Red Hat Enterprise Linux 9

논리 볼륨 구성 및 관리

LVM 논리 볼륨 구성 및 관리 가이드

Last Updated: 2022-08-15

Red Hat Enterprise Linux 9 논리 볼륨 구성 및 관리

LVM 논리 볼륨 구성 및 관리 가이드

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.

법적 공지

초록

이 문서에서는 Red Hat Enterprise Linux 9에서 LVM 논리 볼륨을 관리하는 방법에 대해 설명합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 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

문서 개선을 위한 의견을 보내 주십시오. 문서를 개선할 수 있는 방법에 관해 알려주십시오.

- 특정 문구에 대한 간단한 의견 작성 방법은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **피드백** 버튼이 있는지 확인합니다.
  2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
  3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
  4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성하십시오.
  1. Bugzilla 웹 사이트로 이동하십시오.
  2. 구성 요소로 **문서**를 사용합니다.
  3. **설명** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **버그 제출**을 클릭합니다.

1장. 논리 볼륨 관리 개요

LVM(논리 볼륨 관리)은 논리 스토리지 볼륨을 생성하는 데 도움이 되는 물리 스토리지에 대한 추상화 계층을 생성합니다. 이를 통해 물리적 스토리지를 직접 사용하는 것보다 다양한 방법으로 훨씬 더 유연하게 사용할 수 있습니다.

또한 하드웨어 스토리지 구성은 소프트웨어에서 숨겨지므로 애플리케이션을 중지하거나 파일 시스템을 마운트 해제하지 않고도 크기를 조정하고 이동할 수 있습니다. 이는 운영 비용을 절감할 수 있습니다.

1.1. LVM 아키텍처

다음은 LVM의 구성 요소입니다.

물리 볼륨

PV(물리 볼륨)는 LVM 사용을 위해 지정된 파티션 또는 전체 디스크입니다. 자세한 내용은 LVM 물리 볼륨 관리를 참조하십시오.

볼륨 그룹

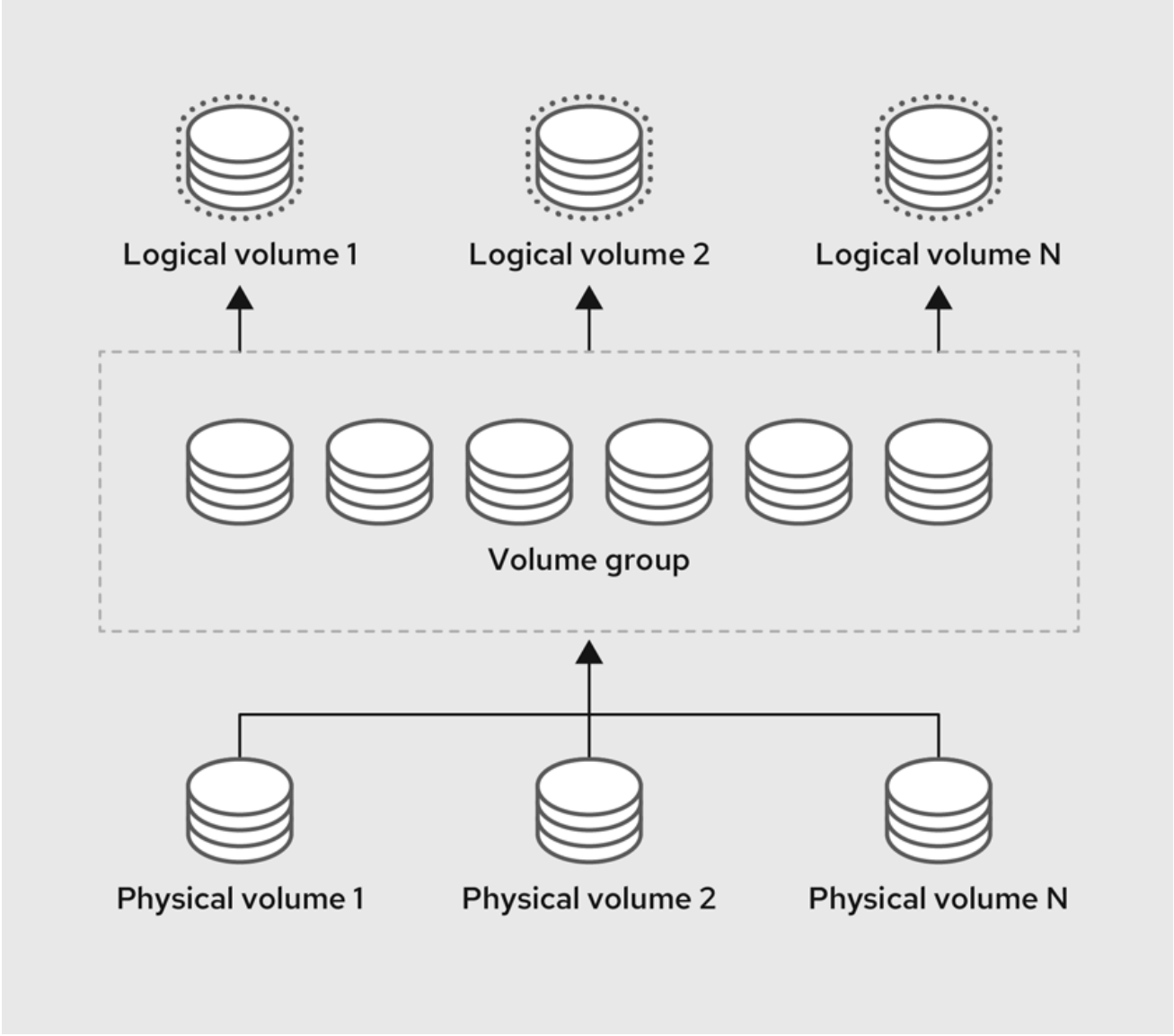
VG(볼륨 그룹)는 PV(물리 볼륨) 컬렉션으로, 논리 볼륨을 할당할 수 있는 디스크 공간 풀을 생성합니다. 자세한 내용은 LVM 볼륨 그룹 관리를 참조하십시오.

논리 볼륨

논리 볼륨은 마운트할 수 있는 스토리지 장치를 나타냅니다. 자세한 내용은 LVM 논리 볼륨 관리를 참조하십시오.

다음 다이어그램은 LVM 구성 요소를 보여줍니다.

그림 1.1. LVM 논리 볼륨 구성 요소



1.2. LVM의 장점

논리 볼륨은 물리 스토리지를 직접 사용하는 것보다 다음과 같은 이점을 제공합니다.

유연한 용량

논리 볼륨을 사용하는 경우 장치 및 파티션을 단일 논리 볼륨에 집계할 수 있습니다. 이 기능을 사용하면 파일 시스템을 단일 대규모 장치처럼 여러 장치에 걸쳐 확장할 수 있습니다.

크기 조정할 수 있는 스토리지 볼륨

기본 장치를 다시 포맷하고 다시 분할하지 않고도 간단한 소프트웨어 명령으로 논리 볼륨을 확장하거나 논리 볼륨을 줄일 수 있습니다.

온라인 데이터 재배치

최신 스토리지 하위 시스템을 배포하려면 시스템이 활성 상태인 동안 데이터를 이동할 수 있습니다. 디스크를 사용하는 동안 디스크에 데이터를 다시 정렬할 수 있습니다. 예를 들어, 제거하기 전에 핫 스왑 가능 디스크를 비울 수 있습니다.

편리한 장치 이름 지정

논리 스토리지 볼륨은 사용자 정의 및 사용자 정의 이름으로 관리할 수 있습니다.

스트라이핑된 볼륨

두 개 이상의 장치에서 데이터를 스트라이핑하는 논리 볼륨을 만들 수 있습니다. 이로 인해 처리량이 크게 증가할 수 있습니다.

RAID 볼륨

논리 볼륨은 편리하게 데이터에 대한 RAID를 구성할 수 있는 방법을 제공합니다. 이로 인해 장치 장애에 대한 보호가 가능하며 성능이 향상됩니다.

볼륨 스냅샷

일관된 백업을 위해 논리 볼륨의 시점 복사본인 스냅샷을 찍거나 실제 데이터에 영향을 주지 않고 변경의 영향을 테스트할 수 있습니다.

씬 볼륨

논리 볼륨은 씬 프로비저닝할 수 있습니다. 이를 통해 사용 가능한 물리 공간보다 큰 논리 볼륨을 만들 수 있습니다.

캐시 볼륨

캐시 논리 볼륨은 SSD 드라이브와 같은 빠른 블록 장치를 사용하여 더 크고 느린 블록 장치의 성능을 향상시킵니다.

