



# Red Hat Enterprise Linux 9

## RHEL で論理ボリュームの重複排除および圧縮

VDO を使用した LVM ストレージ容量の拡張



# Red Hat Enterprise Linux 9 RHEL で論理ボリュームの重複排除および圧縮

VDO を使用した LVM ストレージ容量の拡張

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

## 法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deduplicating\_and\_compressing\_logical\_volumes\_on\_RHEL.ent file |.

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、LVM で Virtual Data Optimizer (VDO) 機能を使用して、RHEL で重複排除した論理ボリュームと圧縮した論理ボリュームを管理する方法を説明します。

## 目次

---

多様性を受け入れるオープンソースの強化 .....	3
RED HAT ドキュメントへのフィードバック (英語のみ) .....	4
第1章 LVM での VDO の概要 .....	5
第2章 LVM-VDO 要件 .....	6
2.1. VDO メモリー要件 .....	6
2.2. VDO ストレージの領域要件 .....	7
2.3. EXAMPLES OF VDO REQUIREMENTS BY PHYSICAL SIZE .....	7
2.4. ストレージスタックの LVM-VDO の配置 .....	8
第3章 重複排除した論理ボリュームと圧縮論理ボリュームの作成 .....	10
3.1. LVM-VDO デプロイメントシナリオ .....	10
3.2. LVM-VDO ボリュームの物理サイズおよび論理サイズ .....	12
3.3. VDO のスラブサイズ .....	12
3.4. VDO のインストール .....	13
3.5. LVM-VDO ボリュームの作成 .....	13
3.6. LVM-VDO ボリュームのマウント .....	14
3.7. LVM-VDO ボリュームでの圧縮および重複排除設定の変更 .....	15
3.8. 仮想データオプティマイザーを使用したシンプロビジョニングの管理 .....	16
第4章 LVM-VDO ボリュームの縮小オプション .....	20
4.1. VDO で DISCARD マウントオプションの有効化 .....	20
4.2. 定期的な TRIM 操作の設定 .....	20



## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#)をご覧ください。

## 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。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
  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隅に **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。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
  1. [Bugzilla](#) の Web サイトに移動します。
  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 LVM での VDO の概要

VDO (Virtual Data Optimizer) 機能は、ストレージ用にインラインのブロックレベルの重複排除、圧縮、およびシンプロビジョニングを提供します。VDO は、LVM シンプロビジョニングのボリュームと同様に、LVM 論理ボリューム (LV) として管理できます。

LVM (LVM-VDO) 上の VDO ボリュームは、以下の論理ボリュームで構成されます。

### VDO プール LV (論理ボリューム)

これは、VDO LV 用のデータを保存し、重複排除して、圧縮するバッキング物理デバイスです。VDO プール LV は、VDO ボリュームの物理サイズを設定します。これは、VDO がディスクに保存できるデータ量です。

現在、各 VDO プール LV は1つの VDO LV のみを保持できます。その結果、VDO は各 VDO LV を重複排除し、圧縮します。つまり、VDO は、複数の VDO LV 間で共有されるデータの重複排除または圧縮ができません。

### VDO LV

これは、VDO プール LV 上の仮想、プロビジョニングされたデバイスです。VDO LV は、VDO ボリュームのプロビジョニングされた論理サイズを設定します。これは、重複排除と圧縮が行われる前にアプリケーションがボリュームに書き込みできるデータ量です。

表1.1 LVM および LVM シンプロビジョニングにおける VDO 内のコンポーネントの比較

	物理デバイス	プロビジョニングされたデバイス
LVM 上の VDO	VDO プール LV (論理ボリューム)	VDO LV
LVM シンプロビジョニング	シンプール	シンボリューム

VDO はシンプロビジョニングされているため、ファイルシステムとアプリケーションは、使用中の論理領域のみを認識し、実際に利用可能な物理領域は認識しません。スクリプトを使用して、実際に利用可能な領域を監視し、使用量がしきい値を超えた場合 (たとえば、VDO プール LV の使用量が 80% になった場合) にアラートを生成します。

## 第2章 LVM-VDO 要件

LVM 上の VDO には、配置とシステムリソースに関する特定の要件があります。

### 2.1. VDO メモリー要件

各 VDO ボリュームには、2つの異なるメモリー要件があります。

#### VDO モジュール

VDO には、固定メモリー 38 MB と変動用に容量を確保する必要があります。

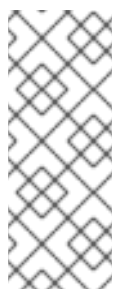
- 設定済みのブロックマップキャッシュサイズ 1 MB ごとに 1.15 MB のメモリー。ブロックマップキャッシュには、少なくとも 150 MB のメモリーが必要です。
- 1 TB の論理領域ごとに 1.6 MB のメモリー
- ボリュームが管理する物理ストレージの 1 TB ごとに 268 MB のメモリー。

