



# Red Hat Enterprise Linux 8

## 論理ボリュームの設定および管理

LVM 論理ボリュームの設定および管理のガイド





## 法律上の通知

Copyright © 2020 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.

## 概要

本書は、Red Hat Enterprise Linux 8 で、LVM 論理ボリュームを管理する方法を説明します。

## 目次

RED HAT ドキュメントへのフィードバック (英語のみ)	5
<b>第1章 論理ボリューム</b>	<b>6</b>
1.1. LVM アーキテクチャーの概要	6
1.2. 物理ボリューム	7
1.3. ボリュームグループ	9
1.4. LVM 論理ボリューム	9
<b>第2章 RHEL システムロールを使用したローカルストレージの管理</b>	<b>17</b>
2.1. ストレージロールの概要	17
2.2. ストレージロールのパラメーター	17
<b>第3章 LVM のデプロイメント</b>	<b>19</b>
3.1. LVM 物理ボリュームの作成	19
3.2. LVM ボリュームグループの作成	19
3.3. LVM 論理ボリュームの作成	19
<b>第4章 LVM 論理ボリュームの設定</b>	<b>21</b>
4.1. CLI コマンドの使用	21
4.2. 3つのディスク上での LVM 論理ボリューム作成	22
4.3. RAID0 (ストライピング) 論理ボリュームの作成	23
4.4. LVM 論理ボリュームの名前の変更	25
4.5. 論理ボリュームからのディスクの削除	26
4.6. 永続的なデバイス番号の設定	28
4.7. LVM エクステントサイズの指定	28
4.8. RHEL システムロールを使用した LVM 論理ボリュームの管理	28
4.9. LVM 論理ボリュームの削除	30
<b>第5章 論理ボリュームのサイズ変更</b>	<b>31</b>
5.1. 論理ボリュームの拡張	31
5.2. 論理ボリュームのファイルシステムの拡張	31
5.3. 論理ボリュームの縮小	32
5.4. ストライプ化論理ボリュームの拡張	33
<b>第6章 LVM ボリュームグループの管理</b>	<b>35</b>
6.1. ボリュームグループ	35
6.2. ボリュームグループの表示	35
6.3. ボリュームグループの統合	36
6.4. ボリュームグループの分割	36
6.5. LVM ボリュームグループの名前変更	37
6.6. ボリュームグループを別のシステムへ移動	38
6.7. ボリュームグループからの物理ボリュームの削除	39
6.8. LVM ボリュームグループの削除	40
6.9. 関連情報	40
<b>第7章 LVM 物理ボリュームの管理</b>	<b>41</b>
7.1. 物理ボリュームとして使用するブロックデバイスのスキャン	41
7.2. 物理ボリュームのパーティションタイプの設定	41
7.3. LVM 物理ボリュームのサイズ変更	42
7.4. 物理ボリュームの削除	42
7.5. ボリュームグループへの物理ボリュームの追加	42
7.6. ボリュームグループからの物理ボリュームの削除	42

<b>第8章 LVM コンポーネントの表示</b> .....	<b>44</b>
8.1. LVM コマンドによる LVM 情報の表示	44
8.2. 物理ボリュームの表示	44
8.3. ボリュームグループの表示	45
8.4. 論理ボリュームの表示	46
<b>第9章 LVM 用のカスタム報告</b> .....	<b>47</b>
9.1. LVM 表示の形式の制御	47
9.2. LVM オブジェクト表示フィールド	48
9.3. LVM 報告のソート	57
9.4. LVM レポート表示への単位の指定	58
9.5. JSON 形式で LVM コマンド結果の表示	59
9.6. LVM コマンドログの表示	60
<b>第10章 RAID 論理ボリュームの設定</b> .....	<b>61</b>
10.1. RAID 論理ボリュームの作成	62
10.2. RAID0 (ストライピング) 論理ボリュームの作成	63
10.3. RAID ボリュームを初期化する速度の制御	65
10.4. リニアデバイスの RAID デバイスへの変換	66
10.5. LVM RAID1 論理ボリュームを LVM リニア論理ボリュームに変換	66
10.6. ミラー化 LVM デバイスを RAID1 デバイスに変換	67
10.7. RAID 論理ボリュームのサイズ変更	68
10.8. 既存の RAID1 デバイスのイメージ数を変更	68
10.9. RAID イメージを複数の論理ボリュームに分割	70
10.10. RAID イメージの分割とマージ	71
10.11. RAID 障害ポリシーの設定	73
10.12. 論理ボリュームで RAID デバイスの交換	74
10.13. RAID 論理ボリュームでのデータ整合性の確認 (RAID スクラビング)	78
10.14. RAID レベルの変更 (RAID テイクオーバー)	79
10.15. RAID ボリュームの属性の変更 (RAID 再成形)	79
10.16. RAID1 論理ボリュームでの I/O 操作の制御	79
10.17. RAID 論理ボリュームのリージョンサイズの変更	80
<b>第11章 スナップショット論理ボリューム</b> .....	<b>81</b>
11.1. スナップショットボリューム	81
11.2. スナップショットボリュームの作成	82
11.3. スナップショットボリュームのマージ	84
<b>第12章 シンプロビジョニングされた論理ボリューム (シンボリューム) の作成および管理</b> .....	<b>85</b>
12.1. シンプロビジョニングされた論理ボリューム (シンボリューム)	85
12.2. シンプロビジョニングされた論理ボリュームの作成	85
12.3. シンプロビジョニングのスナップショットボリューム	88
12.4. シンプロビジョニングのスナップショットボリュームの作成	89
12.5. 削除されているシンプロビジョニングのスナップショットボリュームの追跡および表示	91
<b>第13章 キャッシュを有効にして論理ボリュームのパフォーマンスを改善</b> .....	<b>95</b>
13.1. LVM でのキャッシュの取得方法	95
13.2. LVM キャッシュコンポーネント	95
13.3. 論理ボリュームの DM-CACHE キャッシュの有効化	95
13.4. 論理ボリュームに CACHEPOOL を使用した DM-CACHE キャッシュの有効化	97
13.5. 論理ボリュームの DM-WRITECACHE キャッシュの有効化	98
13.6. 論理ボリュームのキャッシュの無効化	100
<b>第14章 論理ボリュームのアクティブ化</b> .....	<b>101</b>
14.1. 論理ボリュームの自動アクティブ化の制御	101

---

14.2. 論理ボリュームのアクティブ化の制御	102
14.3. 共有論理ボリュームのアクティベーション	103
14.4. 欠落しているデバイスを含む論理ボリュームのアクティブ化	103
<b>第15章 LVM デバイススキャンの制御</b> .....	<b>104</b>
15.1. LVM デバイスフィルター	104
15.2. LVM デバイスフィルター設定の例	104
15.3. LVM デバイスフィルター設定の適用	105
<b>第16章 論理ボリュームに LVM 物理ボリュームを階層化</b> .....	<b>106</b>
<b>第17章 LVM の割り当ての制御</b> .....	<b>107</b>
17.1. LVM の割り当てポリシー	107
17.2. 物理ボリュームでの割り当て防止	108
17.3. CLING 割り当てポリシーを使用した論理ボリュームの拡張	108
<b>第18章 LVM のトラブルシューティング</b> .....	<b>110</b>
18.1. LVM での診断データの収集	110
18.2. 障害の発生した LVM デバイスに関する情報の表示	111
18.3. ボリュームグループから見つからない LVM 物理ボリュームの削除	112
18.4. 破損したメタデータを使用した LVM 物理ボリュームの復旧	113
18.5. 見つからない LVM 物理ボリュームの置き換え	116
18.6. 論理ボリュームの空きエクステントが不足する問題のトラブルシューティング	118
18.7. マルチパス化された LVM デバイスに対する重複した物理ボリューム警告のトラブルシューティング	120



## RED HAT ドキュメントへのフィードバック (英語のみ)

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
  1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上端に **Feedback** ボタンがあることを確認してください。
  2. マウスカーソルで、コメントを追加する部分を強調表示します。
  3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
  4. 表示される手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
  1. [Bugzilla](#) の Web サイトにアクセスします。
  2. Component で **Documentation** を選択します。
  3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
  4. **Submit Bug** をクリックします。

## 第1章 論理ボリューム

ボリューム管理により、物理ストレージに抽象化レイヤーが作成され、論理ストレージボリュームを作成できるようになりました。様々な面で、物理ストレージを直接使用するよりも柔軟性が高くなります。また、ハードウェアストレージ設定がソフトウェアから見えなくなるため、アプリケーションを停止したりファイルシステムをアンマウントしたりせずに、サイズ変更や移動が可能になります。したがって、運用コストが削減できます。

物理ストレージを直接使用する場合と比較して、論理ボリュームには、以下のような利点があります。

- 容量の柔軟性  
論理ボリュームを使用すると、ディスクとパーティションを1つの論理ボリュームに集約できるため、ファイルシステムを複数のディスクに拡張できます。
- サイズ変更可能なストレージプール  
基になるディスクデバイスを再フォーマットしたり、パーティションを再作成したりせずに、簡単なソフトウェアコマンドを使用して論理ボリュームのサイズを拡大または縮小できます。
- オンラインデータ移動  
より新しく、迅速で、障害耐性の高いストレージサブシステムを導入するために、システムがアクティブな状態でもデータを移動できます。データは、ディスクが使用中の場合でもディスクに再配置できます。たとえば、ホットスワップ可能なディスクを削除する前に空にできます。
- 便利なデバイスの命名  
論理ストレージボリュームは、ユーザーが命名をカスタマイズしたグループで管理できます。
- ディスクのストライピング  
2つ以上のディスクにまたがってデータをストライプ化する論理ボリュームを作成できます。これにより、スループットが大幅に向上します。
- ボリュームのミラーリング  
論理ボリュームは、データのミラーを設定する際に便利な方法を提供します。
- ボリュームスナップショット  
論理ボリュームを使用すると、一貫したバックアップが可能なデバイススナップショットを撮ったり、実際のデータに影響を及ぼすことなく変更の影響をテストしたりできます。
- シンボリューム  
論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステンツよりも大きな論理ボリュームを作成できます。
- キャッシュボリューム  
キャッシュ論理ボリュームでは、高速なブロックデバイス (SSD ドライブなど) で構成される小規模な論理ボリュームが使用されるため、頻繁に使用されるブロックを、小規模で高速な論理ボリュームに格納することにより、大規模で低速な論理ボリュームのパフォーマンスが向上します。

### 1.1. LVM アーキテクチャーの概要

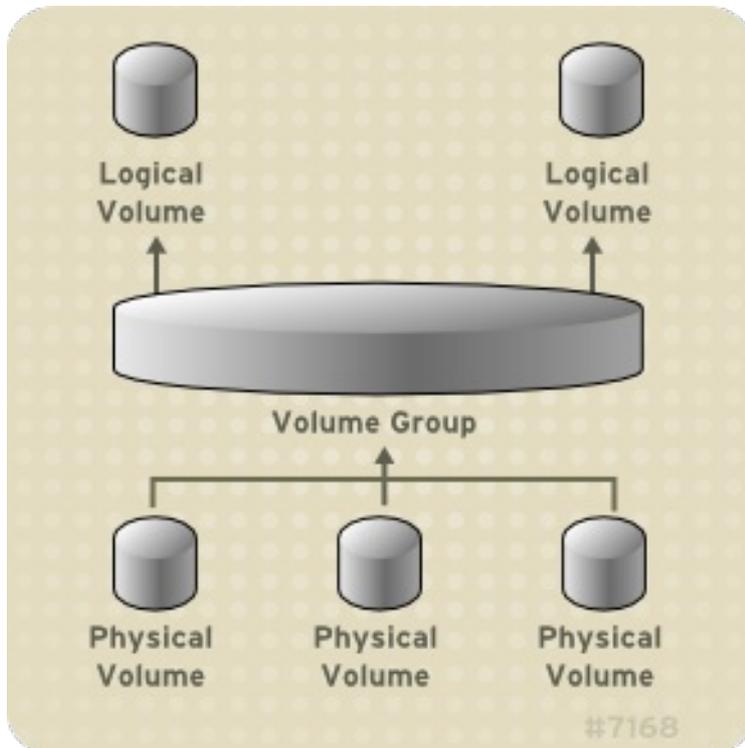
LVM 論理ボリュームの基になる物理ストレージユニットは、パーティション、ディスク全体などのブロックデバイスです。このデバイスは、LVM **物理ボリューム** (Physical Volume: PV) として初期化されます。

LVM 論理ボリュームを作成する場合は、物理ボリュームを**ボリュームグループ** (Volume Group: VG) に

統合します。これによりディスク領域のプールが作成され、そこから LVM 論理ボリューム (Logical Volume: LV) を割り当てます。このプロセスは、ディスクをパーティションに分割する方法に類似しています。論理ボリュームは、ファイルシステムやアプリケーション (データベースなど) に使用されません。

図1.1 「LVM 論理ボリュームのコンポーネント」 は、LVM 論理ボリュームのコンポーネントを簡略に示しています。

図1.1 LVM 論理ボリュームのコンポーネント



## 1.2. 物理ボリューム

LVM 論理ボリュームの基になる物理ストレージユニットは、パーティション、ディスク全体などのブロックデバイスです。LVM 論理ボリューム用にデバイスを使用する場合は、デバイスを物理ボリューム (PV) として初期化する必要があります。ブロックデバイスを物理ボリュームとして初期化すると、デバイスの先頭位置にラベルが付けられます。

LVM ラベルは、デフォルトでは 2 番目の 512 バイトセクターに配置されます。物理ボリュームを作成する場合は、先頭の 4 つのセクターのいずれかにラベルを配置することにより、このデフォルト設定を書き換えることができます。これにより、必要に応じて LVM ボリュームを、このセクターを利用する他のユーザーと併用できるようになります。

デバイスがシステムの起動時に任意の順序で初期化されても、LVM ラベルにより物理デバイスの識別とデバイスの順序付けが正しく行われます。LVM ラベルは、再起動してもクラスター全体で維持されます。

LVM ラベルは、デバイスを LVM 物理ボリュームとして識別するものです。これには、物理ボリューム用のランダムな一意識別子 (UUID) が含まれます。また、ブロックデバイスのサイズもバイト単位で保存し、LVM メタデータがデバイスのどこに保存されているかも記録します。

LVM メタデータには、システムにある LVM ボリュームグループの設定詳細が含まれています。デフォルトでは、メタデータの複製コピーが、ボリュームグループ内で、すべての物理ボリュームの、すべてのメタデータ領域に保存されています。LVM メタデータのサイズは小さく、ASCII 形式が使用されます。

現在、LVM では、各物理ボリュームにメタデータのコピーを1つまたは2つ保存できます。コピーをゼロにすることもできます。デフォルトでは1つ保存されます。物理ボリューム上に保存するメタデータのコピー数を一度設定したら、その数を後で変更することはできません。最初のコピーはデバイスの先頭にあるラベルの後に保存されます。2つ目のコピーがある場合は、デバイスの最後に配置されます。意図したものとは別のディスクに誤って書き込みを行い、ディスクの先頭領域を上書きしてしまった場合でも、デバイス後部にある2つ目のコピーでメタデータを復元できます。

### 1.2.1. LVM 物理ボリュームのレイアウト

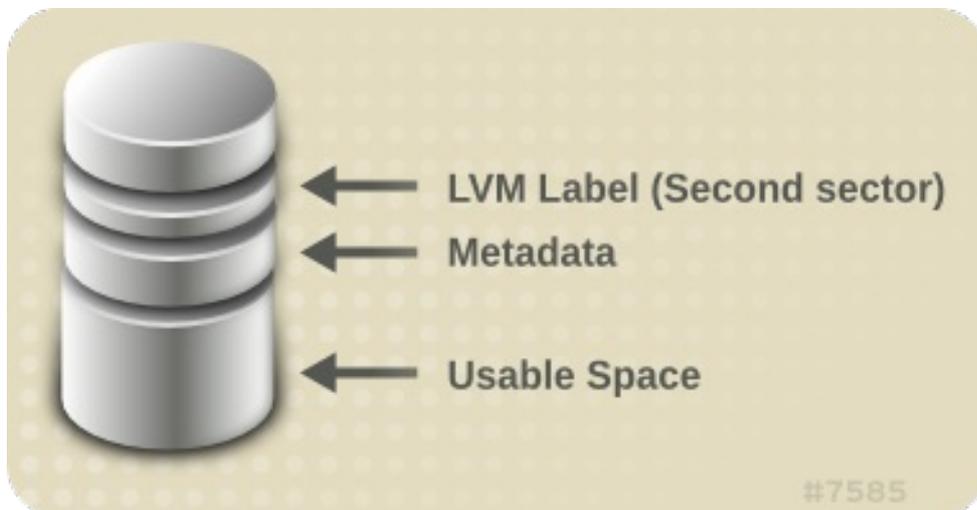
図1.2「物理ボリュームのレイアウト」は、LVM 物理ボリュームのレイアウトを示しています。LVM ラベルが2番目のセクターにあり、その後にメタデータ領域、使用可能なデバイス領域と続きます。



#### 注記

Linux カーネル (および本書) では、セクターのサイズを 512 バイトとしています。

図1.2 物理ボリュームのレイアウト



### 1.2.2. ディスク上の複数パーティション

LVM の使用により、ディスクパーティションから物理ボリュームを作成できます。Red Hat では、以下のような理由により、ディスク全体に対応するパーティションを1つ作成し、1つの LVM 物理ボリュームとしてラベルを付けることを推奨しています。

- 管理上の利便性  
各ディスクが一度だけ表示されると、システムのハードウェアの追跡が簡単になります。これは、特にディスクに障害が発生した場合に役に立ちます。さらに、1つのディスクに物理ボリュームが複数あると、システムの起動時に、不明なパーティションに関するカーネルの警告が発生する可能性があります。
- ストライピングのパフォーマンス  
LVM は、2つの物理ボリュームが同じ物理ディスクにあるかどうかを認識しません。2つの物理ボリュームが同じ物理ディスクにあるときに、ストライプ化された論理ボリュームを作成すると、作成されたボリュームのディスクは同じでも、パーティションは異なる可能性があります。このとき、パフォーマンスは、改善ではなく低下します。

1つのディスクを、複数の LVM 物理ボリュームに分割しないといけない場合があります (推奨はされません)。たとえば、ディスクがほとんどないシステムで、既存システムを LVM ボリュームに移行する場合に、パーティション間でデータを移動しなければならない場合があります。さらに、大容量のディスクが存在し、管理目的で複数のボリュームグループを必要とする場合は、そのディスクにパーティション

ンを設定する必要があります。ディスクに複数のパーティションがあり、そのパーティションがいずれも同じボリュームグループにある場合に、ストライプ化ボリュームを作成するときは、論理ボリュームに追加するパーティションを注意して指定してください。

### 1.3. ボリュームグループ

物理ボリュームはボリュームグループ (VG) に統合されます。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。

ボリュームグループ内で、割り当て可能なディスク領域は、エクステントと呼ばれる固定サイズの単位に分割されます。割り当て可能な領域の最小単位は、1エクステントです。エクステントは、物理ボリュームでは物理エクステントと呼ばれます。

論理ボリュームには、物理エクステントと同じサイズの論理エクステントが割り当てられます。そのため、エクステントのサイズは、ボリュームグループ内のすべての論理ボリュームで同じになります。ボリュームグループは、論理エクステントを物理エクステントにマッピングします。

### 1.4. LVM 論理ボリューム

LVM では、ボリュームグループは論理ボリュームに分割されます。管理者は、標準のディスクパーティションとは異なり、データを破棄せずに論理ボリュームを拡大または縮小できます。ボリュームグループの物理ボリュームが別のドライブまたは RAID アレイにある場合は、ストレージデバイスに論理ボリュームを分散することもできます。

論理ボリュームを、ボリュームに必要なデータよりも小さい容量に縮小すると、データが失われる可能性があります。柔軟性を最大限にするために、現在のニーズに合わせて論理ボリュームを作成し、過剰なストレージ容量を未割り当ての状態にします。必要に応じて、未割り当ての領域を使用するように、論理ボリュームを安全に拡張できます。



#### 重要

AMD システム、Intel システム、ARM システム、および IBM Power Systems サーバーで、ブートローダーは LVM ボリュームを読み取ることができません。/boot パーティション用に、標準的な LVM 以外のディスクパーティションを作成する必要があります。IBM Z では、zipl ブートローダーは、リニアマッピングを使用して LVM 論理ボリュームの /boot に対応します。デフォルトでは、インストールプロセスは常に LVM ボリューム内に /パーティションおよび swap パーティションを作成し、物理ボリュームに別の /boot パーティションを使用します。

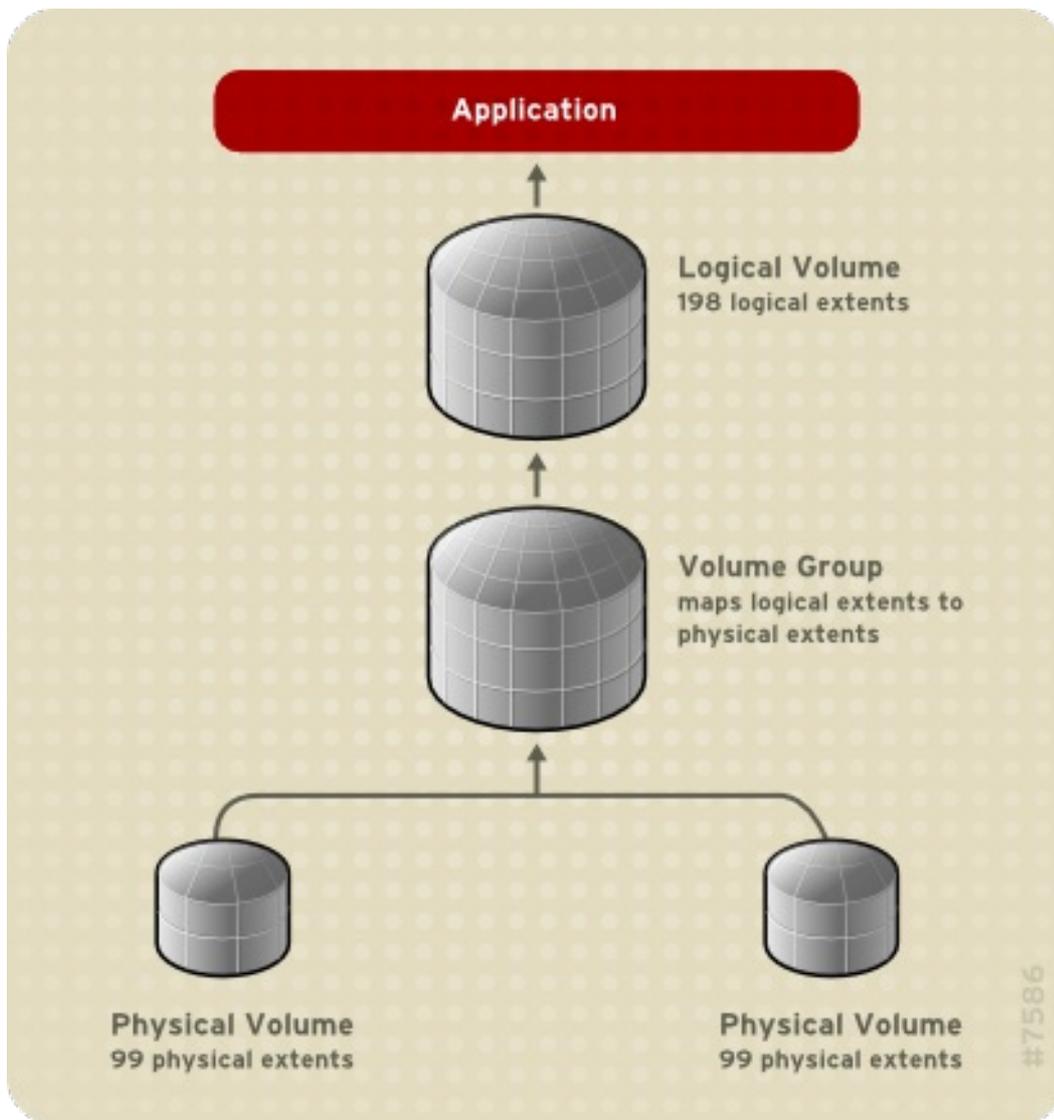
以下のセクションでは、論理ボリュームのタイプを説明します。

#### 1.4.1. リニアボリューム

リニアボリュームは、複数の物理ボリュームの領域を1つの論理ボリュームに統合します。たとえば、60GB ディスクが2つある場合は、120GB の論理ボリュームを作成できます。物理ストレージは連結されます。

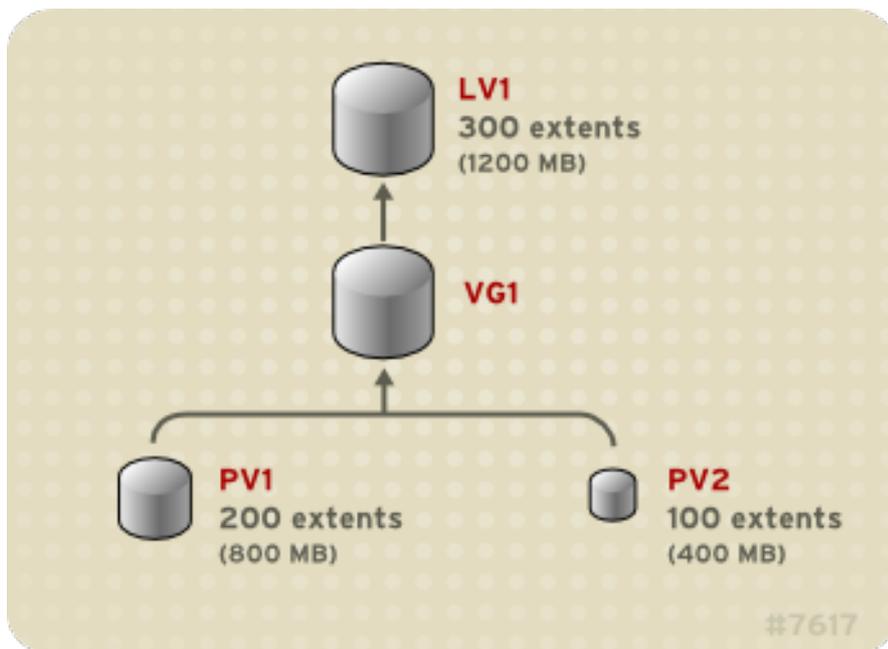
リニアボリュームを作成すると、論理ボリュームの領域に、物理エクステントの範囲が順番に割り当てられます。たとえば、[図1.3「エクステントのマッピング」](#)にあるように、1 から 99 までの論理エクステントが1つ目の物理ボリュームにマッピングされ、100 から 198 までの論理エクステントが2つ目の物理ボリュームにマッピングされます。アプリケーションからは、サイズが198 エクステントのデバイスが1つあるように見えます。

図1.3 エクステントのマッピング



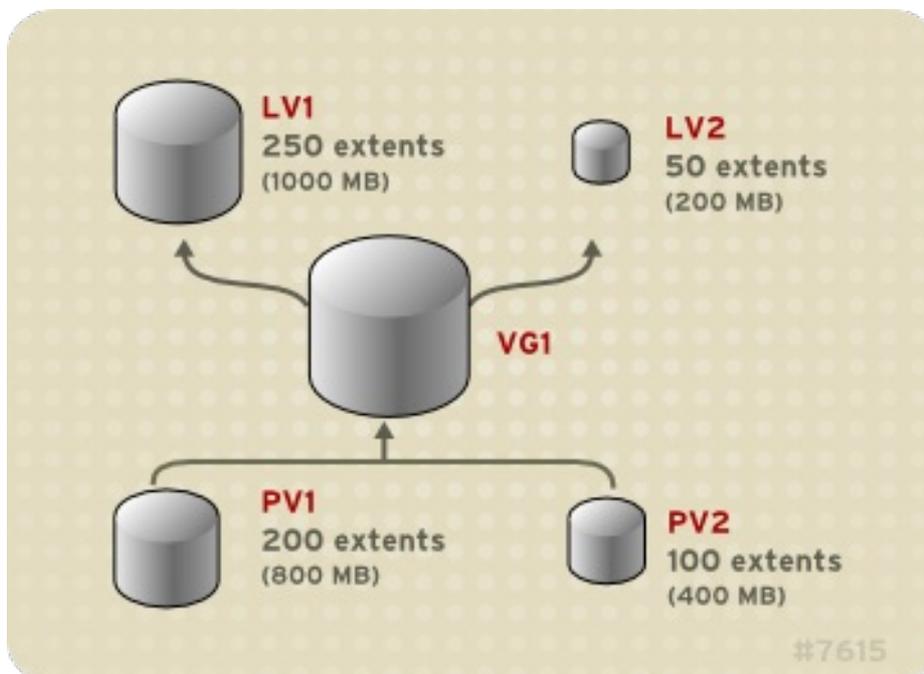
論理ボリュームを構成している各物理ボリュームのサイズは、すべて同じである必要はありません。図1.4「サイズの異なる物理ボリュームを用いたリニアボリューム」は、物理エクステントサイズが4MBのボリュームグループ **VG1** を示しています。このボリュームグループには、**PV1** と **PV2** という2つの物理ボリュームがあります。1エクステントは4MBなので、物理ボリュームが分割される単位は4MBになります。この例では、**PV1** のエクステントサイズは200 (800MB) で、**PV2** のエクステントサイズは100 (400MB) です。リニアボリュームは、エクステントサイズ1から300 (4MBから1200MB) の間で作成できます。この例では、300エクステントのリニアボリューム **LV1** を作成しました。

図1.4 サイズの異なる物理ボリュームを用いたリニアボリューム



物理エクステントのプールから、任意のサイズで、複数のリニア論理ボリュームを設定できます。図1.5「複数の論理ボリューム」は、図1.4「サイズの異なる物理ボリュームを用いたリニアボリューム」と同じボリュームグループを示していますが、ここでは、そのボリュームグループから論理ボリュームを2つ作成します。250 エクステント (1000MB) の **LV1** と、50 エクステント (200MB) の **LV2** です。

図1.5 複数の論理ボリューム



### 1.4.2. ストライプ化論理ボリューム

LVM 論理ボリュームにデータを書き込む際に、ファイルシステムは、基になる物理ボリューム全体にデータを分配します。このとき、ストライプ化論理ボリュームを作成すると、データを物理ボリュームに書き込む方法を制御できます。順次の読み取りおよび書き込みが大量に行われる場合には、これによりデータ I/O の効率を向上できます。

ストライピングは、ラウンドロビン式で、指定した数の物理ボリュームにデータを書き込んでいくことで、パフォーマンスを向上させます。I/O は、ストライピングでは並行して実行されます。これによ

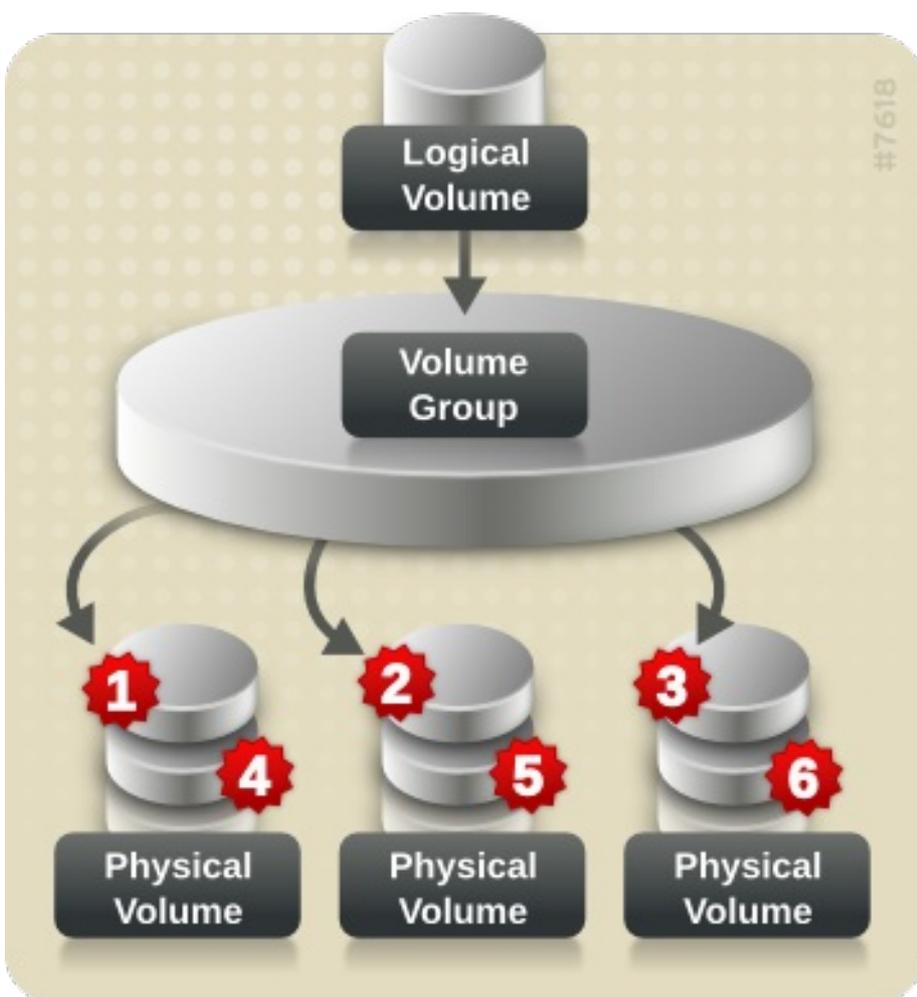
り、ストライプで追加される各物理ボリュームでは、ほぼ直線的なパフォーマンスの向上が期待できません。