2장. LVM 물리 볼륨 관리

PV(물리 볼륨)는 LVM 사용을 위해 지정된 파티션 또는 전체 디스크입니다. LVM 논리 볼륨에 장치를 사용하려면 장치를 물리 볼륨으로 초기화해야 합니다.

물리 볼륨에 전체 디스크 장치를 사용하는 경우 디스크에 파티션 테이블이 없어야 합니다. ASCII 디스크 파티션의 경우 **fdisk** 또는 **cfdisk** 명령 또는 동일한 항목을 사용하여 파티션 ID를 0x8e로 설정해야 합니다. 물리 볼륨에 전체 디스크 장치를 사용하는 경우 디스크에 파티션 테이블이 없어야 합니다. 기존 파티션 테이블을 삭제해야 합니다. 그러면 해당 디스크의 모든 데이터가 효과적으로 삭제됩니다. 와제 **fs -a <PhysicalVolume> 명령을 root로 사용하여 기존 파티션 테이블을 제거할 수 있습니다**.

2.1. 물리 볼륨 개요

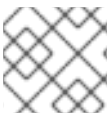
블록 장치를 물리 볼륨으로 초기화하면 장치 시작 근처의 레이블이 지정됩니다. 다음은 LVM 레이블을 설명합니다.

- LVM 레이블은 물리적 장치에 올바른 식별 및 장치 순서를 제공합니다. 레이블이 지정되지 않은 LVM이 아닌 장치는 부팅 중에 시스템에서 검색한 순서에 따라 재부팅 시 이름을 변경할 수 있습니다. LVM 레이블은 재부팅 후에도 클러스터 전체에서 유지됩니다.
- LVM 레이블은 장치를 LVM 물리 볼륨으로 식별합니다. 여기에는 물리 볼륨의 UUID인 임의의 고유 식별자가 포함됩니다. 또한 블록 장치의 크기를 바이트 단위로 저장하고 LVM 메타데이터가 장치에 저장될 위치를 기록합니다.
- 기본적으로 LVM 레이블은 두 번째 512바이트 섹터에 배치됩니다. 물리 볼륨을 생성할 때 처음 4개 섹터에 레이블을 배치하여 이 기본 설정을 덮어쓸 수 있습니다. 필요한 경우 LVM 볼륨이 이러한 섹터의 다른 사용자와 공존할 수 있습니다.

다음은 LVM 메타데이터를 설명합니다.

- LVM 메타데이터에는 시스템에 있는 LVM 볼륨 그룹의 구성 세부 정보가 포함되어 있습니다. 기본적으로 동일한 메타데이터 복사본은 볼륨 그룹 내의 모든 물리 볼륨의 모든 메타데이터 영역에서 유지됩니다. LVM 메타데이터는 작고 ASCII로 저장됩니다.
- 현재 LVM을 사용하면 각 물리 볼륨에 0, 1개 또는 2개의 메타데이터 복사본을 저장할 수 있습니다. 기본값은 1 복사본입니다. 물리 볼륨에서 메타데이터 복사본 수를 구성한 후에는 나중에 해당 번호를 변경할 수 없습니다. 첫 번째 복사본은 레이블 바로 뒤에 장치 시작 부분에 저장됩니다. 두 번째 복사본이 있으면 장치의 끝에 배치됩니다. 의도한 것과 다른 디스크에 작성하여 디스크 시작 부분에 있는 영역을 실수로 덮어쓰는 경우 장치 끝에 있는 두 번째 메타데이터 사본을 통해 메타데이터를 복구할 수 있습니다.

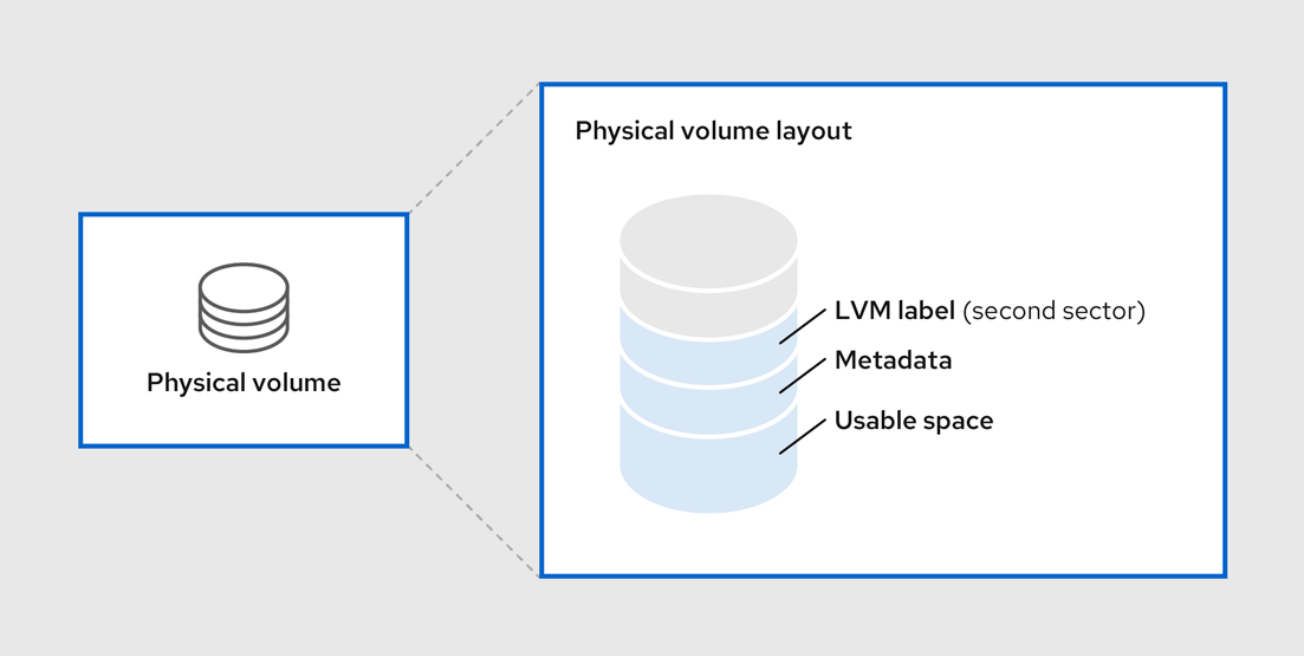
다음 다이어그램에서는 LVM 물리 볼륨의 레이아웃을 보여줍니다. LVM 레이블은 두 번째 섹터에 있고, 메타데이터 영역 다음에 장치에서 사용 가능한 공간이 뒤에 옵니다.



참고

Linux 커널 및 이 문서 전체에서 섹터는 512바이트 크기로 간주됩니다.

그림 2.1. 물리 볼륨 레이아웃



추가 리소스

- 디스크의 여러 파티션

2.2. 디스크의 여러 파티션

LVM을 사용하여 디스크 파티션에서 PV(물리 볼륨)를 만들 수 있습니다.

다음과 같은 이유로 전체 디스크를 포함하는 단일 파티션을 LVM 물리 볼륨으로 표시하는 것이 좋습니다.

관리 편의성

각 실제 디스크가 한 번만 표시되면 시스템에서 하드웨어를 추적하는 것이 더 쉽습니다. 특히 디스크에 오류가 발생하는 경우 이 문제가 발생합니다.

성능 스트라이핑

LVM에서 두 물리 볼륨이 동일한 물리 디스크에 있음을 알릴 수 없습니다. 두 물리 볼륨이 동일한 물리적 디스크에 있는 경우 스트라이핑된 논리 볼륨을 생성하는 경우 동일한 디스크의 여러 파티션에 스트라이프가 있을 수 있습니다. 이로 인해 성능이 향상되는 것이 아니라 성능이 저하됩니다.

RAID 중복

LVM에서 두 물리 볼륨이 동일한 장치에 있는지 확인할 수 없습니다. 두 물리 볼륨이 동일한 장치에 있을 때 RAID 논리 볼륨을 생성하는 경우 성능 및 내결함성이 손실될 수 있습니다.

권장되지는 않지만 디스크를 별도의 LVM 물리 볼륨으로 분할해야 하는 특정 상황이 있을 수 있습니다. 예를 들어 디스크가 거의 없는 시스템에서는 기존 시스템을 LVM 볼륨으로 마이그레이션할 때 파티션 주변 데이터를 이동해야 할 수 있습니다. 또한 대용량 디스크가 있고 관리를 위해 둘 이상의 볼륨 그룹이 필요한 경우 디스크를 분할해야 합니다. 둘 이상의 파티션이 있는 디스크가 있고 이러한 파티션이 모두 동일한 볼륨 그룹에 있는 경우 볼륨을 생성할 때 논리 볼륨에 포함할 파티션을 지정하십시오.