#### UDS インデックス

Universal Deduplication Service (UDS) には、最低 250 MB のメモリーが必要です。このメモリー量は、重複排除が使用するデフォルトの容量です。この値は、インデックスに必要なストレージ容量も左右するので、VDO ボリュームのフォーマット時にこの値を設定してください。

UDS インデックスに必要なメモリーは、インデックスタイプと、重複排除ウィンドウに必要なサイズにより指定されます。

インデックスタイプ	重複排除ウィンドウ	備考
Dense	RAM 1 GB あたり 1 TB	通常、最大 4 TB の物理ストレージには、1 GB の dense インデックスで十分です。
Sparse	RAM 1 GB あたり 10 TB	通常、最大 40 TB の物理ストレージには、1 GB の sparse インデックスで十分です。



#### 注記

VDO ボリュームの最小ディスク使用量は、デフォルト設定のスラブサイズ 2 GB、密インデックス 0.25 を使用した場合、約 4.7 GB が必要です。これにより、0% の重複排除または圧縮で書き込むための 2 GB 弱の物理データが提供されます。

ここでの最小のディスク使用量は、デフォルトのスラブサイズと密インデックスの合計になります。

VDO で推奨されるモードは、UDS の sparse インデックス機能です。データの一時的な局所性に依存し、メモリー内で最も関連するインデックスエントリーのみを保持しようとしています。sparse インデックスでは、UDS は、同じ量のメモリーを使用しながら、dense を使用したときの 10 倍以上長い重複排除ウィンドウを維持できます。

sparse インデックスを使用すると対象範囲が広がりますが、dense インデックスの方が提供する重複排除アドバイスが多くなります。ワークロードの観点では、多くの場合、同じメモリー量であれば、dense インデックスと sparse インデックスにおける重複排除率の差異はごくわずかとなります。

## 関連情報

- [Examples of VDO requirements by physical size](#)

## 2.2. VDO ストレージの領域要件

VDO ボリュームを設定して、最大 256 TB の物理ストレージを使用するように設定できます。データを格納するのに使用できるのは、物理ストレージの一部のみです。このセクションでは、VDO に管理されるボリュームで使用可能なサイズを特定するための計算方法を説明します。

VDO では、2 種類の VDO メタデータと UDS インデックスにストレージが必要です。

- 1 つ目の VDO メタデータは、**物理ストレージ** 4 GB ごとに約 1MB が使用され、スラブごとに 1 MB ずつ使用します。
- 2 つ目の VDO メタデータでは、**論理ストレージ** 1GB ごとに約 1.25 MB が使用され、最も近いスラブに切り上げられます。
- UDS インデックスに必要なストレージの容量は、インデックスの種類と、インデックスに割り当てられている RAM の容量によって異なります。RAM 1GB ごとに、dense の UDS インデックスはストレージを 17 GB 使用し、sparse の UDS インデックスはストレージを 170 GB 使用します。

## 関連情報

- [Examples of VDO requirements by physical size](#)
- [VDO のスラブサイズ](#)

## 2.3. EXAMPLES OF VDO REQUIREMENTS BY PHYSICAL SIZE

以下の表は、基となるボリュームの物理サイズに基づいた、VDO のシステム要件の概算を示しています。それぞれの表には、プライマリストレージ、バックアップストレージなどの、目的のデプロイメントに適した要件が記載されています。

正確な数値は、VDO ボリュームの設定により異なります。

### プライマリストレージのデプロイメント

プライマリストレージの場合、UDS インデックスのサイズは、物理サイズの 0.01% から 25% になります。

表2.1 プライマリストレージのストレージ要件およびメモリー要件

物理サイズ	RAM の使用: UDS	RAM の使用: VDO	ディスク使用率	インデックスタイプ
10GB ~ 1TB	250MB	472MB	2.5GB	Dense
2 ~ 10TB	1GB	3GB	10GB	Dense
	250MB		22GB	Sparse
11 ~ 50TB	2GB	14GB	170GB	Sparse

物理サイズ	RAM の使用: UDS	RAM の使用: VDO	ディスク使用率	インデックスタイプ
51 ~ 100TB	3GB	27GB	255GB	Sparse
101 ~ 256TB	12GB	69GB	1020GB	Sparse

