# Red Hat Enterprise Linux 8

## RHEL에서 논리 볼륨 중복 제거 및 압축

LVM에 VDO를 배포하여 스토리지 용량 증가

Last Updated: 2023-09-26

# Red Hat Enterprise Linux 8 RHEL에서 논리 볼륨 중복 제거 및 압축

LVM에 VDO를 배포하여 스토리지 용량 증가

## 법적 공지

Copyright © 2023 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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® 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 ® 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.

## 초록

LVM(Logical Volume Manager)의 VDO(Virtual Data Optimizer) 기능을 사용하여 중복 및 압축된 논리 볼륨을 관리합니다. VDO는 LVM 씬 프로비저닝 볼륨과 유사하게 LVM의 논리 볼륨(LV) 유형으로 관리할 수 있습니다. LVM(LVM-VDO)에 VDO를 배포하여 블록 액세스, 파일 액세스, 로컬 스토리지 및 원격 스토리지에 중복된 스토리지를 제공할 수 있습니다. 100%가 사용되는 VDO 볼륨의 물리적 공간을 피하기 위해 씬 프로비저닝된 VDO 볼륨을 구성할 수도 있습니다. VDO 볼륨을 LVM으로 가져온 후 LVM 도구를 사용하여 VDO 볼륨을 관리할 수 있습니다.

# 차례

# 보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서 및 웹 속성에서 문제가 있는 언어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

# RED HAT 문서에 관한 피드백 제공

문서에 대한 피드백에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

**특정 문구에 대한 의견 제출**

1. **Multi-page HTML** 형식으로 설명서를 보고 페이지가 완전히 로드된 후 오른쪽 상단 모서리에 **피드백** 버튼이 표시되는지 확인합니다.
2. 커서를 사용하여 주석 처리할 텍스트 부분을 강조 표시합니다.
3. 강조 표시된 텍스트 옆에 표시되는 **피드백 추가** 버튼을 클릭합니다.
4. 의견을 추가하고 **제출** 을 클릭합니다.

**Jira를 통해 피드백 제출 (등록 필요)**

1. Jira 웹 사이트에 로그인합니다.
2. 상단 탐색 모음에서 **생성** 을 클릭합니다.
3. **Summary** (요약) 필드에 설명 제목을 입력합니다.
4. **Description** (설명) 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

# 1장. LVM의 VDO 소개

VDO(가상 데이터 최적화 도구) 기능은 스토리지를 위한 인라인 블록 수준 중복 제거, 압축 및 씬 프로비저닝을 제공합니다. LVM 씬 프로비저닝 볼륨과 유사하게 VDO를 LVM(Logical Volume Manager) 논리 볼륨(LV) 유형으로 관리할 수 있습니다.

LVM(LVM-VDO)의 VDO 볼륨에는 다음 구성 요소가 포함됩니다.

**VDO pool LV**

- 이는 VDO LV의 데이터를 저장, 중복 제거 및 압축하는 백업 물리적 장치입니다. VDO 풀 LV는 VDO 볼륨의 물리적 크기를 설정합니다. 이 볼륨은 VDO 디스크에 저장할 수 있는 데이터 양입니다.
- 현재 각 VDO pool LV는 하나의 VDO LV만 보유할 수 있습니다. 결과적으로 VDO는 각 VDO LV를 별도로 중복하고 압축합니다. 별도의 LV에 저장된 중복 데이터는 동일한 VDO 볼륨의 데이터 최적화의 이점을 누릴 수 없습니다.

**VDO LV**

- VDO 풀 LV 상단에 프로비저닝된 가상 장치입니다. VDO LV는 VDO 볼륨의 프로비저닝된 논리 크기를 설정합니다. 이 볼륨은 중복 제거 및 압축이 발생하기 전에 애플리케이션에 쓸 수 있는 데이터 양입니다.

LVM 씬 프로비저닝 구현의 구조에 이미 익숙한 경우 Table 1.1을 참조하여 VDO의 다양한 측면이 시스템에 어떻게 제공되는지 이해할 수 있습니다.

**표 1.1. LVM 및 LVM 씬 프로비저닝의 VDO의 구성 요소 비교**

| | 물리적 장치 | 프로비저닝된 장치 |
|---|---|---|
| LVM의 VDO | VDO pool LV | VDO LV |
| LVM 씬 프로비저닝 | 씬 풀 | 씬 볼륨 |

VDO는 씬 프로비저닝되므로 파일 시스템과 애플리케이션은 실제 사용 가능한 물리적 공간이 아니라 사용 중인 논리 공간만 볼 수 있습니다. 스크립팅을 사용하여 사용 가능한 물리적 공간을 모니터링하고 사용량이 임계값을 초과하는 경우 경고를 생성합니다. 사용 가능한 VDO 공간 모니터링에 대한 자세한 내용은 Monitoring VDO 섹션을 참조하십시오.

**추가 리소스**

- 스토리지 중복 제거 및 압축
- 씬 프로비저닝 볼륨 생성 및 관리(thin volumes)

# 2장. LVM-VDO 요구 사항

LVM의 VDO에는 배치 및 시스템 리소스에 대한 특정 요구 사항이 있습니다.

## 2.1. VDO 메모리 요구 사항

각 VDO 볼륨에는 두 개의 고유한 메모리 요구 사항이 있습니다.

**VDO 모듈**

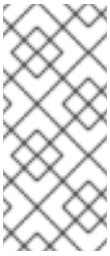
VDO에는 고정된 38MB의 RAM과 몇 가지 변수 양이 필요합니다.

- 구성된 블록 맵 캐시 크기 1MB당 1.15MB의 RAM. 블록 맵 캐시에는 최소 150MB의 RAM이 필요합니다.
- 논리 공간 1TB당 1.6MB의 RAM.
- 볼륨에서 관리하는 물리적 스토리지의 1TB당 268MB의 RAM.