以下は、3つの物理ボリュームにデータがストライプ化されている状態を示しています。この図は、以下のようになります。

- データの1番目のストライプは、1番目の物理ボリュームに書き込まれます。
- データの2番目のストライプは、2番目の物理ボリュームに書き込まれます。
- データの3番目のストライプは、3番目の物理ボリュームに書き込まれます。
- データの4番目のストライプは、1番目の物理ボリュームに書き込まれます。

ストライプ化された論理ボリュームでは、ストライプのサイズは、エクステントのサイズを越えることはできません。

図1.6 3つの物理ボリュームにまたがるデータのストライピング



ストライプ化論理ボリュームは、別のデバイスセットを最初のセットの末端に連結すれば拡張できます。ストライプ化論理ボリュームを拡張するには、ストライプに対応するボリュームグループを構成する、基になる物理ボリュームセットに、十分な空き領域が必要です。たとえば、ボリュームグループ全域を使用している 2 way ストライプがある場合は、そのボリュームグループに物理ボリュームを1つだけ追加しても、ストライプは拡張できません。ボリュームグループには物理ボリュームを2つ以上追加する必要があります。

### 1.4.3. RAID 論理ボリューム

LVM は、RAID0、1、4、5、6、および10に対応します。LVM RAID ボリュームには以下の特徴があります。

- LVM で作成および管理される RAID 論理ボリュームは、MD カーネルドライバーを使用しません。
- RAID1 イメージはアレイから一時的に切り離して、後でアレイにマージし直すことができます。
- LVM RAID ボリュームはスナップショットに対応します。



#### 注記

RAID 論理ボリュームはクラスターには対応していません。RAID 論理ボリュームは、1台のマシンに作成して排他的にアクティブにすることができますが、複数のマシンで同時にアクティブにすることはできません。

### 1.4.4. シンプロビジョニングされた論理ボリューム (シンボリューム)

論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステントよりも大きな論理ボリュームを作成できます。シンプロビジョニングを使用すると、シンプールと呼ばれる、空き領域のストレージプールを管理でき、アプリケーションで必要になった時に任意の数のデバイスに割り当てることができます。後でアプリケーションが実際に論理ボリュームに書き込むときに割り当てられるように、シンプールにバインドするデバイスを作成できます。シンプールは、ストレージ領域をコスト効率よく割り当てることが必要な場合に、動的に拡張できます。



#### 注記

クラスターのノード間では、シンボリュームに対応していません。シンプールとそのすべてのシンボリュームは、1つのクラスターノードで排他的にアクティブにする必要があります。

ストレージ管理者は、シンプロビジョニングを使用することで物理ストレージをオーバーコミットできるため、多くの場合は、追加のストレージを購入する必要がなくなります。たとえば、10人のユーザーから、各自のアプリケーションに使用するファイルシステムをそれぞれ100GB要求された場合、ストレージ管理者は各ユーザーに100GBのファイルシステムを作成します(ただし、実際には100GB未満のストレージが、必要に応じて使用されます)。シンプロビジョニングを使用する場合は、ストレージ管理者がストレージプールを監視し、容量が満杯になり始めたら容量を追加することが重要です。

利用可能な領域をすべて使用できるようにするために、LVM はデータの破棄に対応します。これにより、破棄されたファイルや、その他のブロック範囲で以前に使用された領域を再利用できます。

シンボリュームは、新たに実装されたコピーオンライト (COW) スナップショット論理ボリュームに対応します。これにより、多くの仮想デバイスでシンプール内の同一データを共有できます。

### 1.4.5. スナップショットボリューム

LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。スナップショットの取得後に作成元のデバイスに変更が加えられると、データが変更する前に、これから変更する部分のコピーがスナップショット機能により作成されるため、このコピーを使用して、デバイスの状態を再構築できます。



## 注記

LVM は、シンプロビジョニングのスナップショットに対応します

スナップショットは、スナップショットの作成後に変更したデータ部分のみをコピーするため、スナップショット機能に必要なストレージは最小限になります。たとえば、コピー元がほとんど更新されない場合は、作成元の 3~5% の容量があれば十分にスナップショットを維持できます。



## 注記

ファイルシステムのスナップショットコピーは仮想コピーであり、ファイルシステムのメディアバックアップを実際に作成するわけではありません。スナップショットは、バックアップの代替手順にはなりません。

作成元のボリュームへの変更を保管するために確保する領域は、スナップショットのサイズによって異なります。たとえば、スナップショットを作成してから作成元を完全に上書きした場合に、その変更を保管するのに必要なスナップショットのサイズは、作成元のボリュームと同じか、それ以上になります。スナップショットのサイズは、予想される変更レベルに応じて決定する必要があります。たとえば、`/usr` など、その大部分が読み取り用に使用されるボリュームの短期的なスナップショットに必要な領域は、`/home` のように大量の書き込みが行われるボリュームの長期的なスナップショットに必要な領域よりも小さくなります。

スナップショットが満杯になると、作成元のボリュームの変更を追跡できなくなるため、そのスナップショットは無効になります。スナップショットのサイズは定期的に監視する必要があります。ただし、スナップショットのサイズは完全に変更することが可能なため、ストレージに余裕があれば、スナップショットが停止しないように、スナップショットのボリュームサイズを拡大できます。逆に、スナップショットのボリュームサイズが必要以上に大きければ、そのボリュームのサイズを縮小して、他の論理ボリュームで必要となる領域を確保できます。

スナップショットのファイルシステムを作成しても、作成元への完全な読み取り/書き込みのアクセスは引き続き可能です。スナップショット上のチャンクを変更した場合は、そのチャンクにマークが付けられ、そこには、作成元のボリュームのコピーは入りません。

スナップショット機能にはいくつかの用途があります。

- 最も一般的な用途は、継続的にデータを更新している稼働中のシステムを停止せずに、論理ボリューム上でバックアップを実行する必要がある場合にスナップショットを撮ることです。
- スナップショットのファイルシステムで `fsck` コマンドを実行してファイルシステムの整合性を確認し、作成元のファイルシステムを修復する必要があるかどうかを判断できます。
- スナップショットは読み取りおよび書き込み用であるため、スナップショットを撮ってそのスナップショットにテストを実行することにより、実際のデータに触れることなく、実稼働データにアプリケーションのテストを実行できます。
- LVM ボリュームを作成して、Red Hat の仮想化と併用することが可能です。LVM スナップショットを使用して、仮想ゲストイメージのスナップショットを作成できます。このスナップショットは、最小限のストレージを使用して、既存のゲストの変更や新規ゲストの作成を行う上で利便性の高い方法を提供します。

`lvconvert` コマンドの `--merge` オプションを使用して、スナップショットを作成元のボリュームにマージできます。この機能の用途の1つがシステムロールバックの実行で、データやファイルを紛失した場合や、システムを以前の状態に復元する必要がある場合に行います。スナップショットボリュームのマージ後の論理ボリュームには、作成元のボリューム名、マイナー番号、UUID が付けられ、マージされたスナップショットは削除されます。

### 1.4.6. シンプロビジョニングのスナップショットボリューム

Red Hat Enterprise Linux は、シンプロビジョニングのスナップショットボリュームのサポートを提供します。シンプロビジョニングのスナップショットボリュームにより、多くの仮想デバイスを同じデータボリュームに格納できます。これにより管理が簡略化され、スナップショットボリューム間でのデータ共有が可能になります。

シンボリュームや、LVM スナップショットボリュームの場合、シンプロビジョニングのスナップショットボリュームは、クラスタのノード間では対応していません。スナップショットボリュームは、1つのクラスタノードで排他的にアクティブにする必要があります。

シンプロビジョニングのスナップショットボリュームの利点は以下のとおりです。

- 同じボリュームからのスナップショットが複数ある場合に、シンプロビジョニングのスナップショットボリュームを使用すれば、ディスクの使用量を減らすことができます。
- 作成元が同じスナップショットが複数ある場合は、作成元に1回書き込むことにより、1回のCOW操作でデータを保存できます。作成元のスナップショットの数を増やしても、速度が大幅に低下することはありません。
- シンプロビジョニングのスナップショットボリュームは、別のスナップショットの作成元の論理ボリュームとして使用できます。これにより、再帰的スナップショット(スナップショットのスナップショットのそのまたスナップショットなど)の深度を任意に決定できます。
- シン論理ボリュームのスナップショットにより、シン論理ボリュームを作成することもできます。COW操作が必要になるまで、あるいはスナップショット自体が書き込まれるまで、データ領域は消費されません。
- シンプロビジョニングのスナップショットボリュームは、作成元とともにアクティブにしておく必要はありません。そのため、スナップショットボリュームが多数ある場合は、作成元のみをアクティブにし、スナップショットボリュームはアクティブにしないことができます。
- シンプロビジョニングのスナップショットボリュームの作成元を削除すると、そのボリュームのスナップショットは、それぞれ独立したシンプロビジョニングボリュームになります。したがって、スナップショットとその作成元のボリュームをマージする代わりに、作成元のボリュームを削除し、その独立したボリュームを新たな作成元ボリュームにして、シンプロビジョニングのスナップショットを新たに作成できます。

シンプロビジョニングのスナップショットボリュームを使用する利点は数多くありますが、古いLVMスナップショットボリューム機能の方がニーズに適している場合もあります。

- シンプルのチャンクサイズは変更できません。シンプルのチャンクサイズが大きい場合(1MBなど)や、そのチャンクのサイズでは効率的ではない短期間のスナップショットが必要な場合は、代わりに以前のスナップショット機能を使用できます。
- シンプロビジョニングのスナップショットボリュームのサイズを制限することはできません。スナップショットは、必要な場合はシンプル内の全領域を使用します。これは、ニーズに適さない場合があります。

一般的には、使用するスナップショットの形式を決定する際に、使用しているサイトの特定要件を考慮するようにしてください。

### 1.4.7. キャッシュボリューム

LVMは、高速ブロックデバイス(SSDドライブなど)を、大規模で低速なブロックデバイスのライトバックまたはライトスルーのキャッシュとして使用することに対応します。既存の論理ボリュームのパ

パフォーマンスを改善するためにキャッシュ論理ボリュームを作成したり、大規模で低速なデバイスと共に小規模で高速なデバイスで構成される新規のキャッシュ論理ボリュームを作成したりできます。

## 第2章 RHEL システムロールを使用したローカルストレージの管理

Ansible を使用して LVM とローカルファイルシステム (FS) を管理するには、RHEL 8 で利用可能な RHEL システムロールの1つである **storage** ロールを使用できます。

**storage** ロールを使用すると、ディスク上のファイルシステム、複数のマシンにある論理ボリューム、および RHEL 7.7 以降の全バージョンでのファイルシステムの管理を自動化できます。

RHEL システムロールと、その適用方法の詳細は、「[RHEL システムロールの概要](#)」を参照してください。

### 2.1. ストレージロールの概要

**storage** ロールは以下を管理できます。

- パーティションが分割されていないディスクのファイルシステム
- 論理ボリュームとファイルシステムを含む完全な LVM ボリュームグループ

**storage** ロールを使用すると、次のタスクを実行できます。

- ファイルシステムを作成する
- ファイルシステムを削除する
- ファイルシステムをマウントする
- ファイルシステムをアンマウントする
- LVM ボリュームグループを作成する
- LVM ボリュームグループを削除する
- 論理ボリュームを作成する
- 論理ボリュームを削除する

### 2.2. ストレージロールのパラメーター

本セクションでは、**storage** ロールで使用できる入力変数の表を提供します。

表2.1 ストレージロールのパラメーター

パラメーター	説明	関連情報
storage_volumes	管理対象のパーティションが分割されていない全ディスク上のファイルシステムの一覧	現在、パーティションはサポートされていません。

パラメーター	説明	関連情報
storage_pools	管理するプールの一覧	現在、サポートされている唯一のプールタイプは LVM です。LVM では、プールはボリュームグループ (VG) を表します。各プールの下には、ロールで管理されるボリュームの一覧があります。LVM では、各ボリュームは、ファイルシステムを持つ論理ボリューム (LV) に対応します。
storage_safe_mode	セーフモードを有効または無効にするプールオプション	セーフモードでは、ストレージロールはファイルシステムや LVM 物理ボリュームなどのデータを含むディスクを再フォーマットしませんが、代わりにエラーメッセージでロールが中断します。デフォルトでは、誤って既存のデータを上書きしないように、セーフモードがオンになっています。Red Hat は、セーフモードを有効にして、特定のケースと完全なテストの後にのみ無効にすることが推奨されます。

**storage** ロールは、次の変数に記載されているファイルシステム、ボリューム、およびプールのみを変更します。

- **storage\_pools**
- **storage\_volumes**

### 関連情報

詳細は、**rhel-system-roles** パッケージをインストールして、**/usr/share/doc/rhel-system-roles/storage/** ディレクトリーおよび **/usr/share/ansible/roles/rhel-system-roles.storage/** ディレクトリーを参照してください。

## 第3章 LVM のデプロイメント

以下の手順では、新たにインストールしたオペレーティングシステムの LVM 論理ボリュームの基本設定を説明します。

### 3.1. LVM 物理ボリュームの作成

この手順では、LVM 物理ボリューム (PV) を作成し、ラベルを付ける方法を説明します。

#### 前提条件

- **lvm2** パッケージがインストールされている。

#### 手順

1. 複数の物理ボリュームを作成するには、**pvcreate** コマンドに、スペースで区切られたデバイス名を引数として使用します。

```
# pvcreate /dev/vdb1 /dev/vdb2
```

これにより、ラベルが `/dev/vdb1` および `/dev/vdb2` に配置され、それらが LVM に属する物理ボリュームとしてマークされます。

詳細は、man ページの **pvcreate** を参照してください。

### 3.2. LVM ボリュームグループの作成

この手順では、LVM ボリュームグループを作成する方法を説明します。

#### 前提条件

- **lvm2** パッケージがインストールされている。
- 物理ボリュームが作成されます。「[LVM 物理ボリュームの作成](#)」の手順を参照してください。

#### 手順

1. ボリュームグループを作成するには、以下のコマンドを使用します。

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
```

これにより、**myvg** という名前の VG が作成されます。物理ボリュームの `/dev/vdb1` および `/dev/vdb2` は、ボリューム具ルーム **myvg** のベースストレージレベルです。

詳細は、man ページの **vgcreate** を参照してください。

後で物理ボリュームを拡張して、上のボリュームグループを拡張できます。ボリュームグループを拡張するには、次のコマンドを使用します。

```
# vgextend myvg /dev/vdb3
```

### 3.3. LVM 論理ボリュームの作成

この手順では、LVM 論理ボリュームを作成する方法を説明します。

## 前提条件

- **lvm2** パッケージがインストールされている。
- ボリュームグループが作成されます。「[LVM ボリュームグループの作成](#)」の手順を参照してください。

## 手順

1. 論理ボリュームを作成するには、次のコマンドを使用します。

```
# lvcreate -n lv01 -L 500M myvg
```

**-n** オプションを使用すると、ユーザーは論理ボリューム名を **lv01** に設定できます。この例では、**-L** オプションを使用して Mb 単位で LV のサイズを設定していますが、他の単位を使用することもできます。デフォルトでは、論理ボリュームのタイプはリニアですが、**--type** オプションを使用して必要なタイプを指定できます。



### 重要

ぼぎゅーむグループに要求されるサイズとタイプの空き物理エクステントが十分でない場合、このコマンドは失敗します。

詳細は、man ページの **lvcreate** を参照してください。

## 第4章 LVM 論理ボリュームの設定

以下の手順では、基本的な LVM 管理タスクの例を示します。

### 4.1. CLI コマンドの使用

以下のセクションでは、LVM CLI コマンドの一般的な操作機能を説明します。

#### コマンドラインの引数で単位の指定

コマンドラインの引数でサイズが必要な場合は、常に単位を明示的に指定できます。単位を指定しないと、デフォルトで KB または MB が指定されます。LVM CLI コマンドでは、分数を使用できません。

コマンドライン引数で単位を指定する場合は、LVM が大文字と小文字を区別しません。たとえば、M と m は同じで、2 の累乗 (1024 の倍数) が使用されます。ただし、コマンドで **--units** 引数を指定すると、小文字は、単位が 1024 の倍数であることを示し、その単位は 1000 の倍数であることを示します。

#### ボリュームグループおよび論理ボリュームの指定

LVM CLI コマンドでボリュームグループまたは論理ボリュームを指定する場合は、以下の点に留意してください。

- ここで、コマンドではボリュームグループまたは論理ボリューム名を引数として取り、完全パス名はオプションになります。ボリュームグループ **vg0** 内の **lv0** という論理ボリュームは、**vg0/lv0** として指定できます。
- ボリュームグループの一覧は必須ですが、空の場合は、ボリュームグループの一覧が置き換えられます。
- 論理ボリュームの一覧が必要ですが、ボリュームグループを指定すると、そのボリュームグループにある論理ボリュームがすべて置き換えられます。たとえば、**lvdisplay vg0** コマンドは、ボリュームグループ **vg0** の論理ボリュームをすべて表示します。

#### 出力の詳細レベルを上げる

すべての LVM コマンドは、出力の詳細レベルを上げるために複数回入力できる **-v** 引数を受け入れます。以下の例は、**lvcreate** コマンドのデフォルト出力を示しています。

```
# lvcreate -L 50MB new_vg
Rounding up size to full physical extent 52.00 MB
Logical volume "lv0" created
```

以下のコマンドは、**lvcreate** コマンドの出力と、**-v** 引数を表示します。

```
# lvcreate -v -L 50MB new_vg
Rounding up size to full physical extent 52.00 MB
Archiving volume group "new_vg" metadata (seqno 1).
Creating logical volume lv0
Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 2).
Activating logical volume new_vg/lv0.
activation/volume_list configuration setting not defined: Checking only host tags for new_vg/lv0.
Creating new_vg-lv0
Loading table for new_vg-lv0 (253:0).
Resuming new_vg-lv0 (253:0).
Wiping known signatures on logical volume "new_vg/lv0"
Initializing 4.00 KiB of logical volume "new_vg/lv0" with value 0.
Logical volume "lv0" created
```

引数の **-vv**、**-vvv**、および **-vvvv** を使用すると、表示されるコマンド実行がより詳細になります。 **-vvvv** 引数は、この時点で情報の最大数を提供します。以下の例は、**lvcreate** コマンドで **-vvvv** 引数を指定して、出力の最初の行を示しています。

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913      Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:916      O_DIRECT will be used
#config/config.c:864  Setting global/locking_type to 1
#locking/locking.c:138 File-based locking selected.
#config/config.c:841  Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358 Getting target version for linear
#ioctl/libdm-iface.c:1569 dm version OF [16384]
#ioctl/libdm-iface.c:1569 dm versions OF [16384]
#activate/activate.c:358 Getting target version for striped
#ioctl/libdm-iface.c:1569 dm versions OF [16384]
#config/config.c:864  Setting activation/mirror_region_size to 512
...
```

### LVM CLI コマンドのヘルプの表示

コマンドの **--help** 引数を使用して、LVM CLI コマンドのヘルプを表示できます。

```
# commandname --help
```

コマンドの man ページを表示するには、**man** コマンドを実行します。

```
# man commandname
```

**man lv** コマンドは、LVM に関する一般的なオンライン情報を提供します。

## 4.2.3 つのディスク上での LVM 論理ボリューム作成

この手順例では、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** のディスクで構成される **mylv** という名前の LVM 論理ボリュームを作成します。

1. ボリュームグループのディスクを使用するには、**pvcreate** コマンドで、そのディスクに LVM 物理ボリュームラベルを付けます。



#### 警告

このコマンドは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sdb1" successfully created
Physical volume "/dev/sdc1" successfully created
```

- 作成した LVM 物理ボリュームで構成されるボリュームグループを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
Volume group "myvg" successfully created
```

**vgs** コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
VG #PV #LV #SN Attr VSize VFree
myvg 3 0 0 wz--n- 51.45G 51.45G
```

- 作成したボリュームグループから論理ボリュームを作成します。以下のコマンドを使用して、ボリュームグループ **myvg** から、論理ボリューム **mylv** を作成します。この例では、ボリュームグループの 2 ギガバイトを使用する論理ボリュームが作成されます。

```
# lvcreate -L 2G -n myvg mylv
Logical volume "mylv" created
```

- 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem          1K-blocks  Used Available Use% Mounted on
/dev/mapper/myvg-mylv 1998672  6144  1871288  1% /mnt
```

### 4.3. RAID0 (ストライピング) 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のデータサブボリューム全体に分散します。

RAID0 ボリュームを作成するコマンドの書式は以下のとおりです。

```
lvcreate --type raid0[_meta] --stripes Stripes --stripesize StripeSize VolumeGroup
[PhysicalVolumePath ...]
```

表4.1 RAID0 コマンドの作成に関するパラメーター

パラメーター	説明
<b>--type raid0[_meta]</b>	<b>raid0</b> を指定すると、メタデータボリュームなしで RAID0 ボリュームが作成されます。 <b>raid0_meta</b> を指定すると、メタデータボリュームとともに RAID0 ボリュームが作成されます。RAID0 には耐障害性がないため、RAID1/10 の場合のようにミラーリングされたすべてのデータブロックを格納したり、RAID4/5/6 の場合のようにすべてのパリティブロックを計算して格納する必要はありません。したがって、ミラーリングされたブロックまたはパリティブロックの再同期の進行状態を把握するメタデータボリュームは必要ありません。ただし、RAID0 から RAID4/5/6/10 に変換するには、メタデータボリュームが必要です。 <b>raid0_meta</b> を指定すると、割り当ての失敗を防ぐために、このメタデータが事前に割り当てられます。
<b>--stripes Stripes</b>	論理ボリュームを分散するデバイスの数を指定します。
<b>--stripesize StripeSize</b>	各ストライプのサイズをキロバイト単位で指定します。これは、次のデバイスに移動する前にデバイスに書き込まれるデータの量です。
<b>VolumeGroup</b>	使用するボリュームグループを指定します。
<b>PhysicalVolumePath ...</b>	使用するデバイスを指定します。指定しない場合は、LVM により、 <b>Stripes</b> オプションに指定されているデバイスの数が、各ストライプに1つずつ選択されます。

この手順例では、`/dev/sda1`、`/dev/sdb1`、および `/dev/sdc1` のディスクにデータのストライプを作成する **mylv** という名前の LVM RAID0 論理ボリュームを作成します。

1. **pvcreate** コマンドを使用し、ボリュームグループで LVM 物理ボリュームとして使用するディスクにラベルを付けます。



### 警告

このコマンドは、`/dev/sda1`、`/dev/sdb1`、および `/dev/sdc1` にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sdb1" successfully created
Physical volume "/dev/sdc1" successfully created
```

- ボリュームグループ **myvg** を作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
Volume group "myvg" successfully created
```

**vgs** コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
VG #PV #LV #SN Attr VSize VFree
myvg 3 0 0 wz--n- 51.45G 51.45G
```

- 作成したボリュームグループから、RAID0 論理ボリュームを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** から、RAID0 ボリューム **mylv** が作成されます。この例では、ストライプサイズが 4 キロバイトの 3 つのストライプがある、サイズが 2 ギガバイトの論理ボリュームが作成されます。

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

- RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 525312 4k blocks and 131376 inodes
Filesystem UUID: 9d4c0704-6028-450a-8b0a-8875358c0511
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

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

## 4.4. LVM 論理ボリュームの名前の変更

この手順では、コマンドライン LVM インターフェースを使用して、既存の論理ボリュームの名前を変更します。

### 手順

- 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。

2. クラスター環境に論理ボリュームが存在する場合は、アクティブになっているすべてのノードで、論理ボリュームを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
[root@node-n]# lvchange --activate n vg-name/lv-name
```

3. **lvrename** ユーティリティーを使用して、既存の論理ボリュームの名前を変更します。

```
# lvrename vg-name original-lv-name new-lv-name
```

必要に応じて、デバイスの完全パスを指定できます。

```
# lvrename /dev/vg-name/original-lv-name /dev/vg-name/new-lv-name
```

## 関連情報

- man ページの **lvrename(8)**

## 4.5. 論理ボリュームからのディスクの削除

この手順例では、ディスクを交換するか、または別のボリュームで使用するために、既存の論理ボリュームからディスクを削除する方法を示しています。ディスクを削除する前に、LVM 物理ボリュームのエクステントを、別のディスクまたはディスクセットに移動する必要があります。

### 4.5.1. 既存の物理ボリュームへのエクステントの移動

この例では、論理ボリュームが、ボリュームグループ **myvg** の 4 つの物理ボリュームに分配されています。

```
# pvs -o+pv_used
PV      VG  Fmt Attr PSize PFree Used
/dev/sda1 myvg lvm2 a- 17.15G 12.15G 5.00G
/dev/sdb1 myvg lvm2 a- 17.15G 12.15G 5.00G
/dev/sdc1 myvg lvm2 a- 17.15G 12.15G 5.00G
/dev/sdd1 myvg lvm2 a- 17.15G 2.15G 15.00G
```

この例では、物理ボリューム **/dev/sdb1** からエクステントを移動して、この物理ボリュームをボリュームグループから削除できるようにします。

1. ボリュームグループの別の物理ボリュームに空きエクステントが十分にある場合は、他のオプションを使用せずに、削除するデバイスで **pvmove** コマンドを実行すると、エクステントが他のデバイスに分配されます。  
クラスターで、**pvmove** コマンドは、1 つのノードで排他的にアクティブになる論理ボリュームだけを移動できます。

```
# pvmove /dev/sdb1
/dev/sdb1: Moved: 2.0%
...
/dev/sdb1: Moved: 79.2%
...
/dev/sdb1: Moved: 100.0%
```

**pvmove** コマンドの実行が終了すると、エクステントの分配は次のようになります。

```
# pvs -o+pv_used
PV      VG  Fmt Attr PSize PFree Used
/dev/sda1 myvg lvm2 a- 17.15G 7.15G 10.00G
/dev/sdb1 myvg lvm2 a- 17.15G 17.15G 0
/dev/sdc1 myvg lvm2 a- 17.15G 12.15G 5.00G
/dev/sdd1 myvg lvm2 a- 17.15G 2.15G 15.00G
```

2. **vgreduce** コマンドを使用して、ボリュームグループから物理ボリューム **/dev/sdb1** を削除します。

```
# vgreduce myvg /dev/sdb1
Removed "/dev/sdb1" from volume group "myvg"
# pvs
PV      VG  Fmt Attr PSize PFree
/dev/sda1 myvg lvm2 a- 17.15G 7.15G
/dev/sdb1   lvm2 -- 17.15G 17.15G
/dev/sdc1 myvg lvm2 a- 17.15G 12.15G
/dev/sdd1 myvg lvm2 a- 17.15G 2.15G
```

これでディスクは物理的に削除できるようになり、他のユーザーに割り当てることも可能になります。

#### 4.5.2. 新規ディスクへのエクステンツの移動

この例では、以下のように、ボリュームグループ **myvg** の3つの物理ボリュームに、論理ボリュームが分配されています。

```
# pvs -o+pv_used
PV      VG  Fmt Attr PSize PFree Used
/dev/sda1 myvg lvm2 a- 17.15G 7.15G 10.00G
/dev/sdb1 myvg lvm2 a- 17.15G 15.15G 2.00G
/dev/sdc1 myvg lvm2 a- 17.15G 15.15G 2.00G
```

以下の手順では、**/dev/sdb1** のエクステンツを、新しいデバイス **/dev/sdd1** に移動します。

1. **/dev/sdd1** から、物理ボリュームを新規作成します。

```
# pvcreate /dev/sdd1
Physical volume "/dev/sdd1" successfully created
```

2. 新たに作成した物理ボリューム **/dev/sdd1** を、既存のボリュームグループ **myvg** に追加します。

```
# vgextend myvg /dev/sdd1
Volume group "myvg" successfully extended
# pvs -o+pv_used
PV      VG  Fmt Attr PSize PFree Used
/dev/sda1 myvg lvm2 a- 17.15G 7.15G 10.00G
/dev/sdb1 myvg lvm2 a- 17.15G 15.15G 2.00G
/dev/sdc1 myvg lvm2 a- 17.15G 15.15G 2.00G
/dev/sdd1 myvg lvm2 a- 17.15G 17.15G 0
```

3. **pvmove** を使用して、データを **/dev/sdb1** から **/dev/sdd1** へ移動します。

```
# pvmove /dev/sdb1 /dev/sdd1
/dev/sdb1: Moved: 10.0%
...
/dev/sdb1: Moved: 79.7%
...
/dev/sdb1: Moved: 100.0%

# pvs -o+pv_used
PV      VG  Fmt Attr PSize PFree Used
/dev/sda1  myvg lvm2 a- 17.15G 7.15G 10.00G
/dev/sdb1  myvg lvm2 a- 17.15G 17.15G 0
/dev/sdc1  myvg lvm2 a- 17.15G 15.15G 2.00G
/dev/sdd1  myvg lvm2 a- 17.15G 15.15G 2.00G
```

- データを **/dev/sdb1** から移動したら、この物理ボリュームをボリュームグループから削除できます。

```
# vgreduce myvg /dev/sdb1
Removed "/dev/sdb1" from volume group "myvg"
```

これで、このディスクを別のボリュームグループに再度割り当てたり、システムから削除できるようになりました。

## 4.6. 永続的なデバイス番号の設定

メジャーデバイス番号とマイナーデバイス番号は、モジュールのロード時に動的に割り当てられます。一部のアプリケーションは、ブロックデバイスが常に同じデバイス (メジャーとマイナー) 番号でアクティブにされている場合に、最も効果的に機能します。これは、**lvcreate** コマンドと **lvchange** コマンドで、以下の引数を使用して指定できます。

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

別のデバイスにすでに動的に割り当てられている番号を使用しないように、マイナー番号は大きくします。

NFS を使用してファイルシステムをエクスポートする場合は、そのエクスポートファイルで **fsid** パラメーターを指定すると、LVM 内で永続的なデバイス番号を設定する必要がなくなります。

## 4.7. LVM エクステントサイズの指定

ボリュームグループの作成に物理ボリュームが使用されると、ディスク領域はデフォルトで 4MB のエクステントに分割されます。このエクステントは、論理ボリュームのサイズを拡張/縮小する最小単位です。エクステントの数が多くても、論理ボリュームの I/O パフォーマンスに影響を与えることはありません。

エクステントサイズのデフォルト設定が適切でない場合は、**vgcreate** コマンドに **-s** オプションを使用して、エクステントのサイズを指定できます。**vgcreate** コマンドに **-p** 引数と **-l** 引数を使用すると、ボリュームグループに追加可能な物理ボリュームまたは論理ボリュームの数に制限をかけることができます。

## 4.8. RHEL システムロールを使用した LVM 論理ボリュームの管理

本セクションでは、**storage** ロールを適用して次のタスクを実行する方法を説明します。

- 複数のディスクで構成されるボリュームグループに LVM 論理ボリュームを作成します。
- 論理ボリューム上に特定のラベルを付けて ext4 ファイルシステムを作成します。
- ext4 ファイルシステムを永続的にマウントします。

### 前提条件

- **storage** ロールを含む Ansible Playbook がある。

そのような Playbook を適用する方法は、「[ロールの適用](#)」を参照してください。

#### 4.8.1. 論理ボリュームを管理する Ansible Playbook の例

本セクションでは、Ansible Playbook の例を紹介します。この Playbook は、**storage** ロールを適用して、**myvg** という名前のボリュームグループに **mylv** という LVM 論理ボリュームを作成します。ボリュームグループは、次のディスクで構成されます。