### バックアップストレージのデプロイメント

バックアップストレージの場合、UDS インデックスは、バックアップセットのサイズよりは大きくなりますが、物理サイズと同じか、より小さくなります。バックアップセットや物理サイズが今後大きくなる可能性がある場合は、これをインデックスサイズに組み込んでください。

表.2.2 バックアップストレージのストレージ要件およびメモリ要件

物理サイズ	RAM の使用: UDS	RAM の使用: VDO	ディスク使用率	インデックスタイプ
10GB ~ 1TB	250MB	472MB	2.5 GB	Dense
2 ~ 10TB	2GB	3GB	170GB	Sparse
11 ~ 50TB	10GB	14GB	850GB	Sparse
51 ~ 100TB	20GB	27GB	1700GB	Sparse
101 ~ 256TB	26GB	69GB	3400GB	Sparse

## 2.4. ストレージスタックの LVM-VDO の配置

VDO 論理ボリュームに特定のストレージ層を配置する必要があります。

シックプロビジョニングの層を VDO の上に配置することができますが、その場合はこのようなシックプロビジョニングの保証に頼ることはできません。VDO 層はシンプロビジョニングされているため、シンプロビジョニングの影響は、上記のすべての層に及びます。VDO ボリュームが監視されていない場合には、VDO 層の上にあるシックプロビジョニングのボリュームで、物理領域が不足する可能性があります。

以下の層は、VDO の配下にある場合にサポートされます。VDO 層の上に配置しないでください。

- DM Multipath
- DM Crypt
- ソフトウェア RAID (LVM または MD RAID)

以下の設定には **対応していません**。

- ループバックデバイスの上に VDO

- VDO の上に暗号化されたボリューム
- VDO ボリュームにパーティション作成
- VDO ボリュームの上に RAID (LVM RAID、MD RAID、またはその他のタイプ)
- LVM-VDO への Ceph ストレージのデプロイ

#### 関連情報

- [Stacking LVM volumes knowledgebase article](#)

## 第3章 重複排除した論理ボリュームと圧縮論理ボリュームの作成

VDO 機能を使用する LVM 論理ボリュームを作成して、データを重複排除して圧縮できます。

### 3.1. LVM-VDO デプロイメントシナリオ

LVM 上の VDO (LVM-VDO) は、さまざまな方法でデプロイして、以下に対して、重複排除したストレージを提供できます。

- ブロックアクセス
- ファイルアクセス
- ローカルストレージ
- リモートストレージ

LVM-VDO は、通常の論理ボリューム (LV) として重複排除したストレージを公開するため、そのストレージを標準ファイルシステム、iSCSI および FC のターゲットドライバー、または統合ストレージとして使用できます。

