xfs\_growfs** 将文件系统增大到底层设备所支持的最大大小。

#### 其它资源

- 您系统上的 **xfs\_growfs (8)** 手册页。

## 第 14 章 配置 XFS 错误行为

您可以配置 XFS 文件系统在遇到不同的 I/O 错误时的行为方式。

### 14.1. XFS 中的可配置错误处理

当 I/O 操作期间发生错误时，XFS 文件系统以以下其中一种方式响应：

- XFS 重复重试 I/O 操作，直到操作成功或 XFS 达到设定的限制。  
  
限制是基于重试的最大次数或重试的最长时间。
- XFS 认为错误是永久性的，并停止文件系统上的操作。

您可以配置 XFS 如何对以下错误情况做出响应：

#### EIO

读取或写入时出错

#### ENOSPC

该设备中没有剩余空间

#### ENODEV

无法找到设备

您可以设置重试的最大次数，以及 XFS 认为其是永久错误前的最长时间（以秒为单位）。XFS 在达到任合一个限制时停止重试操作。

您还可以配置 XFS，以便在卸载文件系统时，XFS 会立即取消重试，而不考虑任何其他配置。但此配置可让卸载操作成功，尽管存在持续的错误。

#### 默认行为

每个 XFS 错误情况的默认行为取决于错误上下文。ENODEV 等 XFS 错误都被视为致命且不可恢复的，无论重试次数如何。其默认重试限制为 0。

## 14.2. 特定和未定义的 XFS 错误条件的配置文件

以下目录保存用来控制不同错误条件的 XFS 错误行为的配置文件：

**/sys/fs/xfs/device/error/metadata/EIO/**

对于 EIO 错误情况

**/sys/fs/xfs/device/error/metadata/ENODEV/**

对于 ENODEV 错误情况

**/sys/fs/xfs/device/error/metadata/ENOSPC/**

对于 ENOSPC 错误情况

**/sys/fs/xfs/device/error/default/**

所有其他未定义错误条件的通用配置

每个目录包括以下配置文件来配置重试限制：

**max\_retries**

控制 XFS 重试操作的次数上限。

**retry\_timeout\_seconds**

指定 XFS 停止重试操作后的时间限值（以秒为单位）。

## 14.3. 为特定条件设置 XFS 行为

这个步骤配置了 XFS 如何响应特定的错误条件。

流程

- 

设置重试的最大重试次数、重试时间限制或两者：

- 

要设置重试的最大次数，请将所需的次数写入 `max_retries` 文件：

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/max_retries
```

- 

要设置时间限制，将所需的秒数写入 `retry_timeout_seconds` 文件：

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/retry_timeout_second
```

`value` 是介于 -1 和 C 带符号整数类型的最大可能值之间的数字。64 位 Linux 中是 2147483647。

在这两个限制中，值 -1 用于持续重试，0 用于立即停止。

`device` 是设备的名称，可以在 `/dev/` 目录中找到；例如，`sda`。

#### 14.4. 为未定义条件设置 XFS 行为

此流程配置 XFS 如何对共享一个通用配置的所有未定义的错误情况做出响应。

##### 流程

- 

设置重试的最大重试次数、重试时间限制或两者：

- 

要设置重试的最大次数，请将所需的次数写入 `max_retries` 文件：

```
# echo value > /sys/fs/xfs/device/error/metadata/default/max_retries
```

- 

要设置时间限制，将所需的秒数写入 `retry_timeout_seconds` 文件：

```
# echo value > /sys/fs/xfs/device/error/metadata/default/retry_timeout_seconds
```

**value** 是介于 -1 和 C 带符号整数类型的最大可能值之间的数字。64 位 Linux 中是 2147483647。

在这两个限制中，值 -1 用于持续重试，0 用于立即停止。

**device** 是设备的名称，可以在 `/dev/` 目录中找到；例如，`sda`。

## 14.5. 设置 XFS 卸载行为

这个流程配置 XFS 在卸载文件系统时如何对错误情况做出响应。

如果您在文件系统中设置 `fail_at_unmount` 选项，它会在卸载过程中覆盖所有其他错误配置，并立即卸载文件系统，而不重试 I/O 操作。这允许卸载操作在出现持久错误时也可以成功。



### 警告

在卸载过程启动后，您不能更改 `fail_at_unmount` 值，因为卸载过程会从相应文件系统的 `sysfs` 接口删除配置文件。您必须在文件系统开始卸载前配置卸载行为。

## 流程



启用或禁用 `fail_at_unmount` 选项：



要在文件系统卸载时取消重试所有操作，请启用这个选项：

```
# echo 1 > /sys/fs/xfs/device/error/fail_at_unmount
```



要在文件系统卸载时遵守 `max_retries` 和 `retry_timeout_seconds` 重试限制，请禁用这个选项：



```
# echo 0 > /sys/fs/xfs/device/error/fail_at_unmount
```

***device*** 是设备的名称，可以在 **/dev/** 目录中找到；例如，**sda**.

## 第 15 章 检查和修复文件系统

RHEL 提供可以检查和修复文件系统的文件系统管理工具。这些工具通常被称为 **fsck** 工具，其中 **fsck** 是 **文件系统检查** 的缩写版本。在大多数情况下，这些工具会根据需要在系统引导期间自动运行，但也可以根据需要手动调用。



### 重要

文件系统检查程序只保证跨文件系统的元数据的一致性。它们不知道文件系统中所包含的实际数据，它们不是数据恢复工具。

### 15.1. 需要文件系统检查的场景

如果出现以下情况，可以使用相关的 **fsck** 工具来检查您的系统：

- 系统无法引导
- 特定磁盘上的文件损坏
- 由于不一致，文件系统关闭或变为只读
- 文件系统上的文件无法访问

发生文件系统不一致的原因可能有多种，包括但不限于硬件错误、存储管理错误和软件 **bug**。

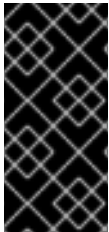


### 重要

文件系统检查工具不能修复硬件问题。如果修复操作成功，文件系统必须是完全可读写的。如果文件系统因为硬件错误而损坏，则必须首先将该文件系统移至好的磁盘，例如，使用 **dd(8)** 工具。

对于日志文件系统，启动时通常需要的所有操作是重播日志（如果需要），此操作通常是一个短操作。

但是，如果发生文件系统不一致或损坏的情况，即使是对于日志记录文件系统，也必须使用文件系统检查程序来修复文件系统。



### 重要

通过将 `/etc/fstab` 中的第 6 字段设为 0，可以在引导时禁用文件系统检查。但是，红帽不建议这样做，除非您在启动时遇到 `fsck` 问题，例如对于非常大的或远程文件系统。

### 其它资源

- [fstab \(5\)、fsck \(8\) 和 dd \(8\) man page](#)

## 15.2. 运行 FSCK 的潜在副作用

通常，运行文件系统检查和修复工具至少可以自动修复发现的一些不一致问题。在某些情况下可能会出现以下问题：

- 如果无法修复，可以丢弃严重损坏的 `inode` 或目录。
- 可能会对文件系统进行大量更改。

要确保不会永久地进行意外或不必要的更改，请确保遵循流程中概述的任何预防步骤。

## 15.3. XFS 中的错误处理机制

这部分论述了 XFS 如何处理文件系统中各种错误。

### 未完全卸载

日志维护文件系统中发生的元数据变化的事务记录。

在系统崩溃、电源故障或其他未完全卸载的情况下，XFS 使用 **journal**（也称为 **log**）来恢复文件系统。挂载 XFS 文件系统时，内核执行日志恢复。

## 损坏

在这种情况下，*损坏* 意味着文件系统中出现以下情况引起的错误，例如：

- 硬件故障
- 存储固件、设备驱动程序、软件堆栈或者文件系统本身的错误
- 导致文件系统部分内容被文件系统之外的内容覆盖的问题

当 XFS 检测到文件系统或文件系统元数据中的损坏时，它可以关闭文件系统，并在系统日志中报告该事件。请注意，如果损坏发生在托管 **/var** 目录的文件系统上，重启后这些日志将不可用。

### 例 15.1. 系统日志条目报告 XFS 崩溃

```
# dmesg --notime | tail -15
```

```
XFS (loop0): Mounting V5 Filesystem
XFS (loop0): Metadata CRC error detected at xfs_agi_read_verify+0xcb/0xf0 [xfs], xfs_agi block
0x2
XFS (loop0): Unmount and run xfs_repair
XFS (loop0): First 128 bytes of corrupted metadata buffer:
00000000027b3b56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000005f9abc7a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000005b0aef35: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000da9d2ded: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000001e265b07: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000006a40df69: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000b272907: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000000e484aac5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
XFS (loop0): metadata I/O error in "xfs_trans_read_buf_map" at daddr 0x2 len 1 error 74
XFS (loop0): xfs_imap_lookup: xfs_ialloc_read_agi() returned error -117, agno 0
XFS (loop0): Failed to read root inode 0x80, error 11
```

当尝试访问损坏的 XFS 文件系统时，用户空间工具通常会报告 *输入/输出错误* 消息。挂载带有损坏日志的 XFS 文件系统会导致挂载失败，并出现以下错误消息：

```
mount: /mount-point: mount(2) system call failed: Structure needs cleaning.
```

您必须手动使用 `xfs_repair` 工具来修复损坏。

#### 其它资源

- [您系统上的 `xfs\_repair` \(8\) 手册页](#)

### 15.4. 使用 XFS\_REPAIR 检查 XFS 文件系统

使用 `xfs_repair` 工具对 XFS 文件系统执行只读检查。与其他文件系统修复工具不同，`xfs_repair` 不会在引导时运行，即使 XFS 文件系统没有被完全卸载。如果是未完全卸载，XFS 会在挂载时重新执行日志，确保一致的文件系统；`xfs_repair` 不能在不先重新挂载脏日志的情况下修复带有脏日志的 XFS 文件系统。



#### 注意

虽然 `xfsprogs` 软件包中有 `fsck.xfs` 二进制文件，但这仅用于满足在引导时查找 `fsck.file` 系统二进制文件的 `initscripts`。`fsck.xfs` 立即退出，退出代码为 0。

#### 流程

1. 通过挂载和卸载文件系统重新显示日志：

```
# mount file-system
# umount file-system
```



#### 注意

如果挂载失败，并带有结构需要清理的错误，则日志已损坏，且无法重播。试运行应发现并报告更多有关磁盘损坏的信息。

2. 使用 `xfs_repair` 工具执行试运行来检查文件系统。打印任何错误并指示将要采取的操作，而不修改文件系统。

```
# xfs_repair -n block-device
```

3.

挂载文件系统：

```
# mount file-system
```

## 其它资源

- 您系统上的 **xfs\_repair** (8) 和 **xfs\_metadump** (8) 手册页

## 15.5. 使用 XFS\_REPAIR 修复 XFS 文件系统

这个过程使用 **xfs\_repair** 工具修复损坏的 XFS 文件系统。

## 流程

1.