- **/dev/sda**
- **/dev/sdb**
- **/dev/sdc**

Playbook は、論理ボリューム上に ext4 ファイルシステムを作成し、ファイルシステムを永続的にマウントします。

```
- hosts: all
vars:
  storage_pools:
    - name: myvg
  disks:
    - sda
    - sdb
    - sdc
  volumes:
    - name: mylv
      size: 2G
      fs_type: ext4
      mount_point: /mnt
roles:
  - rhel-system-roles.storage
```



### 注記

**myvg** という名前のボリュームグループがすでに存在する場合は、論理ボリュームがそれに追加されます。

**myvg** という名前のボリュームグループが存在しない場合は作成されます。

#### 4.8.2. 関連情報

- **storage** ロールの詳細は、「[RHEL のシステムロールを使用したローカルストレージの管理](#)」を参照してください。

## 4.9. LVM 論理ボリュームの削除

この手順では、コマンドライン LVM インターフェースを使用して、既存の論理ボリュームを削除します。

以下のコマンドは、ボリュームグループ **vg-name** から、論理ボリューム **/dev/vg-name/lv-name** を削除します。

### 手順

1. 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。
2. クラスタ環境に論理ボリュームが存在する場合は、アクティブになっているすべてのノードで、論理ボリュームを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
[root@node-n]# lvchange --activate n vg-name/lv-name
```

3. **lvremove** ユーティリティを使用して、論理ボリュームを削除します。

```
# lvremove /dev/vg-name/lv-name  
Do you really want to remove active logical volume "lv-name"? [y/n]: y  
Logical volume "lv-name" successfully removed
```



### 注記

この場合は、論理ボリュームが非アクティブになっていません。削除する前に論理ボリュームを明示的に非アクティブにした場合は、アクティブな論理ボリュームを削除するかどうかを確認するプロンプトが表示されません。

### 関連情報

- [man ページの lvremove\(8\)](#)

## 第5章 論理ボリュームのサイズ変更

論理ボリュームを作成したら、ボリュームのサイズを変更できます。

### 5.1. 論理ボリュームの拡張

論理ボリュームのサイズを拡張するには、**lvextend** コマンドを使用します。

論理ボリュームを拡張する場合は、追加するボリュームの容量、または拡張後のボリュームのサイズを指定できます。

以下のコマンドは、論理ボリューム **/dev/myvg/homevol** を 12 ギガバイトに拡張します。

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

以下のコマンドは、論理ボリューム **/dev/myvg/homevol** に 1 ギガバイト追加します。

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

**lvcreate** コマンドと同様に、**lvextend** コマンドの **-l** 引数を使用して、論理ボリュームの拡張サイズをエクステント数で指定できます。また、この引数を使用してボリュームグループのパーセンテージ、またはボリュームグループに残ってる空き領域をパーセンテージで指定することもできます。以下のコマンドは、論理ボリューム **testlv** を拡張して、ボリュームグループ **myvg** の空き領域をすべて使用するようにします。

```
# lvextend -l +100%FREE /dev/myvg/testlv
Extending logical volume testlv to 68.59 GB
Logical volume testlv successfully resized
```

論理ボリュームを拡張したら、それに合わせてファイルシステムのサイズも拡張する必要があります。

多くのファイルシステムサイズ変更ツールは、デフォルトで、ファイルシステムのサイズを、その下の論理ボリュームのサイズまで拡大するため、2つのコマンドに同じサイズを指定する必要はありません。

### 5.2. 論理ボリュームのファイルシステムの拡張

論理ボリュームのファイルシステムを拡張するには、以下の手順を実行します。

1. 既存のボリュームグループに、論理ボリュームを拡張するのに十分な空き領域があるかどうかを調べます。空き領域が足りない場合は、次の手順を実行します。
  - a. **pvccreate** コマンドを使用して、新しい物理ボリュームを作成します。
  - b. **vgextend** コマンドを使用して、その物理ボリュームを組み込む、拡張対象の論理ボリュームが含まれるボリュームグループを拡張します。

2. ボリュームグループを拡張できたら、**lvresize** コマンドで論理ボリュームを拡張して、ファイルシステムが拡張できるようにします。
3. 論理ボリュームでファイルシステムのサイズを変更します。

**lvresize** コマンドの **-r** オプションを使用すれば、1つのコマンドで、論理ボリュームを拡張し、基になるファイルシステムのサイズを変更できることに注意してください。

### 5.3. 論理ボリュームの縮小

論理ボリュームのサイズを縮小するには、**lvreduce** コマンドを使用します。



#### 注記

GFS2 および XFS のファイルシステムでは縮小に対応していないため、GFS2 または XFS のファイルシステムが含まれる論理ボリュームのサイズは縮小できません。

縮小する論理ボリュームにファイルシステムが含まれている場合は、データの損失を防ぐため、ファイルシステムが、縮小する論理ボリュームにある領域を使用しないようにしてください。そのため、論理ボリュームにファイルシステムが含まれている場合は **lvreduce** コマンドの **--resizefs** オプションを使用することが推奨されます。このオプションを使用すると、**lvreduce** コマンドは論理ボリュームを縮小する前にファイルシステムの縮小を試みます。ファイルシステムの縮小に失敗した場合 (ファイルシステムが満杯であったり、ファイルシステムが縮小に対応していないと失敗します) は、**lvreduce** コマンドの実行に失敗し、論理ボリュームを縮小しません。



#### 警告

ほとんどの場合、**lvreduce** コマンドはデータ損失の可能性を警告し、確認を要求します。しかし、論理ボリュームが非アクティブな状態であったり、**--resizefs** オプションが使用されなかった場合など、警告が表示されない場合があるため、データの損失を防ぐのに確認プロンプトのみを信頼しないようにしてください。

**lvreduce** コマンドの **--test** オプションは、ファイルシステムの確認やファイルシステムのサイズ変更のテストを行わないため、操作が安全な場所を示しません。

以下のコマンドは、ボリュームグループ **vg00** の論理ボリューム **lv01** を 64 メガバイトに縮小します。この例では、**lv01** にはファイルシステムが含まれ、このコマンドにより、論理ボリュームとともにサイズが変更されます。次の例は、コマンドの出力を示しています。

```
# lvreduce --resizefs -L 64M vg00/lv01
fsck from util-linux 2.23.2
/dev/mapper/vg00-lv01: clean, 11/25688 files, 8896/102400 blocks
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vg00-lv01 to 65536 (1k) blocks.
The filesystem on /dev/mapper/vg00-lv01 is now 65536 blocks long.
```

```
Size of logical volume vg00/lv01 changed from 100.00 MiB (25 extents) to 64.00 MiB (16 extents).
Logical volume vg00/lv01 successfully resized.
```

変更するサイズの値の前に「-」記号を指定すると、その値が論理ボリュームの実際のサイズから減算されます。以下の例は、論理ボリュームの絶対サイズを64メガバイトに縮小する代わりに、64メガバイト分のボリュームを縮小する場合に使用するコマンドを示しています。

```
# lvreduce --resizefs -L -64M vg00/lvol1
```

## 5.4. ストライプ化論理ボリュームの拡張

ストライプ化論理ボリュームのサイズを拡大するには、ボリュームグループを構成している物理ボリュームに、ストライプをサポートする十分な空き領域が必要です。たとえば、ボリュームグループ全域を使用する2wayストライプがある場合は、ボリュームグループに物理ボリュームを1つ追加しただけでは、ストライプを拡張することはできません。ボリュームグループには物理ボリュームを2つ以上追加する必要があります。

たとえば、以下の **vgs** コマンドで表示された、2つの物理ボリュームで構成されるボリュームグループ **vg** について考えてみましょう。

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

ボリュームグループの全領域を使用して、ストライプを作成できます。

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
Using default stripesize 64.00 KB
Rounding up size to full physical extent 271.31 GB
Logical volume "stripe1" created
# lvs -a -o +devices
LV   VG   Attr LSize  Origin Snap% Move Log Copy% Devices
stripe1 vg  -wi-a- 271.31G                /dev/sda1(0),/dev/sdb1(0)
```

ボリュームグループの空き領域がなくなっていることに注意してください。

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

以下のコマンドで、ボリュームグループに物理ボリュームをもう1つ追加します。これで、135ギガバイトの領域が追加されます。

```
# vgextend vg /dev/sdc1
Volume group "vg" successfully extended
# vgs
VG #PV #LV #SN Attr VSize VFree
vg  3  1  0 wz--n- 406.97G 135.66G
```

この時点では、ストライプ化論理ボリュームを、ボリュームグループの最大サイズまで拡大することはできません。データをストライプ化するには、基になる物理デバイスが2つ必要です。

```
# lvextend vg/stripe1 -L 406G
Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 406.00 GB
```

```
Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
```

ストライプ化論理ボリュームを拡張するには、もう1つの物理ボリュームを追加してから、論理ボリュームを拡張します。この例では、ボリュームグループに物理ボリュームを2つ追加することにより、ボリュームグループの最大サイズまで、論理ボリュームを拡張できるようになっています。

```
# vgextend vg /dev/sdd1
Volume group "vg" successfully extended
# vgs
VG #PV #LV #SN Attr VSize VFree
vg 4 1 0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 542.00 GB
Logical volume stripe1 successfully resized
```

ストライプ化論理ボリュームを拡張するのに十分な物理デバイスがない場合でも、その拡張部分がストライプ化されなくても問題がないならば、ボリュームの拡張は可能です。ただし、これによりパフォーマンスが一定ではなくなる可能性があります。論理ボリュームに領域を追加する場合、デフォルトの動作では、既存の論理ボリュームの最後のセグメントと同じストライピングパラメーターを使用するようになっていますが、このパラメーターはオーバーライドできます。以下の例では、初回の **lvextend** コマンドが失敗した後に、既存のストライプ化論理ボリュームを拡張して残りの空き領域を使用するようにしています。

```
# lvextend vg/stripe1 -L 406G
Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 406.00 GB
Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

## 第6章 LVM ボリュームグループの管理

本セクションでは、ボリュームグループ管理のさまざまな特徴を実行するコマンドを説明します。

### 6.1. ボリュームグループ

物理ボリュームはボリュームグループ (VG) に統合されます。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。

ボリュームグループ内で、割り当て可能なディスク領域は、エクステントと呼ばれる固定サイズの単位に分割されます。割り当て可能な領域の最小単位は、1エクステントです。エクステントは、物理ボリュームでは物理エクステントと呼ばれます。

論理ボリュームには、物理エクステントと同じサイズの論理エクステントが割り当てられます。そのため、エクステントのサイズは、ボリュームグループ内のすべての論理ボリュームで同じになります。ボリュームグループは、論理エクステントを物理エクステントにマッピングします。

### 6.2. ボリュームグループの表示

LVM ボリュームグループのプロパティを表示するのに使用できるコマンドは、**vgs** および **vgdisplay** の2つです。**vgscan** コマンドは、システムにある、ボリュームグループ用に対応している LVM ブロックデバイスをすべてスキャンしますが、既存のボリュームグループを表示するのにも使用できます。

**vgs** コマンドは、ボリュームグループの情報を設定可能な形式で提供し、1ボリュームグループにつき1行ずつ表示します。**vgs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

**vgdisplay** コマンドは、決められた形式でボリュームグループのプロパティ (サイズ、エクステント、物理ボリュームの数など) を表示します。以下の例は、ボリュームグループ **new\_vg** に対する **vgdisplay** コマンドの出力を示しています。ボリュームグループを指定しないと、既存のボリュームグループがすべて表示されます。

```
# vgdisplay new_vg
--- Volume group ---
VG Name          new_vg
System ID
Format           lvm2
Metadata Areas   3
Metadata Sequence No 11
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV          1
Open LV          0
Max PV           0
Cur PV          3
Act PV           3
VG Size          51.42 GB
PE Size          4.00 MB
Total PE         13164
Alloc PE / Size  13 / 52.00 MB
Free PE / Size   13151 / 51.37 GB
VG UUID          jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

以下の例は、**vgscan** コマンドの出力を示しています。

```
# vgscan
Reading all physical volumes. This may take a while...
Found volume group "new_vg" using metadata type lvm2
Found volume group "officevg" using metadata type lvm2
```

### 6.3. ボリュームグループの統合

2つのボリュームグループを1つのボリュームグループに統合するには、**vgmerge** コマンドを使用します。ボリュームの物理エクステントサイズが同じで、かつ両ボリュームグループの物理ボリュームおよび論理ボリュームのサマリーが「マージ先」ボリュームグループの制限内に収まる場合は、非アクティブな「マージ元」のボリュームを、アクティブまたは非アクティブの「マージ先」ボリュームにマージができます。

以下のコマンドは、非アクティブなボリュームグループ **my\_vg** をアクティブまたは非アクティブなボリュームグループ **databases** にマージして、詳細なランタイム情報を提供します。

```
# vgmerge -v databases my_vg
```

### 6.4. ボリュームグループの分割

この手順例では、3つの物理ボリュームから構成される既存のボリュームグループを使用します。この物理ボリュームに未使用領域が十分にあれば、新たにディスクを追加しなくてもボリュームグループを作成できます。

はじめに、ボリュームグループ **myvg** から論理ボリューム **mylv** を作成します。そのボリュームグループは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** の3つの物理ボリュームで構成されます。

この手順を完了すると、ボリュームグループ **myvg** は、**/dev/sda1** および **/dev/sdb1** で構成されます。次のボリュームグループ **yourvg** は、**/dev/sdc1** で構成されます。

1. **pvscan** コマンドを使用すると、ボリュームグループで現在利用可能な空き領域の容量を確認できます。

```
# pvscan
PV /dev/sda1 VG myvg lvm2 [17.15 GB / 0 free]
PV /dev/sdb1 VG myvg lvm2 [17.15 GB / 12.15 GB free]
PV /dev/sdc1 VG myvg lvm2 [17.15 GB / 15.80 GB free]
Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0 ]
```

2. **pvmove** コマンドを使用して、**/dev/sdc1** で使用中の物理エクステントをすべて **/dev/sdb1** に移動できます。**pvmove** コマンドの実行には時間がかかる場合があります。クラスタで、**pvmove** コマンドは、1つのノードで排他的にアクティブになる論理ボリュームだけを移動できます。

```
# pvmove /dev/sdc1 /dev/sdb1
/dev/sdc1: Moved: 14.7%
/dev/sdc1: Moved: 30.3%
/dev/sdc1: Moved: 45.7%
/dev/sdc1: Moved: 61.0%
```

```

/dev/sdc1: Moved: 76.6%
/dev/sdc1: Moved: 92.2%
/dev/sdc1: Moved: 100.0%

```

データを移動したら、**/dev/sdc1** 上のすべての領域が空き領域になっていることが確認できます。

```

# pvscan
PV /dev/sda1 VG myvg lvm2 [17.15 GB / 0 free]
PV /dev/sdb1 VG myvg lvm2 [17.15 GB / 10.80 GB free]
PV /dev/sdc1 VG myvg lvm2 [17.15 GB / 17.15 GB free]
Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0 ]

```

- 新規ボリュームグループ **yourvg** を作成する場合は、**vgsplit** コマンドを使用して、ボリュームグループ **myvg** を分割します。

以下のコマンドは、ボリュームグループ **myvg** からボリュームグループ **yourvg** を分割し、物理ボリューム **/dev/sdc1** を新しいボリュームグループ **yourvg** に移動します。

```

# lvchange -a n /dev/myvg/mylv
# vgsplit myvg yourvg /dev/sdc1
Volume group "yourvg" successfully split from "myvg"

```

**vgs** コマンドを使用すると、2つのボリュームグループの属性を確認できます。

```

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

```

- ボリュームグループを新しく作成したら、論理ボリューム **yourlv** を新規作成します。

```

# lvcreate -L 5G -n yourlv yourvg
Logical volume "yourlv" created

```

- 論理ボリュームにファイルシステムを新規作成し、そのファイルシステムをマウントします。

```

# mkfs.ext4 /dev/yourvg/yourlv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mount /dev/yourvg/yourlv /mnt

```

## 6.5. LVM ボリュームグループの名前変更

この手順では、コマンドラインの LVM インターフェースを使用して、既存のボリュームグループの名前を変更します。

## 手順

1. クラスタ化環境にボリュームグループが存在する場合は、アクティブになっているすべてのノードでボリュームグループを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
[root@node-n]# vgchange --activate n vg-name
```

2. **vgrename** ユーティリティーを使用して、既存のボリュームグループの名前を変更します。

```
# vgrename original-vg-name new-vg-name
```

必要に応じて、デバイスの完全パスを指定できます。

```
# vgrename /dev/original-vg-name /dev/new-vg-name
```

## 関連情報

- man ページの **vgrename(8)**

## 6.6. ボリュームグループを別のシステムへ移動

LVM ボリュームグループ全体を、別のシステムに移動できます。これを実行するには、**vgexport** コマンドと **vgimport** コマンドの使用が推奨されます。



### 注記

**vgimport** コマンドの **--force** 引数を使用できます。これで物理ボリュームがないボリュームグループをインポートし、**vgreduce --removemissing** コマンドを実行します。

**vgexport** コマンドは、非アクティブのボリュームグループにシステムがアクセスできないようにするため、物理ボリュームの割り当て解除が可能になります。**vgimport** コマンドは、**vgexport** コマンドで非アクティブにしていたボリュームグループに、マシンが再度アクセスできるようにします。

ボリュームグループを2つのシステム間で移行するには、以下の手順に従います。

1. ボリュームグループ内のアクティブなボリュームのファイルにアクセスしているユーザーがないことを確認してから、論理ボリュームをアンマウントします。
2. **vgchange** コマンドで **-a n** 引数を使用して、そのボリュームグループを非アクティブとしてマークします。これによりこのボリュームグループでこれ以上の動作が発生しないようにします。
3. **vgexport** コマンドを使用してボリュームグループをエクスポートします。これにより、削除するシステムからボリュームグループへアクセスできなくなります。ボリュームグループをエクスポートして **pvscan** コマンドを実行すると、以下の例のように、エクスポート先のボリュームグループに物理ボリュームが表示されます。

```
# pvscan
```

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

次にシステムがシャットダウンする時に、ボリュームグループを構成していたディスクを外して、新しいシステムに接続できます。

4. ディスクが新しいシステムに接続したら、**vgimport** コマンドを使用してボリュームグループをインポートし、新しいシステムからアクセスできるようにします。
5. **vgchange** コマンドで **-a y** 引数を使用して、ボリュームグループをアクティブにします。
6. ファイルシステムをマウントして使用できるようにします。

## 6.7. ボリュームグループからの物理ボリュームの削除

ボリュームグループから未使用の物理ボリュームを削除するには、**vgreduce** コマンドを使用します。**vgreduce** コマンドは、空の物理ボリュームを1つまたは複数削除して、ボリュームグループの容量を縮小します。これにより、物理ボリュームが解放され、異なるボリュームグループで使用したり、システムから削除できるようになります。

ボリュームグループから物理ボリュームを削除する前に、**pvdisplay** コマンドを使用して、その物理ボリュームが論理ボリュームで使用されていないことを確認できます。

```
# pvdisplay /dev/hda1

-- Physical volume ---
PV Name      /dev/hda1
VG Name      myvg
PV Size      1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#          1
PV Status    available
Allocatable  yes (but full)
Cur LV      1
PE Size (KByte)  4096
Total PE     499
Free PE      0
Allocated PE  499
PV UUID      Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

物理ボリュームが使用中の場合は、**pvmove** コマンドを使用して、データを別の物理ボリュームに移行する必要があります。その後、**vgreduce** コマンドを使用してその物理ボリュームを削除します。

以下のコマンドは、物理ボリューム **/dev/hda1** を、ボリュームグループ **my\_volume\_group** から取り除きます。

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

論理ボリュームに、障害のある物理ボリュームが含まれる場合は、その論理ボリュームを使用することはできません。見つからない物理ボリュームをボリュームグループから削除します。その物理ボリュームに論理ボリュームが割り当てられていない場合は、**vgreduce** コマンドの **--removemissing** パラメーターを使用できます。

障害が発生した物理ボリュームに、セグメントタイプが **mirror** の論理ボリュームのミラーイメージが

含まれる場合は、**vgreduce --removemissing --mirroronly --force** コマンドを使用して、ミラーからイメージを削除できます。これにより、物理ボリュームのミラーイメージである論理ボリュームのみが削除されます。

## 6.8. LVM ボリュームグループの削除

この手順では、コマンドライン LVM インターフェースを使用して、既存のボリュームグループを削除します。

### 前提条件

- ボリュームグループには論理ボリュームがありません。ボリュームグループから論理ボリュームを削除するには、「[LVM 論理ボリュームの削除](#)」を参照してください。

### 手順

1. クラスタ環境にボリュームグループが存在する場合は、その他のすべてのノードで、ボリュームグループのロックスペースを停止します。削除するノード以外の全ノードで、次のコマンドを実行します。

```
[root@node-n]# vgchange --lockstop vg-name
```

ロックが停止するのを待ちます。

2. ボリュームグループを削除するには、**vgremove** ユーティリティを使用します。

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

### 関連情報

- [man ページの vgremove\(8\)](#)

## 6.9. 関連情報

- [man ページの vgchange\(8\)](#)

## 第7章 LVM 物理ボリュームの管理

LVM 物理ボリュームを管理するために使用できるコマンドや手順には、様々なものがあります。

### 7.1. 物理ボリュームとして使用するブロックデバイスのスキャン

以下の例のように、**lvmdiskscan** コマンドを使用して、物理ボリュームに使用できるブロックデバイスをスキャンできます。

```
# lvmdiskscan
/dev/ram0      [ 16.00 MB]
/dev/sda      [ 17.15 GB]
/dev/root     [ 13.69 GB]
/dev/ram      [ 16.00 MB]
/dev/sda1     [ 17.14 GB] LVM physical volume
/dev/VolGroup00/LogVol01 [ 512.00 MB]
/dev/ram2     [ 16.00 MB]
/dev/new_vg/lvol0 [ 52.00 MB]
/dev/ram3     [ 16.00 MB]
/dev/pkl_new_vg/sparkie_lv [ 7.14 GB]
/dev/ram4     [ 16.00 MB]
/dev/ram5     [ 16.00 MB]
/dev/ram6     [ 16.00 MB]
/dev/ram7     [ 16.00 MB]
/dev/ram8     [ 16.00 MB]
/dev/ram9     [ 16.00 MB]
/dev/ram10    [ 16.00 MB]
/dev/ram11    [ 16.00 MB]
/dev/ram12    [ 16.00 MB]
/dev/ram13    [ 16.00 MB]
/dev/ram14    [ 16.00 MB]
/dev/ram15    [ 16.00 MB]
/dev/sdb      [ 17.15 GB]
/dev/sdb1     [ 17.14 GB] LVM physical volume
/dev/sdc      [ 17.15 GB]
/dev/sdc1     [ 17.14 GB] LVM physical volume
/dev/sdd      [ 17.15 GB]
/dev/sdd1     [ 17.14 GB] LVM physical volume
7 disks
17 partitions
0 LVM physical volume whole disks
4 LVM physical volumes
```

### 7.2. 物理ボリュームのパーティションタイプの設定

ディスクデバイス全体を物理ボリュームに使用している場合は、そのディスクにはパーティションテーブルを含めないでください。ディスクパーティションが DOS の場合は、**fdisk**、**cdisk** などのコマンドを使用して、パーティション ID を 0x8e に設定する必要があります。ディスクデバイス全体に物理ボリュームがある場合は、パーティションテーブルのみを消去する必要がありますが、このとき、そのディスクにあるデータはすべて効果的に破棄されます。以下のコマンドを使用すれば、最初のセクターをゼロで初期化し、既存のパーティションテーブルを削除できます。

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

### 7.3. LVM 物理ボリュームのサイズ変更

なんらかの理由で基になるブロックデバイスのサイズを変更する必要がある場合は、**pvresize** コマンドを使用して LVM のサイズを更新します。このコマンドは、LVM が物理ボリュームを使用しているときに実行できます。

### 7.4. 物理ボリュームの削除

デバイスを LVM で使用する必要がなくなったら、**pvremove** コマンドを使用して LVM ラベルを削除できます。**pvremove** コマンドを実行すると、空の物理ボリュームにある LVM メタデータをゼロにします。

削除する物理ボリュームがボリュームグループの一部になっている場合は、**vgreduce** コマンドで、ボリュームグループから物理ボリュームを取り除く必要があります。

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

### 7.5. ボリュームグループへの物理ボリュームの追加

物理ボリュームを既存ボリュームグループに追加するには、**vgextend** コマンドを使用します。**vgextend** コマンドは、空き物理ボリュームを1つまたは複数追加して、ボリュームグループの容量を増やします。

以下のコマンドは、物理ボリューム **/dev/sdf1** をボリュームグループ **vg1** に追加します。

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

### 7.6. ボリュームグループからの物理ボリュームの削除

ボリュームグループから未使用の物理ボリュームを削除するには、**vgreduce** コマンドを使用します。**vgreduce** コマンドは、空の物理ボリュームを1つまたは複数削除して、ボリュームグループの容量を縮小します。これにより、物理ボリュームが解放され、異なるボリュームグループで使用したり、システムから削除できるようになります。

ボリュームグループから物理ボリュームを削除する前に、**pvdisk** コマンドを使用して、その物理ボリュームが論理ボリュームで使用されていないことを確認できます。

```
# pvdisk /dev/hda1

-- Physical volume ---
PV Name      /dev/hda1
VG Name      myvg
PV Size      1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#          1
PV Status    available
Allocatable  yes (but full)
Cur LV      1
PE Size (KByte)  4096
Total PE     499
Free PE      0
Allocated PE  499
PV UUID      Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

物理ボリュームが使用中の場合は、**pvmove** コマンドを使用して、データを別の物理ボリュームに移行する必要があります。その後、**vgreduce** コマンドを使用してその物理ボリュームを削除します。

以下のコマンドは、物理ボリューム **/dev/hda1** を、ボリュームグループ **my\_volume\_group** から取り除きます。

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

論理ボリュームに、障害のある物理ボリュームが含まれる場合は、その論理ボリュームを使用することはできません。見つからない物理ボリュームをボリュームグループから削除します。その物理ボリュームに論理ボリュームが割り当てられていない場合は、**vgreduce** コマンドの **--removemissing** パラメーターを使用できます。

障害が発生した物理ボリュームに、セグメントタイプが **mirror** の論理ボリュームのミラーイメージが含まれる場合は、**vgreduce --removemissing --mirroronly --force** コマンドを使用して、ミラーからイメージを削除できます。これにより、物理ボリュームのミラーイメージである論理ボリュームのみが削除されます。

## 第8章 LVM コンポーネントの表示

LVM は、LVM コンポーネントを表示し、その表示をカスタマイズする様々な方法を提供します。本セクションでは、基本的な LVM 表示コマンドの使用をまとめています。

### 8.1. LVM コマンドによる LVM 情報の表示

**lvm** コマンドは、LVM への対応および設定に関する情報を表示するのに使用できる組み込みオプションを提供します。

- **lvm devtypes**  
認識されている組み込みブロックデバイスを表示します。
- **lvm formats**  
認識されているメタデータ形式を表示します。
- **lvm help**  
LVM ヘルプテキストを表示します。
- **lvm segtypes**  
認識されている論理ボリュームセグメントタイプを表示します。
- **lvm tags**  
このホストに定義したタグを表示します。
- **lvm version**  
現在のバージョン情報を表示します。

### 8.2. 物理ボリュームの表示

LVM 物理ボリュームのプロパティを表示するのに使用できるコマンドは、**pvs**、**pvdisplay**、および **pvscan** の3つです。

**pvs** コマンドは、物理ボリュームの情報を設定可能な形式で提供し、1物理ボリュームにつき1行ずつ表示します。**pvs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

**pvdisplay** コマンドは、各物理ボリュームの詳細をそれぞれ複数行出力します。物理プロパティ (サイズ、エクステンツ、ボリュームグループなど) が、決められた形式で表示されます。

以下の例は、1つの物理ボリュームについて、**pvdisplay** コマンドで出力した情報です。

```
# pvdisplay
--- Physical volume ---
PV Name           /dev/sdc1
VG Name           new_vg
PV Size           17.14 GB / not usable 3.40 MB
Allocatable       yes
PE Size (KByte)   4096
Total PE          4388
Free PE           4375
Allocated PE      13
PV UUID           Joqlch-yWSj-kuEn-ldwM-01S9-XO8M-mcpsVe
```

**pvscan** コマンドは、システムにある、物理ボリュームで対応している LVM ブロックデバイスをすべてスキャンします。

以下のコマンドでは、検出された物理デバイスがすべて表示されます。

```
# pvscan
PV /dev/sdb2  VG vg0  lvm2 [964.00 MB / 0 free]
PV /dev/sdc1  VG vg0  lvm2 [964.00 MB / 428.00 MB free]
PV /dev/sdc2          lvm2 [964.84 MB]
Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

このコマンドが、特定の物理ボリュームをスキャンしないように、**lvm.conf** ファイルにフィルターを定義できます。

### 8.3. ボリュームグループの表示

LVM ボリュームグループのプロパティを表示するのに使用できるコマンドは、**vgs** および **vgdisplay** の2つです。**vgscan** コマンドは、システムにある、ボリュームグループ用に対応している LVM ブロックデバイスをすべてスキャンしますが、既存のボリュームグループを表示するのにも使用できません。

**vgs** コマンドは、ボリュームグループの情報を設定可能な形式で提供し、1ボリュームグループにつき1行ずつ表示します。**vgs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

**vgdisplay** コマンドは、決められた形式でボリュームグループのプロパティ (サイズ、エクステンツ、物理ボリュームの数など) を表示します。以下の例は、ボリュームグループ **new\_vg** に対する **vgdisplay** コマンドの出力を示しています。ボリュームグループを指定しないと、既存のボリュームグループがすべて表示されます。

```
# vgdisplay new_vg
--- Volume group ---
VG Name          new_vg
System ID
Format           lvm2
Metadata Areas   3
Metadata Sequence No 11
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV          1
Open LV          0
Max PV           0
Cur PV          3
Act PV           3
VG Size          51.42 GB
PE Size          4.00 MB
Total PE         13164
Alloc PE / Size  13 / 52.00 MB
Free PE / Size   13151 / 51.37 GB
VG UUID          jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

以下の例は、**vgscan** コマンドの出力を示しています。

```
# vgscan
```

```
Reading all physical volumes. This may take a while...  
Found volume group "new_vg" using metadata type lvm2  
Found volume group "officevg" using metadata type lvm2
```

## 8.4. 論理ボリュームの表示

LVM 論理ボリュームのプロパティを表示するのに使用できるコマンドは、**lvs**、**lvdisplay**、および **lvscan** の3つです。

**lvs** コマンドは、論理ボリュームの情報を設定可能な形式で提供し、1論理ボリュームにつき1行ずつ表示します。**lvs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

**lvdisplay** コマンドは、決められた形式で、論理ボリュームのプロパティ (サイズ、レイアウト、マッピングなど) を表示します。

以下のコマンドは、**vg00** 内にある **lv02** の属性を示しています。スナップショット論理ボリュームがこの元の論理ボリューム用に作成されている場合、このコマンドはすべてのスナップショット論理ボリュームとそのステータス (アクティブまたは非アクティブ) の一覧を表示します。

```
# lvdisplay -v /dev/vg00/lv02
```

**lvscan** コマンドは、システム内のすべての論理ボリュームをスキャンし、以下の例のように一覧表示します。

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

## 第9章 LVM 用のカスタム報告

LVM では、カスタマイズされたレポートを生成したり、レポートの出力をフィルタリングしたりするための様々な設定およびコマンドラインオプションが提供されます。LVM レポート機能の完全な説明は、man ページの **lvmreport(7)** を参照してください。

コマンドの **pvs**、**lvs**、および **vgs** を使用して、LVM オブジェクトについて簡潔でカスタマイズ可能なレポートを作成できます。このコマンドが生成するレポートには、オブジェクトごとに1行の出力が含まれます。各行には、オブジェクトに関連するプロパティのフィールドについて、順序付けられた一覧が含まれます。レポートするオブジェクトを選択する方法には、物理ボリューム別、ボリュームグループ別、論理ボリューム別、物理ボリュームセグメント別、および論理ボリュームセグメント別の5つの方法があります。

**lvm fullreport** コマンドを使用して、物理ボリューム、ボリュームグループ、論理ボリューム、物理ボリュームセグメント、および論理ボリュームセグメントに関する情報を一度に報告できます。このコマンドとその機能の詳細は、man ページの **lvm-fullreport(8)** を参照してください。

LVM は、LVM コマンドの実行中に収集された操作、メッセージ、および各オブジェクトのステータス(完全なオブジェクト ID 付き)のログが含まれるログレポートをサポートします。LVM ログレポートの詳細は、man ページの **lvmreport(7)** を参照してください。

### 9.1. LVM 表示の形式の制御

コマンドの **pvs**、**lvs**、または **vgs** のどれを使用するかによって、表示されるデフォルトのフィールドセットとソート順序が決定します。このコマンドの出力は、以下の引数を使用して制御できます。

- **-o** 引数を使用すると、表示するフィールドをデフォルト以外に変更できます。たとえば、以下のコマンドは、物理ボリュームの名前とサイズのみを表示します。

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

- **-o** 引数との組み合わせで使用するプラス記号 (+) を使用して、出力にフィールドを追加できます。以下の例は、デフォルトフィールドに加えて、物理ボリュームの UUID を表示しています。