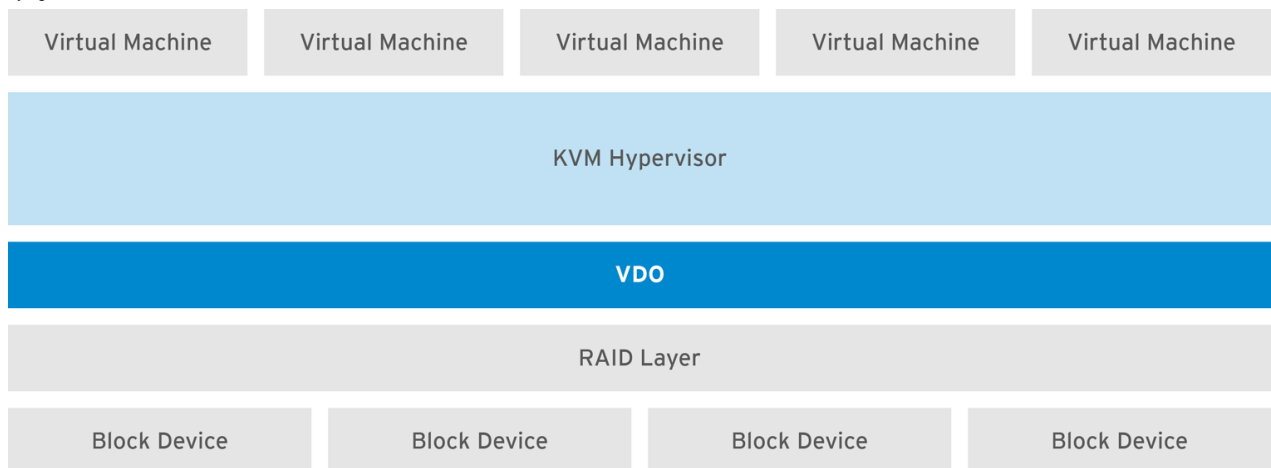


#### 注記

現在、LVM-VDO への Ceph Storage のデプロイメントはサポートされていません。

#### KVM

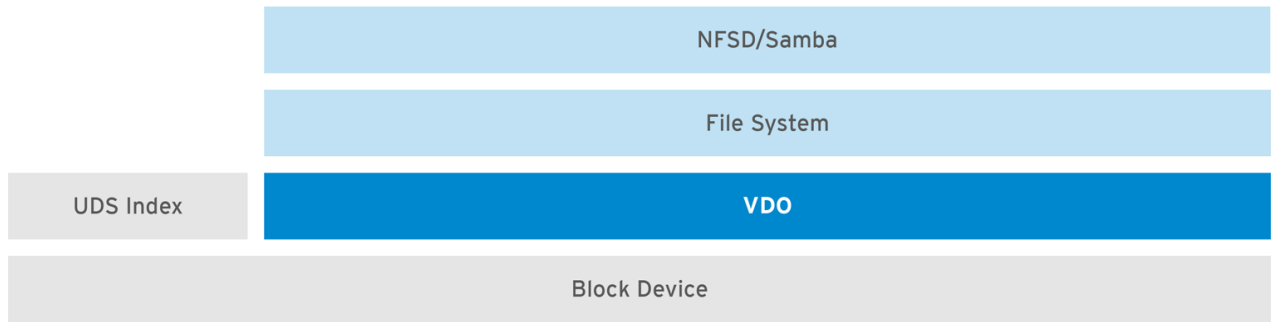
DAS (Direct Attached Storage) を使用して設定した KVM サーバーに LVM-VDO をデプロイできます。



RHEL\_462492\_1117

#### ファイルシステム

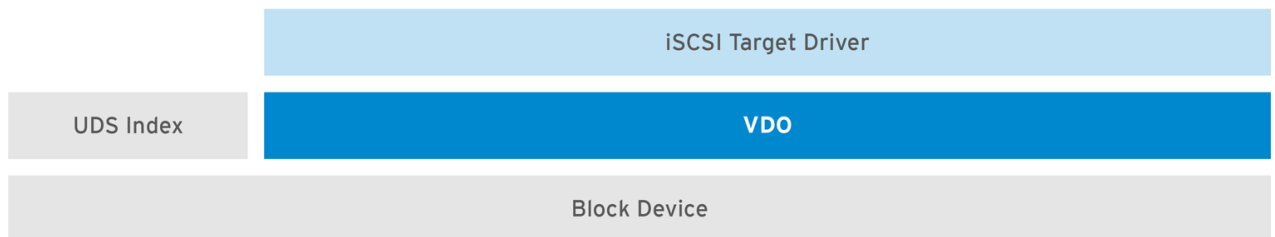
VDO LV にファイルシステムを作成して、NFS サーバーまたは Samba で、NFS ユーザーまたは CIFS ユーザーに公開します。



RHEL\_466924\_0218

### iSCSI ターゲット

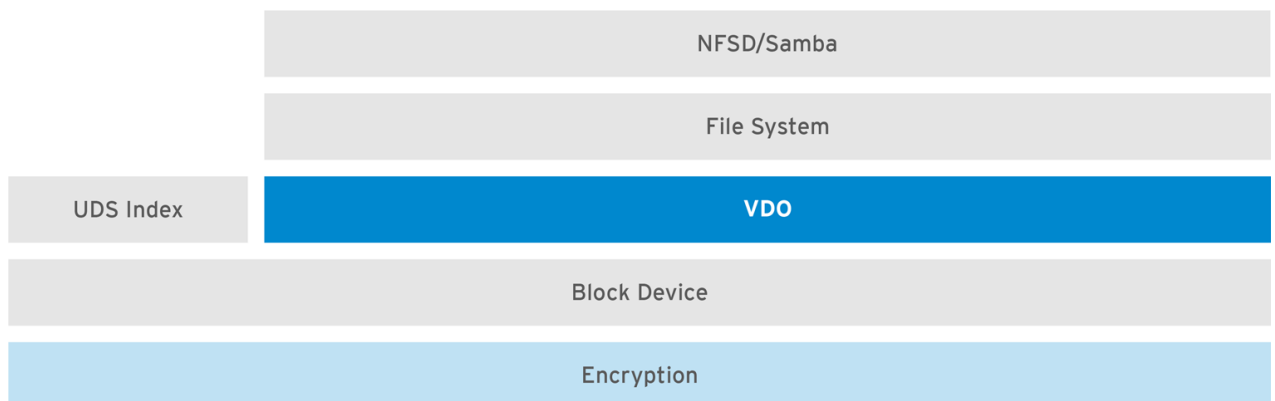
VDO LV 全体を、iSCSI ターゲットとしてリモート iSCSI イニシエーターにエクスポートできます。