LVM에서는 파티션이 아닌 디스크를 물리 볼륨으로 사용할 수 있지만 파티션 없이 PV를 생성하는 것이 혼합 운영 체제 환경에서 문제가 될 수 있으므로 단일 전체 디스크 파티션을 생성하는 것이 좋습니다. 다른 운영 체제는 장치를 무료로 해석하고 드라이브 시작 부분에서 PV 레이블을 덮어쓸 수 있습니다.

2.3. LVM 물리 볼륨 생성

다음 절차에서는 LVM 물리 볼륨(PV)을 만들고 레이블을 지정하는 방법을 설명합니다.

이 절차에서 */dev/vdb1*,*/dev/vdb2* 및 */dev/vdb3* 을 시스템에서 사용 가능한 스토리지 장치로 교체합니다.

사전 요구 사항

- **lvm2** 패키지가 설치되어 있습니다.

절차

1. **pvcreate** 명령에 대한 인수로 공백으로 구분된 장치 이름을 사용하여 여러 개의 물리 볼륨을 생성합니다.

```
# pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Physical volume "/dev/vdb3" successfully created.
```

   그러면 */dev/vdb1*,*/dev/vdb2* 및 */dev/vdb3* 에 레이블이 있으면 LVM에 속하는 물리 볼륨으로 표시합니다.

2. 요구 사항에 따라 다음 명령 중 하나를 사용하여 생성된 물리 볼륨을 확인합니다.

   a. 각 물리 볼륨에 대한 자세한 여러 줄 출력을 제공하는 **pvdisplay** 명령. 크기, 확장 영역, 볼륨 그룹 및 기타 옵션과 같은 물리적 속성을 고정된 형식으로 표시합니다.

```
# pvdisplay
--- NEW Physical volume ---
  PV Name               /dev/vdb1
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb2
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb3
  VG Name
  PV Size               1.00 GiB
[..]
```

   b. **pvs** 명령은 구성 가능한 형태로 물리적 볼륨 정보를 제공하여 물리 볼륨당 한 줄만 표시합니다.

```
# pvs
  PV         VG  Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
/dev/vdb3        lvm2           1020.00m   0
```

   c. **pvscan** 명령은 물리 볼륨에 대해 시스템에서 지원되는 모든 LVM 블록 장치를 검사합니다. **lvm.conf** 파일에 필터를 정의하여 이 명령으로 특정 물리 볼륨 검사를 방지할 수 있습니다.

```
# pvscan
  PV /dev/vdb1                lvm2 [1.00 GiB]
  PV /dev/vdb2                lvm2 [1.00 GiB]
  PV /dev/vdb3                lvm2 [1.00 GiB]
```

추가 리소스

- **pvcreate(8)**, **pvdisplay(8)**, **pvs(8)**, pvs(8) , **pvscan(8)** 및 **lvm(8)** 도움말 페이지

2.4. LVM 물리 볼륨 제거

LVM에서 장치를 더 이상 사용할 필요가 없는 경우 **pvremove** 명령을 사용하여 LVM 레이블을 제거할 수 있습니다. **pvremove 명령을 실행하면** 빈 물리 볼륨의 LVM 메타데이터가 0이 됩니다.

절차

1. 물리 볼륨 제거:

```
# pvremove /dev/vdb3
Labels on physical volume "/dev/vdb3" successfully wiped.
```

2. 기존 물리 볼륨을 보고 필요한 볼륨이 제거되었는지 확인합니다.

```
# pvs
  PV         VG   Fmt  Attr PSize    PFree
/dev/vdb1        lvm2       1020.00m  0
/dev/vdb2        lvm2       1020.00m  0
```

제거하려는 물리 볼륨이 현재 볼륨 그룹의 일부인 경우 **vgreduce** 명령을 사용하여 볼륨 그룹에서 제거해야 합니다. 자세한 내용은 볼륨 그룹에서 물리 볼륨 제거를 참조하십시오.

추가 리소스

- **pvremove(8)** 도움말 페이지

2.5. 추가 리소스

- parted가 있는 디스크에서 파티션 테이블 만들기 .
- **parted(8)** 도움말 페이지.

3장. LVM 볼륨 그룹 관리

VG(볼륨 그룹)는 PV(물리 볼륨) 컬렉션으로, LV(논리 볼륨)를 할당할 수 있는 디스크 공간 풀을 생성합니다.

볼륨 그룹 내에서 할당에 사용할 수 있는 디스크 공간은 확장 영역이라는 고정 크기 단위로 나뉩니다. 익스텐트는 할당할 수 있는 최소 공간 단위입니다. 물리 볼륨 내에서 확장 영역을 물리 확장 영역이라고 합니다.

논리 볼륨은 물리 확장 영역과 동일한 크기의 논리 확장 영역으로 할당됩니다. 따라서 볼륨 그룹의 모든 논리 볼륨에 대해 확장 크기가 동일합니다. 볼륨 그룹은 논리 확장 영역을 물리 확장 영역으로 매핑합니다.

3.1. LVM 볼륨 그룹 생성

이 절차에서는 */dev/vdb1* 및 */dev/vdb 2* 물리 볼륨을 사용하여 LVM 볼륨 그룹(VG) *myvg* 를 만드는 방법을 설명합니다.

사전 요구 사항

- **lvm2** 패키지가 설치되어 있습니다.
- 하나 이상의 물리 볼륨이 생성됩니다. 물리 볼륨 생성에 대한 자세한 내용은 LVM 물리 볼륨 생성을 참조하십시오.

절차

1. 볼륨 그룹을 만듭니다.

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
 Volume group "myvg" successfully created.
```

그러면 *myvg* 의 이름이 있는 VG가 생성됩니다. PVs */dev/vdb1* 및 */dev/vdb2* 는 *myvg* VG의 기본 스토리지 수준입니다.

2. 요구 사항에 따라 다음 명령 중 하나를 사용하여 생성된 볼륨 그룹을 확인합니다.

   a. **vgs** 명령은 볼륨 그룹 정보를 구성 가능한 형태로 제공하여 볼륨 그룹당 하나의 행을 표시합니다.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  myvg   4   1   0 wz--n-   3.98g 1008.00m
```

   b. **vgdisplay** 명령은 크기, 확장 영역, 물리 볼륨 수 및 기타 옵션과 같은 볼륨 그룹 속성을 고정된 형식으로 표시합니다. 다음 예제에서는 *myvg* 볼륨 그룹에 대한 **vgdisplay** 명령의 출력을 보여줍니다. 볼륨 그룹을 지정하지 않으면 기존 볼륨 그룹이 모두 표시됩니다.

```
# vgdisplay myvg _ --- Volume group --- VG Name _myvg
  System ID
  Format                lvm2
  Metadata Areas        4
  Metadata Sequence No  6
  VG Access             read/write
[..]
```

c. **vgscan** 명령은 볼륨 그룹에 대해 시스템에서 지원되는 모든 LVM 블록 장치를 검사합니다.

```
# vgscan
  Found volume group "myvg" using metadata type lvm2
```

3. 선택 사항: 하나 이상의 사용 가능한 물리 볼륨을 추가하여 볼륨 그룹의 용량을 늘립니다.

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

추가 리소스

- **pvcreate(8)**, **vgextend(8)**, **vgdisplay(8)**, **vgs(8)**, **vgscan(8)**, **lvm(8)** 도움말 페이지

3.2. LVM 볼륨 그룹 결합

두 볼륨 그룹을 단일 볼륨 그룹으로 결합하려면 **vgmerge** 명령을 사용합니다. 볼륨의 물리 확장 영역 크기가 동일하고 대상 볼륨 그룹 제한에 맞는 두 볼륨 그룹의 물리 및 논리 볼륨 요약이 동일한 경우 비활성 "소스" 볼륨을 활성 또는 비활성 "대상" 볼륨과 병합할 수 있습니다.

절차

- 자세한 런타임 정보를 제공하여 비활성 *볼륨 그룹 데이터베이스* 를 활성 또는 비활성 볼륨 그룹 *myvg* 에 병합합니다.

```
# vgmerge -v myvg databases
```

추가 리소스

- **vgmerge(8)** 도움말 페이지

3.3. 볼륨 그룹에서 물리 볼륨 제거

볼륨 그룹에서 사용하지 않은 물리 볼륨을 제거하려면 **vgreduce** 명령을 사용합니다. **vgreduce** 명령은 하나 이상의 빈 물리 볼륨을 제거하여 볼륨 그룹의 용량을 줄입니다. 이렇게 하면 물리 볼륨을 다른 볼륨 그룹에서 사용하거나 시스템에서 제거할 수 있습니다.