**UDS 인덱스**

UDS(Universal Deduplication Service)에는 최소 250MB의 RAM이 필요합니다. 이는 중복 제거에서 사용하는 기본 양이기도 합니다. 이 값은 인덱스에 필요한 스토리지 크기에도 영향을 미치므로 VDO 볼륨을 포맷할 때 값을 구성할 수 있습니다.
UDS 인덱스에 필요한 메모리는 인덱스 유형 및 중복 제거 창의 필수 크기에 따라 결정됩니다.

| 인덱스 유형 | 중복 제거 창 | 참고 |
|---|---|---|
| 밀도 | 1GB RAM당 1TB | 1GB 밀도 지수는 일반적으로 최대 4TB의 물리적 스토리지에 충분합니다. |
| 스파스 | 1GB RAM당 10TB | 일반적으로 1GB 스파스 인덱스는 최대 40TB의 물리적 스토리지에 충분합니다. |



**참고**

기본 설정 2GB slab 크기와 0.25 밀도 인덱스를 사용하는 VDO 볼륨의 최소 디스크 사용량에는 approx 4.7GB가 필요합니다. 이는 0% 중복 제거 또는 압축에서 쓸 수 있는 2GB의 물리적 데이터보다 약간 적습니다.

여기에서 최소 디스크 사용은 기본 slab 크기 및 dense 인덱스의 합계입니다.

UDS Sparse Indexing 기능은 VDO에 권장되는 모드입니다. 데이터의 시간적 로컬성에 의존하고 메모리에서 가장 관련성이 높은 인덱스 항목만 유지하려고 합니다. 스파스 인덱스를 사용하면 UDS는 동일한 양의 메모리를 사용하는 동시에 밀도가 있는 것보다 10배 큰 중복 제거 창을 유지할 수 있습니다.

스파스 인덱스가 가장 큰 범위를 제공하지만, 밀도가 더 많은 중복 제거 조언을 제공합니다. 동일한 양의 메모리에 대해 대부분의 워크로드에서 밀도 및 스파스 인덱스 간의 중복 제거 비율의 차이는 무시할 수 있습니다.

**추가 리소스**

- 물리적 크기 VDO 요구 사항의 예

## 2.2. VDO 스토리지 공간 요구 사항

최대 256TB의 물리 스토리지를 사용하도록 VDO 볼륨을 구성할 수 있습니다. 물리적 스토리지의 특정 부분만 데이터를 저장하는 데 사용할 수 있습니다. 이 섹션에서는 VDO 관리 볼륨의 사용 가능한 크기를 결정하는 계산 방법을 제공합니다.

VDO는 두 가지 유형의 VDO 메타데이터 및 UDS 인덱스에 대한 스토리지가 필요합니다.

- 첫 번째 유형의 VDO 메타데이터는 약 1MB의 *물리적 스토리지와* slab당 1MB를 추가로 사용합니다.
- 두 번째 유형의 VDO 메타데이터는 1GB의 *논리 스토리지* 마다 약 1.25MB를 사용하고 가장 가까운 슬랩으로 반올림합니다.
- UDS 인덱스에 필요한 스토리지 양은 인덱스 유형 및 인덱스에 할당된 RAM 크기에 따라 달라집니다. 1GB RAM마다 밀도가 높은 UDS 인덱스는 17GB의 스토리지를 사용하며 스파스 UDS 인덱스는 170GB의 스토리지를 사용합니다.

**추가 리소스**

- 물리적 크기 VDO 요구 사항의 예
- VDO의 Slab 크기

## 2.3. 물리적 크기 VDO 요구 사항의 예

다음 표에서는 기본 볼륨의 물리적 크기에 따라 VDO의 대략적인 시스템 요구 사항을 제공합니다. 각 테이블에는 기본 스토리지 또는 백업 스토리지와 같이 의도한 배포에 적합한 요구 사항이 나열되어 있습니다.

정확한 숫자는 VDO 볼륨 구성에 따라 다릅니다.

**기본 스토리지 배포**

1차 스토리지의 경우 UDS 인덱스는 물리적 크기의 크기에서 25% 사이입니다.

**표 2.1. 기본 스토리지에 대한 스토리지 및 메모리 요구 사항**

| 물리 크기 | RAM 사용량: UDS | RAM 사용량: VDO | 디스크 사용량 | 인덱스 유형 |
|---|---|---|---|---|
| 10GB–1TB | 250MB | 472MB | 2.5GB | 밀도 |
| 2–10TB | 1GB | 3GB | 10GB | 밀도 |
| | 250MB | | 22GB | 스파스 |
| 11–50TB | 2GB | 14GB | 170GB | 스파스 |
| 51–100TB | 3GB | 27GB | 255GB | 스파스 |
| 101–256TB | 12GB | 69GB | 1020GB | 스파스 |

**백업 스토리지 배포**

백업 스토리지의 경우 UDS 인덱스는 백업 세트의 크기를 처리하지만 실제 크기보다 크지는 않습니다. 향후 백업 세트 또는 물리적 크기가 증가할 것으로 예상되는 경우 인덱스 크기로 고려합니다.

**표 2.2. 백업 스토리지에 대한 스토리지 및 메모리 요구 사항**

| 물리 크기 | RAM 사용량: UDS | RAM 사용량: VDO | 디스크 사용량 | 인덱스 유형 |
|---|---|---|---|---|
| 10GB-1TB | 250MB | 472MB | 2.5GB | 밀도 |
| 2-10TB | 2GB | 3GB | 170GB | 스파스 |
| 11-50TB | 10GB | 14GB | 850GB | 스파스 |
| 51-100TB | 20GB | 27GB | 1700GB | 스파스 |
| 101-256TB | 26GB | 69GB | 3400GB | 스파스 |

## 2.4. 스토리지 스택에 LVM-VDO 배치

VDO 논리 볼륨 아래에 특정 스토리지 계층을 배치하고 그 위의 다른 스토리지 계층을 배치해야 합니다.

VDO 상단에 씩 프로비저닝된 계층을 배치할 수 있지만 이 경우 씩 프로비저닝 보장을 사용할 수 없습니다. VDO 계층은 씬 프로비저닝되므로 씬 프로비저닝의 영향은 위의 모든 계층에 적용됩니다. VDO 볼륨을 모니터링하지 않으면 VDO보다 씩 프로비저닝된 볼륨의 물리 공간이 부족할 수 있습니다.