```
# pvs -o +pv_uuid
PV VG Fmt Attr PSize PFree PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G Joqlch-yWSj-kuEn-ldwM-01S9-X08M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-UqkCS
```

- コマンドに **-v** 引数を追加すると、追加のフィールドが含まれます。たとえば、**pvs -v** コマンドは、デフォルトフィールドに加えて、**DevSize** フィールドと **PV UUID** フィールドも表示します。

```
# pvs -v
Scanning for physical volume names
PV VG Fmt Attr PSize PFree DevSize PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
```

```
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G 17.14G Joqlch-yWSj-kuEn-ldwM-01S9-XO8M-
mcpSve
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G 17.14G yfvZK-Cf31-j75k-dECm-0RZ3-0dGW-
tUqkCS
```

- **--noheadings** 引数は、見出し行を表示しません。これはスクリプトを作成する際に便利です。以下の例は、**pv\_name** 引数と共に **--noheadings** 引数を使用して、すべての物理ボリュームの一覧を生成しています。

```
# pvs --noheadings -o pv_name
/dev/sdb1
/dev/sdc1
/dev/sdd1
```

- **--separator separator** 引数は、**区切り文字** を使用して、各フィールドを区切ります。次の例は、**pvs** コマンドのデフォルト出力フィールドを等号 (=) で分割しています。

```
# pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
/dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
/dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

**separator** 引数の使用時にフィールドを配置するには、**--aligned** 引数とともに **separator** 引数を使用します。

```
# pvs --separator = --aligned
PV =VG =Fmt =Attr=PSize =PFree
/dev/sdb1 =new_vg=lvm2=a- =17.14G=17.14G
/dev/sdc1 =new_vg=lvm2=a- =17.14G=17.09G
/dev/sdd1 =new_vg=lvm2=a- =17.14G=17.14G
```

**lvs** コマンドまたは **vgs** コマンドの **-P** 引数を使用して、通常出力では表示されない、障害が発生したボリュームの情報を表示します

表示引数の一覧は、man ページの **pvs(8)**、**vgs(8)**、および **lvs(8)** を参照してください。

ボリュームグループフィールドは、物理ボリューム (および物理ボリュームセグメント) フィールド、または論理ボリューム (および論理ボリュームセグメント) フィールドと混在させることができますが、物理ボリュームフィールドと論理ボリュームフィールドは混在させることができません。たとえば、以下のコマンドは、1つの物理ボリュームにつき1行の出力を表示します。

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

## 9.2. LVM オブジェクト表示フィールド

このセクションでは、**pvs** コマンド、**vgs** コマンド、および **lvs** コマンドを使用して、LVM オブジェクトについて表示できる情報を一覧表示する一連の表を紹介します。

便宜上、フィールド名の接頭辞は、コマンドのデフォルトと一致する場合は省略できます。たとえば、**pvs** コマンドでは、**name** は **pv\_name** 解釈され、**vgs** コマンドでは、**name** は **vg\_name** と解釈されます。

以下のコマンドの実行は、**pvs -o pv\_free** の実行に相当します。

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



## 注記

**pvs**、**vgs**、および **lvs** の出力の属性フィールドにある文字数は、以降のリリースで増える可能性があります。既存の文字フィールドの位置は変更しませんが、新しいフィールドが末尾に追加される可能性があります。相対的な位置を使用して特定の属性文字を検索するスクリプトを作成する場合は、このことを考慮して、フィールドの終点ではなく、フィールドの始点を基点として文字検索を行います。たとえば、**lv\_attr** フィールドの9番目のビットの文字 **p** を検索する場合は、文字列「`^/.....p/`」で指定できます。「`/*p$`」は使用しないでください。

表9.1「**pvs コマンド表示フィールド**」は、**pvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、フィールドの説明を一覧にまとめています。

表9.1 **pvs** コマンド表示フィールド

引数	ヘッダー	説明
<b>dev_size</b>	DevSize	物理ボリュームを作成する基となるデバイスのサイズ
<b>pe_start</b>	1st PE	基となるデバイス内の最初の物理エクステントの開始点までのオフセット
<b>pv_attr</b>	Attr	物理ボリュームのステータス - (a)locatable または e(x)ported
<b>pv_fmt</b>	Fmt	物理ボリュームのメタデータ形式 ( <b>lvm2</b> または <b>lvm1</b> )
<b>pv_free</b>	PFree	物理ボリュームにある残りの空き領域
<b>pv_name</b>	PV	物理ボリュームの名前
<b>pv_pe_alloc_count</b>	Alloc	使用される物理エクステントの数
<b>pv_pe_count</b>	PE	物理エクステントの数
<b>pvseg_size</b>	SSize	物理ボリュームのセグメントサイズ
<b>pvseg_start</b>	Start	物理ボリュームセグメントの最初の物理エクステント

引数	ヘッダー	説明
<b>pv_size</b>	PSize	物理ボリュームのサイズ
<b>pv_tags</b>	PV Tags	物理ボリュームに割り当てられた LVM タグ
<b>pv_used</b>	Used	物理ボリュームで現在使用中の領域の量
<b>pv_uuid</b>	PV UUID	物理ボリュームの UUID

デフォルトで **pvs** コマンドが表示するフィールドは、**pv\_name**、**vg\_name**、**pv\_fmt**、**pv\_attr**、**pv\_size**、および **pv\_free** です。この表示は、**pv\_name** でソートされています。

```
# pvs
PV      VG   Fmt Attr PSize PFree
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new_vg lvm2 a- 17.14G 17.13G
```

**pvs** コマンドに **-v** 引数を使用すると、デフォルトの表示に、**dev\_size** フィールドおよび **pv\_uuid** フィールドが追加されます。

```
# pvs -v
Scanning for physical volume names
PV      VG   Fmt Attr PSize PFree DevSize PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G 17.14G Joqlch-yWSj-kuEn-ldwM-01S9-XO8M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.13G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

**pvs** コマンドに **--segments** 引数を使用すると、各物理ボリュームセグメントの情報を表示します。セグメントはエクステンツの集合です。セグメントの表示は、論理ボリュームがフラグメント化 (断片化) しているかどうかを確認するのに役立ちます。

デフォルトで **pvs --segments** コマンドが表示するフィールドは、**pv\_name**、**vg\_name**、**pv\_fmt**、**pv\_attr**、**pv\_size**、**pv\_free**、**pvseg\_start**、および **pvseg\_size** です。この表示は、物理ボリューム内では **pv\_name** および **pvseg\_size** でソートされています。

```
# pvs --segments
PV      VG      Fmt Attr PSize PFree Start SSize
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 0 1172
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 1172 16
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 1188 1
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 0 26
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 26 24
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 50 26
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 76 24
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 100 26
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 126 24
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 150 22
/dev/sda1 vg      lvm2 a- 17.14G 16.75G 172 4217
```

```

/dev/sdb1 vg      lvm2 a- 17.14G 17.14G  0 4389
/dev/sdc1 vg      lvm2 a- 17.14G 17.14G  0 4389
/dev/sdd1 vg      lvm2 a- 17.14G 17.14G  0 4389
/dev/sde1 vg      lvm2 a- 17.14G 17.14G  0 4389
/dev/sdf1 vg      lvm2 a- 17.14G 17.14G  0 4389
/dev/sdg1 vg      lvm2 a- 17.14G 17.14G  0 4389

```

**pvs -a** コマンドを使用して、LVM が検出し、LVM 物理ボリュームとして初期化していないデバイスを確認できます。

```

# pvs -a
PV                VG  Fmt Attr PSize PFree
/dev/VolGroup00/LogVol01  --  0  0
/dev/new_vg/lvol0        --  0  0
/dev/ram                --  0  0
/dev/ram0                --  0  0
/dev/ram2                --  0  0
/dev/ram3                --  0  0
/dev/ram4                --  0  0
/dev/ram5                --  0  0
/dev/ram6                --  0  0
/dev/root                --  0  0
/dev/sda                 --  0  0
/dev/sdb                 --  0  0
/dev/sdb1                new_vg lvm2 a- 17.14G 17.14G
/dev/sdc                 --  0  0
/dev/sdc1                new_vg lvm2 a- 17.14G 17.09G
/dev/sdd                 --  0  0
/dev/sdd1                new_vg lvm2 a- 17.14G 17.14G

```

表9.2 「vgs 表示フィールド」は、**vgs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。

表9.2 vgs 表示フィールド

引数	ヘッダー	説明
<b>lv_count</b>	#LV	ボリュームグループに含まれる論理ボリュームの数
<b>max_lv</b>	MaxLV	ボリュームグループで許容される論理ボリュームの最大数 (無制限の場合は 0)
<b>max_pv</b>	MaxPV	ボリュームグループで許容される物理ボリュームの最大数 (無制限の場合は 0)
<b>pv_count</b>	#PV	ボリュームグループを定義する物理ボリューム数
<b>snap_count</b>	#SN	ボリュームグループに含まれるスナップショット数
<b>vg_attr</b>	Attr	ボリュームグループのステータス - (w)riteable (書き込み可能)、(r)eadonly (読み取りのみ)、resi(z)eable (サイズ変更可能)、e(x)ported (エクスポート済)、(p)artial (部分的)、および (c)lustered (クラスター化)

引数	ヘッダー	説明
<b>vg_extent_count</b>	#Ext	ボリュームグループの物理エクステントの数
<b>vg_extent_size</b>	Ext	ボリュームグループの物理エクステントのサイズ
<b>vg_fmt</b>	Fmt	ボリュームグループのメタデータ形式 ( <b>lvm2</b> または <b>lvm1</b> )
<b>vg_free</b>	VFree	ボリュームグループの残りの空き領域のサイズ
<b>vg_free_count</b>	Free	ボリュームグループの空き物理エクステントの数
<b>vg_name</b>	VG	ボリュームグループ名
<b>vg_seqno</b>	Seq	ボリュームグループの改訂を示す番号
<b>vg_size</b>	VSize	ボリュームグループのサイズ
<b>vg_sysid</b>	SYS ID	LVM1 システム ID
<b>vg_tags</b>	VG Tags	ボリュームグループに割り当てられた LVM タグ
<b>vg_uuid</b>	VG UUID	ボリュームグループの UUID

デフォルトで **vgs** コマンドが表示するフィールド

は、**vg\_name**、**pv\_count**、**lv\_count**、**snap\_count**、**vg\_attr**、**vg\_size**、および **vg\_free** です。この表示は、**vg\_name** でソートされています。

#### # vgs

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

**vgs** コマンドに **-v** 引数を使用すると、デフォルトの表示に **vg\_extent\_size** および **vg\_uuid** のフィールドが追加されます。

#### # vgs -v

```
Finding all volume groups
Finding volume group "new_vg"
VG Attr Ext #PV #LV #SN VSize VFree VG UUID
new_vg wz--n- 4.00M 3 1 1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

表9.3 「**lvs** 表示フィールド」は、**lvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。



#### 注記

Red Hat Enterprise Linux の最近のリリースでは、**lvs** コマンドの出力にフィールドが追加されている場合があります。ただし、フィールドの順序は同じで、追加のフィールドは出力の最後に表示されます。

表9.3 lvs 表示フィールド

引数	ヘッダー	説明
* <b>chunksize</b> * <b>chunk_size</b>	Chunk	スナップショットボリュームのユニットサイズ
<b>copy_percent</b>	Copy%	ミラー化論理ボリュームの同期のパーセンテージ。さらに <b>pv_move</b> コマンドで物理エクステントを移動する時にも使用されます。
<b>devices</b>	Devices	論理ボリュームを構成するデバイス - 物理ボリューム、論理ボリューム、および物理エクステントと論理エクステントの開始点
<b>lv_ancestors</b>	Ancestors	シンプルスナップショットにおける、論理ボリュームの先祖 (ancestor)
<b>lv_descendants</b>	Descendants	シンプルスナップショットにおける、論理ボリュームの子孫 (descendant)
<b>lv_attr</b>	Attr	<p>論理ボリュームのステータス。論理ボリュームの属性ビットは以下ようになります。</p> <p>* ビット 1 - ボリュームタイプ - (m)irrored (ミラー化)、(M)irrored without initial sync (初期同期なしのミラー化)、(o)rigin (作成元)、(O)rigin with merging snapshot (マージするスナップショットがある作成元)、(r)aid (RAID)、(R)aid without initial sync (初期同期なしの RAID)、(s)napshot (スナップショット)、merging (S)napshot (マージするスナップショット)、(p)vmove (物理ボリュームの移動)、(v)irtual (仮想)、mirror or raid (i)mage (ミラーまたは RAID イメージ)、mirror or raid (l)mage out-of-sync (ミラーまたは RAID イメージの非同期)、mirror (l)og device (ミラーログデバイス)、under (c)onversion (変換中)、thin (V)olume (シンボリューム)、(t)hin pool (シンプル)、(T)hin pool data (シンプルデータ)、raid or thin pool m(e)tadata or pool metadata spare (RAID またはシンプルメタデータもしくはプールメタデータのスペア)</p> <p>* ビット 2 - パーミッション - (w)riteable (書き込み可能)、(r)ead-only (読み取り専用)、(R)ead-only activation of non-read-only volume (読み取り専用でないボリュームを読み取り専用にアクティブ化)</p> <p>* ビット 3 - 割り当てポリシー - (a)nywhere (どこでも)、(c)ontiguous (連続的)、(i)nherited (継承)、c(l)ing (膠着)、(n)ormal (通常)。これは、たとえば <b>pvmove</b> コマンドの実行時など、割り当ての変更に対してボリュームが現在ロックされている場合に大文字になります。</p> <p>* ビット 4 - 固定されたマイナー番号</p> <p>* ビット 5 - ステータス - (a)ctive (アクティブ)、</p>

引数	ヘッダー	説明
		<p>(s)uspended (サスペンド)、(l)invalid snapshot (無効なスナップショット)、invalid (S)uspended snapshot (無効なサスペンドされたスナップショット)、snapshot (m)erge failed (スナップショットのマージが失敗)、suspended snapshot (M)erge failed (サスペンドされたスナップショットのマージが失敗)、mapped (d)evice present without tables (テーブルのないマッピングされたデバイス)、mapped device present with (i)nactive table (非アクティブのテーブルを持つマッピングされたデバイス)</p> <p>* ビット 6 - デバイス開放(o)</p> <p>* ビット 7 - ターゲットタイプ - (m)irror (ミラー)、(r)aid (RAID)、(s)napshot (スナップショット)、(t)hin (シン)、(u)nkown (不明)、(v)irtual (仮想)。これは、同じカーネルターゲットに関連する論理ボリュームをまとめます。たとえば、ミラーイメージ、ミラーログ、ミラー自体が、元のデバイスマッパーのミラーカーネルドライバーを使用する場合は、(m)と表示されます。md raid カーネルドライバーを使用する同等の RAID はすべて (r)と表示されます。元のデバイスマッパードライバーを使用するスナップショットは (s)と表示され、シンプロビジョニングドライバーを使用するシンボリュームのスナップショットは (t)と表示されます。</p> <p>* ビット 8 - 新しく割り当てられたデータブロックは使用前に、ゼロ(z)のブロックで上書きされます。</p> <p>* ビット 9 - ボリュームの正常性 - (p)artial (部分的)、(r)efresh needed (更新が必要)、(m)ismatches exist (不一致が存在)、(w)ritemostly (書き込み多発)。部分的 (p) は、この論理ボリュームが使用する 1 つ以上の物理ボリュームがシステムから欠落していることを表します。更新 (r) は、この RAID 論理ボリュームが使用する 1 つ以上の物理ボリュームで書き込みエラーが発生したことを表します。書き込みエラーは、その物理ボリュームの一時的な障害により引き起こされたか、または物理ボリュームに障害があることを示すかのいずれかの可能性があります。デバイスは更新するか、または置き換える必要があります。不一致 (m) は、RAID 論理ボリュームのアレイに一貫していない部分があることを表します。不整合は、RAID 論理ボリュームで <b>check</b> 操作を開始すると検出されます。(スクラビング操作 <b>check</b> および <b>repair</b> は、<b>lvchange</b> コマンドにより RAID 論理ボリューム上で実行できます。) 書き込み多発 (w) は、write-mostly とマークが付けられた RAID 1 論理ボリュームのデバイスを表します。</p> <p>* ビット 10 - s(k)ip activation (アクティブ化のスキップ - このボリュームには、アクティブ化の実行時にスキップされるようにフラグが設定されます。</p>
<b>lv_kernel_major</b>	KMaj	論理ボリュームの実際のメジャーデバイス番号 (非アクティブの場合は -1)
<b>lv_kernel_minor</b>	KMIN	論理ボリュームの実際のマイナーデバイス番号 (非アクティブの場合は -1)

引数	ヘッダー	説明
<b>lv_major</b>	Maj	論理ボリュームの永続的なメジャーデバイス番号 (未指定の場合は -1)
<b>lv_minor</b>	Min	論理ボリュームの永続的なマイナーデバイス番号 (未指定の場合は -1)
<b>lv_name</b>	LV	論理ボリュームの名前
<b>lv_size</b>	LSize	論理ボリュームのサイズ
<b>lv_tags</b>	LV Tags	論理ボリュームに割り当てられた LVM タグ
<b>lv_uuid</b>	LV UUID	論理ボリュームの UUID
<b>mirror_log</b>	Log	ミラーログが存在するデバイス
<b>modules</b>	Modules	この論理ボリュームを使用するのに必要な、対応するカーネルデバイスマッパーターゲット
<b>move_pv</b>	Move	<b>pvmove</b> コマンドで作成された一時的な論理ボリュームの基となる物理ボリューム
<b>origin</b>	Origin	スナップショットボリュームの作成元のデバイス
<b>* regionsize</b> <b>* region_size</b>	Region	ミラー化論理ボリュームのユニットサイズ
<b>seg_count</b>	#Seg	論理ボリュームのセグメント数
<b>seg_size</b>	SSize	論理ボリュームのセグメントサイズ
<b>seg_start</b>	Start	論理ボリュームのセグメントのオフセット
<b>seg_tags</b>	Seg Tags	論理ボリュームのセグメントに割り当てられた LVM タグ
<b>segtype</b>	タイプ	論理ボリュームのセグメントタイプ (例: ミラー、ストライプ、リニア)
<b>snap_percent</b>	Snap%	使用中スナップショットボリュームの現在のパーセンテージ
<b>stripes</b>	#Str	論理ボリュームのストライプ、またはミラーの数

引数	ヘッダー	説明
* <b>stripesize</b>	Stripe	ストライプ化論理ボリュームのストライプのユニットサイズ
* <b>stripe_size</b>		

デフォルトで **lvs** コマンドが表示するのは以下になります。デフォルトの表示は、ボリュームグループ内では **vg\_name** および **lv\_name** でソートされます。

```
# lvs
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
origin VG owi-a-s--- 1.00g
snap VG swi-a-s--- 100.00m origin 0.00
```

**lvs** コマンドの一般的な用途は、論理ボリュームを構成するデバイスを表示するコマンドに、**devices** を追加することです。また、この例では、**-a** オプションを指定して、RAID ミラーなどの論理ボリュームのコンポーネントである内部ボリュームを、括弧で囲んで表示します。この例には、RAID ボリューム、ストライプのボリューム、シンプルなボリュームが含まれます。

```
# lvs -a -o +devices
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
raid1 VG rwi-a-r--- 1.00g 100.00
raid1_rimage_0(0),raid1_rimage_1(0)
[raid1_rimage_0] VG iwi-a-or--- 1.00g /dev/sde1(7041)
[raid1_rimage_1] VG iwi-a-or--- 1.00g /dev/sdf1(7041)
[raid1_rmeta_0] VG ewi-a-or--- 4.00m /dev/sde1(7040)
[raid1_rmeta_1] VG ewi-a-or--- 4.00m /dev/sdf1(7040)
stripe1 VG -wi-a----- 99.95g /dev/sde1(0),/dev/sdf1(0)
stripe1 VG -wi-a----- 99.95g /dev/sdd1(0)
stripe1 VG -wi-a----- 99.95g /dev/sdc1(0)
[lvol0_pmspare] rhel_host-083 ewi----- 4.00m /dev/vda2(0)
pool00 rhel_host-083 twi-aotz-- <4.79g 72.90 54.69
pool00_tdata(0)
[pool00_tdata] rhel_host-083 Twi-ao---- <4.79g /dev/vda2(1)
[pool00_tmeta] rhel_host-083 ewi-ao---- 4.00m /dev/vda2(1226)
root rhel_host-083 Vwi-aotz-- <4.79g pool00 72.90
swap rhel_host-083 -wi-ao---- 820.00m /dev/vda2(1227)
```

**lvs** コマンドで **-v** 引数を使用して、デフォルトの表示に、**seg\_count**、**lv\_major**、**lv\_minor**、**lv\_kernel\_major**、**lv\_kernel\_minor**、**lv\_uuid** のフィールドを追加します。

```
# lvs -v
Finding all logical volumes
LV VG #Seg Attr LSize Maj Min KMaj KMin Origin Snap% Move Copy% Log Convert LV
UUID
lvol0 new_vg 1 owi-a- 52.00M -1 -1 253 3 LBy1Tz-sr23-Ojsl-LT03-
nHLC-y8XW-EhCI78
newvgsnap1 new_vg 1 swi-a- 8.00M -1 -1 253 5 lvol0 0.20 1ye1OU-1clu-
o79k-20h2-ZGF0-qCJm-Cfbslx
```

**lvs** コマンドの **--segments** 引数を使用して、セグメント情報を強調するデフォルトの列で情報を表示

できます。**segments** 引数を使用する場合、**seg** プレフィックスは必要に応じて使用します。デフォルトで **lvs --segments** コマンドが表示するフィールドは、**lv\_name**、**vg\_name**、**lv\_attr**、**stripes**、**segtype**、および **seg\_size** です。デフォルトの表示は、ボリュームグループ内では **vg\_name**、**lv\_name** でソートされ、論理ボリュームでは **seg\_start** でソートされます。論理ボリュームが断片化されると、このコマンドの出力が表示されます。

```
# lvs --segments
LV   VG      Attr #Str Type  SSize
LogVol00 VolGroup00 -wi-ao 1 linear 36.62G
LogVol01 VolGroup00 -wi-ao 1 linear 512.00M
lv   vg      -wi-a- 1 linear 104.00M
lv   vg      -wi-a- 1 linear 104.00M
lv   vg      -wi-a- 1 linear 104.00M
lv   vg      -wi-a- 1 linear 88.00M
```

**lvs --segments** コマンドで **-v** 引数を使用すると、デフォルトの表示に **seg\_start**、**stripesize**、**chunksize** のフィールドが追加されます。

```
# lvs -v --segments
Finding all logical volumes
LV   VG      Attr Start SSize #Str Type  Stripe Chunk
lv00  new_vg  owi-a- 0 52.00M 1 linear 0 0
newvg snap1 new_vg  swi-a- 0 8.00M 1 linear 0 8.00K
```

以下の1つ目の例は、設定された論理ボリュームが1つあるシステムで実行した **lvs** コマンドのデフォルト出力を示しています。その次の例は、**segments** 引数を指定した **lvs** コマンドのデフォルト出力を表示しています。

```
# lvs
LV VG Attr LSize Origin Snap% Move Log Copy%
lv00 new_vg -wi-a- 52.00M
# lvs --segments
LV VG Attr #Str Type  SSize
lv00 new_vg -wi-a- 1 linear 52.00M
```

### 9.3. LVM 報告のソート

通常、**lvs** コマンド、**vgs** コマンド、または **pvs** コマンドの出力全体をソートして、コラムを正しく配置する場合は、出力を生成して内部に保管する必要があります。**--unbuffered** 引数を指定すると、生成直後にソートされていないままの出力で表示できます。

別の順列のコラム一覧のソートを指定するには、報告コマンドのいずれかと一緒に **-O** 引数を使用します。出力自体に、このフィールドを含める必要はありません。

以下の例では、物理ボリュームの名前、サイズ、および空き領域を表示する **pvs** コマンドの出力を示しています。

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

以下の例では、空き領域のフィールドでソートされた同じ出力を示しています。

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

以下の例では、ソートするフィールドを表示する必要がないことを示しています。

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

逆順でソートするには、**-O** 引数の後で指定するフィールドの先頭に **-** 印を付けます。

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

## 9.4. LVM レポート表示への単位の指定

LVM 報告表示用の単位を指定するには、報告コマンドに **--units** 引数を使用します。バイト (b)、キロバイト (k)、メガバイト (m)、ギガバイト (g)、テラバイト (t)、エクサバイト (e)、ペタバイト (p)、および人間が判読できる表示 (h) を指定できます。デフォルトは人間が判読できる表示です。このデフォルト設定を上書きするには、**/etc/lvm/lvm.conf** ファイルの **global** セクションに **units** パラメーターを設定します。

以下の例では、**pvs** コマンドの出力をデフォルトのギガバイトでなく、メガバイトで指定しています。

```
# pvs --units m
PV      VG   Fmt Attr PSize   PFree
/dev/sda1    lvm2 -- 17555.40M 17555.40M
/dev/sdb1 new_vg lvm2 a- 17552.00M 17552.00M
/dev/sdc1 new_vg lvm2 a- 17552.00M 17500.00M
/dev/sdd1 new_vg lvm2 a- 17552.00M 17552.00M
```

デフォルトでは、単位は 2 の累乗 (1024 の倍数) で表示されます。単位を 1000 の倍数で表示するには、大文字 (B、K、M、G、T、H) で単位を指定できます。

以下のコマンドは、1024 の倍数 (デフォルト) で出力を表示します。

```
# pvs
PV      VG   Fmt Attr PSize   PFree
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G
```

以下のコマンドは、1000 の倍数で出力を表示します。

```
# pvs --units G
```

```
PV    VG    Fmt Attr PSize PFree
/dev/sdb1 new_vg lvm2 a- 18.40G 18.40G
/dev/sdc1 new_vg lvm2 a- 18.40G 18.35G
/dev/sdd1 new_vg lvm2 a- 18.40G 18.40G
```

セクター (512 バイトとして定義) またはカスタム単位も指定できます。

以下の例では、**pvs** コマンドの出力をセクター数として表示しています。

```
# pvs --units s
PV    VG    Fmt Attr PSize  PFree
/dev/sdb1 new_vg lvm2 a- 35946496S 35946496S
/dev/sdc1 new_vg lvm2 a- 35946496S 35840000S
/dev/sdd1 new_vg lvm2 a- 35946496S 35946496S
```

以下の例では、**pvs** コマンドの出力を 4 MB 単位で表示しています。

```
# pvs --units 4m
PV    VG    Fmt Attr PSize  PFree
/dev/sdb1 new_vg lvm2 a- 4388.00U 4388.00U
/dev/sdc1 new_vg lvm2 a- 4388.00U 4375.00U
/dev/sdd1 new_vg lvm2 a- 4388.00U 4388.00U
```

## 9.5. JSON 形式で LVM コマンド結果の表示

LVM 表示コマンドで **--reportformat** オプションを使用して、JSON 形式で出力を表示できます。

以下の例では、標準的なデフォルト形式の **lvs** の出力を示しています。

```
# lvs
LV    VG      Attr    LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
my_raid my_vg    Rwi-a-r--- 12.00m
root  rhel_host-075 -wi-ao---- 6.67g
swap  rhel_host-075 -wi-ao---- 820.00m
```

以下のコマンドは、JSON 形式を指定する場合と同じ LVM 設定の出力を表示します。

```
# lvs --reportformat json
{
  "report": [
    {
      "lv": [
        {"lv_name": "my_raid", "vg_name": "my_vg", "lv_attr": "Rwi-a-r---", "lv_size": "12.00m",
        "pool_lv": "", "origin": "", "data_percent": "", "metadata_percent": "", "move_pv": "", "mirror_log": "",
        "copy_percent": "100.00", "convert_lv": ""},
        {"lv_name": "root", "vg_name": "rhel_host-075", "lv_attr": "-wi-ao----", "lv_size": "6.67g",
        "pool_lv": "", "origin": "", "data_percent": "", "metadata_percent": "", "move_pv": "", "mirror_log": "",
        "copy_percent": "", "convert_lv": ""},
        {"lv_name": "swap", "vg_name": "rhel_host-075", "lv_attr": "-wi-ao----", "lv_size": "820.00m",
        "pool_lv": "", "origin": "", "data_percent": "", "metadata_percent": "", "move_pv": "", "mirror_log": "",
        "copy_percent": "", "convert_lv": ""}
      ]
    }
  ]
}
```

```

    }
  ]
}

```

また、`/etc/lvm/lvm.conf` ファイルで `output_format` 設定を使用して、レポート形式を設定オプションとして設定することもできます。ただし、コマンドラインの `--reportformat` 設定は、この設定よりも優先されます。

## 9.6. LVM コマンドログの表示

レポート指向および処理指向の LVM コマンドを使用して、コマンドログを報告できます (これが `log/report_command_log` 設定で有効になっている場合)。このレポートで表示およびソートするフィールドセットを決定できます。

以下の例では、LVM コマンド向けの完全なログレポートを生成するように LVM を設定します。この例では、論理ボリューム `lvol0` と `lvol1` の両方が、それらの論理ボリュームを含むボリュームグループ `VG` とともに正常に処理されたことを確認できます。

```

# lvmconfig --type full log/command_log_selection
command_log_selection="all"

# lvs
Logical Volume
=====
LV   LSize Cpy%Sync
lvol1 4.00m 100.00
lvol0 4.00m

Command Log
=====
Seq LogType Context  ObjType ObjName ObjGrp  Msg  Errno RetCode
  1 status processing lv   lvol0  vg   success  0    1
  2 status processing lv   lvol1  vg   success  0    1
  3 status processing vg   vg     success  0    1

# lvchange -an vg/lvol1
Command Log
=====
Seq LogType Context  ObjType ObjName ObjGrp  Msg  Errno RetCode
  1 status processing lv   lvol1  vg   success  0    1
  2 status processing vg   vg     success  0    1

```

LVM レポートおよびコマンドログの設定の詳細は、man ページの `lvmreport` を参照してください。

## 第10章 RAID 論理ボリュームの設定

LVM は、RAID0、1、4、5、6、および10に対応します。



### 注記

RAID 論理ボリュームはクラスターには対応していません。RAID 論理ボリュームは、1台のマシンに作成して排他的にアクティブにすることができますが、複数のマシンで同時にアクティブにすることはできません。

RAID 論理ボリュームを作成するには、RAID の種類を **lvcreate** コマンドの **--type** 引数として指定します。表10.1「RAID のセグメントタイプ」は、指定できる RAID セグメントタイプを説明します。LVM で RAID 論理ボリュームを作成した後、他の LVM 論理ボリュームと同じようにボリュームをアクティブ化、変更、削除、表示、および使用することができます。

表10.1 RAID のセグメントタイプ

セグメントタイプ	説明
<b>raid1</b>	RAID1 ミラーリング。-m を指定し、ストライピングを指定しない場合は、 <b>lvcreate</b> コマンドの <b>--type</b> 引数のデフォルト値になります。
<b>raid4</b>	RAID4 専用パリティディスク
<b>raid5</b>	<b>raid5_ls</b> と同じ
<b>raid5_la</b>	* RAID5 left asymmetric * ロータートパリティ 0 + データ継続
<b>raid5_ra</b>	* RAID5 right asymmetric * ロータートパリティ N + データ継続
<b>raid5_ls</b>	* RAID5 left symmetric * ロータートパリティ 0 + データ再起動
<b>raid5_rs</b>	* RAID5 right symmetric * ロータートパリティ N + データ再起動
<b>raid6</b>	<b>raid6_zr</b> と同じ
<b>raid6_zr</b>	* RAID6 zero restart * ロータートパリティゼロ (左から右) + データ再起動