절차

1. 물리 볼륨이 계속 사용 중인 경우 동일한 볼륨 그룹에서 데이터를 다른 물리 볼륨으로 마이그레이션합니다:

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
  ...
  /dev/vdb3: Moved: 79.2%
  ...
  /dev/vdb3: Moved: 100.0%
```

2. 기존 볼륨 그룹의 다른 물리 볼륨에 사용 가능한 확장 영역이 충분하지 않은 경우:

a. */dev/vdb4* 에서 새 물리 볼륨을 생성합니다.

```
# pvcreate /dev/vdb4
  Physical volume "/dev/vdb4" successfully created
```

b. 새로 생성된 물리 볼륨을 *myvg* 볼륨 그룹에 추가합니다.

```
# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended
```

c. */dev/vdb3에서 /dev/vdb 4* 로 데이터를 이동합니다.

```
# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. 볼륨 그룹에서 물리 볼륨 */dev/vdb3* 을 제거합니다.

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

검증

- */dev/vdb3* 물리 볼륨이 *myvg* 볼륨 그룹에서 제거되었는지 확인합니다.

```
# pvs
  PV         VG    Fmt   Attr  PSize       PFree      Used
  /dev/vdb1 myvg  lvm2  a--    1020.00m    0         1020.00m
  /dev/vdb2 myvg  lvm2  a--    1020.00m    0         1020.00m
  /dev/vdb3       lvm2  a--    1020.00m  1008.00m    12.00m
```

추가 리소스

- **vgreduce(8)**, **pvmove(8)** 및 **pvs(8)** 도움말 페이지

3.4. LVM 볼륨 그룹 분할

다음 절차에서는 기존 볼륨 그룹을 분할하는 방법을 설명합니다. 물리 볼륨에 사용되지 않은 공간이 충분한 경우 새 디스크를 추가하지 않고 새 볼륨 그룹을 만들 수 있습니다.

초기 설정에서 *myvg* 볼륨 그룹은 */dev/vdb1*,*/dev/ vdb2* 및 */dev/vdb3* 으로 구성됩니다. 이 절차를 완료하면 *myvg* 볼륨 그룹은 */dev/vdb1* 및 */dev/vdb2* 로 구성되고 두 번째 볼륨 그룹인 *yourvg* 는 */dev/vdb3* 으로 구성됩니다.

사전 요구 사항

- 볼륨 그룹에 충분한 공간이 있습니다. **vgscan** 명령을 사용하여 현재 볼륨 그룹에서 사용할 수 있는 여유 공간을 결정합니다.
- 기존 물리 볼륨의 사용 가능한 용량에 따라 **pvmove** 명령을 사용하여 사용된 모든 물리 확장 영역을 다른 물리 볼륨으로 이동합니다. 자세한 내용은 볼륨 그룹에서 물리 볼륨 제거를 참조하십시오.

절차

1. 기존 볼륨 그룹 *myvg* 를 새 볼륨 그룹 *yourvg* 로 분할하십시오 :

```
# vgsplit myvg yourvg /dev/vdb3
  Volume group "yourvg" successfully split from "myvg"
```

참고

기존 볼륨 그룹을 사용하여 논리 볼륨을 생성한 경우 다음 명령을 사용하여 논리 볼륨을 비활성화합니다.

```
# lvchange -a n /dev/myvg/mylv
```

논리 볼륨 생성에 대한 자세한 내용은 LVM 논리 볼륨 관리를 참조하십시오.

2. 두 볼륨 그룹의 속성을 확인합니다.

```
# 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
```

검증

- vg가 */dev/vdb3* 물리 *볼륨으로 구성된 새로 생성된 볼륨 그룹이 있는지* 확인합니다.

```
# pvs
  PV         VG     Fmt  Attr PSize    PFree    Used
  /dev/vdb1 myvg   lvm2 a--  1020.00m  0        1020.00m
  /dev/vdb2 myvg   lvm2 a--  1020.00m  0        1020.00m
  /dev/vdb3 yourvg lvm2 a--  1020.00m  1008.00m  12.00m
```

추가 리소스

- **vgsplit(8)**, **vgs(8)** 및 **pvs(8)** 도움말 페이지

3.5. LVM 볼륨 그룹 이름 변경

이 절차에서는 기존 볼륨 그룹 *myvg의 이름을 myvg 1* 로 변경합니다.

절차

1. 볼륨 그룹을 비활성화합니다. 클러스터형 볼륨 그룹인 경우 이러한 각 노드에서 다음 명령을 사용하여 활성화된 모든 노드에서 볼륨 그룹을 비활성화합니다.

```
# vgchange --activate n myvg
```

2. 기존 볼륨 그룹의 이름을 변경합니다.

```
# vgrename myvg myvg1
Volume group "myvg" successfully renamed to "myvg1"
```

장치에 대한 전체 경로를 지정하여 볼륨 그룹의 이름을 바꿀 수도 있습니다.

```
# vgrename /dev/myvg /dev/myvg1
```

추가 리소스

- **vgrename(8)** 도움말 페이지

4장. LVM 논리 볼륨 관리

논리 볼륨은 파일 시스템, 데이터베이스 또는 애플리케이션에서 사용할 수 있는 가상, 블록 스토리지 장치입니다. LVM 논리 볼륨을 만들기 위해 PV(물리 볼륨)가 볼륨 그룹(VG)으로 결합됩니다. 그러면 LVM 논리 볼륨(LV)을 할당할 수 있는 디스크 공간 풀이 생성됩니다.

4.1. 논리 볼륨 개요

관리자는 표준 디스크 파티션과 달리 데이터를 삭제하지 않고 논리 볼륨을 늘리거나 줄일 수 있습니다. 볼륨 그룹의 물리 볼륨이 별도의 드라이브 또는 RAID 배열에 있는 경우 관리자는 논리 볼륨을 스토리지 장치에 분배할 수도 있습니다.

논리 볼륨을 필요한 데이터보다 작은 용량으로 줄이는 경우 데이터를 유실할 수 있습니다. 또한 일부 파일 시스템은 축소할 수 없습니다. 유연성을 극대화하려면 논리 볼륨을 생성하여 현재 요구 사항을 충족하고 과도한 스토리지 용량을 할당하지 않은 상태로 둡니다. 필요에 따라 할당되지 않은 공간을 사용하도록 논리 볼륨을 안전하게 확장할 수 있습니다.

중요

AMD, Intel, ARM 시스템 및 IBM Power Systems 서버의 부트 로더는 LVM 볼륨을 읽을 수 없습니다. **/boot** 파티션에 대해 LVM이 아닌 표준 디스크 파티션을 만들어야 합니다. IBM Z에서 **zipl** 부트 로더는 선형 매핑을 사용하여 LVM 논리 볼륨에서 **/boot** 를 지원합니다. 기본적으로 설치 프로세스는 항상 물리 볼륨에 별도의 / 부트 파티션을 사용하여 LVM 볼륨 내에 / 및 스왑 파티션을 만듭니다.

다음은 논리 볼륨의 다양한 유형입니다.

선형 볼륨

선형 볼륨은 하나 이상의 물리 볼륨에서 한 개의 논리 볼륨으로 공간을 집계합니다. 예를 들어, 두 개의 60GB 디스크가 있는 경우 120GB 논리 볼륨을 만들 수 있습니다. 물리 스토리지가 연결되어 있습니다.

스트라이핑된 논리 볼륨

LVM 논리 볼륨에 데이터를 작성할 때 파일 시스템은 기본 물리 볼륨에 데이터를 배치합니다. 스트라이핑된 논리 볼륨을 생성하여 데이터를 물리 볼륨에 작성하는 방법을 제어할 수 있습니다. 대규모 순차적 읽기 및 쓰기의 경우 데이터 I/O의 효율성을 향상시킬 수 있습니다.
스트라이핑은 사전 정의된 수의 물리 볼륨에 라운드 로빈 방식으로 데이터를 작성하여 성능을 향상시킵니다. 스트라이핑을 사용하면 I/O를 병렬로 수행할 수 있습니다. 경우에 따라 스트라이프에 있는 각 추가 물리 볼륨에 대한 선형 성능 향상이 발생할 수 있습니다.

RAID 논리 볼륨

LVM은 RAID 레벨 0, 1, 4, 5, 6, 10을 지원합니다. RAID 논리 볼륨은 클러스터를 인식하지 못합니다.
RAID 논리 볼륨을 생성하면 LVM에서 배열의 모든 데이터 또는 패리티 하위 볼륨에 대해 크기가 1개인 메타데이터 하위 볼륨을 생성합니다.

씬 프로비저닝된 논리 볼륨 (볼륨에서)

씬 프로비저닝된 논리 볼륨을 사용하여 사용 가능한 물리 스토리지보다 큰 논리 볼륨을 생성할 수 있습니다. 씬 프로비저닝된 볼륨 세트를 생성하면 시스템이 요청된 전체 스토리지 용량을 할당하는 대신 사용하는 사항을 할당할 수 있습니다.