다음 계층에서 지원되는 배치는 VDO 아래에 있습니다. VDO 위에 배치하지 마십시오.

- DM Multipath
- DM 암호화
- 소프트웨어 RAID(LVM 또는 MD RAID)

다음 구성은 **지원되지 않습니다**.

- 루프백 장치 상단에 있는 VDO
- VDO 상단에 암호화된 볼륨
- VDO 볼륨의 파티션
- RAID(예: LVM RAID, MD RAID 또는 VDO 볼륨 상단에 있는 기타 유형)
- LVM-VDO에 Ceph Storage 배포

**추가 리소스**

- LVM 볼륨 지식 베이스 스택

# 3장. 중복 제거 및 압축된 논리 볼륨 생성

VDO 기능을 사용하여 데이터를 삭제하고 압축하는 LVM 논리 볼륨을 생성할 수 있습니다.

## 3.1. LVM-VDO 배포 시나리오

다음과 같은 다양한 방법으로 LVM(LVM-VDO)에 VDO를 배포할 수 있습니다.

- 블록 액세스
- 파일 액세스
- 로컬 스토리지
- 원격 스토리지

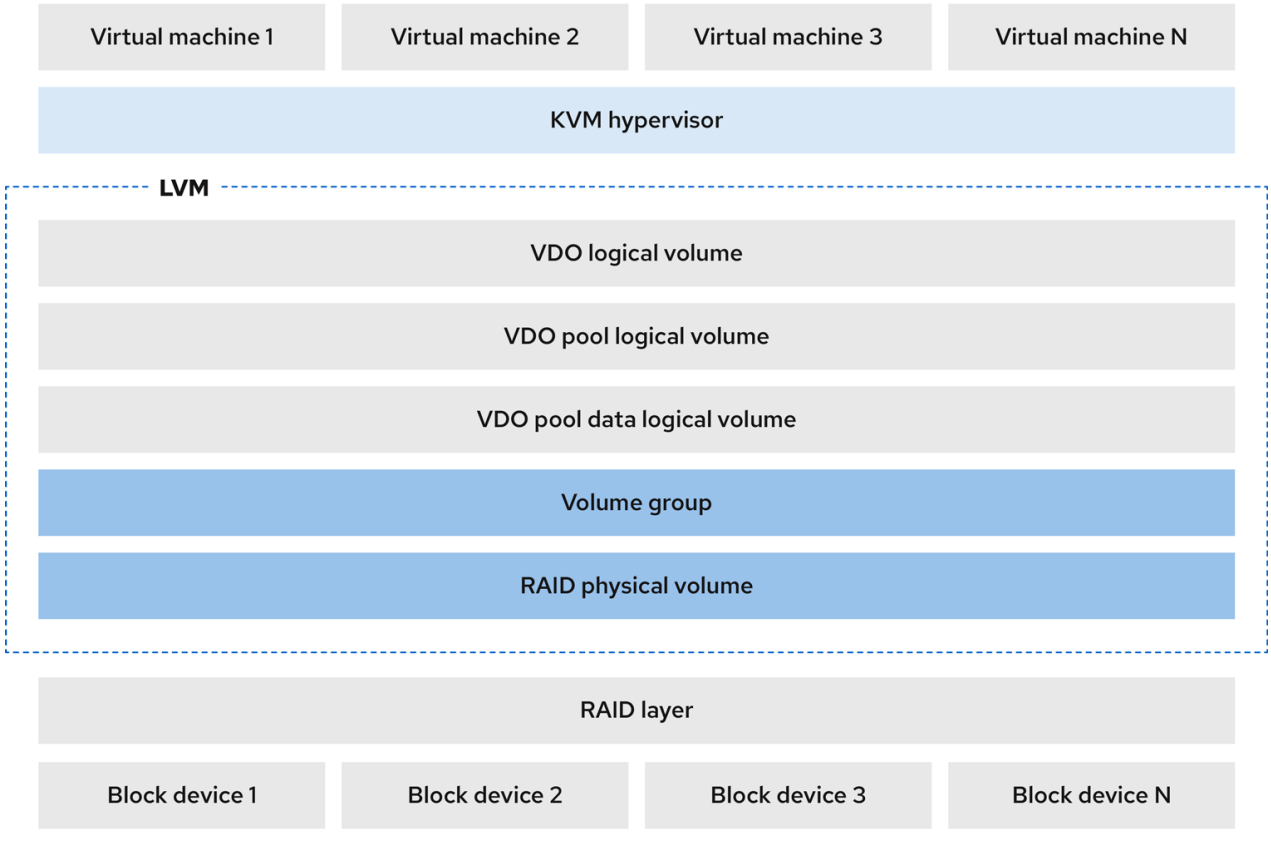
LVM-VDO는 중복 제거 스토리지를 일반 논리 볼륨(LV)으로 노출하므로 표준 파일 시스템, iSCSI 및 FC 대상 드라이버 또는 통합 스토리지와 함께 사용할 수 있습니다.

**참고**

현재 LVM-VDO에 Ceph Storage를 배포하는 것은 지원되지 않습니다.