セグメントタイプ	説明
<b>raid6_nr</b>	<ul style="list-style-type: none"> <li>* RAID6 N restart</li> <li>* ロータートパリティ N (左から右) + データ再起動</li> </ul>
<b>raid6_nc</b>	<ul style="list-style-type: none"> <li>* RAID6 N continue</li> <li>* ロータートパリティ N (左から右) + データを継続</li> </ul>
<b>raid10</b>	<ul style="list-style-type: none"> <li>* ストライピング + ミラーリング。 <b>-m</b> を指定し、1 よりも大きい数をストライプの数として指定すると、これは <b>lvcreate</b> コマンドの <b>--type</b> 引数のデフォルト値になります。</li> <li>* ミラーセットのストライピング</li> </ul>
<b>raid0/raid0_meta</b>	<p>ストライピング。RAID0 では、ストライプサイズの単位で、複数のデータサブボリュームに論理ボリュームデータが分散されます。これは、パフォーマンスを向上させるために使用します。論理ボリュームのデータは、いずれかのデータサブボリュームで障害が発生すると失われます。</p>

大概是、5つのプライマリータイプ (**raid1**、**raid4**、**raid5**、**raid6**、**raid10**) のいずれかを指定すれば対応できるはずですが。RAID 5 または 6 が使用する各種アルゴリズムの詳細は、[http://www.snia.org/sites/default/files/SNIA\\_DDF\\_Technical\\_Position\\_v2.0.pdf](http://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf) に記載されている「**Common RAID Disk Data Format Specification**」の第 4 章を参照してください。

RAID 論理ボリュームを作成するとき、LVM は、データまたはアレイ内のパリティサブボリュームごとに、サイズが 1 エクステントのメタデータサブボリュームを作成します。たとえば、2 方向の RAID1 アレイを作成すると、メタデータサブボリュームが 2 つ (**lv\_rmeta\_0** および **lv\_rmeta\_1**) と、データサブボリュームが 2 つ (**lv\_rimage\_0** および **lv\_rimage\_1**) 作成されます。同様に、3 way ストライプ (および暗黙的なパリティデバイスが 1 つ) の RAID4 を作成すると、メタデータサブボリュームが 4 つ (**lv\_rmeta\_0**、**lv\_rmeta\_1**、**lv\_rmeta\_2**、および **lv\_rmeta\_3**)、データサブボリュームが 4 つ (**lv\_rimage\_0**、**lv\_rimage\_1**、**lv\_rimage\_2**、および **lv\_rimage\_3**) が作成されます。



### 注記

LVM RAID Calculator を使用すると、RAID ストレージで論理ボリュームを作成するコマンドを生成できます。このアプリケーションは、現在のストレージまたは作成予定のストレージについて入力した情報を使用してコマンドを生成します。LVM RAID Calculator アプリケーションは、<https://access.redhat.com/labs/lvmraidcalculator/> で使用できません。

## 10.1. RAID 論理ボリュームの作成

ここでは、異なる種類の RAID 論理ボリュームを作成するコマンドの例を取り上げます。

**-m** 引数にコピー数を指定して、RAID1 アレイを作成できます。同様に、**-i argument** オプションに、RAID 4、5、または 6 の論理ボリュームのストライプ数を指定します。**-l** 引数で、ストライプのサイズを指定することもできます。

以下のコマンドは、ボリュームグループ **my\_vg** 内に、1 ギガバイトの 2 方向 RAID1 アレイ **my\_lv** を作成します。

```
# lvcreate --type raid1 -m 1 -L 1G -n my_lv my_vg
```

以下のコマンドは、ボリュームグループ **my\_vg** に、サイズが1ギガバイトで、名前が **my\_lv** の RAID5 アレイ (ストライプ3つ + 暗黙的なパリティードライブ1つ) を作成します。ストライプ数の指定は、LVM ストライプ化ボリュームの場合と同じように行います。パリティードライブは、正確な数だけ自動的に追加されます。

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

以下のコマンドは、ボリュームグループ **my\_vg** に、サイズが1ギガバイトで、名前が **my\_lv** の RAID6 アレイ (ストライプ3つ + 暗黙的なパリティードライブ2つ) を作成します。

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

## 10.2. RAID0 (ストライピング) 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のデータサブボリューム全体に分散します。

RAID0 ボリュームを作成するコマンドの書式は以下のとおりです。

```
lvcreate --type raid0[_meta] --stripes Stripes --stripesize StripeSize VolumeGroup
[PhysicalVolumePath ...]
```

表10.2 RAID0 コマンドの作成に関するパラメーター

パラメーター	説明
<b>--type raid0[_meta]</b>	<b>raid0</b> を指定すると、メタデータボリュームなしで RAID0 ボリュームが作成されます。 <b>raid0_meta</b> を指定すると、メタデータボリュームとともに RAID0 ボリュームが作成されます。RAID0 には耐障害性がないため、RAID1/10 の場合のようにミラーリングされたすべてのデータブロックを格納したり、RAID4/5/6 の場合のようにすべてのパリティブロックを計算して格納する必要はありません。したがって、ミラーリングされたブロックまたはパリティブロックの再同期の進行状態を把握するメタデータボリュームは必要ありません。ただし、RAID0 から RAID4/5/6/10 に変換するには、メタデータボリュームが必要です。 <b>raid0_meta</b> を指定すると、割り当ての失敗を防ぐために、このメタデータが事前に割り当てられます。

パラメーター	説明
<b>--stripes Stripes</b>	論理ボリュームを分散するデバイスの数を指定します。
<b>--stripesize StripeSize</b>	各ストライプのサイズをキロバイト単位で指定します。これは、次のデバイスに移動する前にデバイスに書き込まれるデータの量です。
<b>VolumeGroup</b>	使用するボリュームグループを指定します。
<b>PhysicalVolumePath ...</b>	使用するデバイスを指定します。指定しない場合は、LVM により、 <b>Stripes</b> オプションに指定されているデバイスの数が、各ストライプに1つずつ選択されます。

この手順例では、`/dev/sda1`、`/dev/sdb1`、および `/dev/sdc1` のディスクにデータのストライプを作成する **mylv** という名前の LVM RAID0 論理ボリュームを作成します。

1. **pvcreate** コマンドを使用し、ボリュームグループで LVM 物理ボリュームとして使用するディスクにラベルを付けます。



#### 警告

このコマンドは、`/dev/sda1`、`/dev/sdb1`、および `/dev/sdc1` にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sdb1" successfully created
Physical volume "/dev/sdc1" successfully created
```

2. ボリュームグループ **myvg** を作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
Volume group "myvg" successfully created
```

**vgs** コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
VG #PV #LV #SN Attr VSize VFree
myvg 3 0 0 wz--n- 51.45G 51.45G
```

3. 作成したボリュームグループから、RAID0 論理ボリュームを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** から、RAID0 ボリューム **mylv** が作成されます。この例

では、ストライプサイズが4キロバイトの3つのストライプがある、サイズが2ギガバイトの論理ボリュームが作成されます。

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

4. RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 525312 4k blocks and 131376 inodes
Filesystem UUID: 9d4c0704-6028-450a-8b0a-8875358c0511
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

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

### 10.3. RAID ボリュームを初期化する速度の制御

RAID10 論理ボリュームを作成する際に、**sync** 操作で論理ボリュームを初期化するのに必要なバックグラウンド I/O は、特に RAID 論理ボリュームを多数作成している場合に、その他の I/O 操作 (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。

RAID 論理ボリュームが初期化される速度は、復旧スロットルを実装することで制御できます。**sync** 操作が実行される速度は、**lvcreate** コマンドの **--minrecoverrate** オプションおよび **--maxrecoverrate** オプションで、この操作の最小および最大 I/O 速度を設定すると制御できます。オプションは以下のよう指定します。

- **--maxrecoverrate Rate[bBsSkMmGg]**  
RAID 論理ボリュームの最大復旧速度を設定し、通常の I/O 操作が押し出されないようにします。**Rate** には、アレイ内の各デバイスに対する1秒あたりのデータ通信量を指定します。サフィックスを指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。復旧速度を 0 に設定すると無制限になります。
- **--minrecoverrate Rate[bBsSkMmGg]**  
RAID 論理ボリュームの最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。**Rate** には、アレイ内の各デバイスに対する1秒あたりのデータ通信量を指定します。サフィックスを指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。

以下のコマンドは、最大速度が128 kiB/sec/device で、サイズが10 ギガバイトのストライプが3つある、2方向の RAID10 アレイを作成します。このアレイは名前は **my\_lv** で、ボリュームグループは **my\_vg** になります。

```
# lvcreate --type raid10 -i 2 -m 1 -L 10G --maxrecoveryrate 128 -n my_lv my_vg
```

RAID のスクラブ操作の最小および最大復旧速度を指定することもできます。

## 10.4. リニアデバイスの RAID デバイスへの変換

既存のリニア論理ボリュームを RAID デバイスに変換するには、**lvconvert** コマンドの **--type** 引数を使用します。

以下のコマンドは、ボリュームグループ **my\_vg** のリニア論理ボリューム **my\_lv** を、2方向の RAID1 アレイに変換します。

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
```

RAID 論理ボリュームは、メタデータとデータサブボリュームのペアで構成されているため、リニアデバイスを RAID1 アレイに変換すると、メタデータサブボリュームが作成され、リニアボリュームが存在する物理ボリューム (のいずれか) にある、作成元の論理ボリュームに関連付けられます。イメージは、メタデータ/データサブボリュームのペアに追加されます。たとえば、作成元のデバイスは以下のとおりです。

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

2方向の RAID1 アレイに変換すると、デバイスには、以下のデータとメタデータサブボリュームのペアが含まれます。

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV      Copy%  Devices
my_lv   6.25  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(0)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rmeta_0]   /dev/sde1(256)
[my_lv_rmeta_1]   /dev/sdf1(0)
```

作成元の論理ボリュームとペアのメタデータイメージを同じ物理ボリュームに配置できないと、**lvconvert** は失敗します。

## 10.5. LVM RAID1 論理ボリュームを LVM リニア論理ボリュームに変換

**lvconvert** コマンドを使用して、既存の RAID1 LVM 論理ボリュームを LVM リニア論理ボリュームに変換する場合は、**-m0** 引数を指定します。これにより、RAID アレイを構成する全 RAID データサブボリュームおよび全 RAID メタデータサブボリュームが削除され、最高レベルの RAID1 イメージがリニア論理ボリュームとして残されます。

以下の例は、既存の LVM RAID1 論理ボリュームを示しています。

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

```

LV          Copy% Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] /dev/sdf1(1)
[my_lv_rmeta_0]  /dev/sde1(0)
[my_lv_rmeta_1]  /dev/sdf1(0)

```

以下のコマンドは、LVM RAID1 論理ボリューム **my\_vg/my\_lv** を、LVM リニアデバイスに変換します。

```

# lvconvert -m0 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV   Copy% Devices
my_lv /dev/sde1(1)

```

LVM RAID1 論理ボリューム を LVM リニアボリュームに変換する場合は、削除する物理ボリュームを指定できます。以下の例は、**/dev/sda1** と **/dev/sdb1** の2つのイメージで構成される LVM RAID1 論理ボリュームのレイアウトを示しています。この例で、**lvconvert** コマンドは **/dev/sda1** を削除して、**/dev/sdb1** をリニアデバイスを構成する物理ボリュームとして残すように指定します。

```

# lvs -a -o name,copy_percent,devices my_vg
LV          Copy% Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0] /dev/sda1(1)
[my_lv_rimage_1] /dev/sdb1(1)
[my_lv_rmeta_0]  /dev/sda1(0)
[my_lv_rmeta_1]  /dev/sdb1(0)
# lvconvert -m0 my_vg/my_lv /dev/sda1
# lvs -a -o name,copy_percent,devices my_vg
LV   Copy% Devices
my_lv /dev/sdb1(1)

```

## 10.6. ミラー化 LVM デバイスを RAID1 デバイスに変換

**lvconvert** コマンドを使用して、セグメントタイプが **mirror** の既存のミラー化 LVM デバイスを、RAID1 LVM デバイスに変換するには、**--type raid1** 引数を指定します。これにより、ミラーサブボリューム (**mimage**) の名前が、RAID サブボリューム (**rimage**) に変更します。また、ミラーログは削除され、対応するデータサブボリュームと同じ物理ボリュームのデータサブボリューム用に、メタデータサブボリューム (**rmeta**) が作成されます。

以下の例は、ミラー化論理ボリューム **my\_vg/my\_lv** のレイアウトを示しています。

```

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

```

以下のコマンドは、ミラー化論理ボリューム **my\_vg/my\_lv** を、RAID1 論理ボリュームに変換します。

```

# lvconvert --type raid1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy% Devices

```

```
my_lv      100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]    /dev/sde1(0)
[my_lv_rimage_1]    /dev/sdf1(0)
[my_lv_rmeta_0]     /dev/sde1(125)
[my_lv_rmeta_1]     /dev/sdf1(125)
```

## 10.7. RAID 論理ボリュームのサイズ変更

RAID 論理ボリュームのサイズは、以下の方法で変更できます。

- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** コマンドまたは **lvextend** コマンドで増やすことができます。これは、RAID イメージの数を変更するものではありません。ストライプ化 RAID 論理ボリュームでは、ストライプ化 RAID 論理ボリュームの作成時と同じ、ストライプを丸める制約が適用されます。
- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** コマンドまたは **lvreduce** コマンドで減らすことができます。これは、RAID イメージの数を変更するものではありません。**lvextend** コマンドでは、ストライプ化 RAID 論理ボリュームの作成時と同じストライプを丸める制約が適用されます。
- **lvconvert** コマンドで **--stripes N** パラメーターを使用すると、ストライプ化 RAID 論理ボリューム (**raid4/5/6/10**) のストライプの数を変更できます。このように、ストライプを追加または削除することで、RAID 論理ボリュームのサイズを増減できます。**raid10** ボリュームにはストライプを追加することしかできないため注意してください。この機能は、同じ RAID レベルを維持しながら、RAID 論理ボリュームの属性を変更できる、RAID の **再成形** 機能の一部になります。RAID 再成形と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再生成する例は、man ページの **lvraid(7)** を参照してください。

## 10.8. 既存の RAID1 デバイスのイメージ数を変更

既存の RAID1 アレイ内のイメージ数は、LVM ミラーリングの初期実装でイメージ数を変更する場合と同様に変更できます。**lvconvert** コマンドを使用して、追加または削除するメタデータ/データサブボリュームのペアの数を指定します。

**lvconvert** コマンドを使用して RAID1 デバイスにイメージを追加する際、追加後のイメージ数を指定するか、デバイスに追加するイメージ数を指定できます。メタデータ/データイメージのペアを置く物理ボリュームを指定することもできます。

メタデータサブボリューム (**rmeta** と呼ばれる) は、対応するデータサブボリューム (**rimage**) と同じ物理デバイスに常に存在します。メタデータ/データのサブボリュームのペアは、**--alloc anywhere** を指定しない限り、RAID アレイにある別のメタデータ/データサブボリュームのペアと同じ物理ボリュームには作成されません。

RAID1 ボリュームにイメージを追加するコマンドの形式は、以下のとおりです。

```
lvconvert -m new_absolute_count vg/lv [removable_PVs]
lvconvert -m +num_additional_images vg/lv [removable_PVs]
```

たとえば、以下のコマンドは、2 方向の RAID1 アレイである LVM デバイス **my\_vg/my\_lv** を表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv      6.25  my_lv_rimage_0(0),my_lv_rimage_1(0)
```

```
[my_lv_rimage_0]    /dev/sde1(0)
[my_lv_rimage_1]    /dev/sdf1(1)
[my_lv_rmeta_0]     /dev/sde1(256)
[my_lv_rmeta_1]     /dev/sdf1(0)
```

以下のコマンドは、2方向の RAID1 デバイス **my\_vg/my\_lv** を、3方向の RAID1 デバイスに変換します。

```
# lvconvert -m 2 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       6.25  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sde1(0)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rimage_2]  /dev/sdg1(1)
[my_lv_rmeta_0]   /dev/sde1(256)
[my_lv_rmeta_1]   /dev/sdf1(0)
[my_lv_rmeta_2]   /dev/sdg1(0)
```

イメージを RAID1 アレイに追加する場合は、イメージに使用する物理ボリュームを指定できます。以下のコマンドは、2方向の RAID1 デバイス **my\_vg/my\_lv** を、3方向の RAID1 デバイスに変換し、物理ボリューム **/dev/sdd1** がアレイに使用されるようにします。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       56.00  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdb1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdb1(0)
# lvconvert -m 2 my_vg/my_lv /dev/sdd1
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       28.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdb1(1)
[my_lv_rimage_2]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdb1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
```

RAID1 アレイからイメージを削除するには、以下のコマンドを使用します。**lvconvert** コマンドを使用して RAID1 デバイスからイメージを削除する場合は、削除後のイメージの合計数を指定するか、デバイスから削除するイメージ数を指定することもできます。または、デバイスを削除する物理ボリュームを指定することもできます。

```
lvconvert -m new_absolute_count vg/lv [removable_PVs]
lvconvert -m -num_fewer_images vg/lv [removable_PVs]
```

また、イメージとその関連付けられたメタデータのサブボリュームを削除すると、それよりも大きな番号のイメージがそのスロットを引き継ぎます。たとえば、**lv\_rimage\_0**、**lv\_rimage\_1**、および **lv\_rimage\_2** で構成される 3 方向の RAID1 アレイから **lv\_rimage\_1** を削除すると、RAID1 アレイを構成するイメージの名前は **lv\_rimage\_0** と **lv\_rimage\_1** になります。サブボリューム **lv\_rimage\_2** の名前が、空のスロットを引き継いで **lv\_rimage\_1** になります。

以下の例は、3方向の RAID1 論理ボリューム **my\_vg/my\_lv** のレイアウトを示しています。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rimage_2]  /dev/sdg1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
[my_lv_rmeta_2]   /dev/sdg1(0)
```

以下のコマンドは、3方向の RAID1 論理ボリュームを、2方向の RAID1 論理ボリュームに変換します。

```
# lvconvert -m1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
```

以下のコマンドは、3方向の RAID1 論理ボリュームを、2方向の RAID1 論理ボリュームに変換し、物理ボリューム **/dev/sde1** からイメージを削除することを指定します。

```
# lvconvert -m1 my_vg/my_lv /dev/sde1
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sdf1(1)
[my_lv_rimage_1]  /dev/sdg1(1)
[my_lv_rmeta_0]   /dev/sdf1(0)
[my_lv_rmeta_1]   /dev/sdg1(0)
```

## 10.9. RAID イメージを複数の論理ボリュームに分割

RAID 論理ボリュームのイメージを分割して新しい論理ボリュームを形成できます。

RAID イメージを分割するコマンドの形式は、以下のとおりです。

```
lvconvert --splitmirrors count -n splitname vg/lv [removable_PVs]
```

既存の RAID1 論理ボリュームから RAID イメージを削除する場合と同様に、RAID データのサブボリューム (およびその関連付けられたメタデータのサブボリューム) をデバイスから削除する場合、それより大きい番号のイメージは、そのスロットを埋めるために番号が変更になります。そのため、RAID アレイを構成する論理ボリュームのインデックス番号は連続する整数となります。



### 注記

RAID1 アレイがまだ同期していない場合は、RAID イメージを分割できません。

以下の例は、2方向の RAID1 論理ボリューム **my\_lv** を、**my\_lv** と **new** の2つのリニア論理ボリュームに分割します。

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

以下の例は、3方向の RAID1 論理ボリューム **my\_lv** を、2方向の RAID1 論理ボリューム **my\_lv** と、リニア論理ボリューム **new** に分割します。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rimage_2]  /dev/sdg1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
[my_lv_rmeta_2]   /dev/sdg1(0)
# lvconvert --splitmirror 1 -n new my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV  Copy%  Devices
my_lv  100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
new    /dev/sdg1(1)
```

## 10.10. RAID イメージの分割とマージ

**lvconvert** コマンドで、**--splitmirrors** 引数とともに **--trackchanges** 引数を使用すると、すべての変更を追跡しながら、RAID1 アレイのイメージを一時的に読み取り専用で分割できます。これにより、イメージの分割後に変更になったアレイの部分のみを再同期する一方で、後でそのイメージをアレイにマージし直すことができます。

RAID イメージを分割する **lvconvert** コマンドの形式は、以下のとおりです。

```
lvconvert --splitmirrors count --trackchanges vg/lv [removable_PVs]
```

**--trackchanges** 引数を使用して RAID イメージを分割する場合、分割するイメージを指定することはできませんが、分割されるボリューム名を変更することはできません。また、分割して作成されたボリュームには以下の制限があります。

- 作成された新規ボリュームは読み取り専用です。

- 新規ボリュームのサイズは変更できません。
- 残りのアレイの名前は変更できません。
- 残りのアレイのサイズは変更できません。
- 新規のボリュームと、残りのアレイを個別にアクティブにすることはできません。

指定した **--trackchanges** 引数を使用して分割したイメージをマージするには、**lvconvert** コマンドに **-merge** 引数を指定して実行します。イメージをマージすると、イメージが分割されてから変更したアレイの部分のみが再同期されます。

RAID イメージをマージする **lvconvert** コマンドの形式は、以下のとおりです。

### lvconvert --merge raid\_image

以下の例は、残りのアレイへの変更を追跡する一方で、RAID1 論理ボリュームを作成し、そのボリュームからイメージを分割します。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdc1(1)
[my_lv_rimage_2]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdc1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
my_lv_rimage_2 split from my_lv for read-only purposes.
Use 'lvconvert --merge my_vg/my_lv_rimage_2' to merge back into my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdc1(1)
my_lv_rimage_2    /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdc1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
```

以下の例は、残りのアレイへの変更の追跡時に、RAID1 ボリュームからイメージを分割します。その後、ボリュームをアレイにマージし直しています。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
lv_rimage_1 split from my_lv for read-only purposes.
Use 'lvconvert --merge my_vg/my_lv_rimage_1' to merge back into my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sdc1(1)
my_lv_rimage_1    /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sdc1(0)
```

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

## 10.11. RAID 障害ポリシーの設定

LVM RAID は、`lvm.conf` ファイルの `raid_fault_policy` フィールドで定義されている詳細設定に基づいて、デバイス障害を自動で処理します。

- `raid_fault_policy` フィールドが `allocate` に設定されている場合、システムは障害が発生したデバイスをボリュームグループの予備のデバイスに置き換えようとします。予備のデバイスがないと、システムログにレポートが送信されます。
- `raid_fault_policy` フィールドが `warn` に設定されている場合、システムは警告を生成して、デバイスが失敗したことがログに示されます。これにより、ユーザーは実施する一連の動作を確認できます。

残りのデバイスで該当するポリシーに対応できる限り、RAID 論理ボリュームは操作を続行します。

### 10.11.1. 「allocate」 RAID 障害ポリシー

以下の例では、`raid_fault_policy` フィールドは `lvm.conf` ファイルで `allocate` に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdf1(1)
[my_lv_rimage_2]  /dev/sdg1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdf1(0)
[my_lv_rmeta_2]   /dev/sdg1(0)
```

`/dev/sde` デバイスが失敗すると、システムログはエラーメッセージを表示します。

```
# grep lvm /var/log/messages
Jan 17 15:57:18 bp-01 lvm[8599]: Device #0 of raid1 array, my_vg-my_lv, has failed.
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994294784: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994376704: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at 0:
Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
4096: Input/output error
Jan 17 15:57:19 bp-01 lvm[8599]: Couldn't find device with uuid
```

```
3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
Jan 17 15:57:27 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is not in-sync.
Jan 17 15:57:36 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is now in-sync.
```

**raid\_fault\_policy** フィールドが **allocate** に設定されているため、障害が発生したデバイスは、ボリュームグループの新しいデバイスに置き換わります。

```
# lvs -a -o name,copy_percent,devices vg
Couldn't find device with uuid 3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
LV          Copy%  Devices
lv          100.00 lv_rimage_0(0),lv_rimage_1(0),lv_rimage_2(0)
[lv_rimage_0] /dev/sdh1(1)
[lv_rimage_1] /dev/sdf1(1)
[lv_rimage_2] /dev/sdg1(1)
[lv_rmeta_0]  /dev/sdh1(0)
[lv_rmeta_1]  /dev/sdf1(0)
[lv_rmeta_2]  /dev/sdg1(0)
```

障害が発生したデバイスを交換しても、LVM は、障害が発生したデバイスが見つけれないと示すことに注意してください。これは、障害が発生したデバイスが、RAID 論理ボリュームからは削除されても、ボリュームグループからは削除されていないためです。**vgreduce --removemissing VG** を実行すると、障害が発生したデバイスをボリュームグループから削除できます。

**raid\_fault\_policy** を **allocate** に設定したにもかかわらず、予備のデバイスがない場合は、割り当てが失敗し、論理ボリュームがそのままの状態になります。割り当てに失敗すると、ドライブが修復され、**lvchange** コマンドの **--refresh** オプションで、障害が発生したデバイスの復元を開始できます。もしくは、障害が発生したデバイスを交換することもできます。

### 10.11.2. 「warn」 RAID 障害ポリシー

以下の例では、**raid\_fault\_policy** フィールドは、**lvm.conf** ファイルで **warn** に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sdh1(1)
[my_lv_rimage_1] /dev/sdf1(1)
[my_lv_rimage_2] /dev/sdg1(1)
[my_lv_rmeta_0]  /dev/sdh1(0)
[my_lv_rmeta_1]  /dev/sdf1(0)
[my_lv_rmeta_2]  /dev/sdg1(0)
```

**/dev/sdh** デバイ스에 障害が発生すると、システムログはエラーメッセージを表示します。ただし、この場合、LVM はイメージの1つを置き換えて、RAID デバイスを自動的に修復しようとはしません。したがって、デバイスに障害が発生したら、**lvconvert** コマンドの **--repair** 引数を使用してデバイスを置き換えることができます。

## 10.12. 論理ボリュームで RAID デバイスの交換

論理ボリュームの RAID デバイスは、**lvconvert** コマンドで置き換えることができます。

- RAID デバイ스에 障害が何も発生しなかった場合は、**lvconvert** コマンドの **--replace** 引数を使用して、デバイスを置き換えます。

- RAID デバイ스에 장애가 발생한 경우는、**lvconvert** 명령의 **--repair** 引数を使用して、 장애가 발생한 디바이스를 교환합니다.

### 10.12.1. 장애 없는 RAID 디바이스의 교환

論理ボリューム의 RAID 디바이스를 교환するには、**lvconvert** 명령의 **--replace** 引数を使用します。このコマンドは、RAID 디바이스가失敗しないと有効ではありません。

**lvconvert --replace** 명령의書式は、以下のとおりです。

```
lvconvert --replace dev_to_remove vg/lv [possible_replacements]
```

以下の例では、RAID1 論理ボリュームを作成した後に、そのボリューム内の 디바이스를交換しています。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdb2(1)
[my_lv_rimage_2]  /dev/sdc1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdb2(0)
[my_lv_rmeta_2]   /dev/sdc1(0)
# lvconvert --replace /dev/sdb2 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       37.50 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sdb1(1)
[my_lv_rimage_1]  /dev/sdc2(1)
[my_lv_rimage_2]  /dev/sdc1(1)
[my_lv_rmeta_0]   /dev/sdb1(0)
[my_lv_rmeta_1]   /dev/sdc2(0)
[my_lv_rmeta_2]   /dev/sdc1(0)
```

以下の例では、RAID1 論理ボリュームを作成した後に、そのボリュームの 디바이스를交換し、交換した 디바이스에使用する物理ボリュームを指定しています。

```
# lvcreate --type raid1 -m 1 -L 100 -n my_lv my_vg
Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv       100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdb1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdb1(0)
# pvs
PV          VG      Fmt Attr PSize  PFree
/dev/sda1  my_vg   lvm2 a-- 1020.00m 916.00m
/dev/sdb1  my_vg   lvm2 a-- 1020.00m 916.00m
/dev/sdc1  my_vg   lvm2 a-- 1020.00m 1020.00m
/dev/sdd1  my_vg   lvm2 a-- 1020.00m 1020.00m
```

```
# lvconvert --replace /dev/sdb1 my_vg/my_lv /dev/sdd1
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv      28.00  my_lv_rimage_0(0),my_lv_rimage_1(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdd1(0)
```

一度に2つ以上のRAIDデバイスを交換するには、以下の例のように複数の **replace** 引数を指定します。

```
# lvcreate --type raid1 -m 2 -L 100 -n my_lv my_vg
Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv      100.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdb1(1)
[my_lv_rimage_2]  /dev/sdc1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdb1(0)
[my_lv_rmeta_2]   /dev/sdc1(0)
# lvconvert --replace /dev/sdb1 --replace /dev/sdc1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
LV          Copy%  Devices
my_lv      60.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sda1(1)
[my_lv_rimage_1]  /dev/sdd1(1)
[my_lv_rimage_2]  /dev/sde1(1)
[my_lv_rmeta_0]   /dev/sda1(0)
[my_lv_rmeta_1]   /dev/sdd1(0)
[my_lv_rmeta_2]   /dev/sde1(0)
```

### 10.12.2. 論理ボリュームに障害が発生した RAID デバイスの交換

RAID は従来の LVM ミラーリングとは異なります。LVM ミラーリングでは、障害が発生したデバイスを削除する必要がありました。削除しないと、ミラー化論理ボリュームがハングします。RAID アレイは、障害があるデバイスがあっても稼働し続けることができます。RAID1 以外の RAID タイプでデバイスを削除すると、レベルが低い RAID に変換されます (たとえば、RAID6 から RAID5、もしくは RAID4 または RAID5 から RAID0)。そのため、障害のあるデバイスを無条件に削除してから交換するのではなく、**lvconvert** コマンドで **--repair** 引数を使用して、RAID ボリュームのデバイスを1回で置き換えることができます。

以下の例では、RAID 論理ボリュームが次のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
LV          Cpy%Sync  Devices
my_lv      100.00  my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0]  /dev/sde1(1)
[my_lv_rimage_1]  /dev/sdc1(1)
[my_lv_rimage_2]  /dev/sdd1(1)
[my_lv_rmeta_0]   /dev/sde1(0)
[my_lv_rmeta_1]   /dev/sdc1(0)
[my_lv_rmeta_2]   /dev/sdd1(0)
```

/dev/sdc デバイ스에 장애가 발생した場合、**lvs** コマンドの出力は以下のようになります。

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

次のコマンドを実行して、障害が発生したコマンドを交換して、論理ボリュームを表示します。

```
# lvconvert --repair my_vg/my_lv
/dev/sdc: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
Faulty devices in my_vg/my_lv successfully replaced.
# lvs -a -o name,copy_percent,devices my_vg
/dev/sdc: open failed: No such device or address
/dev/sdc1: open failed: No such device or address
Couldn't find device with uuid A4kRI2-vlZA-uyCb-cci7-bOod-H5tX-lzH4Ee.
LV          Cpy%Sync Devices
my_lv       43.79 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
[my_lv_rimage_0] /dev/sde1(1)
[my_lv_rimage_1] /dev/sdb1(1)
[my_lv_rimage_2] /dev/sdd1(1)
[my_lv_rmeta_0] /dev/sde1(0)
[my_lv_rmeta_1] /dev/sdb1(0)
[my_lv_rmeta_2] /dev/sdd1(0)
```

障害が発生したデバイスを交換しても、LVM は、障害が発生したデバイスが見つけれないと示すことに注意してください。これは、障害が発生したデバイスが、RAID 論理ボリュームからは削除されても、ボリュームグループからは削除されていないためです。**vgreduce --removemissing VG** を実行すると、障害が発生したデバイスをボリュームグループから削除できます。

デバイス障害が一時的であったり、障害が発生したデバイスの修復が可能な場合は、**lvchange** コマンドの **--refresh** オプションを使用して、障害が発生したデバイスの復旧を開始できます。

次のコマンドは、論理ボリュームを更新します。

```
# lvchange --refresh my_vg/my_lv
```

## 10.13. RAID 論理ボリュームでのデータ整合性の確認 (RAID スクラビング)

LVM は、RAID 論理ボリュームのスクラビングに対応します。RAID スクラビングは、アレイ内のデータおよびパリティブロックをすべて読み込み、それが一貫しているかどうかを確認するプロセスです。

**lvchange** コマンドの **--syncaction** オプションを使用して、RAID スクラビングの操作を開始します。**check** または **repair** のいずれかの操作を指定します。**check** 操作はアレイ全体を対象に、アレイ内の不一致の数を記録しますが、不一致を修復することはありません。**repair** 操作は、不一致を発見した時点で修正を行います。

RAID 論理ボリュームのスクラビングを実行するコマンドの形式は以下のとおりです。