스냅샷 볼륨

LVM 스냅샷 기능을 사용하면 서비스가 중단되지 않고 특정 시점에 장치의 가상 이미지를 생성할 수 있습니다. 스냅샷을 만든 후 원래 장치(원본)를 변경하면 스냅샷 기능이 변경 전과 마찬가지로 변경된 데이터 영역의 복사본을 만들어 장치의 상태를 재구성할 수 있습니다.

씬 프로비저닝된 스냅샷 볼륨

씬 프로비저닝된 스냅샷 볼륨을 사용하여 동일한 데이터 볼륨에 더 많은 가상 장치를 저장할 수 있습니다. 씬 프로비저닝된 스냅샷은 주어진 시간에 캡처하려는 모든 데이터를 복사하지 않으므로 유용합니다.

캐시 볼륨

LVM에서는 느린 큰 블록 장치에 대해 SSD 드라이브와 같은 빠른 블록 장치의 사용을 지원합니다. 사용자는 캐시 논리 볼륨을 생성하여 기존 논리 볼륨의 성능을 향상하거나 크고 느린 장치에 연결된 소형 및 빠른 장치로 구성된 새 캐시 논리 볼륨을 생성할 수 있습니다.

4.2. LVM 논리 볼륨 생성

이 절차에서는 */dev/vdb1*, */dev/vdb2* 및 */dev/vdb 3* 물리 볼륨을 사용하여 생성되는 *myvg* 볼륨 그룹에서 my *lv* LVM LV(논리 볼륨)를 만드는 방법을 설명합니다.

사전 요구 사항

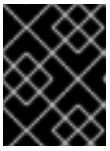
- **lvm2** 패키지가 설치되어 있습니다.
- 볼륨 그룹이 생성됩니다. 자세한 내용은 LVM 볼륨 그룹 생성을 참조하십시오.

절차

1. 논리 볼륨을 만듭니다.

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

n **옵션을** 사용하여 LV 이름을 *mylv* 로 설정하고 **-L** 옵션을 사용하여 Mb 단위로 LV 크기를 설정하지만 다른 단위를 사용할 수 있습니다. LV 유형은 기본적으로 선형이지만 사용자는 **--type** 옵션을 사용하여 원하는 유형을 지정할 수 있습니다.



중요

VG에 요청된 크기 및 유형에 대한 사용 가능한 물리 확장 영역의 수가 충분하지 않으면 명령이 실패합니다.

2. 요구 사항에 따라 다음 명령 중 하나를 사용하여 생성된 논리 볼륨을 확인합니다.

   a. **lvs** 명령은 논리 볼륨 정보를 구성 가능한 형식으로 제공하여 논리 볼륨당 하나의 행을 표시합니다.

```
# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  mylv myvg -wi-ao----   500.00m
```

   b. **lvdisplay** 명령은 크기, 레이아웃 및 매핑과 같은 논리 볼륨 속성을 고정된 형식으로 표시합니다.

```
# lvdisplay -v /dev/myvg/mylv
  --- Logical volume ---
  LV Path                /dev/myvg/mylv
  LV Name                mylv
  VG Name                myvg
```

```
  LV UUID                YTnAk6-kMlT-c4pG-HBFZ-Bx7t-ePMk-7YjhaM
  LV Write Access        read/write
[..]
```

c. **lvscan** 명령은 시스템의 모든 논리 볼륨을 스캔하여 나열합니다.

```
# lvscan
 ACTIVE                 '/dev/myvg/mylv' [500.00 MiB] inherit
```

3. 논리 볼륨에 파일 시스템을 만듭니다. 다음 명령은 논리 **볼륨에 xfs** 파일 시스템을 생성합니다.

```
# mkfs.xfs /dev/myvg/mylv
meta-data=/dev/myvg/mylv       isize=512    agcount=4, agsize=32000 blks
         =                     sectsz=512   attr=2, projid32bit=1
         =                     crc=1        finobt=1, sparse=1, rmapbt=0
         =                     reflink=1
data     =                       bsize=4096   blocks=128000, imaxpct=25
         =                     sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                     sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

4. 논리 볼륨을 마운트하고 파일 시스템 디스크 공간 사용을 보고합니다.

```
# mount /dev/myvg/mylv /mnt

# df -h
Filesystem              1K-blocks  Used   Available Use% Mounted on

/dev/mapper/myvg-mylv   506528    29388  477140     6%   /mnt
```

추가 리소스

- **lvcreate(8)**, **lvdisplay(8)**, **lvs(8)**, **lvscan(8)**, **lvm(8)** 및 **mkfs.xfs(8)** 도움말 페이지

4.3. LVM 논리 볼륨 이름 변경

이 절차에서는 기존 논리 볼륨 *mylv*를 *mylv 1*로 *변경하는* 방법을 설명합니다.

절차

1. 논리 볼륨이 현재 마운트된 경우 볼륨을 마운트 해제합니다.

```
# umount /mnt
```

/mnt 를 마운트 지점으로 바꿉니다.

2. 논리 볼륨이 클러스터형 환경에 있는 경우 활성화된 모든 노드에서 논리 볼륨을 비활성화합니다. 이러한 각 노드에서 다음 명령을 사용합니다.

```
# lvchange --activate n myvg/mylv
```

3. 기존 논리 볼륨의 이름을 변경합니다.

```
# lvrename myvg mylv mylv1
Logical volume "mylv" successfully renamed to "mylv1"
```

장치에 대한 전체 경로를 지정하여 논리 볼륨의 이름을 바꿀 수도 있습니다.

```
# lvrename /dev/myvg/mylv /dev/myvg/mylv1
```

추가 리소스

- **lvrename(8)** 도움말 페이지

4.4. 논리 볼륨에서 디스크 제거

이 절차에서는 디스크를 교체하거나 다른 볼륨의 일부로 디스크를 사용하기 위해 기존 논리 볼륨에서 디스크를 제거하는 방법을 설명합니다.

디스크를 제거하려면 먼저 LVM 물리 볼륨의 확장 영역을 다른 디스크 또는 디스크 집합으로 이동해야 합니다.

절차

1. LV를 사용할 때 물리 볼륨의 사용 및 사용 가능한 공간을 확인합니다.

```
# pvs -o+pv_used
  PV         VG    Fmt    Attr   PSize      PFree     Used
 /dev/vdb1 myvg  lvm2   a--    1020.00m    0        1020.00m
 /dev/vdb2 myvg  lvm2   a--    1020.00m    0        1020.00m
 /dev/vdb3 myvg  lvm2   a--    1020.00m   1008.00m   12.00m
```

2. 데이터를 다른 물리 볼륨으로 이동합니다.

   a. 기존 볼륨 그룹의 다른 물리 볼륨에 사용 가능한 확장 영역이 충분한 경우 다음 명령을 사용하여 데이터를 이동합니다.

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

   b. 기존 볼륨 그룹의 다른 물리 볼륨에 사용 가능한 확장 영역이 충분하지 않은 경우 다음 명령을 사용하여 새 물리 볼륨을 추가하고, 새로 생성된 물리 볼륨을 사용하여 볼륨 그룹을 확장하고, 데이터를 이 물리 볼륨으로 이동합니다.

```
# pvcreate /dev/vdb4
  Physical volume "/dev/vdb4" successfully created

# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended
```

```
# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. 물리 볼륨 제거:

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

논리 볼륨에 오류가 발생한 물리 볼륨이 포함된 경우 해당 논리 볼륨을 사용할 수 없습니다. 볼륨 그룹에서 누락된 물리 볼륨을 제거하려면 누락된 물리 볼륨에 할당된 논리 볼륨이 없는 경우 **vgreduce** 명령의 **--removemissing** 매개변수를 사용할 수 있습니다.

```
# vgreduce --removemissing myvg
```

추가 리소스

- **pvmove(8)**, **vgextend(8)**, **vereduce(8)** 및 **pvs(8)** 도움말 페이지

4.5. LVM 논리 볼륨 제거

이 절차에서는 *myvg 볼륨 그룹에서 기존 논리 볼륨 /dev/myvg/mylv1 을 제거하는 방법을 설명합니다.*

절차

1. 논리 볼륨이 현재 마운트된 경우 볼륨을 마운트 해제합니다.

```
# umount /mnt
```

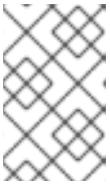
2. 논리 볼륨이 클러스터형 환경에 있는 경우 활성화된 모든 노드에서 논리 볼륨을 비활성화합니다. 이러한 각 노드에서 다음 명령을 사용합니다.