RHEL\_466924\_0218

### 暗号化

DM Crypt などのデバスマッパー (DM) メカニズムは VDO と互換性があります。VDO LV ボリュームの暗号化により、データセキュリティと、VDO LV にある全ファイルシステムが重複排除されるようになります。



RHEL\_466924\_0218



#### 重要

VDO LV で暗号化層を適用すると、データの重複排除が行われてもほとんど行われません。暗号化により、VDO が重複を排除する前に、重複ブロックを変更します。

常に VDO LV の下に暗号化層を配置します。

## 3.2. LVM-VDO ボリュームの物理サイズおよび論理サイズ

このセクションでは、VDO が使用できる物理サイズ、利用可能な物理サイズ、論理サイズを説明します。

### 物理サイズ

これは、VDO プール LV に割り当てられた物理エクステントと同じサイズです。VDO は、以下の目的でこのストレージを使用します。

- 重複排除および圧縮される可能性があるユーザーデータ
- UDS インデックスなどの VDO メタデータ

### 利用可能な物理サイズ

これは、VDO がユーザーデータに使用できる物理サイズの一部です。

これは、物理サイズからメタデータのサイズを引いたものに相当し、スラブサイズの倍数に切り捨てられます。

### 論理サイズ

これは、VDO LV がアプリケーションに提示するプロビジョニングされたサイズです。通常、これは利用可能な物理サイズよりも大きくなります。VDO は現在、絶対最大論理サイズ 4PB の物理ボリュームの最大 254 倍の論理サイズに対応します。

VDO 論理ボリューム (LV) を設定する場合は、VDO LV が提示する論理ストレージのサイズを指定します。アクティブな仮想マシンまたはコンテナをホストする場合、Red Hat は、論理と物理の割合が 10 対 1 のストレージをプロビジョニングすることを推奨します。つまり、物理ストレージを 1 TB 使用している場合は、論理ストレージを 10 TB にします。

`--virtualsize` オプションを指定しないと、VDO はボリュームの比率を **1:1** にプロビジョニングします。たとえば、VDO LV を 20 GB の VDO プール LV 上に配置すると、VDO はデフォルトのインデックスサイズが使用されている場合に UDS インデックス用に 2.5 GB を確保します。残りの 17.5 GB は、VDO メタデータおよびユーザーデータに提供されます。そのため、消費する利用可能なストレージは 17.5 GB を超えません。実際の VDO ボリュームを構成するメタデータにより、これよりも少なくなる可能性があります。

### 関連情報

- [Examples of VDO requirements by physical size](#)

## 3.3. VDO のスラブサイズ

VDO ボリュームの物理ストレージは、複数のスラブに分割されます。各スラブは、物理領域の連続するリージョンです。指定のボリュームのスラブはすべて同じサイズになります。この値は、128 MB の 2 乗で、最大 32 GB にすることができます。

小規模なテストシステムで VDO を評価しやすくするため、デフォルトのスラブサイズは 2GB です。1 つの VDO ボリュームに最大 8192 個のスラブを持たせることができます。したがって、2 GB のスラブを使用したデフォルト設定では、許容される物理ストレージの最大値は 16 TB です。32 GB のスラブを使用する場合、許可される物理ストレージは最大 256 TB です。VDO は、メタデータ用に少なくともスラブ全体を予約するため、予約されたスラブはユーザーデータの格納には使用できません。

スラブサイズは、VDO ボリュームのパフォーマンスには影響しません。

表3.1物理ボリュームサイズによる推奨される VDO スラブサイズ



物理ボリュームのサイズ	推奨されるスラブサイズ
10-99 GB	1 GB
100 GB - 1TB	2 GB
2-256 TB	32 GB



### 注記

VDO ボリュームの最小ディスク使用量は、デフォルト設定のスラブサイズ 2 GB、密インデックス 0.25 を使用した場合、約 4.7 GB が必要です。これにより、0% の重複排除または圧縮で書き込むための 2 GB 弱の物理データが提供されます。

ここでの最小のディスク使用量は、デフォルトのスラブサイズと密インデックスの合計になります。

**lvcreate** コマンドに **--config 'allocation/vdo\_slab\_size\_mb=size-in-megabytes'** オプションを指定すると、スラブサイズを制御できます。

## 3.4. VDO のインストール

この手順では、VDO ボリュームの作成、マウント、および管理に必要なソフトウェアをインストールします。

### 手順

- **vdo** パッケージおよび **kmod-kvdo** パッケージをインストールします。

```
# dnf install vdo kmod-kvdo
```