```
lvchange --syncaction {check|repair} vg/raid_lv
```



### 注記

**lvchange --syncaction repair vg/raid\_lv** 操作は、**lvconvert --repair vg/raid\_lv** と同じ機能を実行しません。**lvchange --syncaction repair** は、アレイでバックグラウンドの同期操作を開始しますが、**lvconvert --repair** 操作は、ミラーまたは RAID 論理ボリュームで障害が発生したデバイスを修復または交換します。

RAID スクラビング操作をサポートするため、**lvs** コマンドは、**raid\_sync\_action** と **raid\_mismatch\_count** の 2 つの新しい出力可能なフィールドに対応します。これらのフィールドは、デフォルトでは出力されません。これらのフィールドを表示するには、以下のように、**lvs** の **-o** パラメーターを使用して指定します。

```
lvs -o +raid_sync_action,raid_mismatch_count vg/lv
```

**raid\_sync\_action** フィールドは、raid ボリュームが現在実行している同期操作を表示します。これには、以下のいずれかの値を使用できます。

- **idle** - すべての同期操作が完了している (何も実行していません)。
- **resync** - アレイを初期化、またはマシン障害後の復旧を実行する。
- **recover** - アレイ内のデバイスを置き換える。
- **check** - アレイの不一致を検索する。
- **repair** - 不一致を検索し、修復する。

**raid\_mismatch\_count** フィールドは、**check** 操作時に検出された不一致の数を表示します。

**lvs** コマンドの **Cpy%Sync** フィールドは、**check** および **repair** を含む **raid\_sync\_action** 操作のいずれかの進捗を出力するようになりました。

**lvs** コマンドの **lv\_attr** フィールドは、RAID スクラビング操作に対応する追加のインジケータを提供するようになりました。このフィールドのビット 9 は、論理ボリュームの正常性を示し、以下のインジケータに対応するようになりました。

- 「(m)ismatches (不一致)」は、RAID 論理ボリュームに不一致があることを示します。この文字は、スクラビング操作で RAID に一貫性がない部分があることを検出した後に表示されません。

- 「(r)efresh (更新)」は、LVM がデバイスラベルを読み取り、デバイスを稼働できると認識した場合でも、RAID アレイのデバイスに障害が発生し、カーネルがこれを障害と認識していることを示します。この論理ボリュームを、デバイスが利用可能になったことをカーネルに通知するために更新する (refresh) か、もしくはデバイスに障害が発生したと思われる場合はデバイスを置き換える (replace) 必要があります。

RAID スクラビング操作を実行する際に、**sync** 操作で必要になるバックグラウンド I/O は、その他の I/O 操作 (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。復旧スロットルを実装して RAID 論理ボリュームのスクラビングを実行する速度を制御できます。

**sync** 操作の実行される速度は、**lvchange** コマンドの **--minrecoveryrate** オプションおよび **--maxrecoveryrate** オプションを使用して、それらの操作の最小および最大 I/O 速度を設定することにより制御できます。オプションは以下のように指定します。

- **--maxrecoveryrate Rate[bBsSkMgG]**  
RAID 論理ボリュームの最大復旧速度を設定し、通常の I/O 操作が押し出されないようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。サフィックスを指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。復旧速度を 0 に設定すると無制限になります。
- **--minrecoveryrate Rate[bBsSkMgG]**  
RAID 論理ボリュームの最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。サフィックスを指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。

## 10.14. RAID レベルの変更 (RAID テイクオーバー)

LVM は、Raid テイクオーバーをサポートします。これは、RAID 論理ボリュームの RAID レベルを別のレベル (たとえば RAID 5 から RAID 6) へ変えることを意味します。RAID レベルの変更は、通常、デバイスの耐障害性を増減したり、論理ボリュームのストライプ化をやり直すために行われます。RAID テイクオーバーには **lvconvert** を使用します。RAID テイクオーバーの詳細と、**lvconvert** を使用して RAID 論理ボリュームを変換する例は、man ページの **lvraid(7)** を参照してください。

## 10.15. RAID ボリュームの属性の変更 (RAID 再成形)

RAID 再成形とは、同じ RAID レベルを維持しつつ、RAID 論理ボリュームの属性を変更することを指します。変更できる属性には、RAID レイアウト、ストライプのサイズ、ストライプの数などがあります。RAID 再成形と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再生成する例は、man ページの **lvraid(7)** を参照してください。

## 10.16. RAID1 論理ボリュームでの I/O 操作の制御

**lvchange** コマンドの **--writemostly** パラメーターおよび **--writebehind** パラメーターを使用して、RAID1 論理ボリュームのデバイスに対する I/O 操作を制御できます。これらのパラメーターを使用する書式は以下のとおりです。

- **--[raid]writemostly PhysicalVolume[:{t|y|n}]**  
RAID1 論理ボリュームのデバイスを **write-mostly** とマークします。これらのドライブのすべての読み取りは、必要でない限り回避されます。このパラメーターを設定することにより、ドライブに対する I/O 操作の回数を最小限に抑えることができます。デフォルトでは、論理ボリュームに指定した物理ボリュームの **write-mostly** 属性を **yes** に設定します。**:n** を物理ボ

リユームに追加して **write-mostly** フラグを削除したり、**:t** を指定して値を切り替えたりできます。 **--writemostly** 引数は、1つのコマンドで複数回指定できるため、1回で論理ボリュームのすべての物理ボリュームで、write-mostly 属性を切り替えることができます。

- **--[raid]writebehind IOCount**

**write-mostly** というマークが付いている RAID1 論理ボリュームのデバイスに許可される、未処理の書き込みの最大数を指定します。この値を上回ると書き込みは同期され、構成要素になっているデバイスへの書き込みがすべて、アレイが書き込みの完了を知らせる前に完了してしまいます。この値をゼロに設定すると、設定はクリアになり、システムが値を任意に選択できるようになります。

## 10.17. RAID 論理ボリュームのリージョンサイズの変更

RAID 論理ボリュームを作成すると、論理ボリュームのリージョンサイズは、`/etc/lvm/lvm.conf` ファイルの **raid\_region\_size** パラメーターの値になります。このデフォルト値は、**lvcreate** コマンドの **-R** オプションで上書きできます。

RAID 論理ボリュームを作成したら、**lvconvert** コマンドの **-R** オプションで、ボリュームのリージョンサイズを変更できます。以下の例では、論理ボリューム **vg/raidlv** のリージョンサイズを 4096K に変更します。リージョンサイズを変更する場合は、RAID ボリュームを同期する必要があります。

```
# lvconvert -R 4096K vg/raid1
```

```
Do you really want to change the region_size 512.00 KiB of LV vg/raid1 to 4.00 MiB? [y/n]: y
Changed region size on RAID LV vg/raid1 to 4.00 MiB.
```

## 第11章 スナップショット論理ボリューム

LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。

### 11.1. スナップショットボリューム

LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。スナップショットの取得後に作成元のデバイスに変更が加えられると、データが変更する前に、これから変更する部分のコピーがスナップショット機能により作成されるため、このコピーを使用して、デバイスの状態を再構築できます。



#### 注記

LVM は、シンプロビジョニングのスナップショットに対応します

スナップショットは、スナップショットの作成後に変更したデータ部分のみをコピーするため、スナップショット機能に必要なストレージは最小限になります。たとえば、コピー元がほとんど更新されない場合は、作成元の3~5%の容量があれば十分にスナップショットを維持できます。



#### 注記

ファイルシステムのスナップショットコピーは仮想コピーであり、ファイルシステムのメディアバックアップを実際に作成するわけではありません。スナップショットは、バックアップの代替手順にはなりません。

作成元のボリュームへの変更を保管するために確保する領域は、スナップショットのサイズによって異なります。たとえば、スナップショットを作成してから作成元を完全に上書きした場合に、その変更を保管するのに必要なスナップショットのサイズは、作成元のボリュームと同じか、それ以上になります。スナップショットのサイズは、予想される変更レベルに応じて決定する必要があります。たとえば、`/usr` など、その大部分が読み取り用に使用されるボリュームの短期的なスナップショットに必要な領域は、`/home` のように大量の書き込みが行われるボリュームの長期的なスナップショットに必要な領域よりも小さくなります。

スナップショットが満杯になると、作成元のボリュームの変更を追跡できなくなるため、そのスナップショットは無効になります。スナップショットのサイズは定期的に監視する必要があります。ただし、スナップショットのサイズは完全に変更することが可能なため、ストレージに余裕があれば、スナップショットが停止しないように、スナップショットのボリュームサイズを拡大できます。逆に、スナップショットのボリュームサイズが必要以上に大きければ、そのボリュームのサイズを縮小して、他の論理ボリュームで必要となる領域を確保できます。

スナップショットのファイルシステムを作成しても、作成元への完全な読み取り/書き込みのアクセスは引き続き可能です。スナップショット上のチャンクを変更した場合は、そのチャンクにマークが付けられ、そこには、作成元のボリュームのコピーは入りません。

スナップショット機能にはいくつかの用途があります。

- 最も一般的な用途は、継続的にデータを更新している稼働中のシステムを停止せずに、論理ボリューム上でバックアップを実行する必要がある場合にスナップショットを撮ることです。
- スナップショットのファイルシステムで `fsck` コマンドを実行してファイルシステムの整合性を確認し、作成元のファイルシステムを修復する必要があるかどうかを判断できます。

- スナップショットは読み取りおよび書き込み用であるため、スナップショットを撮ってそのスナップショットにテストを実行することにより、実際のデータに触れることなく、実稼働データにアプリケーションのテストを実行できます。
- LVM ボリュームを作成して、Red Hat の仮想化と併用することが可能です。LVM スナップショットを使用して、仮想ゲストイメージのスナップショットを作成できます。このスナップショットは、最小限のストレージを使用して、既存のゲストの変更や新規ゲストの作成を行う上で利便性の高い方法を提供します。

**lvconvert** コマンドの **--merge** オプションを使用して、スナップショットを作成元のボリュームにマージできます。この機能の用途の1つがシステムロールバックの実行で、データやファイルを紛失した場合や、システムを以前の状態に復元する必要がある場合に行います。スナップショットボリュームのマージ後の論理ボリュームには、作成元のボリューム名、マイナー番号、UUID が付けられ、マージされたスナップショットは削除されます。

## 11.2. スナップショットボリュームの作成

スナップショットボリュームを作成するには、**lvcreate** コマンドで **-s** 引数を使用します。スナップショットボリュームは書き込み可能です。



### 注記

LVM スナップショットは、クラスターのノード間では対応していません。共有ボリュームグループ内にスナップショットボリュームは作成できません。ただし、共有論理ボリューム上でデータの一貫したバックアップ作成が必要な場合は、ボリュームを排他的にアクティブにした上で、スナップショットを作成できます。



### 注記

RAID 論理ボリュームを対象としたスナップショットに対応しています。

LVM では、作成元のボリュームのサイズよりも大きく、そのボリュームのメタデータを必要とするスナップショットボリュームを作成できません。これよりも大きなスナップショットボリュームを指定しても、システムには、作成元のサイズに必要な大きさのスナップショットボリュームのみが作成されます。

デフォルトで、スナップショットボリュームは、通常のアクティブ化コマンドの実行時に省略されます。

次の手順は、**origin** という名前の論理ボリュームを作成し、その論理ボリュームから、**snap** という名前のスナップショットボリュームを作成します。

1. ボリュームグループ **VG** から、論理ボリューム **origin** を作成します。

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

2. 名前が **snap** で、サイズが 100 MB のスナップショット論理ボリューム **/dev/VG/origin** を作成します。元の論理ボリュームにファイルシステムが含まれている場合は、任意のディレクトリー上でスナップショット論理ボリュームをマウントしてから、そのファイルシステムのコンテンツにアクセスし、元のファイルシステムが更新を継続している間にバックアップを実行できます。

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

3. 論理ボリューム `/dev/VG/origin` のステータスを表示します。ここでは、スナップショットのすべての論理ボリュームとそのステータス (アクティブかどうか) を表示します。

```
# lvsdisplay /dev/VG/origin
--- Logical volume ---
LV Path          /dev/VG/origin
LV Name          origin
VG Name          VG
LV UUID          EsFoBp-CB9H-Epl5-pUO4-Yevi-EdFS-xtFnaF
LV Write Access  read/write
LV Creation host, time host-083.virt.lab.msp.redhat.com, 2019-04-11 14:45:06 -0500
LV snapshot status source of
                  snap [active]
LV Status        available
# open           0
LV Size          1.00 GiB
Current LE       256
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:6
```

4. デフォルトでは `lvs` コマンドは、作成元のボリュームと、使用されているスナップショットボリュームの現在の割合を表示します。以下の例は、スナップショットボリューム作成後の `lvs` コマンドのデフォルト出力を示しています。これには、論理ボリュームを構成するデバイスを含む表示が含まれます。

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



### 警告

作成元ボリュームが変更すると、スナップショットのサイズが増えるため、`lvs` コマンドを使用して、スナップショットボリュームのパーセンテージを定期的に監視して、満杯にならないように確認することが重要です。スナップショットは、100% になると完全に消失します。これは、作成元ボリュームの変更されていない部分に書き込みが行われるため、スナップショットが必ず破損するためです。

スナップショットが満杯になると、スナップショット自体が無効になるだけでなく、そのスナップショットデバイスにマウントされているすべてのファイルシステムのマウントが強制的に解除されます。これにより、マウントポイントへのアクセス時に必ず発生するファイルシステムエラーを回避できます。さらに、`lvm.conf` ファイルに `snapshot_autoextend_threshold` オプションを指定できます。

このオプションにより、スナップショットの残りの領域が、設定されたしきい値を下回ると、常にスナップショットを自動的に拡張できるようになりました。この機能の利用に際しては、ボリュームグループに未割り当ての領域があることが条件になります

LVM では、作成元のボリュームのサイズよりも大きく、そのボリュームのメタデータを必要とするスナップショットボリュームを作成できません。同様に、スナップショットの自動拡張を実行しても、スナップショットに必要なサイズとして計算される最大サイズを超えて拡張されることはありません。スナップショットのサイズが作成元のボリュームを包含できるまで拡大されると、スナップショットの自動拡張は監視されなくなります。

**snapshot\_autoextend\_threshold** および **snapshot\_autoextend\_percent** の設定の詳細は、`/etc/lvm/lvm.conf` ファイルを参照してください。

### 11.3. スナップショットボリュームのマージ

**lvconvert** コマンドの **--merge** オプションを使用して、スナップショットを作成元のボリュームにマージできます。作成元とスナップショットボリュームの両方が閉じている状態になると、マージはただちに開始します。そうでない場合は、作成元またはスナップショットのいずれかがアクティブになり、かつ両方が閉じられている状態に初めてなったときにマージが開始します。root ファイルシステムのように、閉じることができない作成元へのスナップショットのマージは、次に作成元ボリュームがアクティブになるまで行われません。マージが開始すると、マージ後の論理ボリュームには、作成元の名前、マイナー番号、UUID が入ります。マージの進行中、作成元に対する読み取りまたは書き込みは、マージ中のスナップショットに対して実行されているかのように見えます。マージが完了すると、マージされたスナップショットは削除されます。

以下のコマンドは、スナップショットボリューム **vg00/lvol1\_snap** をその作成元にマージします。

```
# lvconvert --merge vg00/lvol1_snap
```

コマンドライン上で複数のスナップショットを指定したり、LVM オブジェクトタグを使用して複数のスナップショットをそれぞれの作成元にマージしたりすることが可能です。以下の例では、論理ボリューム **vg00/lvol1**、**vg00/lvol2**、および **vg00/lvol3** にはすべて **@some\_tag** タグが付きます。以下のコマンドは、この3つのボリュームのスナップショット論理ボリュームを連続的にマージします。マージは **vg00/lvol1**、**vg00/lvol2**、**vg00/lvol3** の順で行われます。**--background** オプションを使用している場合は、すべてのスナップショット論理ボリュームのマージが並行して開始します。

```
# lvconvert --merge @some_tag
```

**lvconvert --merge** コマンドの詳細は、man ページの **lvconvert(8)** を参照してください。

## 第12章 シンプロビジョニングされた論理ボリューム (シンボリューム) の作成および管理

論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステントよりも大きな論理ボリュームを作成できます。

### 12.1. シンプロビジョニングされた論理ボリューム (シンボリューム)

論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステントよりも大きな論理ボリュームを作成できます。シンプロビジョニングを使用すると、シンプールと呼ばれる、空き領域のストレージプールを管理でき、アプリケーションで必要になった時に任意の数のデバイスに割り当てることができます。後でアプリケーションが実際に論理ボリュームに書き込むときに割り当てられるように、シンプールにバインドするデバイスを作成できます。シンプールは、ストレージ領域をコスト効率よく割り当てることが必要がある場合に、動的に拡張できます。



#### 注記

クラスタのノード間では、シンボリュームに対応していません。シンプールとそのすべてのシンボリュームは、1つのクラスタノードで排他的にアクティブにする必要があります。

ストレージ管理者は、シンプロビジョニングを使用することで物理ストレージをオーバーコミットできるため、多くの場合は、追加のストレージを購入する必要がなくなります。たとえば、10人のユーザーから、各自のアプリケーションに使用するファイルシステムをそれぞれ100GB要求された場合、ストレージ管理者は各ユーザーに100GBのファイルシステムを作成します(ただし、実際には100GB未満のストレージが、必要に応じて使用されます)。シンプロビジョニングを使用する場合は、ストレージ管理者がストレージプールを監視し、容量が満杯になり始めたら容量を追加することが重要です。

利用可能な領域をすべて使用できるようにするために、LVMはデータの破棄に対応します。これにより、破棄されたファイルや、その他のブロック範囲で以前に使用された領域を再利用できます。

シンボリュームは、新たに実装されたコピーオンライト (COW) スナップショット論理ボリュームに対応します。これにより、多くの仮想デバイスでシンプール内の同一データを共有できます。

### 12.2. シンプロビジョニングされた論理ボリュームの作成

この手順では、シンプロビジョニングされた論理ボリュームを作成し、拡張するために使用する基本的なコマンドの概要を説明します。LVMシンプロビジョニングの詳細情報と、シンプロビジョニングされた論理ボリュームと共にLVMコマンドおよびユーティリティを使用する方法は、manページの `lvmin(7)` を参照してください。

シンボリュームを作成する場合は、以下のタスクを実行します。

1. `vgcreate` コマンドを使用して、ボリュームグループを作成する。
2. `lvcreate` コマンドを使用して、シンプールを作成する。
3. `lvcreate` コマンドを使用して、シンプール内にシンボリュームを作成する。

`lvcreate` コマンドに `-T` (または `--thin`) オプションを付けて、シンプールまたはシンボリュームを作成できます。また、`lvcreate` の `-T` オプションを使用して、1つのコマンドで同時にプール内にシンプールとシンプロビジョニングされたボリュームの両方を作成することもできます。

以下のコマンドは、`lvcreate` コマンドの `-T` オプションを使用して、`mythinpool` という名前のシンプー

ルを作成します。これは、ボリュームグループ **vg001** 内にあり、サイズは 100M です。物理領域のプールを作成しているため、プールのサイズを指定する必要があります。**lvcreate** コマンドの **-T** オプションは引数を取りません。コマンドで指定するその他のオプションから、作成されるデバイスのタイプが推定されます。

```
# lvcreate -L 100M -T vg001/mythinpool
```

```
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
```

```
Logical volume "mythinpool" created.
```

```
# lvs
```

```
LV      VG   Attr   LSize Pool Origin Data% Meta% Move Log Cpy% Sync Convert
mythinpool vg001 twi-a-tz-- 100.00m          0.00 10.84
```

以下のコマンドは、**lvcreate** コマンドに **-T** オプションを使用して、シンプル **vg001/mythinpool** に **thinvolume** という名前のシンボリュームを作成します。ここでは、ボリュームを含むプールよりも大きなボリュームの仮想サイズを指定している点に注意してください。

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
```

```
WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool vg001/mythinpool (100.00 MiB).
```

```
WARNING: You have not turned on protection against thin pools running out of space.
```

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

```
Logical volume "thinvolume" created.
```

```
# lvs
```

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

以下のコマンドは、**lvcreate** コマンドに **-T** オプションを使用して、プール内にシンプルとシンプロビジョニングされたボリュームを作成します。その際、**lvcreate** コマンドでサイズと仮想サイズの引数を指定します。また、このコマンドは、ボリュームグループ **vg001** にシンプル **mythinpool** を作成し、そのプールにシンプロビジョニングされたボリューム **thinvolume** も作成します。

```
# lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
```

```
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
```

```
WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool vg001/mythinpool (100.00 MiB).
```

```
WARNING: You have not turned on protection against thin pools running out of space.
```

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

```
Logical volume "thinvolume" created.
```

```
# lvs
```

```
LV      VG   Attr   LSize Pool Origin Data% Meta% Move Log Cpy% Sync Convert
mythinpool vg001 twi-aotz-- 100.00m          0.00 10.94
thinvolume vg001 Vwi-a-tz-- 1.00g mythinpool 0.00
```

また、**lvcreate** コマンドの **--thinpool** パラメーターを指定して、シンプルを作成することもできます。**-T** オプションとは異なり、**--thinpool** パラメーターには、作成しているシンプルの論理ボリューム名の引数が必要です。以下の例は、**lvcreate** で **--thinpool** パラメーターを指定して、**mythinpool** という名前のシンプルを作成します。これは、ボリュームグループ **vg001** にあり、サイズは 100M です。

```
# lvcreate -L 100M --thinpool mythinpool vg001
```

```
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
```

```
Logical volume "mythinpool" created.
```

**# lvs**

```
LV      VG   Attr   LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
mythinpool vg001 twi-a-tz-- 100.00m      0.00 10.84
```

チャンクサイズを使用するには、以下の基準を使用します。

- チャンクサイズが小さいほどメタデータが増え、パフォーマンスも低下しますが、スナップショットで領域の使用率が向上します。
- チャンクサイズが大きいほどメタデータ操作は少なくなります、スナップショットの領域効率が低下します。

デフォルトでは、**lvm2** は 64KiB のチャンクサイズで始まり、シンプルメタデータデバイスのサイズが 128MiB を超えると、その値を増やします。これにより、メタデータサイズが圧縮されます。ただし、これにチャンクサイズの値が大きくなる場合があり、スナップショットを使用するための領域の効率が低下します。このような場合は、チャンクのサイズが小さく、メタデータサイズが大きくなる方が適しています。

ボリュームデータサイズが TiB の範囲にある場合は、サポートされる最大サイズであるメタデータサイズとして ~15.8GiB を使用し、要件に応じてチャンクサイズを設定します。ただし、ボリュームのデータサイズを拡張し、チャンクサイズを小さくする必要がある場合には、メタデータサイズを拡大できないことに注意してください。

**警告**

Red Hat は、デフォルト値よりも小さいチャンクサイズを設定することは推奨していません。チャンクサイズが小さすぎる場合で、ボリュームのメタデータの容量が不足すると、ボリュームはデータを作成できません。論理ボリュームを監視して、拡張またはメタデータボリュームが完全に満杯になる前にストレージをさらに作成します。メタデータ用の領域が不足しないように、シンプルのチャンクサイズは十分な大きさになるようにしてください。

プールの作成で、ストライピングに対応しています。以下のコマンドは、2つの 64 kB のストライプがあり、チャンクサイズが 256 kB のボリュームグループ **vg001** に、**pool** という名前の 100M のシンプールを作成します。また、1T のシンボリューム **vg00/thin\_lv** も作成します。

```
# lvcreate -i 2 -l 64 -c 256 -L 100M -T vg00/pool -V 1T --name thin_lv
```

**lvextend** コマンドを使用して、シンボリュームのサイズを拡張できます。ただし、シンプルのサイズを縮小することはできません。

以下のコマンドは、既存のシンプルのサイズ (100M) を変更し、100M 拡張します。

```
# lvextend -L+100M vg001/mythinpool
Extending logical volume mythinpool to 200.00 MiB
Logical volume mythinpool successfully resized
# lvs
LV      VG   Attr   LSize  Pool  Origin Data%  Meta%  Move Log Copy% Convert
mythinpool vg001 twi-a-tz 200.00m      0.00
thinvolume vg001 Vwi-a-tz 1.00g mythinpool 0.00
```

他の論理ボリュームのタイプと同様に、**lvrename** を使用してボリューム名を変更したり、**lvremove** を使用してボリュームを削除したりできます。また、**lvs** コマンドおよび **lvdisplay** コマンドを使用して、ボリュームの情報を表示することもできます。

デフォルトでは、**lvcreate** コマンドは、計算式 ( $\text{Pool\_LV\_size} / \text{Pool\_LV\_chunk\_size} * 64$ ) から、シンプルのメタデータ論理ボリュームのサイズを設定します。スナップショットが大量にある場合や、シンプルのサイズが小さく、後で急激に大きくなることが予測される場合は、**lvcreate** コマンドの **--poolmetadatasize** パラメーターで、シンプルのメタデータボリュームのデフォルト値を大きくしないといけない場合があります。シンプルのメタデータ論理ボリュームで対応している値は 2MiB ~ 16GiB です。

**lvconvert** コマンドの **--thinpool** パラメーターを使用して、既存の論理ボリュームをシンプルボリュームに変換できます。既存の論理ボリュームをシンプルボリュームに変換する場合は、**lvconvert** コマンドの **--thinpool** パラメーターとともに **--poolmetadata** パラメーターを使用して、既存の論理ボリュームをシンプルボリュームのメタデータボリュームに変換する必要があります。



### 注記

論理ボリュームをシンプルボリュームまたはシンプルメタデータボリュームに変換すると、論理ボリュームのコンテンツが破棄されます。なぜなら、**lvconvert** は、デバイスのコンテンツを保存するのではなく、コンテンツを上書きするためです。

以下の例は、ボリュームグループ **vg001** の既存の論理ボリューム **lv1** を、シンプルボリュームに変換します。また、ボリュームグループ **vg001** の既存の論理ボリューム **lv2** を、そのシンプルボリュームのメタデータボリュームに変換します。

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

## 12.3. シンプロビジョニングのスナップショットボリューム

Red Hat Enterprise Linux は、シンプロビジョニングのスナップショットボリュームのサポートを提供します。シンプロビジョニングのスナップショットボリュームにより、多くの仮想デバイスを同じデータボリュームに格納できます。これにより管理が簡略化され、スナップショットボリューム間でのデータ共有が可能になります。

シンボリュームや、LVM スナップショットボリュームの場合、シンプロビジョニングのスナップショットボリュームは、クラスターのノード間では対応していません。スナップショットボリュームは、1つのクラスターノードで排他的にアクティブにする必要があります。

シンプロビジョニングのスナップショットボリュームの利点は以下のとおりです。

- 同じボリュームからのスナップショットが複数ある場合に、シンプロビジョニングのスナップショットボリュームを使用すれば、ディスクの使用量を減らすことができます。
- 作成元が同じスナップショットが複数ある場合は、作成元に1回書き込むことにより、1回のCOW操作でデータを保存できます。作成元のスナップショットの数を増やしても、速度が大幅に低下することはありません。
- シンプロビジョニングのスナップショットボリュームは、別のスナップショットの作成元の論理ボリュームとして使用できます。これにより、再帰的スナップショット(スナップショットのスナップショットのそのまたスナップショットなど)の深度を任意に決定できます。

- シン論理ボリュームのスナップショットにより、シン論理ボリュームを作成することもできます。COW 操作が必要になるまで、あるいはスナップショット自体が書き込まれるまで、データ領域は消費されません。
- シンプロビジョニングのスナップショットボリュームは、作成元とともにアクティブにしておく必要はありません。そのため、スナップショットボリュームが多数ある場合は、作成元のみをアクティブにし、スナップショットボリュームはアクティブにしないことができます。
- シンプロビジョニングのスナップショットボリュームの作成元を削除すると、そのボリュームのスナップショットは、それぞれ独立したシンプロビジョニングボリュームになります。したがって、スナップショットとその作成元のボリュームをマージする代わりに、作成元のボリュームを削除し、その独立したボリュームを新たな作成元ボリュームにして、シンプロビジョニングのスナップショットを新たに作成できます。

シンプロビジョニングのスナップショットボリュームを使用する利点は数多くありますが、古い LVM スナップショットボリューム機能の方がニーズに適している場合もあります。

- シンプルのチャンクサイズは変更できません。シンプルのチャンクサイズが大きい場合 (1MB など) や、そのチャンクのサイズでは効率的ではない短期間のスナップショットが必要な場合は、代わりに以前のスナップショット機能を使用できます。
- シンプロビジョニングのスナップショットボリュームのサイズを制限することはできません。スナップショットは、必要な場合はシンプル内の全領域を使用します。これは、ニーズに適さない場合があります。

一般的には、使用するスナップショットの形式を決定する際に、使用しているサイトの特定要件を考慮するようにしてください。

## 12.4. シンプロビジョニングのスナップショットボリュームの作成

Red Hat Enterprise Linux は、シンプロビジョニングのスナップショットボリュームのサポートを提供します。



### 注記

このセクションでは、シンプロビジョニングのスナップショットボリュームを作成し、拡張するために使用する基本的なコマンドの概要を説明します。LVM シンプロビジョニングの詳細情報と、シンプロビジョニングされた論理ボリュームと共に LVM コマンドおよびユーティリティーを使用する方法は、man ページの `lvmtthin(7)` を参照してください。



### 重要

シンプロビジョニングのスナップショットボリュームを作成する場合、ボリュームのサイズは指定しません。サイズパラメーターを指定すると、作成されるスナップショットはシンプロビジョニングのスナップショットボリュームにはならず、データを保管するためにシンプルを使用することもあります。たとえば、`lvcreate -s vg/thinvolume -L10M` コマンドは、作成元ボリュームがシンボリュームであっても、シンプロビジョニングのスナップショットを作成しません。

シンプロビジョニングのスナップショットは、シンプロビジョニングされた作成元ボリューム用に作成するか、またはシンプロビジョニングされていない作成元ボリューム用にも作成できます。

**lvcreate** コマンドで、**--name** オプションを使用してスナップショットボリューム名を指定できます。以下のコマンドは、シンプロビジョニングされた論理ボリューム **vg001/thinvolume** で、シンプロビジョニングのスナップショットボリューム (**mynsnapshot1**) を作成します。

```
# lvcreate -s --name mynsnapshot1 vg001/thinvolume
Logical volume "mynsnapshot1" created
# lvs
LV      VG      Attr  LSize Pool   Origin  Data% Move Log Copy% Convert
mynsnapshot1 vg001  Vwi-a-tz 1.00g mythinpool thinvolume 0.00
mythinpool vg001  twi-a-tz 100.00m          0.00
thinvolume vg001  Vwi-a-tz 1.00g mythinpool          0.00
```

シンプロビジョニングのスナップショットボリュームには、他のシンボリュームと同じ特性があります。ボリュームのアクティブ化、拡張、名前変更、削除、さらにはスナップショット作成も個別に行うことができます。

デフォルトで、スナップショットボリュームは、通常のアクティブ化コマンドの実行時に省略されます。論理ボリュームのアクティブ化を制御する方法は、「[論理ボリュームのアクティブ化](#)」を参照してください。

シンプロビジョニングされていない論理ボリュームの、シンプロビジョニングされたスナップショットを作成することもできます。シンプロビジョニングされていない論理ボリュームはシンプール内に含まれていないため、**外部の作成元**と呼ばれます。外部の作成元ボリュームは、複数の異なるシンプールからであっても、多くのシンプロビジョニングのスナップショットボリュームで使用でき、共有できます。外部の作成元は、シンプロビジョニングのスナップショットが作成される際に非アクティブであり、かつ読み取り専用である必要があります。

外部の作成元のシンプロビジョニングされたスナップショットを作成する場合は、**--thinpool** オプションを指定する必要があります。以下のコマンドは、読み取り専用の非アクティブなボリューム **origin\_volume** のシンプロビジョニングのスナップショットボリュームを作成します。このシンプロビジョニングのスナップショットボリュームの名前は **mythinsnap** です。論理ボリューム **origin\_volume** は、既存のシンプール **vg001/pool** を使用する、ボリュームグループ **vg001** 内のシンプロビジョニングのスナップショットボリューム **mythinsnap** に対する外部の作成元になります。作成元ボリュームは、スナップショットボリュームと同じボリュームグループに存在するため、作成元の論理ボリュームを指定する場合にボリュームグループを指定する必要はありません。

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

以下のコマンドにあるように、最初のスナップショットボリュームの2番目のシンプロビジョニングのスナップショットボリュームを作成できます。

```
# lvcreate -s vg001/mythinsnap --name my2ndthinsnap
```

**lvs** コマンドのレポートフィールド **lv\_ancestors** および **lv\_descendants** を指定すると、シンプロビジョニングされたスナップショット論理ボリュームのすべての先祖 (ancestor) と子孫 (descendant) をそれぞれ表示できます。