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

3. **lvremove** 유틸리티를 사용하여 논리 볼륨을 제거합니다.

```
# lvremove /dev/myvg/mylv1

Do you really want to remove active logical volume "mylv1"? [y/n]: y
Logical volume "mylv1" successfully removed
```



참고

이 경우 논리 볼륨이 비활성화되지 않았습니다. 제거하기 전에 논리 볼륨을 명시적으로 비활성화한 경우 활성 논리 볼륨을 제거할지 여부를 확인하는 메시지가 표시되지 않습니다.

추가 리소스

- **lvremove(8)** 도움말 페이지

4.6. LVM 볼륨 그룹 제거

다음 절차에서는 기존 볼륨 그룹을 제거하는 방법을 설명합니다.

사전 요구 사항

- 볼륨 그룹에 논리 볼륨이 없습니다. 볼륨 그룹에서 논리 볼륨을 제거하려면 LVM 논리 볼륨 제거를 참조하십시오.

절차

1. 볼륨 그룹이 클러스터된 환경에 있는 경우 다른 모든 노드에서 볼륨 **그룹의 잠금** 공간을 중지합니다. 제거를 수행하는 노드를 제외한 모든 노드에서 다음 명령을 사용합니다.

```
# vgchange --lockstop vg-name
```

   잠금이 중지될 때까지 기다립니다.

2. 볼륨 그룹 제거:

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

추가 리소스

- **vgremove(8)** 도움말 페이지

5장. 논리 볼륨의 크기 수정

논리 볼륨을 만든 후에는 볼륨 크기를 수정할 수 있습니다.

5.1. 논리 볼륨 및 파일 시스템 확장

이 절차에서는 논리 볼륨을 확장하고 동일한 논리 볼륨에서 파일 시스템을 확장하는 방법을 설명합니다.

논리 볼륨의 크기를 늘리려면 **lvextend** 명령을 사용합니다. 논리 볼륨을 확장할 때 볼륨을 확장할 양 또는 확장 후 원하는 크기를 나타낼 수 있습니다.

사전 요구 사항

1. 파일 시스템이 있는 기존 LV(논리 볼륨)가 있습니다. the **df -Th** 명령을 사용하여 파일 시스템 유형을 확인합니다.
   LV 및 파일 시스템을 만드는 방법에 대한 자세한 내용은 LVM 논리 볼륨 생성을 참조하십시오.

2. 볼륨 그룹에 LV 및 파일 시스템을 확장할 충분한 공간이 있습니다. **vgs -o 이름,vgfree** 명령을 사용하여 사용 가능한 공간을 확인합니다.

절차

1. 선택 사항: 볼륨 그룹에 LV를 늘리기 위한 공간이 충분하지 않은 경우 다음 명령을 사용하여 볼륨 그룹에 새 물리 볼륨을 추가합니다.

   ```
   # vgextend myvg /dev/vdb3
   Physical volume "/dev/vdb3" successfully created.
   Volume group "myvg" successfully extended
   ```

   자세한 내용은 LVM 볼륨 그룹 생성을 참조하십시오.

2. 볼륨 그룹이 충분히 크기 때문에 요구 사항에 따라 다음 단계 중 하나를 실행합니다.

   a. 제공된 크기로 LV를 확장하려면 다음 명령을 사용합니다.

   ```
   # lvextend -L 3G /dev/myvg/mylv
   Size of logical volume myvg/mylv changed from 2.00 GiB (512 extents) to 3.00 GiB (768 extents).
   Logical volume myvg/mylv successfully resized.
   ```

참고

lvextend 명령의 **-r** 옵션을 사용하여 논리 볼륨을 확장하고 기본 파일 시스템의 크기를 단일 명령으로 조정할 수 있습니다.

```
# lvextend -r -L 3G /dev/myvg/mylv
```

주의

동일한 인수와 함께 **lvresize** 명령을 사용하여 논리 볼륨을 확장할 수도 있지만 이 명령은 실수로 축소되지는 않습니다.

b. *mylv* 논리 볼륨을 확장하여 *myvg* 볼륨 그룹의 할당되지 않은 모든 공간을 채우려면 다음 명령을 사용합니다.

```
# lvextend -l +100%FREE /dev/myvg/mylv
 Size of logical volume myvg/mylv changed from 10.00 GiB (2560 extents) to 6.35 TiB (1665465 extents).
 Logical volume myvg/mylv successfully resized.
```

lvcreate 명령과 마찬가지로 **lvextend** 명령의 **-l** 인수를 사용하여 논리 볼륨의 크기를 늘리기 위해 확장 영역 수를 지정할 수 있습니다. 이 인수를 사용하여 볼륨 그룹의 백분율 또는 볼륨 그룹에서 사용 가능한 나머지 공간의 백분율을 지정할 수도 있습니다.

3. **r** 옵션을 **lvextend** 명령과 함께 사용하지 않는 경우 LV를 확장하고 단일 명령으로 파일 시스템의 크기를 조정한 다음 다음 명령을 사용하여 논리 볼륨에서 파일 시스템의 크기를 조정합니다.

```
xfs_growfs /mnt/mnt1/
meta-data=/dev/mapper/myvg-mylv  isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 262144 to 524288
```

참고

d 옵션이 없으면 **xfs_growfs** 는 파일 시스템을 기본 장치에서 지원하는 최대 크기로 확장합니다. **자세한 내용은 XFS 파일 시스템의 크기 승격을 참조하십시오.**

ext4 파일 시스템 크기를 조정하려면 ext4 파일 시스템 Resizing an ext4 파일 시스템을 참조하십시오.

검증

- **다음 명령을 사용하여 파일 시스템이 확장되고 있는지 확인합니다.**

```
# df -Th
Filesystem          Type      Size  Used Avail Use% Mounted on
devtmpfs            devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs               tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs               tmpfs     1.9G  8.6M  1.9G   1% /run
tmpfs               tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/rhel-root xfs        45G  3.7G   42G   9% /
/dev/vda1           xfs      1014M  369M  646M  37% /boot
tmpfs               tmpfs     374M     0  374M   0% /run/user/0
/dev/mapper/myvg-mylv xfs       2.0G   47M  2.0G   3% /mnt/mnt1
```

추가 리소스

- vgextend(8), lvextend(8) 및 xfs_growfs(8) 도움말 페이지

5.2. 논리 볼륨 축소

lvreduce 명령을 사용하여 논리 볼륨의 크기를 줄일 수 있습니다.

참고

GFS2 또는 XFS 파일 시스템에서 축소는 지원되지 않으므로 GFS2 또는 XFS 파일 시스템이 포함된 논리 볼륨의 크기를 줄일 수 없습니다.

감소하는 논리 볼륨에 파일 시스템이 포함된 경우 데이터 손실을 방지하기 위해 파일 시스템이 감소하는 논리 볼륨의 공간을 사용하지 않도록 해야 합니다. 따라서 논리 볼륨에 파일 시스템이 포함된 경우 lvreduce 명령의 --resizefs 옵션을 사용하는 것이 좋습니다.

이 옵션을 사용하면 lvreduce 명령은 논리 볼륨을 축소하기 전에 파일 시스템을 줄이려고 시도합니다. 파일 시스템이 축소되거나 파일 시스템이 축소를 지원하지 않는 경우와 같이 파일 시스템이 축소되는 경우와 같이 파일 시스템이 축소되는 경우, lvreduce 명령이 실패하고 논리 볼륨을 축소하지 않습니다.

주의

대부분의 경우 **lvreduce** 명령은 가능한 데이터 손실에 대해 경고하고 확인을 요청합니다. 그러나 논리 볼륨이 비활성 상태이거나 **--resizefs** 옵션이 사용되지 않는 경우와 같이 이러한 프롬프트가 표시되지 않으므로 데이터 손실을 방지하기 위해 이러한 확인 프롬프트를 사용하지 않아야 합니다.

lvreduce 명령의 **--test** 옵션을 사용하면 작업이 안전한 위치는 표시되지 않습니다. 이 옵션은 파일 시스템을 확인하거나 파일 시스템 크기 조정을 테스트하지 않기 때문입니다.

절차

- my *vg* 볼륨 그룹의 *mylv* 논리 볼륨을 64MB로 축소하려면 다음 명령을 사용합니다.

```
# lvreduce --resizefs -L 64M myvg/mylv
fsck from util-linux 2.37.2
/dev/mapper/myvg-mylv: clean, 11/25688 files, 4800/102400 blocks
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mapper/myvg-mylv to 65536 (1k) blocks.
The filesystem on /dev/mapper/myvg-mylv is now 65536 (1k) blocks long.

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

이 예에서 *mylv* 에는 논리 볼륨과 함께 크기 조정되는 파일 시스템이 포함되어 있습니다.

- **resize** 값 앞에 **-** 기호를 지정하면 논리 볼륨의 실제 크기에서 값이 뺀 것을 나타냅니다. 논리 볼륨을 절대 크기 64MB로 축소하려면 다음 명령을 사용합니다.