## 3.5. LVM-VDO ボリュームの作成

この手順では、VDO プール LV に VDO 論理ボリューム (LV) を作成します。

### 前提条件

- VDO ソフトウェアをインストールしている。詳細は、[Installing VDO](#) を参照してください。
- 空きストレージ容量を持つ LVM ボリュームグループがシステムに存在する。

### 手順

1. **vdo1** などの VDO LV の名前を選択します。システムの VDO LV ごとに、異なる名前とデバイスを使用する必要があります。  
以下の手順では、**vdo-name** をその名前に置き換えます。
2. VDO LV を作成します。

```
# lvcreate --type vdo \  
--name vdo-name
```

```
--size physical-size
--virtualsize logical-size \
vg-name
```

- **vg-name** を、VDO LV を配置する既存の LVM ボリュームグループの名前に置き換えます。
- **logical-size** を、VDO LV が含まれる論理ストレージのサイズに置き換えます。
- 物理サイズが 16TiB を超える場合は、以下のオプションを指定して、ボリューム上のスラブサイズを 32GiB に増やします。

```
--config 'allocation/vdo_slab_size_mb=32768'
```

物理サイズが 16TiB を超えるデフォルトのスラブサイズ 2GiB を使用すると、**lvcreate** コマンドは以下のエラーで失敗します。

```
ERROR - vdoformat: formatVDO failed on '/dev/device': VDO Status: Exceeds maximum
number of slabs supported
```

### 例3.1 コンテナストレージ用の VDO LV の作成

たとえば、1TB の VDO プール LV で、コンテナストレージ用に VDO LV を作成するには、次のコマンドを実行します。

```
# lvcreate --type vdo \
  --name vdo1
  --size 1T
  --virtualsize 10T \
  vg-name
```



#### 重要

VDO ボリュームの作成中に問題が発生した場合は、ボリュームを削除します。

### 3. VDO LV にファイルシステムを作成します。

- XFS ファイルシステムの場合:

```
# mkfs.xfs -K /dev/vg-name/vdo-name
```

- ext4 ファイルシステムの場合:

```
# mkfs.ext4 -E nodiscard /dev/vg-name/vdo-name
```

#### 関連情報

- **lvmvdo(7)** の man ページ

## 3.6. LVM-VDO ボリュームのマウント

この手順では、手動で、または永続的に、LVM-VDO ボリュームにファイルシステムをマウントします。

#### 前提条件

- LVM-VDO ボリュームがシステム上にある。詳細は、[Creating an LVM-VDO volume](#) を参照してください。

#### 手順

- LVM-VDO ボリュームに手動でファイルシステムをマウントするには、以下のコマンドを使用します。

```
# mount /dev/vg-name/vdo-name mount-point
```

- 起動時にファイルシステムを自動的にマウントするように設定するには、`/etc/fstab` ファイルに以下の行を追加します。

- XFS ファイルシステムの場合:

```
/dev/vg-name/vdo-name mount-point xfs defaults 0 0
```

- ext4 ファイルシステムの場合:

```
/dev/vg-name/vdo-name mount-point ext4 defaults 0 0
```

LVM-VDO ボリュームが、iSCSI などのネットワークを必要とするブロックデバイスに配置されている場合は、`_netdev` マウントオプションを追加します。iSCSI や、ネットワークを必要とするその他のブロックデバイスの `_netdev` マウントオプションに関する情報は、man ページの `systemd.mount(5)` を参照してください。

#### 関連情報

- `systemd.mount(5)` man ページ

### 3.7. LVM-VDO ボリュームでの圧縮および重複排除設定の変更

この手順では、VDO プール論理ボリューム (LV) の圧縮および重複排除を有効または無効にします。



#### 注記

圧縮と重複排除はデフォルトで有効になっています。

#### 前提条件

- LVM-VDO ボリュームがシステム上にある。

#### 手順

1. 論理ボリュームで圧縮および重複排除が有効かどうかを調べるには、次のコマンドを実行します。

```
# lvs -o+vdo_compression,vdo_deduplication
```

- 稼働中の VDOPoolLV の重複排除インデックスと圧縮のステータスを確認します。

```
# lvs -o+vdo_compression_state,vdo_index_state
```

**vdo\_index\_state** は、**error**、**close**、**opening**、**closing**、**online** および **offline** として表示されます。

- VDOPoolLV の圧縮を有効または無効にするには、以下を行います。

```
# lvchange --compression y|n vg-name/vdopoolname
```