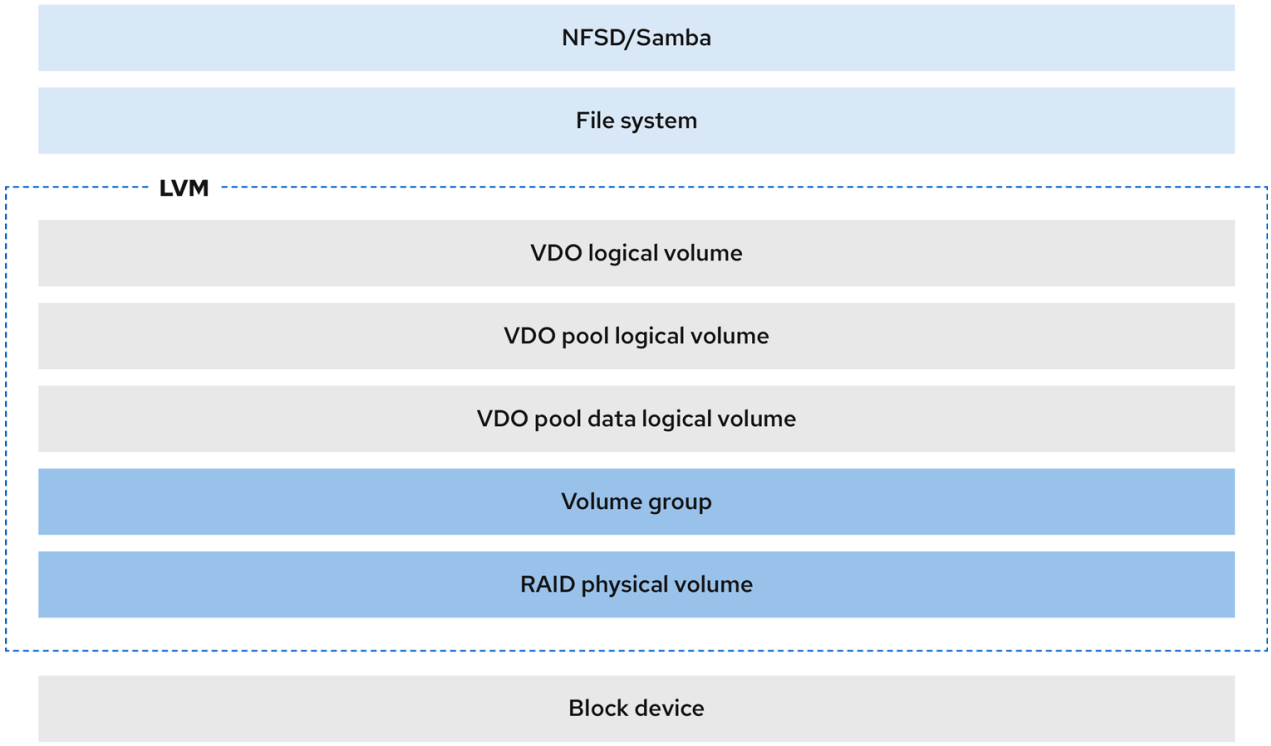
**KVM**

직접 연결된 스토리지로 구성된 KVM 서버에 LVM-VDO를 배포할 수 있습니다.



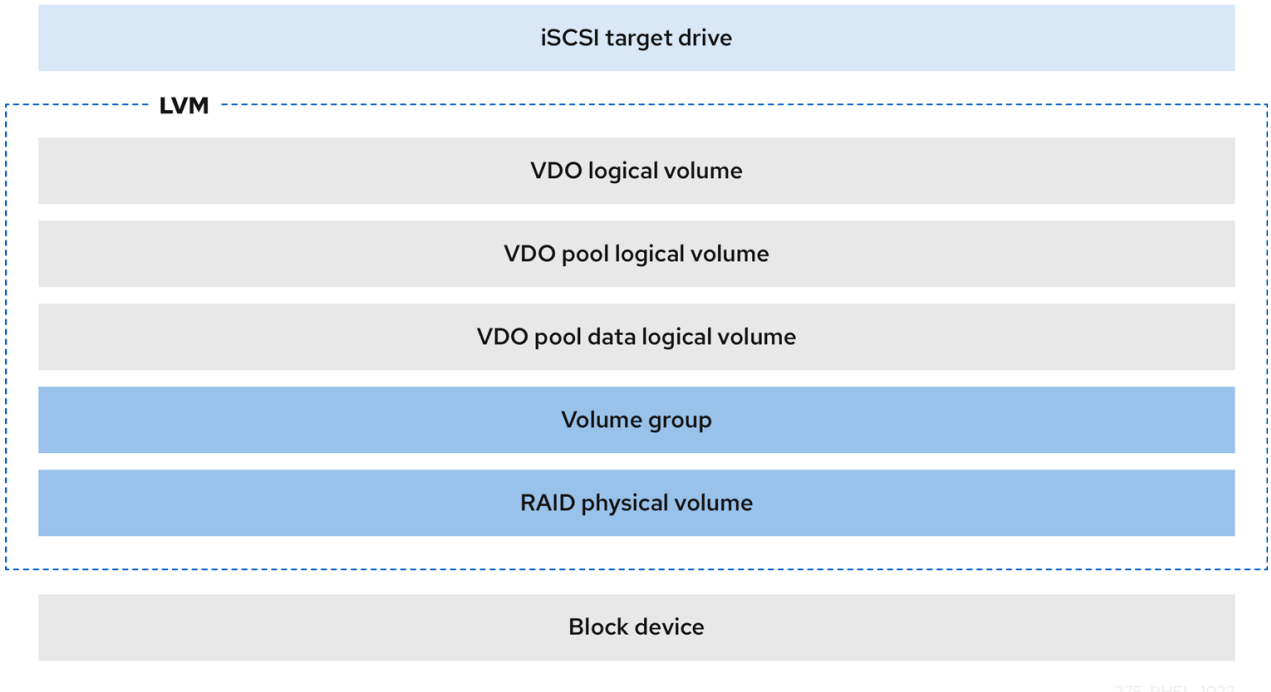

**파일 시스템**

VDO LV 상단에 파일 시스템을 생성하고 NFS 서버 또는 Samba를 사용하여 NFS 또는 CIFS 사용자에게 노출할 수 있습니다.