```
# lvreduce --resizefs -L -64M myvg/mylv
```

추가 리소스

- **lvreduce(8)** 도움말 페이지

6장. 논리 볼륨 스냅샷

LVM 스냅샷 기능을 사용하면 서비스 중단 없이 특정 즉시 */dev/sda* 와 같은 볼륨의 가상 이미지를 생성할 수 있습니다.

6.1. 스냅샷 볼륨 개요

스냅샷을 만든 후 원래 볼륨(원본)을 수정하면 스냅샷 기능을 통해 볼륨 상태를 재구성할 수 있도록 변경 전에 수정된 데이터 영역을 복사합니다. 스냅샷을 만들면 원본에 대한 전체 읽기 및 쓰기 권한이 그대로 유지됩니다.

스냅샷은 스냅샷을 만든 후 변경된 데이터 영역만 복사하므로 스냅샷 기능에는 최소한의 스토리지 용량이 필요합니다. 예를 들어 거의 업데이트되지 않는 오리진을 사용하면 원본 용량의 **3 ~ 3 %**가 스냅샷을 유지하기에 충분합니다. 백업 프로시저를 대체하는 것은 아닙니다. 스냅샷 복사본은 가상 복사본이며 실제 미디어 백업이 아닙니다.

스냅샷의 크기는 원본 볼륨에 대한 변경 사항을 저장하기 위한 공간 집합 크기를 제어합니다. 예를 들어 스냅샷을 만든 다음 원본을 완전히 덮어쓰는 경우 최소한 변경 사항을 유지하는 원본 볼륨만큼 스냅샷이 커야 합니다. 스냅샷의 크기를 정기적으로 모니터링해야 합니다. 예를 들어 **/usr** 과 같은 읽기-대부분 볼륨의 수명이 짧은 스냅샷에는 **/home** 과 같은 여러 쓰기가 포함되어 있기 때문에 볼륨의 장기 스냅샷보다 적은 공간이 필요합니다.

스냅샷이 가득 차면 원본 볼륨에서 변경 사항을 더 이상 추적할 수 없으므로 스냅샷이 유효하지 않습니다. 그러나 사용량이 **snapshot_autoextend_threshold** 값을 초과할 때마다 스냅샷을 자동으로 확장하도록 **LVM**을 구성하여 스냅샷이 유효하지 않게 될 수 있습니다. 스냅샷은 완전히 다시 설정할 수 있으며 다음 작업을 수행할 수 있습니다.

- 스토리지 용량이 있는 경우 스냅샷 볼륨의 크기를 늘려 손실되지 않도록 할 수 있습니다.
- 스냅샷 볼륨이 필요한 것보다 크면 볼륨의 크기를 줄여 다른 논리 볼륨에 필요한 공간을 확보할 수 있습니다.

스냅샷 볼륨은 다음과 같은 이점을 제공합니다.

- 대부분의 경우 데이터를 지속적으로 업데이트하는 라이브 시스템을 중단하지 않고 논리 볼륨에서 백업을 수행해야 하는 경우 스냅샷을 만듭니다.

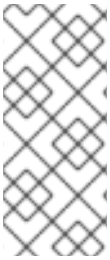
- 스냅샷 파일 시스템에서 **fsck** 명령을 실행하여 파일 시스템의 무결성을 확인하고 원본 파일 시스템에 파일 시스템을 복구해야 하는지 확인할 수 있습니다.

- 스냅샷은 읽기/쓰기이므로 실제 데이터를 사용하지 않고 스냅샷을 만들고 스냅샷에 대해 테스트를 실행하여 프로덕션 데이터에 대해 애플리케이션을 테스트할 수 있습니다.

- **Red Hat Virtualization**과 함께 사용할 **LVM** 볼륨을 생성할 수 있습니다. **LVM** 스냅샷을 사용하여 가상 게스트 이미지의 스냅샷을 생성할 수 있습니다. 이러한 스냅샷을 사용하면 기존 게스트를 쉽게 수정하거나 추가 스토리지를 최소화하여 새 게스트를 만들 수 있습니다.

6.2. 원본 볼륨의 스냅샷 생성

원래 볼륨의 스냅샷을 생성하는 데 필요한 크기가 뒤에 **-s** 또는 **--size** 인수와 함께 **lvcreate** 명령을 사용합니다. 볼륨 스냅샷은 쓸 수 있습니다. 기본적으로 스냅샷 볼륨은 씬 프로비저닝된 스냅샷과 비교하여 일반 활성화 명령 중에 원본으로 활성화됩니다. **LVM**은 원본 볼륨의 크기와 볼륨에 필요한 메타데이터 크기보다 큰 스냅샷 볼륨 생성을 지원하지 않습니다. 이 것보다 큰 스냅샷 볼륨을 지정하면 **LVM**에서 원본 크기에 필요한 스냅샷 볼륨을 만듭니다.



참고

클러스터의 노드는 **LVM** 스냅샷을 지원하지 않습니다. 공유 볼륨 그룹에서 스냅샷 볼륨을 생성할 수 없습니다. 그러나 공유 논리 볼륨에서 일관된 데이터 백업을 생성해야 하는 경우 볼륨을 독점적으로 활성화한 다음 스냅샷을 만들 수 있습니다.

다음 절차에서는 ***origin*** 이라는 원래 논리 볼륨과 ***snap*** 라는 이 원래 볼륨의 스냅샷 볼륨을 생성합니다.

사전 요구 사항

- **VG** ***001*** *볼륨 그룹을 생성* 했습니다. 자세한 내용은 **LVM** 볼륨 그룹 생성을 참조하십시오.

절차

1. 볼륨 그룹 ***vg001*** 에서 ***origin*** 이라는 논리 볼륨을 만듭니다.

```
# lvcreate -L 1G -n origin vg001
Logical volume "origin" created.
```

2.

크기가 *100MB* 인 /dev/vg001/origin의 *snap* of */dev/vg001/origin* 이라는 스냅샷 논리 볼륨을 생성합니다.

```
# lvcreate --size 100M --name snap --snapshot /dev/vg001/origin
  Logical volume "snap" created.
```

원래 논리 볼륨에 파일 시스템이 포함된 경우 원래 파일 시스템이 계속 업데이트되는 동안 파일 시스템의 콘텐츠에 액세스하여 백업을 실행하기 위해 임의의 디렉터리에 **snapshot** 논리 볼륨을 마운트할 수 있습니다.

3.

사용 중인 스냅샷 볼륨의 현재 백분율 및 원본 볼륨을 표시합니다.

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---  1.00g                                       /dev/sde1(0)
  snap vg001  swi-a-s--- 100.00m     origin 0.00                            /dev/sde1(256)
```

모든 스냅샷 논리 볼륨 및 **lvdisplay** */dev/vg001/origin* 명령을 사용하여 활성 또는 비활성과 같은 논리 볼륨 */dev/vg001/origin*의 상태를 표시할 수도 있습니다.

주의

원본 볼륨이 변경되면 스냅샷 볼륨의 크기가 정기적으로 증가하기 때문에 **lvs** 명령으로 스냅샷 볼륨의 비율이 가득 차지 않도록 정기적으로 모니터링하는 것이 중요합니다. 원본의 변경되지 않은 부분에 대한 쓰기 작업은 스냅샷을 손상시키지 않고 성공하지 못하므로 **100%** 가득 차 있는 스냅샷이 완전히 손실됩니다.

4.

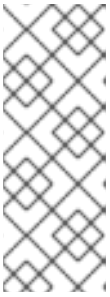
사용량이 **snapshot_autoextend_threshold** 값을 초과하면 스냅샷을 자동으로 확장하도록 **LVM**을 구성하여 스냅샷이 **100%** 가득 차면 유효하지 않게 할 수 있습니다. **/etc/lvm.conf** 파일에서 **snapshot_autoextend_threshold** 및 **snapshot_autoextend_percent** 옵션의 기존 값을 보고 요구 사항에 따라 편집합니다.

다음 예제에서는 **snapshot_autoextend_threshold** 옵션을 스냅샷 볼륨 확장 요구 사항에 따라 **100**보다 작은 값 및 **snapshot_autoextend_percent** 옵션을 값으로 설정합니다.

```
# vi /etc/lvm.conf
snapshot_autoextend_threshold = 70
snapshot_autoextend_percent = 20
```

다음 명령을 실행하여 이 스냅샷을 수동으로 확장할 수도 있습니다.

```
# lvextend -L+100M /dev/vg001/snap
```



참고

이 기능을 사용하려면 볼륨 그룹에 할당되지 않은 공간이 필요합니다. 스냅샷의 자동 확장은 스냅샷에 필요한 최대 계산된 크기 이상으로 스냅샷 볼륨의 크기를 늘리지 않습니다. 스냅샷이 원본을 덮을 정도로 충분히 크게 증가하면 자동 확장을 위해 더 이상 모니터링되지 않습니다.