使用 **xfs\_metadump** 工具在修复前为诊断或测试目的创建元数据镜像。如果损坏是由软件 bug 导致的，则预修复文件系统元数据映像对于支持调查非常有用。预修复镜像中出现的损坏模式有助于分析根本原因。

- 使用 **xfs\_metadump** 调试工具将 XFS 文件系统上的元数据复制到文件。如果需要发送大的 metadump 文件来支持，可使用标准压缩工具来压缩生成的 metadump 文件，以减少文件大小。

```
# xfs_metadump block-device metadump-file
```

2.

通过重新挂载文件系统来重新显示日志：

```
# mount file-system
# umount file-system
```

3.

使用 **xfs\_repair** 工具来修复卸载的文件系统：

- 如果挂载成功，则不需要额外的选项：

```
# xfs_repair block-device
```

- 如果挂载失败，带有 **Structure needs cleaning** 错误，日志会破坏且无法重复显示。使用 **-L** 选项（**强制日志归零**）来清除日志：



#### 警告

该命令会导致崩溃时正在进行的所有元数据更新丢失，这可能会造成严重的文件系统损坏和数据丢失。只有在无法重播日志时，才应将其作为最后的手段。

```
# xfs_repair -L block-device
```

4.

挂载文件系统：

```
# mount file-system
```

#### 其它资源

- 您系统上的 **xfs\_repair (8)** 手册页

## 15.6. EXT2、EXT3 和 EXT4 中的处理机制出错

**ext2**、**ext3** 和 **ext4** 文件系统使用 **e2fsck** 工具来执行文件系统检查和修复。文件名 **fsck.ext2**、**fsck.ext3** 和 **fsck.ext4** 是 **e2fsck** 工具的硬链接。这些二进制文件在引导时自动运行，其行为因正在检查的文件系统和文件系统的状态而异。

对于不是元数据日志记录文件系统的 **ext2** 和没有日志的 **ext4** 文件系统，会调用完整的文件系统检查和修复。

对于带有元数据日志的 **ext3** 和 **ext4** 文件系统，日志将在用户空间中重播，从实用工具退出。这是默认操作，因为日志重播确保崩溃后文件系统的一致性。

如果这些文件系统在挂载时遇到元数据不一致的情况，它们会在文件系统超级块中记录此事实。如果

**e2fsck** 发现文件系统标记有这样的错误，**e2fsck** 会在重播日志（如果存在）后执行全面的检查。

## 其它资源

- [系统中 fsck \(8\) 和 e2fsck \(8\) 手册页](#)

## 15.7. 使用 E2FCK 检查 EXT2、EXT3 或者 EXT4 文件系统

这个流程使用 **e2fsck** 工具检查 **ext2**、**ext3** 或 **ext4** 文件系统。

## 流程

1. 通过重新挂载文件系统来重新显示日志：

```
# mount file-system  
# umount file-system
```

2. 执行空运行检查文件系统。

```
# e2fsck -n block-device
```



### 注意

打印任何错误并指示将要采取的操作，而不修改文件系统。稍后一致性检查阶段可能会打印额外的错误，因为在修复模式下运行时，它会发现可能在早期阶段已经修复了的的不一致问题。

## 其它资源

- [系统上的 e2image \(8\) 和 e2fsck \(8\) man page](#)

## 15.8. 使用 E2FCK 修复 EXT2、EXT3 或者 EXT4 文件系统

这个流程使用 **e2fsck** 工具修复损坏的 **ext2**、**ext3** 或 **ext4** 文件系统。



## 流程

1.

保存文件系统镜像以进行支持调查。如果损坏是由软件 **bug** 导致的，则预修复文件系统元数据映像对于支持调查非常有用。预修复镜像中出现的损坏模式有助于分析根本原因。



### 注意

严重损坏的文件系统可能会导致元数据镜像创建出现问题。

- 

如果要为测试目的创建镜像，请使用 **-r** 选项来创建与文件系统本身大小相同的稀疏文件。然后 **e2fsck** 可以直接对生成的文件进行操作。

```
# e2image -r block-device image-file
```

- 

如果您要创建要存档或提供用于诊断的镜像，请使用 **-Q** 选项，该选项可创建适合于传输的更紧凑的文件格式。

```
# e2image -Q block-device image-file
```

2.

通过重新挂载文件系统来重新显示日志：

```
# mount file-system
# umount file-system
```

3.

自动修复文件系统。如果需要用户干预，**e2fsck** 指明其输出中未修复的问题，并在退出代码中反映此状态。

```
# e2fsck -p block-device
```

## 其它资源

- 

系统上的 **e2image** (8) 手册页

- 

系统上的 **e2fsck** (8) 手册页

## 第 16 章 挂载文件系统

作为系统管理员，您可以在系统上挂载文件系统以访问其上的数据。

### 16.1. LINUX 挂载机制

以下是在 Linux 中挂载文件系统的基本概念。

在 Linux、UNIX 和类似的操作系统中，不同分区和可移动设备（例如，CD、DVD 或者 USB 闪存）上的文件系统可以附加到目录树中的某个点（挂载点），然后再次分离。虽然文件系统挂载在目录上，但无法访问该目录的原始内容。

请注意，Linux 不会阻止您将文件系统挂载到已附加了文件系统的目录。

挂载时，您可以通过以下方法识别设备：

- 通用唯一标识符(UUID)：例如，`UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb`
- 卷标签：例如，`LABEL=home`
- 到非持久性块设备的完整路径：例如，`/dev/sda3`

当您使用 `mount` 命令挂载文件系统时，如果没有提供所有必需的信息，即设备名称、目标目录或文件系统类型，`mount` 工具会读取 `/etc/fstab` 文件的内容，以检查其中是否列出了给定的文件系统。`/etc/fstab` 文件包含设备名称列表、所选文件系统要挂载的目录，以及文件系统类型和挂载选项。因此，当挂载在 `/etc/fstab` 中指定的文件系统时，以下命令语法就足够了：

- 使用挂载点挂载：

```
# mount directory
```

- 使用块设备挂载：

```
# mount device
```

## 其它资源

- 在您的系统中的 **mount (8)** 手册页
- [如何列出永久命名属性，如 UUID。](#)

## 16.2. 列出当前挂载的文件系统

使用 **findmnt** 工具列出命令行中的所有当前挂载的文件系统。

## 流程

- 要列出所有挂载的文件系统，请使用 **findmnt** 工具：

```
$ findmnt
```

- 要将列出的文件系统限制为特定的文件系统类型，请添加 **--types** 选项：

```
$ findmnt --types fs-type
```

例如：

### 例 16.1. 只列出 XFS 文件系统

```
$ findmnt --types xfs
```

TARGET	SOURCE	FSTYPE	OPTIONS
/	/dev/mapper/luks-5564ed00-6aac-4406-bfb4-c59bf5de48b5	xfs	rw,relatime
—/boot	/dev/sda1	xfs	rw,relatime
—/home	/dev/mapper/luks-9d185660-7537-414d-b727-d92ea036051e	xfs	rw,relatime

## 其它资源

- [在您的系统中的 findmnt \(8\) 手册页](#)

### 16.3. 使用 MOUNT 挂载文件系统

使用 **mount** 工具挂载文件系统。

#### 先决条件

- 验证您选择的挂载点上没有挂载文件系统：

```
$ findmnt mount-point
```

#### 流程

1. 要附加某个文件系统，请使用 **mount** 工具：

```
# mount device mount-point
```

#### 例 16.2. 挂载 XFS 文件系统

例如：要挂载由 **UUID** 识别的本地 **XFS** 文件系统：

```
# mount UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /mnt/data
```

2. 如果 **mount** 无法自动识别文件系统类型，请使用 **--types** 选项来指定：

```
# mount --types type device mount-point
```

#### 例 16.3. 挂载 NFS 文件系统

例如：要挂载远程 **NFS** 文件系统：

```
# mount --types nfs4 host:/remote-export /mnt/nfs
```

## 其它资源

- [在您的系统中的 mount \(8\) 手册页](#)

## 16.4. 移动挂载点

使用 `mount` 实用程序将挂载的文件系统的挂载点更改为不同的目录。

### 流程

1. 要更改挂载文件系统的目录：

```
# mount --move old-directory new-directory
```

#### 例 16.4. 移动本地文件系统

例如，将挂载在 `/mnt/userdirs/` 目录的文件系统移动到 `/home/` 挂载点：

```
# mount --move /mnt/userdirs /home
```

2. 验证文件系统是否已如预期移动：

```
$ findmnt  
$ ls old-directory  
$ ls new-directory
```

## 其它资源

- [在您的系统中的 mount \(8\) 手册页](#)

## 16.5. 使用 UMOUNT 卸载文件系统

使用 `umount` 实用程序卸载文件系统。

### 流程

1.

使用以下命令之一卸载文件系统：

•

通过挂载点：

```
# umount mount-point
```

•

通过设备：

```
# umount device
```

如果命令失败并显示类似如下的错误，这意味着文件系统正在使用，因为进程正在使用其上的资源：

```
umount: /run/media/user/FlashDrive: target is busy.
```

2.