以下の例では、下記の点を前提としています。

- **stack1** は、ボリュームグループ **vg001** で元となるボリュームです。
- **stack2** は、**stack1** のスナップショットです。
- **stack3** は、**stack2** のスナップショットです。
- **stack4** は、**stack3** のスナップショットです。

さらに

- **stack5** も、**stack2** のスナップショットです。
- **stack6** は、**stack5** のスナップショットです。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
LV      Ancestors          Descendants
stack1
stack2  stack1             stack3,stack4,stack5,stack6
stack3  stack2,stack1         stack4
stack4  stack3,stack2,stack1
stack5  stack2,stack1         stack6
stack6  stack5,stack2,stack1
pool
```

## 注記

**lv\_ancestors** フィールドおよび **lv\_descendants** フィールドは、既存の依存関係を表示しますが削除されたエントリは追跡しません。このチェーンの最中にエントリが削除されると、依存関係チェーンが壊れるためです。たとえば、この設定例から論理ボリューム **stack3** を削除すると、以下のように表示されます。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
LV      Ancestors          Descendants
stack1
stack2  stack1             stack5,stack6
stack4
stack5  stack2,stack1         stack6
stack6  stack5,stack2,stack1
pool
```

ただし、削除した論理ボリュームを追跡して表示するようにシステムを設定できます。 **lv\_ancestors\_full** フィールドおよび **lv\_descendants\_full** フィールドを指定することで、このボリュームを含む完全依存チェーンを表示できます。

## 12.5. 削除されているシンプロビジョニングのスナップショットボリュームの追跡および表示

**lvm.conf** 設定ファイルで **record\_lvs\_history** メタデータオプションを有効にして、削除したシンプロビジョニングのスナップショットとシン論理ボリュームを追跡するように設定します。これにより、元の依存関係チェーンから削除し、**過去** の論理ボリュームになった論理ボリュームを含む、完全にシンプロビジョニングされたスナップショットの依存関係チェーンを表示できます。

**lvm.conf** 設定ファイルで **lvs\_history\_retention\_time** メタデータオプションを使用し、保持時間 (秒) を指定して、決められた期間、システムに過去のボリュームを保持するように設定できます。

過去の論理ボリュームでは、削除された論理ボリュームを単純化したものが保持され、以下の、ボリュームのレポートフィールドが含まれます。

- **lv\_time\_removed** - 論理ボリュームの削除時間
- **lv\_time** - 論理ボリュームの作成時間

- **lv\_name** - 論理ボリュームの名前
- **lv\_uuid** - 論理ボリュームの UUID
- **vg\_name** - 論理ボリュームを含むボリュームグループ

ボリュームを削除すると、過去の論理ボリューム名には頭にハイフンが付きます。たとえば、論理ボリューム **lvvol1** を削除すると、過去のボリューム名は **-lvvol1** となります。過去の論理ボリュームは再アクティベートすることができません。

**record\_lvs\_history** メタデータオプションを有効にしても、**lvremove** コマンドの **--nohistory** オプションを指定して論理ボリュームを削除すれば、過去の論理ボリュームを個別に保持しないようにできます。

ボリューム表示に過去の論理ボリュームを含むには、LVM 表示コマンドに **-H|--history** オプションを指定します。**-H** オプションとともに、レポートフィールド **lv\_full\_ancestors** および **lv\_full\_descendants** を指定すると、過去のボリュームを含む、完全にシンプロビジョニングされたスナップショットの依存関係チェーンを表示できます。

以下のコマンド群は、過去の論理ボリュームを表示および管理する例を示します。

1. **lvm.conf** ファイルに **record\_lvs\_history=1** を設定して、過去の論理ボリュームを保持していることを確認します。このメタデータオプションは、デフォルトでは有効ではありません。
2. 以下のコマンドを入力して、シンプロビジョニングのスナップショットチェーンを表示します。  
この例では、以下のように設定されています。

- **lvvol1** は元となるボリュームで、チェーンの中で最初のボリュームになります。
- **lvvol2** は、**lvvol1** のスナップショットです。
- **lvvol3** は、**lvvol2** のスナップショットです。
- **lvvol4** は、**lvvol3** のスナップショットです。
- **lvvol5** も、**lvvol3** のスナップショットです。  
この例の **lvs** 表示コマンドに **-H** オプションを追加しても、シンプロビジョニングのスナップショットボリュームは削除されていないため、過去の論理ボリュームは表示されないことに注意してください。

```
# lvs -H -o name,full_ancestors,full_descendants
LV  FAncestors    FDescendants
lvvol1          lvvol2,lvvol3,lvvol4,lvvol5
lvvol2 lvvol1      lvvol3,lvvol4,lvvol5
lvvol3 lvvol2,lvvol1  lvvol4,lvvol5
lvvol4 lvvol3,lvvol2,lvvol1
lvvol5 lvvol3,lvvol2,lvvol1
pool
```

3. スナップショットチェーンから論理ボリューム **lvvol3** を削除してから再度 **lvs** コマンドを実行し、過去の論理ボリュームが、先祖 (ancestor) と子孫 (descendant) とともに、どのように表示されるかを確認します。

```
# lvremove -f vg/lvol3
Logical volume "lvvol3" successfully removed
```

```
# lvs -H -o name,full_ancestors,full_descendants
```

```
LV  FAncestors  FDescendants
lvol1          lvol2,-lvol3,lvol4,lvol5
lvol2 lvol1      -lvol3,lvol4,lvol5
-lvol3 lvol2,lvol1  lvol4,lvol5
lvol4 -lvol3,lvol2,lvol1
lvol5 -lvol3,lvol2,lvol1
pool
```

4. 過去のボリュームが削除された時間を表示する場合は、**lv\_time\_removed** レポートフィールドを使用できます。

```
# lvs -H -o name,full_ancestors,full_descendants,time_removed
```

```
LV  FAncestors  FDescendants  RTime
lvol1          lvol2,-lvol3,lvol4,lvol5
lvol2 lvol1      -lvol3,lvol4,lvol5
-lvol3 lvol2,lvol1  lvol4,lvol5  2016-03-14 14:14:32 +0100
lvol4 -lvol3,lvol2,lvol1
lvol5 -lvol3,lvol2,lvol1
pool
```

5. **vgname/lvname** フォーマットを以下の例のように指定すると、表示コマンドで過去の論理ボリュームを個別に参照できます。**lv\_attr** フィールドの5番目のビットを **h** に設定して、ボリュームが過去のボリュームであることを示していることに注意してください。

```
# lvs -H vg/-lvol3
```

```
LV  VG  Attr  LSize
-lvol3 vg  ---h---- 0
```

6. ボリュームにライブの子孫がないと、LVM は過去の論理ボリュームを保持しません。これは、スナップショットチェーンの最後に論理ボリュームを削除すると、論理ボリュームが過去の論理ボリュームとして保持されないことを示しています。

```
# lvremove -f vg/lvol5
```

```
Automatically removing historical logical volume vg/-lvol5.
Logical volume "lvol5" successfully removed
```

```
# lvs -H -o name,full_ancestors,full_descendants
```

```
LV  FAncestors  FDescendants
lvol1          lvol2,-lvol3,lvol4
lvol2 lvol1      -lvol3,lvol4
-lvol3 lvol2,lvol1  lvol4
lvol4 -lvol3,lvol2,lvol1
pool
```

7. 以下のコマンドを実行して、ボリューム **lvol1** および **lvol2** を削除します。次に **lvs** コマンドを実行して、ボリュームが削除されるとどのように表示されるかを確認します。

```
# lvremove -f vg/lvol1 vg/lvol2
```

```
Logical volume "lvol1" successfully removed
Logical volume "lvol2" successfully removed
```

```
# lvs -H -o name,full_ancestors,full_descendants
```

```
LV  FAncestors  FDescendants
-lvol1          -lvol2,-lvol3,lvol4
-lvol2 -lvol1      -lvol3,lvol4
```

```
-lvol3 -lvol2,-lvol1    lvol4
lvol4 -lvol3,-lvol2,-lvol1
pool
```

- 過去の論理ボリュームを完全に削除したら、**lvremove** コマンドを再度実行します。以下の例のように、ハイフンが追加された、過去のボリューム名を指定します。

```
# lvremove -f vg/-lvol3
Historical logical volume "lvol3" successfully removed
# lvs -H -o name,full_ancestors,full_descendants
LV   FAncestors  FDescendants
-lvol1          -lvol2,lvol4
-lvol2 -lvol1    lvol4
lvol4 -lvol2,-lvol1
pool
```

- 子孫にライブボリュームが含まれるチェーンがある場合に限り、過去の論理ボリュームは保持されます。これは、以下の例のように、そのボリュームにリンクされている子孫がない場合に、過去の論理ボリュームを削除すると、チェーンの論理ボリュームがすべて削除されることを意味します。

```
# lvremove -f vg/lvol4
Automatically removing historical logical volume vg/-lvol1.
Automatically removing historical logical volume vg/-lvol2.
Automatically removing historical logical volume vg/-lvol4.
Logical volume "lvol4" successfully removed
```

## 第13章 キャッシュを有効にして論理ボリュームのパフォーマンスを改善

LVM 論理ボリュームにキャッシュを追加して、パフォーマンスを向上できます。LVM は、SSD などの高速なデバイスを使用して、論理ボリュームに I/O 操作をキャッシュします。

以下の手順では、高速デバイスから特別な論理ボリュームを作成し、この特別な論理ボリュームを元の論理ボリュームに接続して、パフォーマンスを向上させます。

### 13.1. LVM でのキャッシュの取得方法

LVM は、以下のようなキャッシュの取得方法を提供します。論理ボリューム上のさまざまなタイプの I/O パターンに適しています。

#### dm-cache

このメソッドは、高速なボリュームで頻繁に使用されるデータをキャッシュして、このようなデータへのアクセス時間を短縮します。このメソッドは、読み取りおよび書き込みの両方の操作をキャッシュします。

**dm-cache** メソッドは、**cache** タイプの論理ボリュームを作成します。

#### dm-writecache

このメソッドは、書き込み操作のみをキャッシュします。高速なボリュームは書き込み操作を保存し、それらをバックグラウンドで低速なディスクに移行します。高速ボリュームは通常 SSD または永続メモリー (PMEM) ディスクです。

**dm-writecache** メソッドは、**writecache** タイプの論理ボリュームを作成します。

### 13.2. LVM キャッシュコンポーネント

論理ボリュームのキャッシュを有効にすると、LVM は元のボリュームの名前を変更および非表示にし、元の論理ボリュームで構成される新しい論理ボリュームを表示します。新しい論理ボリュームの構成は、キャッシュ方法と、**cachevol** オプションまたは **cachepool** オプションを使用しているかどうかによって異なります。

**cachevol** オプションおよび **cachepool** オプションは、キャッシングコンポーネントの配置に対するさまざまなレベルの制御を公開します。

- **cachevol** オプションを使用すると、高速なデバイスは、データブロックのキャッシュされたコピーとキャッシュ管理用のメタデータの両方を保存します。
- **cachepool** オプションを使用すると、別のデバイスはデータブロックのキャッシュコピーとキャッシュ管理用のメタデータを保存できます。  
**dm-writecache** メソッドは、**cachepool** と互換性がありません。

すべての設定において、LVM は、結果として作成される1つのデバイスを公開し、すべてのキャッシングコンポーネントをグループ化します。作成されるデバイスは、元の低速な論理ボリュームと同じ名前になります。

### 13.3. 論理ボリュームの DM-CACHE キャッシュの有効化

この手順では、**dm-cache** メソッドを使用して、論理ボリュームで一般的に使用されるデータのキャッシュを有効にします。

## 前提条件

- システムに、**dm-cache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

## 手順

1. 高速デバイスに **cachevol** ボリュームを作成します。

```
# lvcreate --size cachevol-size --name fastvol vg /dev/fast-pv
```

以下の値を置き換えます。

### **cachevol-size**

**5G** などの **cachevol** ボリュームのサイズ

### **fastvol**

**cachevol** ボリュームの名前

### **vg**

ボリュームグループ名

### **/dev/fast-pv**

高速ブロックデバイスへのパス (例: **/dev/sdf1**)

2. **cachevol** ボリュームをメインの論理ボリュームに接続して、キャッシュを開始します。

```
# lvconvert --type cache --cachevol fastvol vg/main-lv
```

以下の値を置き換えます。

### **fastvol**

**cachevol** ボリュームの名前

### **vg**

ボリュームグループ名

### **main-lv**

低速な論理ボリュームの名前

## 検証手順

- 新たに作成されたデバイスを確認します。

```
# lvs --all --options +devices vg
```

```
LV          Pool          Type  Devices
main-lv     [fastvol_cvol] cache  main-lv_corig(0)
[fastvol_cvol]          linear /dev/fast-pv
[main-lv_corig]          linear /dev/slow-pv
```

## 関連情報

- この手順と、管理例を含むその他の詳細は、man ページの **lvmmcache(7)** を参照してください。

## 13.4. 論理ボリュームに **CACHEPOOL** を使用した **DM-CACHE** キャッシュの有効化

この手順では、キャッシュデータとキャッシュメタデータ論理ボリュームを個別に作成し、ボリュームをキャッシュプールに統合することができます。

### 前提条件

- システムに、**dm-cache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

### 手順

1. 高速デバイスに **cachepool** ボリュームを作成します。

```
# lvcreate --type cache-pool --size cachepool-size --name fastpool vg /dev/fast
```

以下の値を置き換えます。

#### **cachepool-size**

**cachepool** のサイズ (例: **5G**)

#### **fastpool**

**cachepool** ボリュームの名前

#### **vg**

ボリュームグループ名

#### **/dev/fast**

高速ブロックデバイスへのパス (例: **/dev/sdf1**)



### 注記

**--poolmetadata** オプションを使用して、**cache-pool** の作成時にプールメタデータの場所を指定できます。

2. キャッシュを開始するために、メイン論理ボリュームに **cachepool** をアタッチします。

```
# lvconvert --type cache --cachepool fastpool vg/main
```

以下の値を置き換えます。

#### **fastpool**

**cachepool** ボリュームの名前

#### **vg**

ボリュームグループ名

#### **main**

低速な論理ボリュームの名前

### 検証手順

- 新たに作成されたデバイスを確認します。

```
# lvs --all --options +devices vg

LV          Pool          Type    Devices
[fastpool_cpoo]          cache-pool fastpool_pool_cdata(0)
[fastpool_cpoo_cdata]          linear   /dev/sdf1(4)
[fastpool_cpoo_cmeta]          linear   /dev/sdf1(2)
[lvol0_pmspare]          linear   /dev/sdf1(0)
main        [fastpooool_cpoo] cache    main_corig(0)
[main_corig]          linear   /dev/sdf1(0)
```

## 関連情報

- man ページの **lvcreate(8)**
- man ページの **lvmcache(7)**
- man ページの **lvconvert(8)**

## 13.5. 論理ボリュームの DM-WRITECACHE キャッシュの有効化

この手順では、**dm-writecache** メソッドを使用して、論理ボリュームへの書き込み I/O 操作のキャッシュを有効にします。

### 前提条件

- システムに、**dm-writecache** を使用した高速化したい低速な論理ボリュームがある。
- 低速な論理ボリュームを含むボリュームグループには、高速ブロックデバイスに未使用の物理ボリュームも含まれます。

### 手順

1. 低速な論理ボリュームがアクティブな場合は、非アクティブにします。

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

以下の値を置き換えます。

#### **vg**

ボリュームグループ名

#### **main-lv**

低速な論理ボリュームの名前

2. 高速なデバイス上に非アクティブな **cachevol** ボリュームを作成します。

```
# lvcreate --activate n --size cachevol-size --name fastvol vg /dev/fast-pv
```

以下の値を置き換えます。

#### **cachevol-size**

5G などの **cachevol** ボリュームのサイズ

**fastvol**

**cachevol** ボリュームの名前

**vg**

ボリュームグループ名

**/dev/fast-pv**

高速ブロックデバイスへのパス (例: **/dev/sdf1**)

3. **cachevol** ボリュームをメインの論理ボリュームに接続して、キャッシュを開始します。

```
# lvconvert --type writecache --cachevol fastvol vg/main-lv
```

以下の値を置き換えます。

**fastvol**

**cachevol** ボリュームの名前

**vg**

ボリュームグループ名

**main-lv**

低速な論理ボリュームの名前

4. 作成された論理ボリュームをアクティベートします。

```
# lvchange --activate y vg/main-lv
```

以下の値を置き換えます。

**vg**

ボリュームグループ名

**main-lv**

低速な論理ボリュームの名前

## 検証手順

- 新たに作成されたデバイスを確認します。

```
# lvs --all --options +devices vg

LV          VG Attr  LSize  Pool           Origin         Data%  Meta%  Move Log
Cpy%Sync  Convert Devices
main-lv     vg Cwi-a-C--- 500.00m [fastvol_cv] [main-lv_worig] 0.00
main-lv_worig(0)
[fastvol_cv] vg Cwi-aoC--- 252.00m
/dev/sdc1(0)
[main-lv_worig] vg owi-aoC--- 500.00m
/dev/sdb1(0)
```

## 関連情報

- 管理例を含む情報は、man ページの **lvmcache(7)** を参照してください。

## 13.6. 論理ボリュームのキャッシュの無効化

この手順では、論理ボリュームで現在有効な **dm-cache** キャッシュまたは **dm-writocache** キャッシュを無効にします。

### 前提条件

- キャッシュは、論理ボリュームで有効になります。

### 手順

1. 論理ボリュームを非アクティブにします。

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

以下の値を置き換えます。

#### **vg**

ボリュームグループ名

#### **main-lv**

キャッシュが有効な論理ボリュームの名前

2. **cachevol** ボリュームまたは **cachepool** ボリュームの割り当てを解除します。

```
# lvconvert --splitcache vg/main-lv
```

以下の値を置き換えます。

#### **vg**

ボリュームグループ名

#### **main-lv**

キャッシュが有効な論理ボリュームの名前

### 検証手順

- 論理ボリュームが接続されていないことを確認します。

```
# lvs --all --options +devices [replaceable]_vg_
```

```
LV   Attr   Type  Devices
fastvol -wi----- linear /dev/fast-pv
main-lv -wi----- linear /dev/slow-pv
```

### 関連情報

- man ページの **lvmcache(7)**

## 第14章 論理ボリュームのアクティブ化

アクティブ状態の論理ボリュームは、ブロックデバイスを介して使用できます。アクティブになっている論理ボリュームにはアクセスにでき、変更できます。論理ボリュームを作成すると、デフォルトでアクティブになります。

個々の論理ボリュームを非アクティブにしてカーネルが認識しないようにする必要がある状況はさまざまです。個々の論理ボリュームは、**lvchange** コマンドの **-a** オプションを使用してアクティブまたは非アクティブにできます。

個々の論理ボリュームを非アクティブにするには、以下のコマンドを実行します。

```
lvchange -an vg/lv
```

個々の論理ボリュームをアクティブにするには、以下のコマンドを実行します。

```
lvchange -ay vg/lv
```

**vgchange** コマンドの **-a** オプションを使用して、ボリュームグループの論理ボリュームをすべてアクティブまたは非アクティブにできます。これは、ボリュームグループの個々の論理ボリュームに **lvchange -a** コマンドを実行するのと同じです。

ボリュームグループの論理ボリュームをすべて非アクティブにするには、以下のコマンドを実行します。

```
vgchange -an vg
```

ボリュームグループの論理ボリュームをすべてアクティブにするには、以下のコマンドを実行します。

```
vgchange -ay vg
```

### 14.1. 論理ボリュームの自動アクティブ化の制御

論理ボリュームの自動アクティブ化は、システム起動時に論理ボリュームをイベントベースで自動的にアクティブにすることを指します。システムでデバイスが利用可能になると(デバイスのオンラインイベント)、**systemd/udev** は、各デバイスに **lvm2-pvscan** サービスを実行します。このサービスは、**named** デバイスを読み込む **pvscan --cache -aay device** コマンドを実行します。デバイスがボリュームグループに属している場合、**pvscan** コマンドは、そのボリュームグループに対する物理ボリュームがすべて、そのシステムに存在するかどうかを確認します。存在する場合は、このコマンドが、そのボリュームグループにある論理ボリュームをアクティブにします。

**/etc/lvm/lvm.conf** 設定ファイルで以下の設定オプションを使用して、論理ボリュームの自動アクティブ化を制御できます。

- **global/event\_activation**  
**event\_activation** が無効になっている場合、**systemd/udev** は、システムの起動時に存在する物理ボリュームでのみ、論理ボリュームを自動アクティブにします。すべての物理ボリュームが表示されていないと、一部の論理ボリュームが自動的にアクティブにならない場合もあります。
- **activation/auto\_activation\_volume\_list**  
**auto\_activation\_volume\_list** を空のリストに設定すると、自動アクティベーションは完全に無効になります。特定の論理ボリュームとボリュームグループに **auto\_activation\_volume\_list** を設定すると、自動アクティベーションは、設定した論理ボリュームに制限されます。

このオプションの設定は、`/etc/lvm/lvm.conf` 設定ファイルを参照してください。

## 14.2. 論理ボリュームのアクティブ化の制御

以下の方法で、論理ボリュームのアクティブ化を制御できます。

- `/etc/lvm/conf` ファイルの `activation/volume_list` 設定で行います。これにより、どの論理ボリュームをアクティブにするかを指定できます。このオプションの使用方法は `/etc/lvm/lvm.conf` 設定ファイルを参照してください。
- 論理ボリュームのアクティブ化スキップフラグで行います。このフラグが論理ボリュームに設定されていると、通常のアクティベーションコマンド時にそのボリュームがスキップされます。

以下の方法で、論理ボリュームのアクティブ化スキップフラグを設定できます。

- `lvcreate` コマンドの `-kn` オプションまたは `--setactivationskip n` オプションを指定すると、論理ボリュームの作成時にアクティブ化スキップフラグをオフにできます。
- `lvchange` コマンドの `-kn` オプションまたは `--setactivationskip n` オプションを指定すると、既存の論理ボリュームのアクティブ化スキップフラグをオフにできます。
- `lvchange` コマンドの `-ky` オプションまたは `--setactivationskip y` オプションを指定すると、オフになっているボリュームに対して、アクティベーションスキップフラグを再度オンにできます。

このアクティブ化スキップフラグが論理ボリュームに設定されているかを確認するには、`lvs` コマンドを実行します。以下のような `k` 属性が表示されます。

```
# lvs vg/thin1s1
LV      VG Attr      LSize Pool Origin
thin1s1  vg  Vwi---tz-k 1.00t pool0 thin1
```

標準オプション `-ay` または `--activate y` の他に、`-K` オプションまたは `--ignoreactivationskip` オプションを使用して、`k` 属性セットで論理ボリュームをアクティブにできます。

デフォルトでは、シンプロビジョニングのスナップショットボリュームに、作成時にアクティブ化スキップのフラグが付いています。`/etc/lvm/lvm.conf` ファイルの `auto_set_activation_skip` 設定で、新たに作成した、シンプロビジョニングのスナップショットボリュームの、デフォルトのアクティブ化スキップ設定を制御できます。

以下のコマンドは、アクティブ化スキップフラグが設定されているシンプロビジョニングされたスナップショット論理ボリュームをアクティベートします。

```
# lvchange -ay -K VG/SnapLV
```

以下のコマンドは、アクティブ化スキップフラグがない、シンプロビジョニングされたスナップショットを作成します。

```
# lvcreate --type thin -n SnapLV -kn -s ThinLV --thinpool VG/ThinPoolLV
```

以下のコマンドは、スナップショット論理ボリュームから、アクティブ化スキップフラグを削除します。

```
# lvchange -kn VG/SnapLV
```

### 14.3. 共有論理ボリュームのアクティベーション

以下のように、**lvchange** コマンドおよび **vgchange** コマンドの **-a** オプションを使用して、共有論理ボリュームの論理ボリュームのアクティブ化を制御できます。

コマンド	アクティベーション
<b>lvchange -ay e</b>	共有論理ボリュームを排他モードでアクティベートし、1台のホストのみが論理ボリュームをアクティベートできるようにします。アクティベーションが失敗 (論理ボリュームが別のホストでアクティブの場合に発生する可能性あり) すると、エラーが報告されます。
<b>lvchange -asy</b>	共有モードで共有論理ボリュームをアクティブにし、複数のホストが論理ボリュームを同時にアクティブにできるようにします。アクティブできない場合は (その論理ボリュームは別のホストで排他的にアクティブになっている場合に発生する可能性あり)、エラーが報告されます。スナップショットなど、論理タイプが共有アクセスを禁止している場合は、コマンドがエラーを報告して失敗します。複数のホストから同時に使用できない論理ボリュームタイプには、シン、キャッシュ、RAID、およびスナップショットがあります。
<b>lvchange -an</b>	論理ボリュームを非アクティブにします。

### 14.4. 欠落しているデバイスを含む論理ボリュームのアクティブ化

**lvchange** コマンドで、**activation\_mode** パラメーターを以下のいずれかの値に設定することで、デバイスが見つからない論理ボリュームをアクティブにするように設定できます。

アクティベーションモード	意味
complete	物理ボリュームが見つからない論理ボリュームのみをアクティベートすることを許可します。これが最も制限的なモードです。
degraded	物理ボリュームが見つからない RAID 論理ボリュームをアクティベートすることを許可します。
partial	物理ボリュームが見つからない論理ボリュームをすべてアクティベートすることを許可します。このオプションは、回復または修復にのみ使用してください。

**activation\_mode** のデフォルト値は、**/etc/lvm/lvm.conf** ファイルの **activation\_mode** 設定により決まります。詳細情報は、man ページの **lvraid(7)** を参照してください。

## 第15章 LVM デバイススキャンの制御

`/etc/lvm/lvm.conf` ファイルにフィルターを設定することで、LVM デバイスのスキャンを制御できます。`lvm.conf` ファイルのフィルターは、`/dev` ディレクトリーのデバイス名に適用される一連の単純な正規表現で構成され、見つかった各ブロックデバイスを許可するかどうかを指定します。

### 15.1. LVM デバイスフィルター

LVM ツールは、`/dev` ディレクトリー内のデバイスをスキャンし、そのデバイスすべてで LVM メタデータを確認します。`/etc/lvm/lvm.conf` ファイルのフィルターは、LVM がスキャンするデバイスを制御します。

フィルターは、`/dev` ディレクトリーのスキャン、または `/etc/lvm/lvm.conf` ファイルの `dir` キーワードで指定されたディレクトリーによって検出される各デバイスに LVM が適用されるパターンの一覧です。パターンは、任意の文字で区切られた正規表現で、その前に、`accept` の場合は `a`、`reject` の場合は `r` が付いています。デバイスに一致する最初の正規表現は、LVM がデバイスを許可するか、拒否（無視）するかを判断します。LVM は、パターンに一致しないデバイスを受け付けます。

以下は、すべてのデバイスをスキャンするフィルターのデフォルト設定です。

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

### 15.2. LVM デバイスフィルター設定の例

以下の例は、LVM がスキャンするデバイスを制御するフィルターの使用を示しています。



#### 警告

ここに示す例では、意図せずにシステムにある追加デバイスと一致する場合があります。システムの推奨プラクティスを表していないことがあることに注意してください。たとえば、`a/loop/` は `a.*loop.*/` と同等であり、`/dev/solooperation/lvol1` に一致します。

- 以下のフィルターは、検出されたすべてのデバイスを追加します。設定ファイル内で設定されているフィルターはないため、これはデフォルトの動作になります。

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

- 以下のフィルターは、ドライブにメディアが入っていない場合の遅延を回避するために `cdrom` デバイスを削除します。

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

- 以下のフィルターはすべてのループデバイスを追加して、その他のすべてのブロックデバイスを削除します。

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

- 以下のフィルターはすべてのループデバイスと IDE デバイスを追加して、その他のすべてのブロックデバイスを削除します。

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

- 以下のフィルターは1番目の IDE ドライブにパーティション 8 のみを追加して、他のすべてのブロックデバイスを削除します。

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

### 15.3. LVM デバイスフィルター設定の適用

この手順では、LVM スキャンするデバイスを制御する LVM デバイスフィルターの設定を変更します。

#### 前提条件

- 使用するデバイスフィルターパターンを準備します。

#### 手順

1. `/etc/lvm/lvm.conf` ファイルを変更せずに、デバイスフィルターパターンをテストします。LVM コマンドに、`--config 'devices{ filter = [ your device filter pattern ] }'` オプションを指定して使用します。以下に例を示します。

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

2. `/etc/lvm/lvm.conf` 設定ファイルで `filter` オプションを編集して、新しいデバイスフィルターパターンを使用します。
3. 新しい設定で、使用する物理ボリュームまたはボリュームグループがないことを確認します。

```
# pvscan
```

```
# vgscan
```

4. 再起動時に LVM が必要なデバイスのみをスキャンするように `initramfs` ファイルシステムを再構築します。

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

## 第16章 論理ボリュームに LVM 物理ボリュームを階層化

LVM を設定して、論理ボリュームに物理ボリュームを作成できるようにします。

デフォルトでは、LVM コマンドは、システムの論理ボリュームをスキャンしません。このデフォルトの動作には、次の利点があります。

- システムにアクティブな論理ボリュームが多数ある場合は、LVM コマンドごとに追加の時間が必要になり、パフォーマンスに悪影響を及ぼし、望まない遅延やタイムアウトが発生します。
- 論理ボリュームにゲスト仮想マシンのイメージの物理ボリュームが含まれている場合、ホストは通常、ゲストに属する階層化された物理ボリュームをスキャンまたは使用しません。ただし、ゲストの仮想マシンの物理ボリュームが、ホストから見える SCSI デバイスに直接存在する場合は、ホストの LVM がその物理ボリュームにアクセスできないようにするために、[15 章 LVM デバイススキャンの制御](#)に従って、フィルターを設定する必要があります。

意図的に論理ボリューム層に物理ボリューム層を載せる場合に、論理ボリュームをスキャンすることが必要になる場合があります。これにより、論理ボリュームで **pvcreate** コマンドを実行できます。すべての論理ボリュームをスキャンするように LVM を設定するには、`/etc/lvm/lvm.conf` ファイルの **scan\_lvs** 設定オプションを **scan\_lvs=1** に設定します。LVM コマンドがスキャンする論理ボリュームを制限するには、[15 章 LVM デバイススキャンの制御](#)に従って、`/etc/lvm/lvm.conf` 設定ファイルにデバイスフィルターを設定できます。

## 第17章 LVM の割り当ての制御

デフォルトでは、ボリュームグループは、同じ物理ボリューム上に並行ストライプを配置しないなど、常識的な規則に従って物理エクステントを割り当てます。これは、**normal** 割り当てポリシーです。**vgcreate** コマンドで **--alloc** 引数を使用して、割り当てポリシーの **contiguous**、**anywhere**、または **cling** を指定します。一般的に、**normal** 以外の割り当てポリシーが必要となるのは、通常とは異なる、標準外のエクステント割り当てを必要とする特別なケースのみです。

### 17.1. LVM の割り当てポリシー

LVM の操作で物理エクステントを1つまたは複数の論理ボリュームに割り当てる必要がある場合、割り当ては以下のように行われます。

- ボリュームグループで割り当てられていない物理エクステントのセットが、割り当てのために生成されます。コマンドラインの末尾に物理エクステントの範囲を指定すると、指定した物理ボリュームの中で、その範囲内で割り当てられていない物理エクステントだけが、割り当て用エクステントとして考慮されます。
- 割り当てポリシーは順番に試行されます。最も厳格なポリシー (**contiguous**) から始まり、最後は **--alloc** オプションで指定した割り当てポリシーか、特定の論理ボリュームやボリュームグループにデフォルトとして設定されている割り当てポリシーが試行されます。割り当てポリシーでは、埋める必要がある空の論理ボリューム領域の最小番号の論理エクステントから始まり、割り当てポリシーによる制限に従って、できるだけ多くの領域の割り当てを行います。領域が足りなくなると、LVM は次のポリシーに移動します。