- VDOPoolLV の重複排除を有効または無効にするには、以下を行います。

```
# lvchange --deduplication y|n vg-name/vdopoolname
```

## 関連情報

- **lvmvdo(7)** の man ページ

## 3.8. 仮想データオプティマイザーを使用したシンプロビジョニングの管理

VDO ボリュームの物理領域の使用率が 100% に近づいている状況に対応するため、物理領域の将来の拡張に備えて、シンプロビジョニングされた VDO ボリュームを設定することができます。**lvcreate** オペレーションで **-l 100%FREE** を使用する代わりに、たとえば '95%FREE' を使用して、必要に応じて後で回復するための領域が確保されていることを確認します。この手順では、以下の問題を解決する方法を説明します。

- ボリュームの領域が不足しています。
- ファイルシステムが読み取り専用モードになります。
- ボリュームにより報告された ENOSPC



### 注記

VDO ボリュームで物理領域の高使用率に対処する最善の方法は、未使用のファイルを削除し、オンライン廃棄または **fstrim** を使用して、その未使用ファイルが使用しているブロックを破棄することです。VDO ボリュームの物理スペースは、デフォルトのスラブサイズが 2 GB の VDO ボリュームの場合は 16 TB、最大スラブサイズが 32 GB の VDO ボリュームの場合は 256 TB である 8192 スラブにのみ拡張できます。

以下のすべての手順で、**myvg** と **myvdo** を、それぞれボリュームグループ名および VDO 名に置き換えます。

## 前提条件

1. **lvm2** パッケージ、**vdo** パッケージ、および **kmod-kvdo** パッケージをインストールしている。
2. 空きストレージ容量を持つ LVM ボリュームグループがシステムに存在する。
3. **lvcreate --type vdo --name myvdo myvg -L logical-size-of-pool --virtualsize virtual-size-of-vdo** コマンドを使用した、シンプロビジョニングされた VDO ボリューム。詳細は、[Creating an LVM-VDO volume](#) を参照してください。

## 手順

1. シンプロビジョニングの VDO ボリュームに最適な論理サイズを判断します。

```
# vdostats myvg-vpool0-vpool
```

```
Device          1K-blocks Used   Available Use% Space saving%
myvg-vpool0-vpool 104856576 29664088 75192488 28% 69%
```

領域の節約率を計算するには、以下の式を使用します。

```
Savings ratio = 1 / (1 - Space saving%)
```

この例では、

- 約 80GB のデータセットでは、約 **3.22:1** 領域の削減率が得られます。
  - データセットのサイズにこの比率を乗算すると、同じ領域節約でより多くのデータが VDO ボリュームに書き込まれている場合に、論理サイズが 256GB になる可能性があります。
  - この数を 200GB まで下げると、同じ領域節約率の場合、物理領域の空き領域に安全なマージンがある論理サイズになります。
2. VDO ボリュームの空き物理領域を監視します。

```
# vdostats myvg-vpool0-vpool
```

このコマンドは定期的に行うことで、VDO ボリュームの使用済み物理領域および空き物理領域の監視を行うことができます。

3. オプション:利用可能な

`/usr/share/doc/vdo/examples/monitor/monitor_check_vdostats_physicalSpace.pl` スクリプトを使用して、VDO ボリュームの物理領域の使用状況に関する警告を表示します。

```
# /usr/share/doc/vdo/examples/monitor/monitor_check_vdostats_physicalSpace.pl myvg-vpool0-vpool
```

4. VDO ボリュームを作成すると、`dmeventd` 監視サービスは、VDO ボリュームの物理領域の使用を監視します。VDO ボリュームを作成または起動すると、デフォルトで有効になります。VDO ボリュームの監視中に、`journalctl` コマンドを使用してログに `dmeventd` の出力を表示します。

```
lvm[8331]: Monitoring VDO pool myvg-vpool0-vpool.
```

```
...
```

```
lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 84.63% full.
```

```
lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 91.01% full.
```

```
lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 97.34% full.
```

5. 利用可能な物理領域がほぼ不足している VDO ボリュームを修正します。VDO ボリュームに物理領域を追加できても、ボリュームを拡張する前に領域がいっぱいになった場合は、そのボリュームへの I/O を一時的に停止する必要があるかもしれません。ボリュームへの I/O を一時的に停止するには、以下の手順を実行します。ここで、VDO ボリューム `myvdo` には、`/users/homeDir` パスにマウントされたファイルシステムが含まれます。