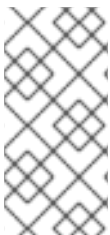
如果文件系统正在使用，请使用 **fuser** 工具来确定哪个进程正在访问它。例如：

```
$ fuser --mount /run/media/user/FlashDrive /run/media/user/FlashDrive: 18351
```

之后，停止使用文件系统的进程，并尝试再次卸载它。

## 16.6. 在 WEB 控制台中挂载和卸载文件系统

为了能够在 RHEL 系统中使用分区，您需要在分区中作为一个设备挂载文件系统。



### 注意

您还可以卸载文件系统，RHEL 系统将会停止使用它。卸载文件系统可让您删除、删除或重新格式化设备。

### 先决条件

•

**cockpit-storaged** 软件包已安装在您的系统上。

- 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](#)。
- 如果要卸载文件系统，请确保系统没有使用存储在分区中的任何文件、服务或应用程序。

## 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. 点 **Storage** 选项卡。
3. 在 **Storage** 表中，选择您要从中删除分区的卷。
4. 在 **GPT partitions** 部分中，点击您要挂载或卸载其文件系统的分区旁边的菜单按钮 **⋮**。
5. 点 **Mount** 或 **Unmount**。

## 16.7. 常用挂载选项

下表列出了 **mount** 实用程序的最常见选项。您可以使用以下语法应用这些挂载选项：

```
# mount --options option1,option2,option3 device mount-point
```

表 16.1. 常用挂载选项

选项	描述
<b>async</b>	对文件系统启用异步输入和输出操作。
<b>auto</b>	使用 <b>mount -a</b> 命令使文件系统被自动挂载。
<b>defaults</b>	为 <b>async,auto,dev,exec,nouser,rw,suid</b> 选项提供别名。
<b>exec</b>	允许在特定文件系统中执行二进制文件。
<b>loop</b>	将镜像挂载为 loop 设备。
<b>noauto</b>	默认行为禁用使用 <b>mount -a</b> 命令对文件系统进行自动挂载。
<b>noexec</b>	不允许在特定文件系统中执行二进制文件。
<b>nouser</b>	不允许普通用户（即 root 用户）挂载和卸载文件系统。
<b>remount</b>	如果已经挂载文件系统，则会重新挂载文件系统。
<b>ro</b>	仅挂载文件系统以读取。
<b>rw</b>	挂载文件系统以进行读和写操作。
<b>user</b>	允许普通用户（即 root 用户）挂载和卸载该文件系统。



## 第 17 章 在多个挂载点共享挂载

作为系统管理员，您可以重复挂载点以便从多个目录中访问文件系统。

### 17.1. 共享挂载的类型

您可以使用多种共享挂载。当您在共享挂载点挂载另一个文件系统时，这两种文件系统之间的区别就是这种情况。共享挂载使用 *共享子树* 功能实现。

可用的挂载类型如下：

#### **private**

这个类型不接收或转发任何传播事件。

当您在重复或者原始挂载点下挂载另一个文件系统时，不会反映在另一个文件系统中。

#### **shared**

这个类型会为给定挂载点创建准确的副本。

当挂载点被标记为 **shared** 挂载时，原始挂载点中的任何挂载都会反映在其中，反之亦然。

这是根文件系统的默认挂载类型。

#### **slave**

此类型会创建给定挂载点的有限重复。

当挂载点标记为 **slave** 挂载时，原始挂载点中的任何挂载都会反映在该挂载点中，但 **slave** 挂载中的任何挂载都没有反映在其原始挂载中。

#### **unbindable**

此类型可防止给定挂载点被复制。

## 其它资源

- [Linux Weekly News 上的 \*Shared subtrees\* 文章](#)

## 17.2. 创建私有挂载点副本

将挂载点复制为专用挂载。您稍后挂载到复制或原始挂载点下的文件系统不会反映在另一个文件系统中。

### 流程

1. 从原始挂载点创建虚拟文件系统(VFS)节点：

```
# mount --bind original-dir original-dir
```

2. 将原始挂载点标记为私有：

```
# mount --make-private original-dir
```

或者，要更改所选挂载点以及其下的所有挂载点的挂载类型，请使用 `--make-rprivate` 选项，而不是 `--make-private` 选项。

3. 创建副本：

```
# mount --bind original-dir duplicate-dir
```

### 例 17.1. 将 `/media` 复制到 `/mnt` 作为专用挂载点

1. 从 `/media` 目录创建 VFS 节点：

```
# mount --bind /media /media
```

2. 将 `/media` 目录标记为私有：

```
# mount --make-private /media
```

3.

在 **/mnt** 中创建副本：

```
# mount --bind /media /mnt
```

4.

现在可以验证 **/media** 和 **/mnt** 共享内容，但 **/media** 中的挂载内容没有出现在 **/mnt** 中。例如，如果 **CD-ROM** 驱动器包含非空介质，并且 **/media/cdrom/** 目录存在，请使用：

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI GPL isolinux LiveOS
# ls /mnt/cdrom
#
```

5.

还可以验证 **/mnt** 目录中挂载的文件系统没有反映在 **/media** 中。例如，如果插入了使用 **/dev/sdc1** 设备的非空 **USB** 闪存驱动器，并且 **/mnt/flashdisk/** 目录存在，请使用：

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US publican.cfg
```

## 其它资源



在您的系统中的 **mount (8)** 手册页

## 17.3. 创建共享挂载点副本

将挂载点复制为共享挂载。您稍后挂载到原始目录或副本下的文件系统始终反映在其它文件系统中。

## 流程

1.

从原始挂载点创建虚拟文件系统(VFS)节点：

```
# mount --bind original-dir original-dir
```

2.

将原始挂载点标记为共享：

```
# mount --make-shared original-dir
```

或者，要更改所选挂载点及其下的所有挂载点的挂载类型，请使用 `--make-rshared` 选项，而不是 `--make-shared`。

3.

创建副本：

```
# mount --bind original-dir duplicate-dir
```

### 例 17.2. 将 `/media` 重复到 `/mnt` 作为共享挂载点

要使 `/media` 和 `/mnt` 目录共享相同的内容：

1.

从 `/media` 目录创建 VFS 节点：

```
# mount --bind /media /media
```

2.

将 `/media` 目录标记为共享：

```
# mount --make-shared /media
```

3.

在 `/mnt` 中创建副本：

```
# mount --bind /media /mnt
```

4.

现在，可以验证 `/media` 中的挂载是否也出现在 `/mnt` 中。例如，如果 CD-ROM 驱动器包含非空介质，并且 `/media/cdrom/` 目录存在，请使用：

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI GPL isolinux LiveOS
# ls /mnt/cdrom
EFI GPL isolinux LiveOS
```

5.

同样，还可以验证 `/mnt` 目录中挂载的任何文件系统是否反映在 `/media` 中。例如，如果插入了使用 `/dev/sdc1` 设备的非空 USB 闪存驱动器，并且 `/mnt/flashdisk/` 目录存在，请使

用：

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
en-US publican.cfg
# ls /mnt/flashdisk
en-US publican.cfg
```

## 其它资源

- 在您的系统中的 **mount (8)** 手册页

## 17.4. 创建从挂载点副本

将挂载点复制为 **slave** 挂载类型。您稍后挂载在原始挂载点下的文件系统将反映在副本中，而不是反过来。

## 流程

1. 从原始挂载点创建虚拟文件系统(VFS)节点：

```
# mount --bind original-dir original-dir
```

2. 将原始挂载点标记为共享：

```
# mount --make-shared original-dir
```

或者，要更改所选挂载点和其下的所有挂载点的挂载类型，请使用 **--make-rshared** 选项，而不是 **--make-shared**。

3. 创建副本，并将其标记为 **slave** 类型：

```
# mount --bind original-dir duplicate-dir
# mount --make-slave duplicate-dir
```

**例 17.3.** 将 **/media** 复制到 **/mnt** 作为从挂载点

这个示例演示了如何使 **/media** 目录的内容也出现在 **/mnt** 中，但 **/mnt** 目录中的任何挂载都不会反映在 **/media** 中。

1.

从 **/media** 目录创建 VFS 节点：

```
# mount --bind /media /media
```

2.

将 **/media** 目录标记为共享：

```
# mount --make-shared /media
```

3.

在 **/mnt** 中创建副本，并将其标记为 **slave**：

```
# mount --bind /media /mnt
# mount --make-slave /mnt
```

4.

验证 **/media** 中的挂载是否也出现在 **/mnt** 中。例如，如果 **CD-ROM** 驱动器包含非空介质，并且 **/media/cdrom/** 目录存在，请使用：

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI GPL isolinux LiveOS
# ls /mnt/cdrom
EFI GPL isolinux LiveOS
```

5.

还要验证 **/mnt** 目录中挂载的文件系统是否没有反映在 **/media** 中。例如，如果插入了使用 **/dev/sdc1** 设备的非空 **USB** 闪存驱动器，并且 **/mnt/flashdisk/** 目录存在，请使用：

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US publican.cfg
```

## 其它资源

- [在您的系统中的 mount \(8\) 手册页](#)

## 17.5. 防止挂载点重复

将挂载点标记为 **unbindable**，以便无法在另一个挂载点中复制挂载点。

### 流程

- 要将挂载点的类型改为 **unbindable** 挂载，请使用：

```
# mount --bind mount-point mount-point
# mount --make-unbindable mount-point
```

或者，要更改所选挂载点和其下的所有挂载点的挂载类型，请使用 **--make-runbindable** 选项，而不是 **--make-unbindable** 选项。

重复此挂载的任何后续尝试都会失败，并显示以下错误：

```
# mount --bind mount-point duplicate-dir

mount: wrong fs type, bad option, bad superblock on mount-point,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail or so
```

### 例 17.4. 防止 /media 被复制

- 要防止 /media 目录被共享，请使用：

```
# mount --bind /media /media
# mount --make-unbindable /media
```

### 其它资源

- 在您的系统中的 **mount (8)** 手册页

## 第 18 章 永久挂载文件系统

作为系统管理员，您可以永久地挂载文件系统以配置不可移动的存储。

### 18.1. /ETC/FSTAB 文件

使用 `/etc/fstab` 配置文件控制文件系统的永久挂载点。`/etc/fstab` 文件中的每一行定义了文件系统的挂载点。

它包括六个字段，用空格分开：

1. 由持久属性或 `/dev` 目录中的路径标识的块设备。
2. 挂载该设备的目录。
3. 该设备的文件系统。
4. 文件系统的挂载选项，其包括在引导时使用默认选项挂载分区的 `defaults` 选项。挂载选项字段也识别 `x-systemd.option` 格式的 `systemd` 挂载单元选项。
5. `dump` 工具的备份选项。
6. `fsck` 工具的检查顺序。

#### 注意

`systemd-fstab-generator` 将 `/etc/fstab` 文件中的条目动态转换为 `systemd-mount` 单元。`systemd` 会在手动激活过程中从 `/etc/fstab` 自动挂载 LVM 卷，除非 `systemd-mount` 单元已被屏蔽。

#### 例 18.1. `/etc/fstab` 中的 `/boot` 文件系统



块设备	挂载点	File system	选项	Backup	检查
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b	/boot	xfs	defaults	0	0

systemd 服务从 /etc/fstab 中的条目自动生成挂载单元。

其它资源

- [fstab \(5\) 和 systemd.mount \(5\) 手册页](#)

18.2. 在 /ETC/FSTAB 中添加文件系统

在 /etc/fstab 配置文件中为文件系统配置持久挂载点。

流程

1. 找到文件系统的 UUID 属性：

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

例如：

例 18.2. 查看分区的 UUID

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

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
sda1	xfs	Boot	ea74bbec-536d-490c-b8d9-5b40bbd7545b	/boot

2. 如果挂载点目录不存在，请创建它：

```
# mkdir --parents mount-point
```

3.