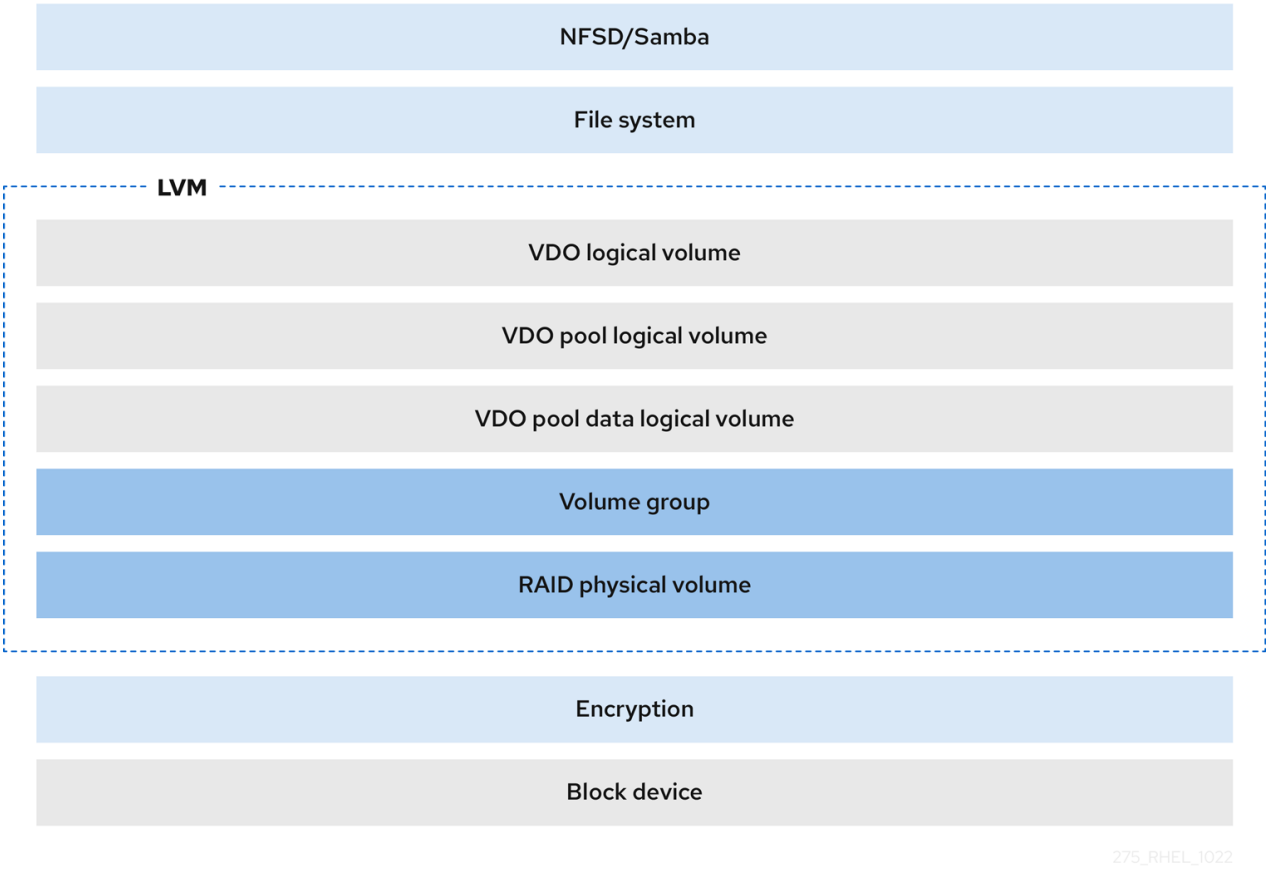

### iSCSI 대상

VDO LV 전체를 iSCSI 대상으로 원격 iSCSI 이니시에이터로 내보낼 수 있습니다.




### Encryption

DM Crypt와 같은 DM(Device Mapper) 메커니즘은 VDO와 호환됩니다. VDO LV 볼륨을 암호화하면 데이터 보안이 보장되고 VDO LV 위의 파일 시스템은 여전히 중복 제거됩니다.




**중요**

VDO LV 위의 암호화 계층을 적용하면 데이터 중복 제거가 거의 없습니다. VDO에서 중복을 제거하기 전에 암호화를 사용하면 중복 블록이 서로 다릅니다.

항상 VDO LV 아래에 암호화 계층을 배치합니다.

## 3.2. LVM-VDO 볼륨의 물리 및 논리적 크기

이 섹션에서는 VDO가 활용할 수 있는 물리적 크기, 사용 가능한 물리적 크기 및 VDO에 대해 설명합니다.

**물리 크기**

이 크기는 VDO 풀 LV에 할당된 물리 확장 영역과 동일합니다. VDO는 다음을 위해 이 스토리지를 사용합니다.

- 중복 제거 및 압축 가능한 사용자 데이터
- UDS 인덱스와 같은 VDO 메타데이터

**사용 가능한 물리 크기**

이것은 VDO가 사용자 데이터에 사용할 수 있는 물리적 크기의 부분입니다.
실제 크기와 동일한 값으로 반올림되며 slab 크기의 배수로 반올림됩니다.

**논리 크기**

이는 VDO LV가 애플리케이션에 제공하는 프로비저닝된 크기입니다. 일반적으로 사용 가능한 실제 크기보다 큽니다. VDO는 현재 절대 최대 논리 크기가 4 PB인 물리 볼륨의 최대 254배의 논리 크기를 지원합니다.
VDO LV(논리 볼륨)를 설정할 때 VDO LV에서 제공하는 논리 스토리지 크기를 지정합니다. 활성 VM 또는 컨테이너를 호스팅할 때, Red Hat은 10:1 논리적 비율과 물리적인 비율로 스토리지를 프로비저닝하는 것이 좋습니다. 즉, 1TB의 물리적 스토리지를 사용하는 경우 10TB의 논리적 스토리지가 됩니다.

virtual **size** 옵션을 지정하지 않으면 VDO는 볼륨을 **1:1** 비율로 프로비저닝합니다. 예를 들어, 20GB VDO 풀 LV 상단에 VDO LV를 배치하면 기본 인덱스 크기가 사용되는 경우 VDO는 UDS 인덱스용으로 2.5GB를 예약합니다. 나머지 17.5GB는 VDO 메타데이터 및 사용자 데이터에 대해 제공됩니다. 따라서 사용할 수 있는 스토리지가 17.5GB를 넘지 않으며 실제 VDO 볼륨을 구성하는 메타데이터로 인해 더 적을 수 있습니다.

**추가 리소스**

- 물리적 크기 VDO 요구 사항의 예

## 3.3. VDO의 슬랩 크기

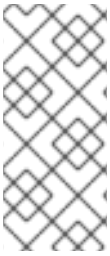
VDO 볼륨의 물리 스토리지는 여러 슬랩으로 나뉩니다. 각 조각은 물리적 공간의 인접 지역입니다. 지정된 볼륨에 대한 모든 슬래브는 크기가 같으며 2의 배수 128MB 최대 32GB의 거듭제곱이 될 수 있습니다.