- a. ファイルシステムをフリーズします。

```
# xfs_freeze -f /users/homeDir

# vgextend myvg /dev/vdc2

# lvextend -l new_size myvg/vpool0-name

# xfs_freeze -u /users/homeDir
```

- b. ファイルシステムをアンマウントします。

```
# umount /users/homeDir

# vgextend myvg /dev/vdc2

# lvextend -l new_size myvg/vpool0-name

# mount -o discard /dev/myvg/myvdo /users/homeDir
```



### 注記

キャッシュされたデータでファイルシステムをアンマウントまたはフリーズすると、キャッシュされたデータの書き込みが発生します。これにより、VDO ボリュームの物理領域がいっぱいになる可能性があります。VDO ボリュームの空き物理領域の監視しきい値を設定する場合は、ファイルシステムのデータキャッシュの最大量を考慮してください。

6. ファイルシステムが使用しなくなったブロックは、**fstrim** ユーティリティを使用してクリーンアップできます。VDO ボリュームにマウントしたファイルシステムに対して **fstrim** を実行すると、そのボリュームの空き物理領域が増える可能性があります。**fstrim** ユーティリティは、VDO ボリュームに破棄を送信します。これは、以前使用したブロックへの参照を削除するのに使用されます。これらのブロックのいずれかが単一参照されている場合は、物理領域が使用可能になります。

- a. VDO の統計を確認して、現在の空き領域の量を確認します。

```
# vdostats --human-readable myvg-vpool0-vpool

Device      Size Used Available Use% Space saving%
myvg-vpool0-vpool 100.0G 95.0G 5.0G 95% 73%
```

- b. 未使用ブロックの破棄

```
# fstrim /users/homeDir
```

- c. VDO ボリュームの空き物理領域を表示します。

```
# vdostats --human-readable myvg-vpool0-vpool

Device      Size Used Available Use% Space saving%
myvg-vpool0-vpool 100.0G 30.0G 70.0G 30% 43%
```

この例では、ファイルシステムで **fstrim** を実行した後、破棄により VDO ボリュームで使用する物理領域 65G が返されました。



### 注記

重複排除と圧縮のレベルが低いボリュームを破棄すると、重複排除と圧縮のレベルが高いボリュームを破棄するよりも、物理領域を確保できるようになります。重複排除と圧縮のレベルが高いボリュームでは、単に未使用のブロックを破棄する場合よりも、物理領域を確保するためにより詳細なクリーンアップが必要になる可能性があります。

## 第4章 LVM-VDO ボリュームの縮小オプション

ファイルシステムは、**discard** オプションでマウントできます。これにより、VDO ボリュームに、未使用領域が通知されます。もう1つの方法として、**fstrim** アプリケーション (オンデマンドでの破棄) または **mount -o discard** コマンドを実行して即座に破棄します。

**fstrim** アプリケーションを使用する場合には、管理者は追加のプロセスをスケジュールおよび監視する必要がありますが、**mount -o discard** コマンドを使用すると、可能な場合には、領域の即時に復元できます。

このオプションのパフォーマンスへの影響は非常に大きく可能性があるため、**fstrim** アプリケーションを使用する場合は **discard** マウントオプションではなく未使用のブロックを破棄することが推奨されます。このため、**nodiscard** がデフォルトです。

### 4.1. VDO で DISCARD マウントオプションの有効化

この手順では、VDO ボリュームで **discard** オプションを有効にします。

#### 前提条件

- LVM-VDO ボリュームがシステム上にある。

#### 手順

- ボリュームで **discard** を有効にします。

```
# mount -o discard /dev/vg-name/vdo-name mount-point
```

#### 関連情報

- man ページの **xfstools(5)**、**mount(8)**、および **lvmvdo(7)**

### 4.2. 定期的な TRIM 操作の設定

この手順では、システムで TRIM 操作のスケジューリングを有効にします。

#### 前提条件

- LVM-VDO ボリュームがシステム上にある。

#### 手順

- タイマーを有効にして起動します。

```
# systemctl enable --now fstrim.timer
```

#### 検証

- タイマーが有効化されていることを確認します。

```
# systemctl list-timers fstrim.timer
```



#### 例4.1 検証手順での出力候補

```
# systemctl list-timers fstrim.timer
NEXT          LEFT    LAST PASSED UNIT    ACTIVATES
Mon 2021-05-10 00:00:00 EDT 5 days left n/a  n/a    fstrim.timer fstrim.service
```



#### 注記

**fstrim.timer** は、マウントされている全ファイルシステムで実行されるため、VDO ボリュームへの参照は表示されません。

#### 関連情報

- **fstrim(8)** の man ページ