以 **root** 用户身份，编辑 **/etc/fstab** 文件，并为文件系统添加一行，由 **UUID** 标识。

例如：

**例 18.3. /etc/fstab 中的 /boot 挂载点**

```
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot xfs 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.

尝试挂载文件系统来验证配置是否正常工作：

```
# mount mount-point
```

## 其它资源



[持久性命名属性概述](#)

## 第 19 章 根据需要挂载文件系统

作为系统管理员，您可以将 NFS 等文件系统配置为按需自动挂载。

### 19.1. AUTOFS 服务

**autofs** 服务可以自动挂载和卸载文件系统（按需），从而节省了系统资源。它可用于挂载文件系统，如 NFS、AFS、SMBFS、CIFS、CIFS 和本地文件系统。

使用 `/etc/fstab` 配置进行永久挂载的一个缺点是，无论用户访问挂载的文件系统的频率有多低，系统都必须投入资源来保持装载的文件系统。例如，当系统同时维护多个系统的 NFS 挂载时，这可能会影响系统性能。

`/etc/fstab` 的替代方法是使用基于内核的 **autofs** 服务。它由以下组件组成：

- 实施文件系统的内核模块，以及
- 执行所有其他功能的用户空间服务。

#### 其它资源

- 系统中的 **autofs** (8) 手册页

### 19.2. AUTOFS 配置文件

本节描述了 **autofs** 服务所使用的配置文件的用法和语法。

#### 主映射文件

**autofs** 服务使用 `/etc/auto.master`（主映射）作为其默认的主配置文件。这可以通过使用 `/etc/autofs.conf` 配置文件中的 **autofs** 配置以及名称服务开关(NSS)机制来将其更改为使用其他受支持的网络源和名称。

所有 **on-demand** 挂载点都必须在主映射中配置。挂载点、主机名、导出的目录和选项都可以在一组文件（或其他支持的网络源）中指定，而不必为每个主机手动配置它们。

主映射文件列出了 **autofs** 控制的挂载点，以及它们相应的配置文件或网络来源（称为自动挂载映射）。**master** 映射的格式如下：

```
mount-point map-name options
```

使用这种格式的变量有：

### ***mount-point***

**autofs** 挂载点；例如，`/mnt/data`。

### ***map-file***

映射源文件，其中包含挂载点列表以及应该挂载这些挂载点的文件系统的位置。

### ***options***

如果提供了这个选项，则它们适用于给定映射中的所有条目（如果它们本身没有指定选项的话）。

#### **例 19.1. /etc/auto.master 文件**

以下是 `/etc/auto.master` 文件中的一个示例行：

```
/mnt/data /etc/auto.data
```

### **映射文件**

映射文件配置单个 **on-demand** 挂载点的属性。

如果目录不存在，自动挂载程序会创建它们。如果在自动挂载程序启动之前目录已存在，则自动挂载程序在退出时不会删除它们。如果指定了超时，则如果在超时时间内没有访问该目录，则目录会被自动卸载。

映射的一般格式与主映射类似。但是，**options** 字段会出现在挂载点和位置之间，而不是像 **master** 映射那样在条目的末尾：

```
mount-point options location
```

使用这种格式的变量有：

### **mount-point**

这指的是 **autofs** 挂载点。这可以是间接挂载的单个目录名称，也可以是直接挂载的挂载点的完整路径。每个直接和间接映射条目键（*挂载点*）后面都跟着一个以空格分隔的偏移目录列表（每个子目录名称都以 / 开头），这就是所谓的多挂载条目。

### **options**

在提供这个选项时，这些选项将附加到主映射条目选项（如果有的话），或者如果配置条目 **append\_options** 设为 **no**，则使用这些选项代替主映射选项。

### **location**

这指的是文件系统的位置，如本地文件系统路径（对于以 / 开头的映射名称，前面带有 **Sun** 映射格式转义字符：）、**NFS** 文件系统或其他有效的文件系统位置。

## 例 19.2. 映射文件

以下是映射文件（例如 **/etc/auto.misc**）中的一个示例：

```
payroll -fstype=nfs4 personnel:/exports/payroll
sales -fstype=xfs :/dev/hda4
```

映射文件中的第一列指示 **autofs** 挂载点：来自名为 **personnel** 的服务器的 **sales** 和 **payroll**。第二列指示 **autofs** 挂载的选项。第三列显示挂载源。

根据给定的配置，**autofs** 挂载点将是 **/home/payroll** 和 **/home/sales**。通常省略 **-fstype=** 选项，如果文件系统是 **NFS**，则不需要该选项，如果系统默认是 **NFS** 挂载的 **NFSv4**，则包括 **NFSv4** 的挂载。

使用给定配置时，如果进程需要访问 **autofs** 卸载的目录，如 **/home/payroll/2006/July.sxc**，则 **autofs** 服务会自动挂载该目录。

## amd 映射格式

**autofs** 服务也识别 **amd** 格式的映射配置。如果要重复使用为 **am-utils** 服务编写的现有的自动挂载程序配置（已从 Red Hat Enterprise Linux 中删除），这将非常有用。

但是，红帽建议使用前面章节中描述的更简单的 **autofs** 格式。

## 其它资源

- **autofs (5)、autofs.conf (5) 和 auto.master (5) man page**
- **/usr/share/doc/autofs/README.amd-maps 文件**

## 19.3. 配置 AUTOFS 挂载点

使用 **autofs** 服务配置按需挂载点。

## 先决条件

- 安装 **autofs** 软件包：

```
# yum install autofs
```

- 启动并启用 **autofs** 服务：

```
# systemctl enable --now autofs
```

## 流程

1. 为位于 **/etc/auto.*identifier*** 的按需挂载点创建一个映射文件。使用标识挂载点的名称替换 ***identifier***。

2. 在映射文件中，输入挂载点、选项和位置字段，如 [autofs 配置文件部分中所述](#)。
3. 注册主映射文件中的映射文件，如 [autofs 配置文件 部分中所述](#)。
4. 允许服务重新读取配置，以便它可以管理新配置的 **autofs** 挂载：

```
# systemctl reload autofs.service
```

5. 尝试访问 **on-demand** 目录中的内容：

```
# ls automounted-directory
```

## 19.4. 使用 AUTOFS 服务自动挂载 NFS 服务器用户主目录

配置 **autofs** 服务，以自动挂载用户主目录。

### 先决条件

- 已安装 **autofs** 软件包。
- **autofs** 服务已启用并正在运行。

### 流程

1. 通过在需要挂载用户主目录的服务器上编辑 **/etc/auto.master** 文件，指定映射文件的挂载点和位置。要做到这一点，请在 **/etc/auto.master** 文件中添加以下行：

```
/home /etc/auto.home
```

2. 在需要挂载用户主目录的服务器中创建名为 **/etc/auto.home** 的映射文件，并使用以下参数编辑该文件：

```
* -fstype=nfs,rw,sync host.example.com:/home/&
```

您可以跳过 *fstype* 参数，因为它默认为 *nfs*。如需更多信息，请参阅您系统中的 *autofs* (5) 手册页。

3.

重新载入 *autofs* 服务：

```
# systemctl reload autofs
```

## 19.5. 覆盖或添加 AUTOFS 站点配置文件

有时覆盖客户端系统上特定挂载点的站点默认值会很有用。

### 例 19.3. 初始条件

例如，请考虑以下情况：

- 自动挂载程序映射存储在 NIS 中，*/etc/nsswitch.conf* 文件具有以下指令：

```
automount: files nis
```

- *auto.master* 文件包含：

```
+auto.master
```

- NIS *auto.master* 映射文件包含：

```
/home auto.home
```

- NIS *auto.home* 映射包含：

```
beth fileserver.example.com:/export/home/beth
joe fileserver.example.com:/export/home/joe
* fileserver.example.com:/export/home/&
```



- **autofs 配置选项 BROWSE\_MODE 设为 yes :**

```
BROWSE_MODE="yes"
```

- **文件映射 /etc/auto.home 不存在。**

## 流程

这部分描述了从不同服务器挂载主目录的示例，并使用所选条目增强 **auto.home**。

### 例 19.4. 从不同服务器挂载主目录

根据上述条件，假设客户端系统需要覆盖 NIS 映射 **auto.home**，并从其他服务器挂载主目录。

- **在这种情况下，客户端需要使用以下 /etc/auto.master 映射 :**

```
/home /etc/auto.home
+auto.master
```

- **/etc/auto.home 映射包含条目 :**

```
* host.example.com:/export/home/&
```

由于自动挂载程序仅处理第一次出现的挂载点，即包含 **/etc/auto.home** 内容的 **/home** 目录，而不是 NIS **auto.home** 映射。

### 例 19.5. 仅使用所选条目增强 auto.home

或者，使用几个条目来增加站点范围的 **auto.home** 映射：

1. **创建一个 /etc/auto.home 文件映射，并在其中放置新条目。在结尾处，包含 NIS auto.home 映射。然后 /etc/auto.home 文件映射类似：**

```
mydir someserver:/export/mydir
+auto.home
```

2.

有了这些 NIS `auto.home` 映射条件，列出 `/home` 目录输出的内容：

```
$ ls /home

beth joe mydir
```

最后一个示例按预期工作，因为 `autofs` 不包含与正在读取的文件映射同名的文件映射的内容。因此，`autofs` 转到 `nsswitch` 配置中的下一个映射源。

## 19.6. 使用 LDAP 存储自动挂载器映射

配置 `autofs` 以将自动挂载程序映射存储在 LDAP 配置中，而不是存储在 `autofs` 映射文件中。

### 先决条件

- 必须在所有配置的系统安装 LDAP 客户端程序库，以便从 LDAP 检索自动挂载器映射。在 Red Hat Enterprise Linux 上，`openldap` 软件包应作为 `autofs` 软件包的依赖项自动安装。

### 流程

1.

要配置 LDAP 访问，请修改 `/etc/openldap/ldap.conf` 文件。确保为您的站点正确设置了 `BASE`、`URI` 和 `schema` 选项。

2.

`rfc2307bis` 草案中描述了最近建立的用于在 LDAP 中存储自动映射的模式。要使用此模式，请在 `/etc/autofs.conf` 配置文件中通过删除模式定义中的注释字符来设置它。例如：

#### 例 19.6. 设置 `autofs` 配置

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

3.

确保配置中所有其他模式条目都被注释了。rfc2307bis 模式的 automountKey 属性替换 rfc2307 模式的 cn 属性。以下是 LDAP 数据交换格式(LDIF)配置的一个示例：

### 例 19.7. LDIF 配置

```
# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.master

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
automountKey: /home
automountInformation: auto.home

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: foo
automountInformation: filer.example.com:/export/foo

# /, auto.home, example.com
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

### 其它资源



[rfc2307bis 草案](#)

## 19.7. 使用 SYSTEMD.AUTOMOUNT 在 /ETC/FSTAB 按需挂载文件系统

当在 `/etc/fstab` 中定义挂载点时，请使用自动挂载 `systemd` 单元根据需要挂载文件系统。您必须为每个挂载添加自动挂载单元并启用它。

### 流程

- 1.

添加所需的 `fstab` 条目，如 [永久挂载文件系统](#) 中所述。例如：

```
/dev/disk/by-id/da875760-edb9-4b82-99dc-5f4b1ff2e5f4 /mount/point xfs defaults 0 0
```

2.

将 **x-systemd.automount** 添加到上一步中创建的条目的 **options** 字段中。

3.