기본 slab 크기는 작은 테스트 시스템에서 VDO를 쉽게 평가할 수 있도록 2GB입니다. 단일 VDO 볼륨은 최대 8192 개의 슬래브를 보유할 수 있습니다. 따라서 2GB의 slab이 있는 기본 구성에서 허용되는 최대 물리적 스토리지는 16TB입니다. 32GB slab을 사용하는 경우 허용되는 최대 물리 스토리지는 256TB입니다. VDO는 항상 메타데이터를 위해 최소 하나의 slab을 예약하므로 예약된 슬랩을 사용자 데이터를 저장하는 데 사용할 수 없습니다.

Slab 크기는 VDO 볼륨의 성능에 영향을 미치지 않습니다.

**표 3.1. 물리 볼륨 크기 별 권장 VDO 슬래브 크기**

| 물리 볼륨 크기 | 권장 슬랩 크기 |
|---|---|
| 10~99GB | 1GB |
| 100GB - 1TB | 2GB |
| 2-256 TB | 32GB |



**참고**

기본 설정 2GB slab 크기와 0.25 밀도 인덱스를 사용하는 VDO 볼륨의 최소 디스크 사용량에는 approx 4.7GB가 필요합니다. 이는 0% 중복 제거 또는 압축에서 쓸 수 있는 2GB의 물리적 데이터보다 약간 적습니다.

여기에서 최소 디스크 사용은 기본 slab 크기 및 dense 인덱스의 합계입니다.

**lvcreate** 명령에 **--config 'allocation/vdo_slab_size_mb=*size-in-megabytes*'** 옵션을 제공하여 slab 크기를 제어할 수 있습니다.

## 3.4. VDO 설치

이 절차에서는 VDO 볼륨을 생성, 마운트 및 관리하는 데 필요한 소프트웨어를 설치합니다.

**절차**

- VDO 소프트웨어를 설치합니다.

```
# yum install lvm2 kmod-kvdo vdo
```

## 3.5. LVM-VDO 볼륨 생성

이 절차에서는 VDO 풀 LV에 VDO 논리 볼륨(LV)을 생성합니다.

**사전 요구 사항**

- VDO 소프트웨어를 설치합니다. 자세한 내용은 VDO 설치를 참조하십시오.
- 사용 가능한 스토리지 용량이 있는 LVM 볼륨 그룹이 시스템에 있습니다.

**절차**

1. VDO LV의 이름을 선택합니다(예: **vdo1).** 시스템에서 각 VDO LV에 다른 이름 및 장치를 사용해야 합니다.
   다음 모든 단계에서 *vdo-name* 을 이름으로 바꿉니다.
2. VDO LV를 생성합니다.

```
# lvcreate --type vdo \
        --name vdo-name
        --size physical-size
        --virtualsize logical-size \
        vg-name
```

   - *vg-name* 을 VDO LV를 배치하려는 기존 LVM 볼륨 그룹의 이름으로 바꿉니다.
   - *logical-size* 를 VDO LV가 제공하는 논리 스토리지 양으로 바꿉니다.
   - 물리적 크기가 16TiB보다 크면 다음 옵션을 추가하여 볼륨의 slab 크기를 32GiB로 늘립니다.

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

     16TiB보다 큰 실제 크기에서 2GiB의 기본 slab 크기를 사용하는 경우 **lvcreate** 명령이 실패하고 다음 오류가 발생합니다.

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

**예 3.1. 컨테이너 스토리지를 위한 VDO LV 생성**

예를 들어 1TB VDO 풀 LV에서 컨테이너 스토리지용 VDO LV를 생성하려면 다음을 사용합니다.

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

**중요**

VDO 볼륨을 생성할 때 오류가 발생하면 볼륨을 제거하여 정리합니다.

3. VDO LV에 파일 시스템을 생성합니다.
   - XFS 파일 시스템의 경우:

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

   - ext4 파일 시스템의 경우:

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

**추가 리소스**

- **lvmvdo(7)** 도움말 페이지

## 3.6. LVM-VDO 볼륨 마운트

이 절차에서는 LVM-VDO 볼륨에 파일 시스템을 수동으로 또는 영구적으로 마운트합니다.

**사전 요구 사항**

- LVM-VDO 볼륨이 시스템에 있습니다. 자세한 내용은 LVM-VDO 볼륨 생성을 참조하십시오.

**절차**

- LVM-VDO 볼륨에 파일 시스템을 수동으로 마운트하려면 다음을 사용합니다.

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

- 부팅 시 자동으로 마운트되도록 파일 시스템을 구성하려면 **/etc/fstab** 파일에 행을 추가합니다.
  - XFS 파일 시스템의 경우:

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

  - ext4 파일 시스템의 경우:

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

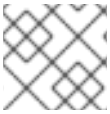
LVM-VDO 볼륨이 iSCSI와 같은 네트워크가 필요한 블록 장치에 있는 경우 **_netdev** 마운트 옵션을 추가합니다. 네트워크가 필요한 iSCSI 및 기타 블록 장치의 경우 **_netdev** 마운트 옵션에 대한 정보는 **systemd.mount(5)** 도움말 페이지를 참조하십시오.

**추가 리소스**

- **systemd.mount(5)** 도움말 페이지

## 3.7. LVM-VDO 볼륨에서 압축 및 중복 제거 설정 변경

이 절차에서는 VDO 풀 LV(논리 볼륨)의 압축 및 중복 제거를 활성화하거나 비활성화합니다.



**참고**

압축 및 중복 제거는 기본적으로 활성화됩니다.

**사전 요구 사항**

- LVM-VDO 볼륨이 시스템에 있습니다.

**절차**

1. 압축 및 중복 제거가 논리 볼륨에서 활성화되었는지 확인하려면 다음을 수행하십시오.

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

2. 실행 중인 활성 VDOPoolLV의 중복 제거 인덱스의 압축 및 상태를 확인합니다.