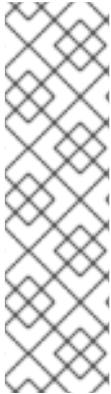
割り当てポリシーの制限は以下のとおりです。

- **contiguous** の割り当てポリシーでは、論理ボリュームの最初の論理エクステントを除いたすべての論理エクステントは、その直前の論理エクステントに物理的に隣接している必要があります。論理ボリュームがストライプ化またはミラー化されている場合は、**contiguous** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レグ) に個別に適用されます。
- **cling** の割り当てポリシーでは、既存の論理ボリュームに追加する論理エクステントに使用される物理ボリュームが、その論理ボリュームにある別の (1つ以上の) 論理エクステントですでに使用されている必要があります。設定パラメーター **allocation/cling\_tag\_list** が定義されており、一覧表示されているいずれかのタグが2つの物理ボリュームにある場合、この2つの物理ボリュームは一致すると見なされます。これにより、割り当てのために、同様のプロパティ (物理的な場所など) が設定されている物理ボリュームのグループにタグを付け、その物理ボリュームを同等なものとして処理できます。論理ボリュームがストライプ化またはミラー化されると、**cling** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レグ) に個別に適用されます。
- **normal** の割り当てポリシーは、並列の論理ボリューム (異なるストライプまたはミラーイメージ/レグ) 内の同じオフセットで、その並列の論理ボリュームにすでに割り当てられている論理エクステントと同じ物理ボリュームを共有する物理エクステントは選択しません。ミラーデータを保持するために、論理ボリュームと同時にミラーログを割り当てる場合、**normal** の割り当てポリシーでは、最初にログとデータに対して、それぞれ別の物理ボリュームを選択しようとします。異なる物理ボリュームを選択できず、かつ **allocation/mirror\_logs\_require\_separate\_pvs** 設定パラメーターが0に設定されている場合は、データの一部とログが物理ボリュームを共有できるようになります。

また、シンプルメタデータを割り当てる場合も、**normal** の割り当てポリシーはミラーログを割り当てる場合と同じようになりますが、設定パラメーターは **allocation/thin\_pool\_metadata\_require\_separate\_pvs** の値が適用されます。

- 割り当て要求を満たすのに十分な空きエクステントがあっても、**normal** の割り当てポリシーで使用されない場合は、たとえ同じ物理ボリュームに2つのストライプを配置することでパフォーマンスが低下しても、**anywhere** 割り当てポリシーがその空きエクステントを使用します。

割り当てポリシーは、**vgchange** コマンドで変更できます。



### 注記

定義された割り当てポリシーに従って、このセクションで説明されている以上のレイアウトの操作が必要な場合は、今後のバージョンでコードが変更する可能性があることに注意してください。たとえば、割り当て可能な空き物理エクステントの数が同じ2つの空の物理ボリュームをコマンドラインで指定する場合、現行バージョンのLVMでは、それが表示されている順番に使用が検討されます。ただし、今後のリリースで、そのプロパティが変更されない保証はありません。特定の論理ボリューム用に特定のレイアウトを取得することが重要な場合は、各手順に適用される割り当てポリシーに基づいてLVMがレイアウトを決定することがないように、**lvcreate** と **lvconvert** を順に使用してレイアウトを構築してください。

特定のケースで、割り当てプロセスがどのように行われているかを確認するには、コマンドに **-vvvv** オプションを追加するなどして、デバッグログ出力を表示します。

## 17.2. 物理ボリュームでの割り当て防止

**pvchange** コマンドを使用すると、1つまたは複数の物理ボリュームの空き領域で物理エクステントが割り当てられないように設定できます。これは、ディスクエラーが発生した場合や、物理ボリュームを取り除く場合に必要となる可能性があります。

以下のコマンドは、**/dev/sdk1** での物理エクステントの割り当てを無効にします。

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

**pvchange** コマンドで **-xy** 引数を使用すると、無効にされていた割り当てを許可できます。

## 17.3. CLING 割り当てポリシーを使用した論理ボリュームの拡張

LVM ボリュームを拡張する際には、**lvextend** コマンドの **--alloc cling** オプションを使用して、**cling** 割り当てポリシーを指定できます。このポリシーにより、既存の論理ボリュームの最終セグメントと同じ物理ボリュームの領域が選択されます。物理ボリューム上に十分な領域がなく、タグの一覧が **/etc/lvm/lvm.conf** ファイルで定義されている場合には、LVM が、その物理ボリュームにいずれかのタグが付けられているかを確認し、既存エクステントと新規エクステント間で、物理ボリュームのタグを適合させようとしています。

たとえば、使用している論理ボリュームが、1つのボリュームグループ内の2サイト間でミラー化されている場合は、**@site1** タグと **@site2** タグを使用し、サイトの場所に応じて物理ボリュームにタグを付けることができます。この場合は、**lvm.conf** ファイル内に以下の行を指定します。

```
cling_tag_list = [ "@site1", "@site2" ]
```

以下の例では、**lvm.conf** ファイルが変更されて、次のような行が追加されています。

```
cling_tag_list = [ "@A", "@B" ]
```

また、この例では、`/dev/sdb1`、`/dev/sdc1`、`/dev/sdd1`、`/dev/sde1`、`/dev/sdf1`、`/dev/sdg1`、および `/dev/sdh1` の物理ボリュームで構成されるボリュームグループ **taft** が作成されています。この物理ボリュームには、**A**、**B**、および **C** のタグが付けられています。この例では、**C** のタグは使用されていませんが、LVM がタグを使用して、ミラーレグに使用する物理ボリュームを選択することを示しています。

```
# pvs -a -o +pv_tags /dev/sd[bcdefgh]
PV      VG  Fmt Attr PSize PFree PV Tags
/dev/sdb1 taft lvm2 a-- 15.00g 15.00g A
/dev/sdc1 taft lvm2 a-- 15.00g 15.00g B
/dev/sdd1 taft lvm2 a-- 15.00g 15.00g B
/dev/sde1 taft lvm2 a-- 15.00g 15.00g C
/dev/sdf1 taft lvm2 a-- 15.00g 15.00g C
/dev/sdg1 taft lvm2 a-- 15.00g 15.00g A
/dev/sdh1 taft lvm2 a-- 15.00g 15.00g A
```

以下のコマンドは、ボリュームグループ **taft** から 10 ギガバイトのミラー化ボリュームを作成します。

```
# lvcreate --type raid1 -m 1 -n mirror --nosync -L 10G taft
WARNING: New raid1 won't be synchronised. Don't read what you didn't write!
Logical volume "mirror" created
```

以下のコマンドは、ミラーレグおよび RAID メタデータのサブボリュームに使用されるデバイスを表示します。

```
# lvs -a -o +devices
LV      VG  Attr      LSize Log Cpy%Sync Devices
mirror  taft Rwi-a-r--- 10.00g 100.00 mirror_rimage_0(0),mirror_rimage_1(0)
[mirror_rimage_0] taft iwi-aor--- 10.00g /dev/sdb1(1)
[mirror_rimage_1] taft iwi-aor--- 10.00g /dev/sdc1(1)
[mirror_rmeta_0] taft ewi-aor--- 4.00m /dev/sdb1(0)
[mirror_rmeta_1] taft ewi-aor--- 4.00m /dev/sdc1(0)
```

以下のコマンドは、ミラー化ボリュームのサイズを拡張します。**cling** 割り当てポリシーで、同じタグが付いた物理ボリュームを使用して、ミラーレグが拡張される必要があることを示します。

```
# lvextend --alloc cling -L +10G taft/mirror
Extending 2 mirror images.
Extending logical volume mirror to 20.00 GiB
Logical volume mirror successfully resized
```

以下に表示したコマンドは、レグとして同一のタグが付いた物理ボリュームを使用してミラーレグが拡張されているのを示しています。**C** のタグが付いた物理ボリュームは無視される点に注意してください。

```
# lvs -a -o +devices
LV      VG  Attr      LSize Log Cpy%Sync Devices
mirror  taft Rwi-a-r--- 20.00g 100.00 mirror_rimage_0(0),mirror_rimage_1(0)
[mirror_rimage_0] taft iwi-aor--- 20.00g /dev/sdb1(1)
[mirror_rimage_0] taft iwi-aor--- 20.00g /dev/sdg1(0)
[mirror_rimage_1] taft iwi-aor--- 20.00g /dev/sdc1(1)
[mirror_rimage_1] taft iwi-aor--- 20.00g /dev/sdd1(0)
[mirror_rmeta_0] taft ewi-aor--- 4.00m /dev/sdb1(0)
[mirror_rmeta_1] taft ewi-aor--- 4.00m /dev/sdc1(0)
```

## 第18章 LVM のトラブルシューティング

LVM ツールを使用して、LVM ボリュームおよびグループのさまざまな問題のトラブルシューティングを行うことができます。

### 18.1. LVM での診断データの収集

LVM コマンドが想定どおりに機能しない場合は、以下の方法で診断情報を収集できます。

#### 手順

- 以下の方法を使用して、さまざまな診断データを収集します。
  - **-vvvv** 引数を LVM コマンドに追加して、コマンドの出力の詳細レベルを増やします。
  - `/etc/lvm/lvm.conf` 設定ファイルの **log** セクションで、**level** オプションの値を増やします。これにより、LVM がシステムログにより多くの情報を提供します。
  - 問題が論理ボリュームのアクティブ化に関連する場合は、アクティブ化中に LVM がログメッセージをログに記録できるようにします。
    - i. `/etc/lvm/lvm.conf` 設定ファイルの **log** セクションで **activation = 1** オプションを設定します。
    - ii. LVM コマンドに **-vvvv** オプションを付けて実行します。
    - iii. コマンドの出力を確認します。
    - iv. **activation** オプションを **0** にリセットします。  
オプションを **0** にリセットしないと、メモリー不足の状況でシステムが応答しなくなる可能性があります。

- 診断目的で情報ダンプを表示します。

```
# lvmdump
```

- 追加のシステム情報を表示します。

```
# lvs -v
```

```
# pvs --all
```

```
# dmsetup info --columns
```

- `/etc/lvm/backup/` ディレクトリーの最後の LVM メタデータのバックアップと、`/etc/lvm/archive/` ディレクトリー内のアーカイブバージョンを確認します。
- 現在の設定情報を確認します。

```
# lvmconfig
```

- `/run/lvm/hints` キャッシュファイルで、物理ボリュームを持つデバイスを記録します。

## 関連情報

- man ページの `lvmdump(8)`

## 18.2. 障害の発生した LVM デバイスに関する情報の表示

ボリュームが失敗した理由を特定するのに役立つ、障害の発生した LVM ボリュームに関する情報を表示できます。

### 手順

- `vgs` ユーティリティーまたは `lvs` ユーティリティーを使用して、障害が発生したボリュームを表示します。

#### 例18.1 障害が発生したボリュームグループ

この例では、ボリュームグループ `vg` を構成するデバイスのいずれかが失敗しています。ボリュームグループは使用できませんが、障害が発生したデバイスに関する情報を表示できます。

```
# vgs --options +devices

/dev/sdb: open failed: No such device or address
/dev/sdb: open failed: No such device or address
WARNING: Couldn't find device with uuid 42B7bu-YCmp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG vg is missing PV 42B7bu-YCmp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
WARNING: Couldn't find all devices for LV vg/linear while checking used and assumed devices.
WARNING: Couldn't find all devices for LV vg/striped while checking used and assumed devices.
VG #PV #LV #SN Attr  VSize VFree Devices
vg  2  2  0 wz-pn- <3.64t <3.60t [unknown](0)
vg  2  2  0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/sdc1(0)
```

#### 例18.2 リニアおよびストライプの論理ボリュームの障害

この例では、障害が発生したデバイスにより、ボリュームグループ内のリニア論理ボリュームとストライプ化論理ボリュームの両方が失敗していました。コマンドの出力には、障害が発生した論理ボリュームが表示されます。

```
# lvs --all --options +devices

/dev/sdb: open failed: No such device or address
/dev/sdb: open failed: No such device or address
WARNING: Couldn't find device with uuid 42B7bu-YCmp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG vg is missing PV 42B7bu-YCmp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
WARNING: Couldn't find all devices for LV vg/linear while checking used and assumed devices.
WARNING: Couldn't find all devices for LV vg/striped while checking used and assumed devices.
```

```

LV   VG Attr   LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
linear vg -wi-a---p- 20.00g                [unknown](0)
stripe vg -wi-a---p- 20.00g                [unknown]
(5120),/dev/sdc1(0)

```

### 例18.3 ミラー化論理ボリュームのログに障害が発生しました。

以下の例は、ミラー化論理ボリュームのログが失敗した場合の **vgs** ユーティリティーおよび **lvs** ユーティリティーからのコマンドの出力を示しています。

```

# vgs --all --options +devices

VG   #PV #LV #SN Attr   VSize VFree Devices
corey 4 4 0 rz-pnc 1.58T 1.34T my_mirror_mimage_0(0),my_mirror_mimage_1(0)
corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
corey 4 4 0 rz-pnc 1.58T 1.34T unknown device(0)
corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdb1(0)

# lvs --all --options +devices

LV           VG   Attr   LSize  Origin Snap% Move Log           Copy% Devices
my_mirror    corey mwi-a- 120.00G my_mirror_mlog 1.95
my_mirror_mimage_0(0),my_mirror_mimage_1(0)
[my_mirror_mimage_0] corey iwi-ao 120.00G unknown device(0)
[my_mirror_mimage_1] corey iwi-ao 120.00G /dev/sdb1(0)
[my_mirror_mlog] corey lwi-ao 4.00M /dev/sdd1(0)

```

## 18.3. ボリュームグループから見つからない LVM 物理ボリュームの削除

物理ボリュームに障害が発生した場合は、ボリュームグループ内の残りの物理ボリュームをアクティブにし、その物理ボリュームを使用していたすべての論理ボリュームをボリュームグループから削除できます。

### 手順

1. ボリュームグループ内の残りの物理ボリュームをアクティベートします。

```
# vgchange --activate y --partial volume-group
```

2. 削除する論理ボリュームを確認します。

```
# vgreduce --removemissing --test volume-group
```

3. ボリュームグループから、失われた物理ボリュームを使用していた論理ボリュームをすべて削除します。

```
# vgreduce --removemissing --force volume-group
```

4. 必要に応じて、保持する論理ボリュームを誤って削除した場合には、**vgreduce** 操作を元に戻すことができます。

```
# vgcfgrestore volume-group
```



### 警告

シンプールの削除すると、LVM は操作を元に戻すことができません。

## 18.4. 破損したメタデータを使用した LVM 物理ボリュームの復旧

物理ボリュームのボリュームグループメタデータ領域が誤って上書きされたり、破棄されたりする場合は、メタデータ領域が正しくないことを示すエラーメッセージか、システムが特定の UUID を持つ物理ボリュームを見つけることができないことを示すエラーメッセージが表示されます。物理ボリュームのメタデータ領域を書き換えて、物理ボリュームからデータを復旧できる場合があります。

### 18.4.1. LVM ボリュームに欠落または破損したメタデータがあることを検出

以下の例は、物理ボリュームのメタデータ領域が見つからないか、破損しているかを確認できるコマンド出力を示しています。

#### 手順

- 論理ボリュームを一覧表示しようとします。

```
# lvs --all --options +devices
```

#### 例18.4 不足または破損したメタデータを持つ出力

この例では、特定の論理ボリュームが、メタデータが欠落しているか破損している物理ボリュームに配置されています。

```
Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
Couldn't find all physical volumes for volume group VG.
Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
Couldn't find all physical volumes for volume group VG.
...
```

### 18.4.2. 見つからない LVM 物理ボリュームのメタデータの検索

この手順では、物理ボリュームが見つからないか、破損している、アーカイブされた最新のメタデータを見つけます。

#### 手順

1. 物理ボリュームを含むボリュームグループのアーカイブされたメタデータファイルを検索します。

アーカイブされたメタデータファイルは、`/etc/lvm/archive/volume-group-name_backup-number.vg` パスにあります。ボリュームグループの番号が最も高い、既知の有効なメタデータファイルの最後のものを選択します。

2. 物理ボリュームの UUID を検索します。以下の方法のいずれかを使用します。

- 論理ボリュームを一覧表示します。

```
# lvs --all --options +devices

Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
```

- アーカイブされたメタデータファイルを確認します。ボリュームグループ設定の `physical_volumes` セクションで、`id =` のラベルが付いた値として UUID を検索します。
- `--partial` オプションを使用してボリュームグループを非アクティブにします。

```
# vgchange --activate n --partial volume-group-name

PARTIAL MODE. Incomplete logical volumes will be processed.
WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG raid_sanity is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
0 logical volume(s) in volume group "raid_sanity" now active
```

### 18.4.3. LVM 物理ボリュームでのメタデータの復元

この手順では、破損したり、新しいデバイスに置き換えたりする物理ボリュームのメタデータを復元します。



#### 警告

作業用の LVM 論理ボリュームでこの手順を実行しないでください。誤った UUID を指定すると、データが失われることとなります。

#### 前提条件

- 見つからない物理ボリュームのメタデータを特定している。詳細は「[見つからない LVM 物理ボリュームのメタデータの検索](#)」を参照してください。

#### 手順

1. 物理ボリュームでメタデータを復元します。

```
# pvcreate --uuid physical-volume-uuid \
  --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \
  block-device
```



## 注記

コマンドは、LVM メタデータ領域のみを上書きし、既存のデータ領域には影響を与えません。

### 例18.5 /dev/sdh1 での物理ボリュームの復元

以下の例では、以下のプロパティーで /dev/sdh1 デバイスを物理ボリュームとしてラベル付けします。

- **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk** の UUID
- **VG\_00050.vg** に含まれるメタデータ情報 (ボリュームグループの最新のアーカイブメタデータ)

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \
  --restorefile /etc/lvm/archive/VG_00050.vg \
  /dev/sdh1

...
Physical volume "/dev/sdh1" successfully created
```

2. ボリュームグループのメタデータを復元します。

```
# vgcfgrestore volume-group-name

Restored volume group volume-group-name
```

3. ボリュームグループの論理ボリュームを表示します。

```
# lvs --all --options +devices volume-group-name
```

現在、論理ボリュームは非アクティブです。以下に例を示します。

```
LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
stripe VG  -wi--- 300.00G                /dev/sdh1 (0),/dev/sda1(0)
stripe VG  -wi--- 300.00G                /dev/sdh1 (34728),/dev/sdb1(0)
```

4. 論理ボリュームのセグメントタイプが RAID またはミラーの場合は、論理ボリュームを再同期します。

```
# lvchange --resync volume-group-name/logical-volume-name
```

5. 論理ボリュームを非アクティブにします。

```
# lvchange --activate y /dev/volume-group-name/logical-volume-name
```

6. ディスク上の LVM メタデータが、それを上書きしたものと同じかそれ以上のスペースを使用する場合は、この手順で物理ボリュームを回復できます。メタデータを上書きしたものがメタデータ領域を超えると、ボリューム上のデータが影響を受ける可能性があります。fsck コマンドを使用してデータを復元できる場合があります。

## 検証手順

- アクティブな論理ボリュームを表示します。

```
# lvs --all --options +devices

LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
stripe VG   -wi-a- 300.00G                /dev/sdh1 (0),/dev/sda1(0)
stripe VG   -wi-a- 300.00G                /dev/sdh1 (34728),/dev/sdb1(0)
```

## 18.5. 見つからない LVM 物理ボリュームの置き換え

物理ボリュームに障害が発生した場合や、置き換える必要がある場合は、新しい物理ボリュームにラベルを付けて、既存のボリュームグループで失われたボリュームを置き換えることができます。

### 前提条件

- 物理ボリュームを新しいストレージデバイスに置き換えている。  
(このステップの配置は再評価します)

### 18.5.1. 見つからない LVM 物理ボリュームのメタデータの検索

この手順では、物理ボリュームが見つからないか、破損している、アーカイブされた最新のメタデータを見つけます。

### 手順

1. 物理ボリュームを含むボリュームグループのアーカイブされたメタデータファイルを検索します。  
アーカイブされたメタデータファイルは、**/etc/lvm/archive/volume-group-name\_backup-number.vg** パスにあります。ボリュームグループの番号が最も高い、既知の有効なメタデータファイルの最後のものを選択します。
2. 物理ボリュームの UUID を検索します。以下の方法のいずれかを使用します。
  - 論理ボリュームを一覧表示します。

```
# lvs --all --options +devices

Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
```

- アーカイブされたメタデータファイルを確認します。ボリュームグループ設定の **physical\_volumes** セクションで、**id =** のラベルが付いた値として UUID を検索します。
- **--partial** オプションを使用してボリュームグループを非アクティブにします。

```
# vgchange --activate n --partial volume-group-name

PARTIAL MODE. Incomplete logical volumes will be processed.
WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
WARNING: VG raid_sanitty is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
0 logical volume(s) in volume group "raid_sanitty" now active
```

## 18.5.2. LVM 物理ボリュームでのメタデータの復元

この手順では、破損したり、新しいデバイスに置き換えたりする物理ボリュームのメタデータを復元します。



### 警告

作業用の LVM 論理ボリュームでこの手順を実行しないでください。誤った UUID を指定すると、データが失われることになります。

### 前提条件

- 見つからない物理ボリュームのメタデータを特定している。詳細は「[見つからない LVM 物理ボリュームのメタデータの検索](#)」を参照してください。

### 手順

1. 物理ボリュームでメタデータを復元します。

```
# pvcreate --uuid physical-volume-uuid \  
--restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \  
block-device
```



### 注記

コマンドは、LVM メタデータ領域のみを上書きし、既存のデータ領域には影響を与えません。

#### 例18.6 /dev/sdh1 での物理ボリュームの復元

以下の例では、以下のプロパティで **/dev/sdh1** デバイスを物理ボリュームとしてラベル付けします。

- **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk** の UUID
- **VG\_00050.vg** に含まれるメタデータ情報 (ボリュームグループの最新のアーカイブメタデータ)

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \  
--restorefile /etc/lvm/archive/VG_00050.vg \  
/dev/sdh1  
  
...  
Physical volume "/dev/sdh1" successfully created
```

2. ボリュームグループのメタデータを復元します。

```
# vgcfgrestore volume-group-name

Restored volume group volume-group-name
```

3. ボリュームグループの論理ボリュームを表示します。

```
# lvs --all --options +devices volume-group-name
```

現在、論理ボリュームは非アクティブです。以下に例を示します。

```
LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
stripe VG   -wi--- 300.00G                /dev/sdh1 (0),/dev/sda1(0)
stripe VG   -wi--- 300.00G                /dev/sdh1 (34728),/dev/sdb1(0)
```

4. 論理ボリュームのセグメントタイプが RAID またはミラーの場合は、論理ボリュームを再同期します。

```
# lvchange --resync volume-group-name/logical-volume-name
```

5. 論理ボリュームを非アクティブにします。

```
# lvchange --activate y /dev/volume-group-name/logical-volume-name
```

6. ディスク上の LVM メタデータが、それを上書きしたものと同じかそれ以上のスペースを使用する場合は、この手順で物理ボリュームを回復できます。メタデータを上書きしたものがメタデータ領域を超えると、ボリューム上のデータが影響を受ける可能性があります。fsck コマンドを使用してデータを復元できる場合があります。

## 検証手順

- アクティブな論理ボリュームを表示します。

```
# lvs --all --options +devices

LV   VG   Attr LSize  Origin Snap%  Move Log Copy%  Devices
stripe VG   -wi-a- 300.00G                /dev/sdh1 (0),/dev/sda1(0)
stripe VG   -wi-a- 300.00G                /dev/sdh1 (34728),/dev/sdb1(0)
```

## 18.6. 論理ボリュームの空きエクステン트가不足する問題のトラブルシューティング

論理ボリュームを作成しようとする、ボリュームグループに十分な空き容量があると思われる場合でも、**Insufficient free extents** エラーメッセージが表示されることがあります。ボリュームグループに論理ボリュームを作成できるようこのエラーのトラブルシューティングを行うことができます。

### 18.6.1. ボリュームグループ

物理ボリュームはボリュームグループ (VG) に統合されます。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。

ボリュームグループ内で、割り当て可能なディスク領域は、エクステントと呼ばれる固定サイズの単位に分割されます。割り当て可能な領域の最小単位は、1エクステントです。エクステントは、物理ボリュームでは物理エクステントと呼ばれます。

論理ボリュームには、物理エクステントと同じサイズの論理エクステントが割り当てられます。そのため、エクステントのサイズは、ボリュームグループ内のすべての論理ボリュームで同じになります。ボリュームグループは、論理エクステントを物理エクステントにマッピングします。

### 18.6.2. LVM 出力の丸めエラー

ボリュームグループの領域使用量を報告する LVM コマンドは、報告された数を 2 進法に切り上げ、人間が判読できる出力を提供します。これには、**vgdisplay** ユーティリティーおよび **vgs** ユーティリティーが含まれます。

丸めの結果、報告された空き領域の値は、ボリュームグループが提供する物理エクステントよりも大きくなる可能性があります。報告された空き領域のサイズの論理ボリュームを作成しようとする、以下のエラーが発生する可能性があります。

#### Insufficient free extents

エラーを回避するには、ボリュームグループの空き物理エクステントの数を調べる必要があります。これは、空き領域の正確な値です。次に、エクステントの数を使用して、論理ボリュームを正常に作成できます。

### 18.6.3. LVM ボリューム作成時の丸めエラーの防止

LVM 論理ボリュームを作成する際に、丸めエラーが発生しないように、論理ボリュームのサイズを指定できます。

#### 手順

1. ボリュームグループの空き物理エクステントの数を検索します。

```
# vgdisplay volume-group-name
```

#### 例18.7 ボリュームグループの空きエクステント

たとえば、以下のボリュームグループには 8780 個のの空き物理エクステントがあります。

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

2. 論理ボリュームを作成します。ボリュームサイズをバイトではなくエクステントに入力します。

#### 例18.8 エクステントの数を指定して論理ボリュームを作成

```
# lvcreate --extents 8780 --name testlv testvg
```

#### 例18.9 残りの領域をすべて使用する論理ボリュームの作成

または、論理ボリュームを拡張して、ボリュームグループ内の残りの空き領域の割合を使用できます。以下に例を示します。

```
# lvcreate --extents 100%FREE --name testlv2 testvg
```

## 検証手順

- ボリュームグループが使用するエクステントの数を確認します。

```
# vgs --options +vg_free_count,vg_extent_count

VG   #PV #LV #SN Attr   VSize  VFree Free #Ext
testvg 2  1  0 wz--n- 34.30G  0    0 8780
```

## 18.7. マルチパス化された LVM デバイスに対する重複した物理ボリューム警告のトラブルシューティング

マルチパスストレージで LVM を使用する場合は、ボリュームグループまたは論理ボリュームの一覧を表示する LVM コマンドを実行すると、以下のようなメッセージが表示される場合があります。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/dm-5 not /dev/sdd
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowerb not /dev/sde
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sddlmap not /dev/sdf
```

これらの警告のトラブルシューティングにより、LVM が警告を表示する理由を理解し、または警告を非表示にできます。

### 18.7.1. 重複した PV 警告の原因

Device Mapper Multipath (DM Multipath)、EMC PowerPath、または Hitachi Dynamic Link Manager (HDLM) などのマルチパスソフトウェアがシステム上のストレージデバイスを管理すると、特定の論理ユニット (LUN) への各パスが異なる SCSI デバイスとして登録されます。マルチパスソフトウェアは、各パスにマップする新しいデバイスを作成します。各 LUN には、同じ基礎となるデータを参照する `/dev` ディレクトリーに複数のデバイスノードがあるため、すべてのデバイスノードには同じ LVM メタデータが含まれます。

表18.1異なるマルチパスソフトウェアでのデバイスマッピングの例

マルチパスソフトウェア	LUN への SCSI パス	マルチパスデバイスパスへのマッピング
DM Multipath	<code>/dev/sdb</code> および <code>/dev/sdc</code>	<code>/dev/mapper/mpath1</code> または <code>/dev/mapper/mpatha</code>
EMC PowerPath		<code>/dev/emcpowera</code>
HDLM		<code>/dev/sddlmap</code>

複数のデバイスノードが原因で、LVM ツールは同じメタデータを複数回検出し、複製として報告します。

### 18.7.2. PV の重複警告が発生した場合

LVM は、以下のいずれかのケースで重複した PV 警告を表示します。

- 出力に表示される 2 つデバイスは、両方とも同じデバイスへの単一パスです。
- 出力に表示される 2 つのデバイスは、両方ともマルチパスマップです。

#### 同じデバイスへの単一パス

以下の例は、重複デバイスが、同じデバイスへの両方の単一パスである、重複した PV の警告を示しています。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/sdd not /dev/sdf
```

**multipath -ll** コマンドを使用して現在の DM Multipath トポロジーを一覧表示すると、同じマルチパスマップの下に **/dev/sdd** と **/dev/sdf** の両方を確認できます。

これらの重複メッセージは警告のみで、LVM 操作が失敗しているわけではありません。代わりに、LVM が物理ボリュームとしてデバイスのいずれかのみを使用して他を無視していることを警告します。

メッセージは、LVM が誤ったデバイスを選択するか、ユーザーが警告を中断していることを示す場合は、フィルターを適用できます。フィルターは、物理ボリュームに必要なデバイスのみを検索し、マルチパスデバイスへの基礎となるパスを省略するように LVM を設定します。その結果、警告が表示されなくなりました。

#### マルチパスマップ

以下の例は、両方のマルチパスマップである 2 つのデバイスに対する重複した物理ボリューム警告を示しています。重複した物理ボリュームは、同じデバイスへの異なるパスではなく、2 つのデバイスに置かれます。

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/mapper/mpatha not /dev/mapper/mpathc
```

```
Found duplicate PV GDjTZf7Y03GJHjteqOwrye2dcSCjdaUi: using /dev/emcpowera not /dev/emcpowerh
```

この状況は、同じデバイスへの両方の単一パスであるデバイスに対する重複する警告よりも複雑です。これらの警告は、多くの場合、マシンがアクセスできないデバイス (LUN クローンやミラーなど) にアクセスしていることを意味します。

マシンから削除するデバイスが分からないと、この状況は復旧できない可能性があります。Red Hat は、この問題に対処するために Red Hat テクニカルサポートにお問い合わせください。

### 18.7.3. LVM デバイスフィルター

LVM ツールは、**/dev** ディレクトリー内のデバイスをスキャンし、そのデバイスすべてで LVM メタデータを確認します。**/etc/lvm/lvm.conf** ファイルのフィルターは、LVM がスキャンするデバイスを制御します。

フィルターは、**/dev** ディレクトリーのスキャン、または **/etc/lvm/lvm.conf** ファイルの **dir** キーワードで指定されたディレクトリーによって検出される各デバイスに LVM が適用されるパターンの一覧です。パターンは、任意の文字で区切られた正規表現で、その前に、**accept** の場合は **a**、**reject** の場合は

`r` が付いています。デバイスに一致する最初の正規表現は、LVM がデバイスを許可するか、拒否 (無視) するかを判断します。LVM は、パターンに一致しないデバイスを受け付けます。

以下は、すべてのデバイスをスキャンするフィルターのデフォルト設定です。

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

#### 18.7.4. PV の重複警告を防ぐ LVM デバイスフィルターの例

以下の例は、1つの論理ユニット (LUN) への複数のストレージパスによって引き起こされる、重複した物理ボリュームの警告を回避する LVM デバイスフィルターを示しています。

設定するフィルターには、LVM がメタデータをチェックする必要があるすべてのデバイスが含まれている必要があります。たとえば、`root` ボリュームグループのあるローカルのハードドライブや、マルチパスを設定したデバイスなどです。マルチパスデバイスへの基礎となるパス (`/dev/sdb`、`/dev/sdd` など) を拒否すると、マルチパスデバイス自体で一意的な各メタデータ領域が一度検出されるため、重複した物理ボリュームの警告を回避できます。

- このフィルターは、最初のハードドライブと DM Multipath デバイスの次のパーティションを受け入れますが、その他のパーティションはすべて拒否します。

```
filter = [ "a/dev/sda2$", "a/dev/mapper/mpath.*", "r|.*)" ]
```

- このフィルターは、すべての HP SmartArray 全コントローラーと、EMC PowerPath デバイスを許可します。

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

- このフィルターは、最初の IDE ドライブとマルチパスデバイス上のパーティションをすべて受け入れます。

```
filter = [ "a/dev/hda.*", "a/dev/mapper/mpath.*", "r|.*)" ]
```

#### 18.7.5. LVM デバイスフィルター設定の適用

この手順では、LVM スキャンするデバイスを制御する LVM デバイスフィルターの設定を変更します。

##### 前提条件

- 使用するデバイスフィルターパターンを準備します。

##### 手順

- `/etc/lvm/lvm.conf` ファイルを変更せずに、デバイスフィルターパターンをテストします。LVM コマンドに、`--config 'devices{ filter = [ your device filter pattern ] }'` オプションを指定して使用します。以下に例を示します。

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

- `/etc/lvm/lvm.conf` 設定ファイルで `filter` オプションを編集して、新しいデバイスフィルターパターンを使用します。
- 新しい設定で、使用する物理ボリュームまたはボリュームグループがないことを確認します。

```
# pvscan
```

```
# vgscan
```

- 再起動時に LVM が必要なデバイスのみをスキャンするように **initramfs** ファイルシステムを再構築します。

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

#### 18.7.6. 関連情報

- 15章 [LVM デバイススキャンの制御](#)