加载新创建的单元，以便您的系统注册新配置：

```
# systemctl daemon-reload
```

4.

启动自动挂载单元：

```
# systemctl start mount-point.automount
```

## 验证

1.

检查 **mount-point.automount** 是否正在运行：

```
# systemctl status mount-point.automount
```

2.

检查自动挂载的目录是否有所需的内容：

```
# ls /mount/point
```

## 其它资源

- 在您的系统上 **systemd.automount (5)** 和 **systemd.mount (5)** 手册页
- [管理 systemd](#)

## 19.8. 使用 SYSTEMD.AUTOMOUNT 通过挂载单元根据需要挂载文件系统

当挂载点由挂载单元定义时，请使用自动挂载 **systemd** 单元来按需挂载文件系统。您必须为每个挂载添加自动挂载单元并启用它。

## 流程

1. 创建挂载单元。例如：

```
mount-point.mount
[Mount]
What=/dev/disk/by-uuid/f5755511-a714-44c1-a123-cfde0e4ac688
Where=/mount/point
Type=xfs
```

2. 创建一个名称与挂载单元相同的单元文件，但扩展 **.automount**。

3. 打开文件并创建 **[Automount]** 部分。将 **Where=** 选项设置为挂载路径：

```
[Automount]
Where=/mount/point
[Install]
WantedBy=multi-user.target
```

4. 加载新创建的单元，以便您的系统注册新配置：

```
# systemctl daemon-reload
```

5. 启用并启动自动挂载单元：

```
# systemctl enable --now mount-point.automount
```

## 验证

1. 检查 **mount-point.automount** 是否正在运行：

```
# systemctl status mount-point.automount
```

2. 检查自动挂载的目录是否有所需的内容：

```
# ls /mount/point
```

## 其它资源

- 在您的系统上 **systemd.automount (5)** 和 **systemd.mount (5)** 手册页
- [管理 systemd](#)

## 第 20 章 使用 IDM 中的 SSSD 组件来缓存 AUTOFS 映射

系统安全服务守护进程(SSSD)是一种系统服务，来访问远程服务目录和身份验证机制。当网络连接较慢时，数据缓存非常有用。要将 SSSD 服务配置为缓存 autofs 映射，请按照本节中的以下步骤操作。

### 20.1. 手动配置 AUTOFS，来将 IDM 服务器用作 LDAP 服务器

配置 autofs，以将 IdM 服务器用作 LDAP 服务器。

#### 流程

1. 编辑 `/etc/autofs.conf` 文件，来指定 autofs 搜索的模式属性：

```
#
# Other common LDAP naming
#
map_object_class = "automountMap"
entry_object_class = "automount"
map_attribute = "automountMapName"
entry_attribute = "automountKey"
value_attribute = "automountInformation"
```



#### 注意

用户可以在 `/etc/autofs.conf` 文件中小写和大写形式写入属性。

2. 可选：指定 LDAP 配置。有两种方法可以做到这一点：最简单的方法是让自动挂载服务自行发现 LDAP 服务器和位置：

```
ldap_uri = "ldap:///dc=example,dc=com"
```

这个选项要求 DNS 包含可发现服务器的 SRV 记录。

或者，明确设置要使用的 LDAP 服务器，以及用于 LDAP 搜索的基本 DN：

```
ldap_uri = "ldap://ipa.example.com"
search_base = "cn=location,cn=automount,dc=example,dc=com"
```

3.

编辑 `/etc/autofs_ldap_auth.conf` 文件，以便 `autofs` 允许客户端通过 `IdM LDAP` 服务器进行身份验证。

•

将 `authrequired` 更改为 `yes`。

•

将主体设置为 `IdM LDAP` 服务器( `host/FQDN@REALM`)的 `Kerberos` 主机主体。主体名称用于连接 `IdM` 目录，来作为 `GSS` 客户端身份验证的一部分。

```
<autofs_ldap_sasl_conf
  usetls="no"
  tlsrequired="no"
  authrequired="yes"
  authtype="GSSAPI"
  clientprinc="host/server.example.com@EXAMPLE.COM"
/>
```

有关主机主体的更多信息，请参阅在 [在 IdM 中使用规范化的 DNS 主机名](#)。

如有必要，请运行 `klist -k` 来获取确切的主机主体信息。

## 20.2. 配置 SSSD 来缓存 AUTOFS 映射

`SSSD` 服务可用于缓存存储在 `IdM` 服务器上的 `autofs` 映射，而无需配置 `autofs` 以使用 `IdM` 服务器。

### 先决条件

•

`sssd` 软件包已安装。

### 流程

1.

打开 `SSSD` 配置文件：

```
# vim /etc/sssd/sssd.conf
```



2.

将 **autofs** 服务添加到由 **SSSD** 处理的服务列表中。

```
[sssd]
domains = ldap
services = nss,pam,autofs
```

3.

创建一个新的 **[autofs]** 部分。您可以将此留空，因为 **autofs** 服务的默认设置适用于大多数基础架构。

```
[nss]

[pam]

[sudo]

[autofs]

[ssh]

[pac]
```

如需更多信息，请参阅系统中的 **sssd.conf** 手册页。

4.

可选：为 **autofs** 条目设置搜索基础。默认情况下，这是 **LDAP** 搜索库，但可以在 **ldap\_autofs\_search\_base** 参数中指定子树。

```
[domain/EXAMPLE]

ldap_search_base = "dc=example,dc=com"
ldap_autofs_search_base = "ou=automount,dc=example,dc=com"
```

5.

重启 **SSSD** 服务：

```
# systemctl restart sssd.service
```

6.

检查 **/etc/nsswitch.conf** 文件，以便 **SSSD** 被列为自动挂载配置的源：

```
automount: sss files
```

7.

重启 **autofs** 服务：

```
# systemctl restart autofs.service
```

8.

通过列出用户的 **/home** 目录来测试配置，假设 **/home** 有一个主映射条目：

```
# ls /home/userName
```

如果这没有挂载远程文件系统，请检查 **/var/log/messages** 文件是否有错误。如有必要，通过将 **logging** 参数设为 **debug** 来提高 **/etc/sysconfig/autofs** 文件的 **debug** 级别。

## 第 21 章 为 ROOT 文件系统设置只读权限

有时，您需要使用只读权限挂载 **root** 文件系统(/)。示例用例包括在系统意外断电后增强安全性或确保数据完整性。

### 21.1. 始终保留写入权限的文件和目录

要使系统正常工作，一些文件和目录需要保留写权限。当 **root** 文件系统以只读模式挂载时，这些文件将使用 **tmpfs** 临时文件系统挂载到 **RAM** 中。

这些文件和目录的默认集合是从 **/etc/rwtab** 文件中读取的。请注意，需要 **read-root** 软件包才能使您的系统中存在此文件。

```
dirs /var/cache/man
dirs /var/gdm
<content truncated>

empty /tmp
empty /var/cache/foomatic
<content truncated>

files /etc/adjtime
files /etc/ntp.conf
<content truncated>
```

**/etc/rwtab** 文件中的条目遵循以下格式：

```
copy-method path
```

在这个语法中：

- 用指定如何将文件或者目录复制到 **tmpfs** 的关键字之一替换 *copy-method*。
- 使用到文件或目录的路径替换 *path*。

**/etc/rwtab** 文件可识别将文件或目录复制到 **tmpfs** 的以下方法：

## empty

一个空路径被复制到 **tmpfs**。例如：

```
empty /tmp
```

## dirs

目录树被复制到 **tmpfs**。例如：

```
dirs /var/run
```

## files

将文件或目录树完整地复制到 **tmpfs**。例如：

```
files /etc/resolv.conf
```

在向 **/etc/rwtab.d/** 添加自定义路径时，也适用相同的格式。

## 21.2. 将 ROOT 文件系统配置为在引导时使用只读权限挂载

使用这个流程时，根文件系统将以只读方式安装在所有后续引导上。

### 流程

1. 在 **/etc/sysconfig/readonly-root** 文件中，将 **READONLY** 选项设置为 **yes**，以只读方式挂载文件系统：

```
READONLY=yes
```

2. 在 **/etc/fstab** 文件中的 **root** 条目(**/**)中添加 **ro** 选项：

```
/dev/mapper/luks-c376919e... / xfs x-systemd.device-timeout=0,ro 1 1
```

3. 启用 **ro** 内核选项：

```
# grubby --update-kernel=ALL --args="ro"
```

4.

确定禁用了 **rw kernel** 选项：

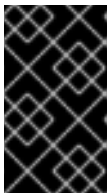
```
# grubby --update-kernel=ALL --remove-args="rw"
```

5.

如果您需要在 **tmpfs** 文件系统中添加需要挂载具有写权限的文件和目录，请在 **/etc/rwtab.d/** 目录中创建一个文本文件，并将配置放在其中。

例如：要将 **/etc/example/file** 文件挂载为具有写权限，请将此行添加到 **/etc/rwtab.d/example** 文件中：

```
files /etc/example/file
```



**重要**

对 **tmpfs** 中的文件和目录所做的更改不会在启动后保留。

6.

重启系统以应用更改。

## 故障排除

- 

如果您错误地将 **root** 文件系统挂载为具有只读权限，则可以使用以下命令再次将其重新挂载为具有读写权限：

```
# mount -o remount,rw /
```

## 第 22 章 对带有配额的 XFS 限制存储空间的使用

您可以使用磁盘配额来限制用户或组群可用的磁盘空间量。您还可以定义一个警告级别，在用户消耗太多磁盘空间或分区已满前通知系统管理员。

XFS 配额子系统管理对磁盘空间（块）和文件(inode)使用情况的限制。XFS 配额控制或报告在用户、组群、目录或项目级别使用这些项目的使用情况。组和项目配额只适用于旧的非默认 XFS 磁盘格式。

在按目录或按项目管理时，XFS 管理与特定项目相关联的目录层次结构的磁盘使用情况。

### 22.1. 磁盘配额

在大多数计算环境中，磁盘空间不会是有限的。quota 子系统提供控制磁盘空间使用的机制。

您可以为独立用户和本地文件系统上的用户组群配置磁盘配额。这样就使得可以将分配给用户特定文件(如电子邮件)的空间与分配给用户所从事的项目的空间分开来管理。配额子系统在用户超过分配的限制时警告用户，但会为当前的工作提供一些额外的空间（硬限制/软限制）。

如果实施了配额，您需要检查是否超过了配额，并确保配额准确。如果用户重复超过配额或者持续达到其软限制，则系统管理员可以帮助用户确定如何使用较少的磁盘空间或增加用户的磁盘配额。

您可以通过配额设置来控制：

- 消耗的磁盘块数量。
- 内节点数，这是在 UNIX 文件系统中包含文件信息的数据结构。由于 inode 存储与文件相关的信息，因此这允许控制可创建的文件数。

### 22.2. XFS\_QUOTA 工具

您可以使用 xfs\_quota 工具来管理 XFS 文件系统上的配额。另外，您可以使用关闭了限制强制的 XFS 文件系统作为有效的磁盘用量记帐系统。

**XFS 配额系统**在许多方面与其他文件系统不同。最重要的是，**XFS 将配额信息视为文件系统元数据**，并使用日志记录来提供更高级别的一致性保证。

## 其它资源

- [您系统上的 xfs\\_quota \(8\) 手册页](#)

## 22.3. XFS 中的文件系统配额管理

**XFS 配额子系统管理对磁盘空间（块）和文件(inode)使用情况的限制。XFS 配额控制或报告在用户、组群、目录或项目级别使用这些项目的使用情况。组和项目配额只适用于旧的非默认 XFS 磁盘格式。**

**在按目录或按项目管理时，XFS 管理与特定项目相关联的目录层次结构的磁盘使用情况。**

## 22.4. 为 XFS 启用磁盘配额

**在 XFS 文件系统上为用户、组和项目启用磁盘配额。启用配额后，xfs\_quota 工具可用来设置限制并报告磁盘使用情况。**

## 流程

1. **为用户启用配额：**

```
# mount -o uquota /dev/xvdb1 /xfs
```

**使用 uqnoenforce 替换 uquota，以允许在不强制实施任何限制的情况下报告使用情况。**

2. **为组群启用配额：**

```
# mount -o gquota /dev/xvdb1 /xfs
```

**使用 gqnoenforce 替换 gquota，以允许在不强制实施任何限制的情况下报告使用情况。**

3.