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

**vdo_index_state** 는 **오류,닫기,열기, 닫기 ,온라인 및 오프라인 으로 표시할 수 있습니다.**

3. **VDOPoolLV의 압축을 활성화하거나 비활성화하려면 다음을 수행합니다.**

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

4. **VDOPoolLV에 대한 중복 제거를 활성화하거나 비활성화하려면 다음을 수행합니다.**

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

**추가 리소스**

- **lvmvdo(7) 도움말 페이지**

**3.8. 가상 데이터 최적화 도구를 사용하여 씬 프로비저닝 관리**

VDO 볼륨의 물리적 공간 사용량이 100%에 적용되는 조건을 해결하기 위해 씬 프로비저닝된 VDO 볼륨을 구성하여 물리 공간의 향후 확장을 준비할 수 있습니다. 예를 들어, lvcreate 작업에서 -l 100%FREE를 사용하는 대신 '95%FREE'를 사용하여 나중에 복구할 수 있도록 예약된 공간이 있는지 확인합니다. 이 절차에서는 다음 문제를 해결하는 방법을 설명합니다.

- 볼륨이 공간이 부족합니다.
- 파일 시스템이 읽기 전용 모드로 전환
- 볼륨에서 보고한 ENOSPC

참고

VDO 볼륨에서 높은 물리적 공간 사용을 처리하는 가장 좋은 방법은 사용하지 않는 파일을 삭제하고 온라인 삭제 또는 fstrim 프로그램을 사용하여 사용되지 않은 파일에서 사용하는 블록을 삭제하는 것입니다. VDO 볼륨의 물리 공간은 기본 slab 크기가 2GB인 VDO 볼륨의 경우 16TB인 8192 slabs로, 최대 32GB 크기의 VDO 볼륨의 경우 256TB로 증가할 수 있습니다.

다음 모든 단계에서 *myvg* 및 *myvdo*를 각각 볼륨 그룹 및 VDO 이름으로 교체합니다.

사전 요구 사항

1. VDO 소프트웨어를 설치합니다. 자세한 내용은 VDO 설치를 참조하십시오.
2. 사용 가능한 스토리지 용량이 있는 LVM 볼륨 그룹이 시스템에 있습니다.
3. lvcreate --type ProfileBundle --name myvdo myvg -L logical-size-of-pool --virtualsize virtual-size-of-vdo 명령을 사용하는 씬 프로비저닝 VDO 볼륨. 자세한 내용은 LVM-VDO 볼륨 생성을 참조하십시오.

절차

1. 씬 프로비저닝된 VDO 볼륨의 최적 논리 크기를 결정합니다.

```
# vdostats myvg-vpool0-vpool

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

공간 절감 비율을 계산하려면 다음 수식을 사용하십시오.

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

이 예에서, In this example,

- 약 3.22:1 공간 절약 비율은 약 80GB의 데이터 세트에 있습니다.
- 데이터 세트 크기를 비율으로 곱하면 동일한 공간 절약이 있는 더 많은 데이터가 VDO 볼륨에 기록되면 256GB의 잠재적인 논리 크기를 얻을 수 있습니다.
- 이 수치를 200GB로 조정하면 동일한 공간 절약 비율을 고려하여 여유 물리적 공간이 안전한 논리 크기를 얻을 수 있습니다.

2. VDO 볼륨에서 사용 가능한 물리 공간을 모니터링합니다.

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

이 명령은 주기적으로 실행되어 VDO 볼륨의 사용 가능한 물리 공간을 모니터링할 수 있습니다.

3. 선택 사항: 사용 가능한 /usr/share/doc/vdo/examples/monitor/monitor_check_vdostats_physicalSpace.pl 스크립트를 사용하여 VDO 볼륨의 물리 공간 사용에 대한 경고를 확인합니다.

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

4. VDO 볼륨을 생성할 때 dmeventd 모니터링 서비스는 VDO 볼륨의 물리 공간 사용량을 모니터링합니다. VDO 볼륨을 만들거나 시작할 때 기본적으로 활성화되어 있습니다.

journalctl 명령을 사용하여 VDO 볼륨을 모니터링하는 동안 로그에 dmeventd 의 출력을 확

인합니다.

```
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.

사용 가능한 물리 공간이 거의 없는 VDO 볼륨을 해결합니다. VDO 볼륨에 물리적 공간을 추가할 수 있지만 볼륨을 확장하기 전에 볼륨 공간이 가득 차면 I/O를 볼륨에 일시적으로 중지해야 할 수 있습니다.

볼륨에서 I/O를 일시적으로 중지하려면 다음 단계를 실행합니다. 여기서 VDO 볼륨 *myvdo* 에는 */users/homeDir* 경로에 마운트된 파일 시스템이 포함되어 있습니다.

a.

파일 시스템을 정지합니다.

```
# xfs_freeze -f /users/homeDir

# vgextend myvg /dev/vdc2

# lvextend -l new_size myvg/vpool0-name

# xfs_freeze -u /users/homeDir
```

b.

파일 시스템을 마운트 해제합니다.

```
# umount /users/homeDir

# vgextend myvg /dev/vdc2

# lvextend -l new_size myvg/vpool0-name

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

참고

캐시된 데이터가 있는 파일 시스템을 마운트 해제하거나 해제하면 캐시된 데이터의 쓰기가 수행되므로 VDO 볼륨의 물리적 공간을 채울 수 있습니다. VDO 볼륨에서 사용 가능한 물리 공간에 대한 모니터링 임계값을 설정할 때 캐시된 파일 시스템 데이터의 최대 양을 고려하십시오.

6.

파일 시스템에서 더 이상 사용하지 않는 블록은 **fstrim** 유틸리티를 사용하여 정리할 수 있습니다. VDO 볼륨 상단에 마운트된 파일 시스템에 대해 **fstrim** 을 실행하면 해당 볼륨의 사용 가능한 물리 공간이 증가할 수 있습니다. **fstrim** 유틸리티를 사용하면 VDO 볼륨에 삭제가 전송되고 이전에 사용된 블록에 대한 참조를 제거하는 데 사용됩니다. 이러한 블록 중 하나가 단일 참조인 경우 물리적 공간을 사용할 수 있습니다.

a. VDO 통계를 확인하여 현재 사용 가능한 공간이 무엇인지 확인합니다.