为项目启用配额：

```
# mount -o pquota /dev/xvdb1 /xfs
```

将 **pquota** 替换为 **pqnoenforce**，以允许在不强制实施任何限制的情况下报告使用情况。

4.

或者，也可以在 **/etc/fstab** 文件中包含配额挂载选项。以下示例显示了 **/etc/fstab** 文件中用来分别在 **XFS** 文件系统上为用户、组和项目启用配额的条目。这些示例还使用读写权限挂载文件系统：

```
# vim /etc/fstab
/dev/xvdb1 /xfs xfs rw,quota 0 0
/dev/xvdb1 /xfs xfs rw,gquota 0 0
/dev/xvdb1 /xfs xfs rw,prjquota 0 0
```

## 其它资源

•

您系统上的 **XFS (5)** 和 **xfs\_quota (8)** 手册页

## 22.5. 报告 XFS 使用量

使用 **xfs\_quota** 工具设置限制并报告磁盘使用情况。默认情况下，**xfs\_quota** 以交互方式运行，并处于基本模式。基本模式子命令只是报告使用情况，适用于所有用户。

## 先决条件

•

为 **XFS** 文件系统启用配额。请参阅为 **XFS** 启用磁盘配额。

## 流程

1.

启动 **xfs\_quota shell**：

```
# xfs_quota
```

2.

显示给定用户的使用情况和限制：

■



```
xfs_quota> quota username
```

3.

显示块和内节点的空闲和已使用的数量：

```
xfs_quota> df
```

4.

运行 **help** 命令来显示 **xfs\_quota** 可使用的基本命令。

```
xfs_quota> help
```

5.

指定 **q** 来退出 **xfs\_quota**。

```
xfs_quota> q
```

#### 其它资源

- 您系统上的 **xfs\_quota (8)** 手册页

## 22.6. 修改 XFS 配额限制

启动带有 **-x** 选项的 **xfs\_quota** 工具，来启用专家模式，并运行管理员命令，该命令允许修改配额系统。此模式的子命令允许实际限制的配置，并且仅可提供给具有升级特权的用户使用。

#### 先决条件

- 为 XFS 文件系统启用配额。请参阅为 [XFS 启用磁盘配额](#)。

#### 流程

1.

启动带有 **-x** 选项的 **xfs\_quota shell**，来启用专家模式：

```
# xfs_quota -x /path
```

2.

报告具体文件系统的配额信息：

■

```
xfs_quota> report /path
```

例如，若要显示 **/home** 的配额报告示例（在 **/dev/blockdevice** 上），请使用命令 **report -h /home**。此时会显示类似如下的输出：

```
User quota on /home (/dev/blockdevice)
Blocks
User ID    Used  Soft  Hard Warn/Grace
-----
root        0    0    0 00 [-----]
testuser  103.4G    0    0 00 [-----]
```

3.

**修改配额限制：**

```
xfs_quota> limit isoft=500m ihard=700m user
```

例如，要为用户 **john**（其主目录为 **/home/john**）软和硬 **inode** 数限制分别设置为 **500** 和 **700**，请使用以下命令：

```
# xfs_quota -x -c 'limit isoft=500 ihard=700 john' /home/
```

在这种情况下，传递 **mount\_point**，这是挂载的 **xfs** 文件系统。

4.

**显示 xfs\_quota -x 可用的专家命令：**

```
xfs_quota> help
```

验证

•

**验证配额限制是否已修改：**

```
xfs_quota> report -i -u
User quota on /home (/dev/loop0)
Inodes
User ID    Used  Soft  Hard  Warn/ Grace
-----
root        3    0    0 00 [-----]
testuser    2   500   700 00 [-----]
```

其它资源

- [您系统上的 xfs\\_quota \(8\) 手册页](#)

## 22.7. 为 XFS 设置项目限制

为项目控制的目录配置限值。

### 流程

1. 将项目控制的目录添加到 `/etc/projects`。例如，以下命令将唯一 ID 为 11 的 `/var/log` 路径添加到 `/etc/projects`：您的项目 ID 可以是任何映射到项目的数字值。

```
# echo 11:/var/log >> /etc/projects
```

2. 将项目名称添加到 `/etc/projid`，来将项目 ID 映射到项目名称。例如，以下命令将名为 `logfiles` 的项目与上一步中定义的项目 ID 11 关联。

```
# echo logfiles:11 >> /etc/projid
```

3. 初始化项目目录。例如，以下命令初始化项目目录 `/var`：

```
# xfs_quota -x -c 'project -s logfiles' /var
```

4. 为使用初始化目录的项目配置配额：

```
# xfs_quota -x -c 'limit -p bhard=1g logfiles' /var
```

### 其它资源

- [xfs\\_quota \(8\)](#), [projid \(5\)](#), 和 [projects \(5\) man pages](#)

## 第 23 章 对带有配额的 EXT4 限制存储空间使用

在分配前，您必须在系统中启用磁盘配额。您可以为每个用户、每个组或每个项目分配磁盘配额。但是，如果设置了软限制，您可以在一个可配置的期间内（称为宽限期）超过这些配额。

### 23.1. 安装配额工具

您必须安装 **quota RPM** 软件包才能实现磁盘配额。

#### 流程

- 安装 **quota** 软件包：

```
# yum install quota
```

### 23.2. 在创建文件系统时启用配额功能

在创建文件系统时启用配额。

#### 流程

1. 在创建文件系统时启用配额：

```
# mkfs.ext4 -O quota /dev/sda
```



#### 注意

默认仅启用和初始化用户和组群配额。

2. 更改创建文件系统时的默认设置：

```
# mkfs.ext4 -O quota -E quotatype=usrquota:grpquota:prjquota /dev/sda
```

3.

挂载文件系统：

```
# mount /dev/sda
```

## 其它资源

•

您系统上的 **ext4 (5)** 手册页。

## 23.3. 在现有文件系统中启用配额功能

使用 **tune2fs** 命令在现有文件系统中启用配额功能。

## 流程

1.

卸载文件系统：

```
# umount /dev/sda
```

2.

在现有文件系统中启用配额：

```
# tune2fs -O quota /dev/sda
```



注意

默认只初始化用户和组群配额。

3.

更改默认值：

```
# tune2fs -Q usrquota,grpquota,prjquota /dev/sda
```

4.

挂载文件系统：

```
# mount /dev/sda
```

## 其它资源

- [您系统上的 ext4 \(5\) 手册页。](#)

## 23.4. 启用配额强制

在不使用任何额外选项挂载文件系统后，默认启用配额记帐，但不启用配额强制。

### 先决条件

- 启用配额功能，并初始化默认配额。

### 流程

- 通过 **quotaon** 为用户配额启用配额强制：

```
# mount /dev/sda /mnt
```

```
# quotaon /mnt
```



#### 注意

可以使用 **usrquota**、**grpquota** 或 **prjquota** 挂载选项在挂载时启用配额强制。

```
# mount -o usrquota,grpquota,prjquota /dev/sda /mnt
```

- 在所有文件系统中启用用户、组群和项目配额：

```
# quotaon -vaugP
```

- 如果未指定 **-u**、**-g** 或 **-P** 选项，则仅启用用户配额。
- 如果只指定 **-g** 选项，则只启用组配额。

- 如果只指定 **-P** 选项，则只启用项目配额。

- 为特定文件系统（如 `/home`）启用配额：

```
# quotaon -vugP /home
```

#### 其它资源

- 系统中的 `quotaon (8)` 手册页

### 23.5. 为每个用户分配配额

磁盘配额通过 `edquota` 命令分配给用户。



#### 注意

`edquota` 使用由 `EDITOR` 环境变量定义的文本编辑器。要更改编辑器，请将 `~/.bash_profile` 文件中的 `EDITOR` 环境变量设为您选择的编辑器的完整路径。

#### 先决条件

- 用户必须在设置用户配额前存在。

#### 流程

1. 为用户分配配额：

```
# edquota username
```

使用您要为其分配配额的用户替换 `username`。

例如，如果您为 `/dev/sda` 分区启用配额，并执行命令 `quota testuser`，则会在系统配置的默认编辑器中显示以下内容：

```
Disk quotas for user testuser (uid 501):
Filesystem blocks soft hard inodes soft hard
/dev/sda 44043 0 0 37418 0 0
```

2.

更改所需限制。

如果值为 0，则代表没有设定限制。在文本编辑器中更改它们。

例如，下面显示了 **testuser** 的软和硬限制，它们分别被设置为 **50000** 和 **55000**。

```
Disk quotas for user testuser (uid 501):
Filesystem blocks soft hard inodes soft hard
/dev/sda 44043 50000 55000 37418 0 0
```

- 第一列是启用了配额的文件系统的名称。
- 第二列显示目前该用户使用的块数。
- 下面的两列是为该用户在文件系统中设定软限制和硬限制。
- **inodes** 列显示用户当前使用的 **inodes** 数。
- 最后两列是为该用户在文件系统中设定软和硬的内节点限制。
  - 硬块限制是用户或者组群可以使用的绝对最大磁盘空间量。达到这个限制后，就无法再使用其他磁盘空间。
  - 软块限制定义可以使用的最大磁盘空间量。然而，与硬限制不同，在一定时间内可以超过软限制。这段时间被称为**宽限期**。宽限期可以用秒、分钟、小时、天、周或月表示。

验证



- 验证是否为该用户设定了配额：

```
# quota -v testuser
Disk quotas for user testuser:
Filesystem blocks quota limit grace files quota limit grace
/dev/sda    1000* 1000 1000      0   0   0
```

### 23.6. 为每个组群分配配额

您可以根据组群分配配额。

#### 先决条件

- 组群在设定组群配额前必须已经存在。

#### 流程

1. 设置组群配额：

```
# edquota -g groupname
```

例如，要为 **devel** 组设置组配额：

```
# edquota -g devel
```

这个命令在文本编辑器中显示该组群的现有配额：

```
Disk quotas for group devel (gid 505):
Filesystem blocks soft hard inodes soft hard
/dev/sda    440400  0   0 37418  0   0
```

2. 修改限制并保存文件。

#### 验证

- 验证是否设定了组群配额：

```
# quota -vg groupname
```

### 23.7. 为每个项目分配配额

您可以为每个项目分配配额。

#### 先决条件

- 在您的文件系统中启用了项目配额。

#### 流程

1. 将项目控制的目录添加到 `/etc/projects`。例如，以下命令将唯一 ID 为 11 的 `/var/log` 路径添加到 `/etc/projects`：您的项目 ID 可以是任何映射到项目的数字值。

```
# echo 11:/var/log >> /etc/projects
```

2. 将项目名称添加到 `/etc/projid`，来将项目 ID 映射到项目名称。例如，以下命令将名为 `Logs` 的项目与上一步中定义的 ID 为 11 的项目相关联：

```
# echo Logs:11 >> /etc/projid
```

3. 设置所需的限制：

```
# edquota -P 11
```



#### 注意

您可以通过其项目 ID（本例中为 11）或其名称（本例中为 `Logs`）来选择项目。

4. 使用 `quotaon`，启用配额强制：

请参阅 [启用配额强制](#)。

## 验证

- 验证是否设置了项目配额：

```
# quota -vP 11
```



### 注意

您可以使用项目 ID 或项目名称验证。

## 其它资源

- [edquota \(8\)](#), [projid \(5\)](#), 和 [projects \(5\) man page](#)

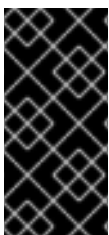
## 23.8. 为软限制设置宽限期

如果给定配额具有软限制，您可以编辑宽限期，这是可以超过软限制的时间。您可以为用户、组或项目设置宽限期。

## 流程

- 编辑宽限期：

```
# edquota -t
```



### 重要

虽然其它 **edquota** 命令针对特定用户、组或项目的配额操作，但 **-t** 选项在每个启用了配额的文件系统上操作。

## 其它资源

- [系统上的 edquota \(8\) 手册页](#)

## 23.9. 关闭文件系统配额

使用 **quotaoff** 来在指定的文件系统上关闭磁盘配额强制。执行此命令后可启用配额核算。

### 流程

- 关闭所有用户和组群配额：

```
# quotaoff -vaugP
```

- 如果未指定 **-u**、**-g** 或 **-P** 选项，则仅禁用用户配额。
- 如果只指定 **-g** 选项，则只禁用组配额。
- 如果只指定 **-P** 选项，则只禁用项目配额。
- **-v** 开关会在命令执行时显示详细状态信息。

### 其它资源

- [系统中的 quotaoff \(8\) 手册页](#)

## 23.10. 报告磁盘配额

使用 **repquota** 工具创建磁盘配额报告。

### 流程

1. 运行 **repquota** 命令：

■

```
# repquota
```

例如，命令 `repquota /dev/sda` 产生此输出：

```
*** Report for user quotas on device /dev/sda
Block grace time: 7days; Inode grace time: 7days
  Block limits  File limits
User  used soft hard grace used soft hard grace
-----
root  --   36   0   0         4   0   0
kristin --  540   0   0        125   0   0
testuser -- 440400 500000 550000      37418   0   0
```

2.

查看所有启用了配额的文件系统的磁盘用量报告：

```
# repquota -augP
```

每个用户后显示的 `--` 符号确定是否超过了块或 `inode` 限制。如果超过了任何一个软限制，则 `+` 字符会出现在相应的 `-` 字符的位置。第一个 `-` 字符表示块限制，第二个表示 `inode` 限制。

`grace` 列通常为空白。如果超过了软限制，则该列包含的时间规格等同于宽限期中剩余的时间量。如果宽限期过期了，则 `none` 会出现在其位置上。

其它资源

有关详细信息，请参阅 `repquota(8)` 手册页。

## 第 24 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使用未使用的物理块。

### 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。

### 24.1. 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，并实时触发，而无需用户干预。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** 服务定期运行的批量操作。

**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。

## 24.2. 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
```

#### 其它资源

- [您系统上的 fstrim \(8\) 手册页](#)

### 24.3. 启用在线块丢弃

您可以执行在线块丢弃操作，以便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** 文件的挂载条目中。

#### 其它资源

- [在您的系统中 mount \(8\) 和 fstab \(5\) 手册页](#)

### 24.4. 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'
```

其它资源

•

[/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) file

•

[/usr/share/doc/rhel-system-roles/storage/](#) directory

## 24.5. 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 25 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。技术预览功能不被红帽产品服务等级协议(SLA)支持，且功能可能并不完善。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更多信息，请参阅 <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**

## 25.1. 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。

## 25.2. 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。

## 25.3. 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
```

## 25.4. 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
```

## 25.5. 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。

## 25.6. 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
```

## 25.7. 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。

## 25.8. 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**。

## 25.9. 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的应用程序，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- 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- eb7ae9e1e540	

## 25.10. 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。

**其它资源**

- [使用基于策略的解密配置加密卷的自动解锁](#)

**25.11. 将 STRATIS 池绑定到 TPM**

当您为加密的 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。

## 25.12. 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
```

## 25.13. 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。

#### 其它资源

- [将加密的 Stratis 池绑定到 NBDE](#)
- [将加密的 Stratis 池绑定到 TPM](#)

### 25.14. 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。

停止的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
```

**25.15. 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 25.1. 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
```

**其它资源**

- [挂载 Stratis 文件系统](#)

**25.16. 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。

### 25.17. 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。

## 25.18. 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

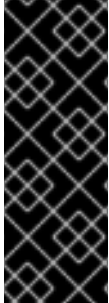
从加密的 **Stratis** 池永久挂载 **Stratis** 文件系统可能会导致引导过程停止，直到提供密码为止。如果池使用任何无人值守机制进行加密，例如 **NBDE** 或 **TPM2**，则 **Stratis** 池将自动解锁。如果没有，用户需要在控制台中输入密码。

### 其它资源

- [永久挂载文件系统](#)

## 第 26 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。技术预览功能不被红帽产品服务等级协议(SLA)支持，且功能可能并不完善。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更多信息，请参阅 <https://access.redhat.com/support/offerings/techpreview>。

### 26.1. 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
```

#### 其它资源

- [系统中 Stratis \(8\) 手册页](#)

### 26.2. 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 27 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。技术预览功能不被红帽产品服务等级协议(SLA)支持，且功能可能并不完善。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更多信息，请参阅 <https://access.redhat.com/support/offerings/techpreview>。

## 27.1. 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
```

## 其它资源

- 系统中 **Stratis (8)** 手册页

## 27.2. 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

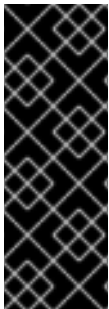
1. **登录到 RHEL 8 web 控制台。**
2. **点 Storage。**
3. **在 Storage 表中，点您要查看的 Stratis 池。**

**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 28 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。技术预览功能不被红帽产品服务等级协议(SLA)支持，且功能可能并不完善。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更多信息，请参阅 <https://access.redhat.com/support/offerings/techpreview>。

### 28.1. 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。

### 28.2. 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**。

## 其它资源

- **系统中 Stratis (8) 手册页**

## 28.3. 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
```

## 其它资源

- [挂载 Stratis 文件系统](#)
- [在您的系统中的 mount \(8\) 手册页](#)

## 28.4. 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。

#### 其它资源

- [系统中 Stratis \(8\) 手册页](#)

## 28.5. 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：

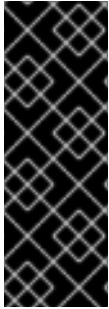
```
# stratis filesystem destroy my-pool my-fs-snapshot
```

#### 其它资源

- [系统中 Stratis \(8\) 手册页](#)

## 第 29 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。技术预览功能不被红帽产品服务等级协议(SLA)支持，且功能可能并不完善。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更多信息，请参阅 <https://access.redhat.com/support/offerings/techpreview>。

### 29.1. 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
```

## 其它资源

- [系统中 Stratis \(8\) 手册页](#)

## 29.2. 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**。

### 29.3. 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
```

其它资源

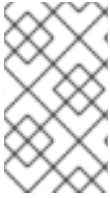
•

系统中 **Stratis (8)** 手册页

## 29.4. 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**。

## 第 30 章 开始使用 EXT3 文件系统

作为系统管理员，您可以创建、挂载、调整大小、备份和恢复 ext3 文件系统。ext3 文件系统基本上是一个 ext2 文件系统的改进版本。

### 30.1. EXT3 文件系统的特性

下面是 ext3 文件系统的特性：

- **可用性：**在意外的电源故障或系统崩溃后，由于提供了日志记录而不需要进行文件系统检查。默认的日志大小需要大约 1 秒钟才能恢复，具体取决于硬件的速度，



**注意**

ext3 中唯一支持的日志模式是 `data=ordered`（默认）。如需更多信息，请参阅红帽知识库解决方案 [RHEL 是否支持 EXT journaling 选项 "data=writeback"?](#)。

- **数据完整性：**ext3 文件系统可防止在意外的电源失败或系统崩溃时丢失数据完整性。
- **速度：**尽管可以多次写一些数据，但在大多数情况下，ext3 的吞吐量要高于 ext2，因为 ext3 的日志优化了硬盘磁头的运动。
- **轻松转换：**从 ext2 迁移到 ext3 很容易，而且不需要重新格式化就可以获得健壮的日志文件系统的好处。

#### 其它资源

- [系统中的 ext3 手册页](#)

### 30.2. 创建 EXT3 文件系统

作为系统管理员，您可以使用 `mkfs.ext3` 命令在块设备上创建 ext3 文件系统。

## 先决条件

- 您磁盘中的一个分区。有关创建 MBR 或 GPT 分区的详情，请参考 [使用 parted 在磁盘上创建分区表](#)。

+ 或者，使用 LVM 或者 MD 卷。

## 流程

1.

要创建一个 ext3 文件系统：

- 对于常规分区设备、LVM 卷、MD 卷或者类似的设备，使用以下命令：