```
# 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. 사용되지 않는 블록 삭제:

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

c. VDO 볼륨의 사용 가능한 물리 공간을 확인합니다.

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


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

이 예제에서는 파일 시스템에서 **fstrim** 을 실행한 후 VDO 볼륨에서 사용할 물리적 공간의 65G를 반환할 수 있었습니다.

참고

낮은 수준의 중복 제거 및 압축으로 볼륨을 삭제하면 중복 제거 및 압축 수준이 높은 볼륨을 삭제하는 것보다 물리적 공간을 회수할 수 있습니다. 높은 수준의 중복 제거 및 압축이 있는 볼륨은 이미 사용되지 않은 블록을 버리는 것보다 더 광범위한 정리를 필요로 할 수 있습니다.

# 4장. 기존 VDO 볼륨을 LVM으로 가져오기

**VDO 관리자가 생성한 VDO 볼륨을 LVM으로 가져올 수 있습니다. 결과적으로 LVM 툴을 사용하여 볼륨을 논리 볼륨으로 관리할 수 있습니다.**

참고

가져오기 작업은 취소할 수 없습니다. 기존 VDO 볼륨을 LVM으로 변환한 후 VDO 데이터 액세스는 LVM 명령을 통해서만 액세스할 수 있으며 VDO 관리자는 더 이상 볼륨을 제어하지 않습니다.

사전 요구 사항

- VDO 소프트웨어를 설치합니다. 자세한 내용은 VDO 설치를 참조하십시오.

절차

1. VDO 관리자가 생성한 기존 VDO 볼륨을 논리 볼륨으로 변환합니다. 다음 명령에서 이름이 볼륨 그룹 이름, ***lv -name*** 을 논리 볼륨 이름으로, ***/dev/sdg1*** 을 VDO 장치로 바꿉니다.

```
# lvm_import_vdo --name vg-name/lv-name /dev/sdg1


Convert VDO device "/dev/sdg1" to VDO LV "vg-name/lv-name"? [y|N]: Yes
Stopping VDO vdo-name
Converting VDO vdo-name
    Opening /dev/disk/by-id/scsi-36d094660575ece002291bd67517f677a-part1 exclusively
    Loading the VDO superblock and volume geometry
    Checking the VDO state
    Converting the UDS index
    Converting the VDO
    Conversion completed for '/dev/disk/by-id/scsi-36d094660575ece002291bd67517f677a-
part1': VDO is now offset by 2097152 bytes
 Physical volume "/dev/sdg1" successfully created.
 Volume group "vg-name" successfully created
 WARNING: Logical volume vg-name/lv-name_vpool not zeroed.
 Logical volume "lv-name_vpool" created.
 WARNING: Converting logical volume vg-name/lv-name_vpool to VDO pool volume
WITHOUT formating.
 WARNING: Using invalid VDO pool data MAY DESTROY YOUR DATA!
 Logical volume "lv-name" created.
 Converted vg-name/lv-name_vpool to VDO pool volume and created virtual vg-name/lv-
name VDO volume.
```

2. 선택 사항: VDO LV에 파일 시스템을 만듭니다.

3. 선택 사항: LVM-VDO 볼륨을 마운트합니다. 자세한 내용은 LVM-VDO 볼륨 마운트를 참조하십시오.

검증

- LVM 장치를 나열하여 VDO 볼륨을 LVM으로 가져오는 것이 성공했는지 확인합니다.

```
# lvs -a -o +devices
  LV                VG      Attr       LSize  Pool         Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  lv-name           vg-name vwi-a-v--- 25.00g lv-name_vpool        0.00
lv-name_vpool(0)
  lv-name_vpool     vg-name dwi------- <1.82t                     0.31                            lv-
name_vpool_vdata(0)
  [lv-name_vpool_vdata] vg-name Dwi-ao---- <1.82t
/dev/sdg1(0)
```

추가 리소스

- lvm_import_vdo(8), lvmvdo(7), and systemd.mount(5) man pages

# 5장. LVM-VDO 볼륨의 TRIM 옵션

VDO 볼륨에 사용되지 않는 공간을 알리는 **discard** 옵션을 사용하여 파일 시스템을 마운트할 수 있습니다. 또 다른 옵션은 즉시 삭제를 위해 온디맨드 삭제 또는 **mount -o discard** 명령인 **fstrim** 애플리케이션을 사용하는 것입니다.

**fstrim** 애플리케이션을 사용하는 경우 관리자는 추가 프로세스를 예약하고 모니터링해야 하는 반면 **mount -o discard** 명령을 사용하면 가능한 경우 즉시 공간을 복구할 수 있습니다.

현재 **fstrim** 애플리케이션을 사용하여 **discard** 마운트 옵션이 아닌 사용하지 않는 블록을 폐기하는 것이 좋습니다. 이 옵션의 성능에 미치는 영향은 매우 심각할 수 있기 때문입니다. 이러한 이유로 **nodiscard** 가 기본값입니다.

## 5.1. VDO에서 삭제 마운트 옵션 활성화

이 절차에서는 **VDO** 볼륨에서 삭제 옵션을 활성화합니다.

사전 요구 사항

- **LVM-VDO** 볼륨이 시스템에 있습니다.

절차

- 볼륨에서 삭제 를 활성화합니다.

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

추가 리소스

- **XFS(5), mount(8)** 및 **lvmvdo(7)** 도움말 페이지

## 5.2. 정기적인 TRIM 작업 설정

이 절차에서는 시스템에서 예약된 **TRIM** 작업을 활성화합니다.

사전 요구 사항

- **LVM-VDO 볼륨이 시스템에 있습니다.**

**절차**

- **타이머를 활성화하고 시작합니다.**

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

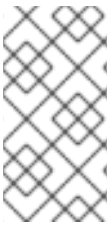
**검증**

- **타이머가 활성화되었는지 확인합니다.**

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

**예 5.1. 확인 절차의 출력 가능**

```
# 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
```



**참고**

**fstrim.timer 는 마운트된 모든 파일 시스템에서 실행되므로 VDO 볼륨에 대한 참조가 표시되지 않습니다.**

**추가 리소스**

- **fstrim(8) 도움말 페이지**