```
# mkfs.ext3 /dev/block_device
```

使用到块设备的路径替换 `/dev/block_device`。

例如：`/dev/sdb1`、`/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a` 或 `/dev/my-volgroup/my-lv`。一般说来，默认选项适用于大多数使用场景。

- 对于条带块设备（如 RAID5 阵列），可以在创建文件系统时指定条带几何结构。使用正确的条带几何结构可提高 ext3 文件系统的性能。例如，要在 4k-块文件系统上创建跨距为 64k（即 16 x 4096）的文件系统，请使用以下命令：

```
# mkfs.ext3 -E stride=16,stripe-width=64 /dev/block_device
```

在给定示例中：

- **stride=value**：指定 RAID 块大小
- **stripe-width=value**：指定 RAID 设备中数据磁盘的数量，或者条带中的条带单元

的数量。

#### 注意

- 

在创建文件系统时指定 **UUID**：

```
# mkfs.ext3 -U UUID /dev/block_device
```

使用您要设置的 **UUID** 替换 **UUID**：例如，7cd65de3-e0be-41d9-b66d-96d749c02da7。

使用 **ext3** 文件系统的路径替换 **/dev/*block\_device***，以便将 **UUID** 添加给它：例如 **/dev/sda8**。

- 

在创建文件系统时指定标签：

```
# mkfs.ext3 -L label-name /dev/block_device
```

2.

查看所创建的 **ext3** 文件系统：

```
# blkid
```

#### 其它资源

- 

系统中的 **ext3** 和 **mkfs.ext3** 手册页

### 30.3. 挂载 EXT3 文件系统

作为系统管理员，您可以使用 **mount** 工具挂载 **ext3** 文件系统。

#### 先决条件

- 

**ext3** 文件系统。有关创建 **ext3** 文件系统的详情，请参考 [创建 ext3 文件系统](#)。

## 流程

1. 要创建一个挂载点来挂载文件系统：

```
# mkdir /mount/point
```

使用创建分区挂载点的目录名替换 */mount/point*。

2. 挂载 **ext3** 文件系统：

- 要挂载一个没有额外选项的 **ext3** 文件系统：

```
# mount /dev/block_device /mount/point
```

- 要永久挂载文件系统，请参阅 [永久挂载文件系统](#)。

3. 查看挂载的文件系统：

```
# df -h
```

## 其它资源

- 在您的系统中 挂载、**ext3** 和 **fstab** man page
- [挂载文件系统](#)

## 30.4. 重新定义 EXT3 文件系统大小

作为系统管理员，您可以使用 **resize2fs** 工具调整 **ext3** 文件系统的大小。**resize2fs** 工具以文件系统块大小为单位读取大小，除非使用后缀表示特定的单位。以下后缀代表特定的单元：

- **s**（扇区） - 512 字节扇区

- K(KB)- 1,024 字节
- M (兆字节) - 1,048,576 字节
- G(GB)- 1,073,741,824 字节
- T(TB)- 1,099,511,627,776 字节

### 先决条件

- ext3 文件系统。有关创建 ext3 文件系统的详情，请参考 [创建 ext3 文件系统](#)。
- 调整大小后可保留文件系统的基本块设备。

### 流程

1. 要重新定义 ext3 文件系统大小，请执行以下步骤：

- 要缩小并增大卸载的 ext3 文件系统的大小：

```
# umount /dev/block_device
# e2fsck -f /dev/block_device
# resize2fs /dev/block_device size
```

使用块设备的路径替换 `/dev/block_device`，例如 `/dev/sdb1`。

使用 `s`、`K`、`M`、`G` 和 `T` 后缀将 `size` 替换为所需的调整大小的值。

- 使用 `resize2fs` 命令挂载时，可以增大 ext3 文件系统：

```
# resize2fs /mount/device size
```



### 注意

扩展时 **size** 参数是可选的（通常是多余的）。**resize2fs** 会自动扩展来填充容器的可用空间，通常是逻辑卷或分区。

2.

查看重新定义大小的文件系统：

```
# df -h
```

### 其它资源



在您的系统上 **resize2fs**、**e2fsck** 和 **ext3 man page**



## 第 31 章 EXT4 文件系统入门

作为系统管理员，您可以创建、挂载、调整大小、备份和恢复 ext4 文件系统。ext4 文件系统是 ext3 文件系统的可扩展扩展。使用 Red Hat Enterprise Linux 8，它可以支持的最大的文件大小为 16 TB，支持的最大的文件系统大小为 50 TB。

### 31.1. EXT4 文件系统的特性

以下是 ext4 文件系统的特性：

- - 使用数据块：ext4 文件系统使用数据块，这可在使用大型文件时提高性能，并减少大型文件的元数据开销。
- - Ext4 相应地标记未分配的块组和 inode 表部分，这允许在文件系统检查期间跳过块组和表部分。它可快速进行文件系统检查，随着文件系统大小的增加，该检查将变得更加有益。
- - 元数据校验和：默认情况下，在 Red Hat Enterprise Linux 8 中启用了这个功能。
- - ext4 文件系统的分配特性：
    - 持久性预分配
    - 延迟分配
    - 多块分配
    - 条状化分配
- - 扩展属性(xattr)：这允许系统为每个文件关联多个额外名称和值对。

- **配额日志：**这避免了崩溃后需要很长时间的配额一致性检查。



#### 注意

**ext4** 中唯一支持的日志模式是 **data=ordered**（默认）。如需更多信息，请参阅红帽知识库解决方案 [RHEL 是否支持 EXT journaling 选项 "data=writeback"?](#)。

- **亚秒时间戳** - 这为次秒提供时间戳。

#### 其它资源

- [系统中的 ext4 手册页](#)

### 31.2. 创建 EXT4 文件系统

作为系统管理员，您可以使用 **mkfs.ext4** 命令在块设备上创建 **ext4** 文件系统。

#### 先决条件

- 您磁盘中的一个分区。有关创建 **MBR** 或 **GPT** 分区的详情，请参考 [使用 parted 在磁盘上创建分区表](#)。
- 另外，还可使用 **LVM** 或者 **MD** 卷。

#### 流程

1. 要创建 **ext4** 文件系统：
  - 对于常规分区设备、**LVM** 卷、**MD** 卷或者类似的设备，使用以下命令：

```
# mkfs.ext4 /dev/block_device
```

使用到块设备的路径替换 `/dev/block_device`。

例如：`/dev/sdb1`、`/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a` 或 `/dev/my-volgroup/my-lv`。一般说来，默认选项适用于大多数使用场景。

- 对于条带块设备（如 RAID5 阵列），可以在创建文件系统时指定条带几何结构。使用正确的条带几何结构可提高 ext4 文件系统的性能。例如，要在 4k-块文件系统中创建跨距为 64k（即 16 x 4096）的文件系统，请使用以下命令：

```
# mkfs.ext4 -E stride=16,stripe-width=64 /dev/block_device
```

在给定示例中：

- **stride=value**：指定 RAID 块大小
- **stripe-width=value**：指定 RAID 设备中数据磁盘的数量，或者条带中的条带单元的数量。

#### 注意

- 在创建文件系统时指定 UUID：

```
# mkfs.ext4 -U UUID /dev/block_device
```

使用您要设置的 UUID 替换 **UUID**：例如，`7cd65de3-e0be-41d9-b66d-96d749c02da7`。

使用 ext4 文件系统的路径替换 `/dev/block_device`，来将 UUID 添加给它：例如 `/dev/sda8`。

- 在创建文件系统时指定标签：

```
# mkfs.ext4 -L label-name /dev/block_device
```

2.

查看创建的 **ext4** 文件系统：

```
# blkid
```

## 其它资源

- [系统中的 ext4 和 mkfs.ext4 man page](#)

## 31.3. 挂载 EXT4 文件系统

作为系统管理员，您可以使用 **mount** 工具挂载 **ext4** 文件系统。

## 先决条件

- **ext4** 文件系统。有关创建 **ext4** 文件系统的详情，请参考 [创建 ext4 文件系统](#)。

## 流程

1.

要创建一个挂载点来挂载文件系统：

```
# mkdir /mount/point
```

使用创建分区挂载点的目录名替换 */mount/point*。

2.

挂载 **ext4** 文件系统：

- 要挂载一个没有额外选项的 **ext4** 文件系统：

```
# mount /dev/block_device /mount/point
```

- 要永久挂载文件系统，请参阅 [永久挂载文件系统](#)。

3.

查看挂载的文件系统：

```
# df -h
```

#### 其它资源

- 在您的系统中 挂载、**ext4** 和 **fstab man page**
- [挂载文件系统](#)

### 31.4. 重新定义 EXT4 文件系统大小

作为系统管理员，您可以使用 **resize2fs** 工具调整 **ext4** 文件系统的大小。**resize2fs** 工具以文件系统块大小为单位读取大小，除非使用后缀表示特定的单位。以下后缀代表特定的单元：

- **s**（扇区） - 512 字节扇区
- **K(KB)**- 1,024 字节
- **M**（兆字节） - 1,048,576 字节
- **G(GB)**- 1,073,741,824 字节
- **T(TB)**- 1,099,511,627,776 字节

#### 先决条件

- **ext4** 文件系统。有关创建 **ext4** 文件的详情，请参考 [创建 ext4 文件系统](#)。
- 调整大小后可保留文件系统的基本块设备。

## 流程

1.

要重新定义 **ext4** 文件系统大小，请执行以下步骤：



要缩小并增大卸载的 **ext4** 文件系统的大小：

```
# umount /dev/block_device
# e2fsck -f /dev/block_device
# resize2fs /dev/block_device size
```

使用块设备的路径替换 **/dev/block\_device**，例如 **/dev/sdb1**。

使用 **s**、**K**、**M**、**G** 和 **T** 后缀将 **size** 替换为所需的调整大小的值。



可以使用 **resize2fs** 命令在挂载时增大 **ext4** 文件系统：

```
# resize2fs /mount/device size
```



### 注意

扩展时 **size** 参数是可选的（通常是多余的）。**resize2fs** 会自动扩展来填充容器的可用空间，通常是逻辑卷或分区。

2.

查看重新定义大小的文件系统：

```
# df -h
```

## 其它资源



在您的系统上 **resize2fs**、**e2fsck** 和 **ext4** man page

## 31.5. 和 EXT4 和 XFS 一起使用的工具比较

这部分比较用于完成 ext4 和 XFS 文件系统中常用任务的工具。

任务	ext4	XFS
创建文件系统	mkfs.ext4	mkfs.xfs
文件系统检查	e2fsck	xfs_repair
重新定义文件系统大小	resize2fs	xfs_growfs
保存文件系统的镜像	e2image	xfs_metadump 和 xfs_mdrestore
标签或者调整文件系统	tune2fs	xfs_admin
备份文件系统	dump 和 restore	xfsdump 和 xfsrestore
配额管理	quota	xfs_quota
文件映射	filefrag	xfs_bmap