추가 리소스

- lvcreate(8), lvextend(8), lvs(8) 매뉴얼 페이지
- /etc/lvm/lvm.conf 파일

6.3. 원래 볼륨에 스냅샷 병합

lvconvert 명령을 --merge 옵션과 함께 사용하여 스냅샷을 원래(원본) 볼륨에 병합합니다. 데이터 또는 파일이 손실되거나 시스템을 이전 상태로 복원해야 하는 경우 시스템 롤백을 수행할 수 있습니다. 스냅샷 볼륨을 병합한 후 결과 논리 볼륨에 원본 볼륨의 이름, 마이너 번호, UUID가 있습니다. 병합이 진행 중인 동안 원본에 대한 읽기 또는 쓰기는 병합되는 스냅샷으로 지시된 대로 나타납니다. 병합이 완료되면 병합된 스냅샷이 제거됩니다.

origin 및 snapshot 볼륨이 모두 열려 있지 않고 활성 상태가 아니면 병합이 즉시 시작됩니다. 그렇지 않으면 병합은 origin 또는 snapshot이 활성화되고 둘 다 닫은 후에 시작됩니다. 원본 볼륨이 활성화된 후(예: 루트 파일 시스템) 종료할 수 없는 원본으로 스냅샷을 병합할 수 있습니다.

절차

1. 스냅샷 볼륨을 병합합니다. 다음 명령은 스냅샷 볼륨 *vg001/snap* 을 원본으로 병합합니다.

```
# lvconvert --merge vg001/snap
Merging of volume vg001/snap started.
  vg001/origin: Merged: 100.00%
```

2. *origin 볼륨을 확인합니다.*

```
# lvs -a -o +devices
  LV     VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001 owi-a-s---  1.00g                                      /dev/sde1(0)
```

추가 리소스

- *lvconvert(8) 도움말 페이지*

7장. 씬 프로비저닝된 볼륨 생성 및 관리(오인 볼륨)

Red Hat Enterprise Linux는 씬 프로비저닝된 스냅샷 볼륨 및 논리 볼륨을 지원합니다.

논리 볼륨 및 스냅샷 볼륨은 씬 프로비저닝할 수 있습니다.

- 씬 프로비저닝된 논리 볼륨을 사용하여 사용 가능한 물리 스토리지보다 큰 논리 볼륨을 생성할 수 있습니다.
- 씬 프로비저닝된 스냅샷 볼륨을 사용하면 동일한 데이터 볼륨에 더 많은 가상 장치를 저장할 수 있습니다.

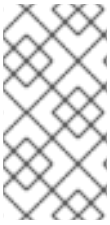
7.1. 씬 프로비저닝 개요

최신 스토리지 스택은 이제 두꺼운 프로비저닝과 씬 프로비저닝 중에서 선택할 수 있는 기능을 제공합니다.

- 두꺼운 프로비저닝은 실제 사용과 관계없이 블록이 할당되는 기존의 블록 스토리지의 동작을 제공합니다.
- 씬 프로비저닝은 데이터를 저장하는 물리적 장치보다 크기가 커질 수 있는 대규모 블록 스토리지 풀을 프로비저닝할 수 있는 기능을 부여합니다. 개별 블록이 실제로 사용될 때까지 할당되지 않으므로 오버 프로비저닝이 가능합니다. 동일한 풀을 공유하는 씬 프로비저닝 장치가 여러 개 있는 경우 이러한 장치를 오버 프로비저닝할 수 있습니다.

씬 프로비저닝을 사용하면 물리적 스토리지를 과도하게 커밋할 수 있으며 대신 씬 풀이라는 여유 공간 풀을 관리할 수 있습니다. 애플리케이션에 필요한 경우 임의의 수의 장치에 이 씬 풀을 할당할 수 있습니다. 스토리지 공간을 비용 효율적으로 할당하는 데 필요할 때 씬 풀을 동적으로 확장할 수 있습니다.

예를 들어, **10**명의 사용자가 애플리케이션에 대해 **100GB** 파일 시스템을 요청하는 경우, 각 사용자에 대해 **100GB** 파일 시스템으로 표시되는 항목을 만들 수 있지만 필요한 경우에만 사용되는 실제 스토리지에서는 지원을 받을 수 있습니다.



참고

씬 프로비저닝을 사용할 때는 스토리지 풀을 모니터링하고 사용 가능한 물리 공간이 부족해질 때 용량을 추가하는 것이 중요합니다.

다음은 씬 프로비저닝 장치를 사용할 때 몇 가지 이점입니다.

- *사용 가능한 물리 스토리지보다 큰 논리 볼륨을 생성할 수 있습니다.*
- *동일한 데이터 볼륨에 더 많은 가상 장치를 저장할 수 있습니다.*
- *논리적으로 확장될 수 있고 자동으로 확장되어 데이터 요구사항을 지원할 수 있는 파일 시스템을 생성할 수 있으며, 사용되지 않은 블록은 풀의 파일 시스템에서 사용하기 위해 풀로 반환됩니다.*

다음은 씬 프로비저닝 장치를 사용할 때의 잠재적인 단점입니다.

- *씬 프로비저닝된 볼륨에는 사용 가능한 물리적 스토리지가 부족하여 실행할 위험이 있습니다. 기본 스토리지가 과도하게 프로비저닝된 경우 사용 가능한 물리적 스토리지가 부족하여 중단될 수 있습니다. 예를 들어 백업용으로 **1T** 물리적 스토리지만 사용하여 씬 프로비저닝된 스토리지 **10T**를 생성하면 **1T**가 모두 소진된 후 볼륨을 사용할 수 없거나 쓸 수 없게 됩니다.*
- *씬 프로비저닝 장치 이후 볼륨에 볼륨을 전송하지 않으면 사용 계정이 정확하지 않습니다. 예를 들어 **-o discard** 마운트 옵션 없이 파일 시스템을 배치하고 씬 프로비저닝된 장치 상단에 **fstrim** 을 실행하지 않는 경우 이전에 사용된 스토리지가 할당되지 않습니다. 이 경우 실제로 사용하지 않아도 시간이 지남에 따라 프로비저닝된 전체 양을 사용하게 됩니다.*
- *사용 가능한 물리적 공간이 부족하려면 논리 및 물리적 사용을 모니터링해야 합니다.*
- *쓰기(**CoW**) 작업의 복사는 스냅샷을 사용하여 파일 시스템에서 느려질 수 있습니다.*
- *데이터 블록은 여러 파일 시스템 간에 혼합되어 있어 최종 사용자에게 표시되지 않는 경우에도 기본 스토리지의 임의의 액세스 제한 사항을 유발할 수 있습니다.*

7.2. 씬 프로비저닝된 논리 볼륨 생성

씬 프로비저닝된 논리 볼륨을 사용하여 사용 가능한 물리 스토리지보다 큰 논리 볼륨을 생성할 수 있습니다. 씬 프로비저닝된 볼륨 세트를 생성하면 시스템에서 요청된 전체 스토리지 용량을 할당하는 대신 사용자가 사용하는 항목을 할당할 수 있습니다.

lvcreate 명령의 **-T** 또는 **--thin** 옵션을 사용하면 씬 풀 또는 씬 볼륨을 생성할 수 있습니다. 단일 명령으로 씬 풀과 씬 볼륨을 동시에 생성하려면 **lvcreate** 명령의 **-T** 옵션을 사용할 수도 있습니다. 이 절차에서는 씬 프로비저닝된 논리 볼륨을 생성하고 확장하는 방법을 설명합니다.

사전 요구 사항

- 볼륨 그룹을 생성했습니다. 자세한 내용은 *LVM 볼륨 그룹 생성을* 참조하십시오.

절차

1. 씬 풀을 생성합니다.

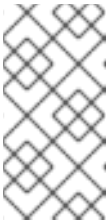
```
# lvcreate -L 100M -T vg001/mythinpool
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

물리 공간 풀을 생성하므로 풀 크기를 지정해야 합니다. **lvcreate** 명령의 **-T** 옵션은 인수를 사용하지 않습니다. 명령을 사용하여 추가한 다른 옵션에서 생성할 장치 유형을 결정합니다. 다음 예와 같이 추가 매개변수를 사용하여 씬 풀을 생성할 수도 있습니다.

- **lvcreate** 명령의 **--thinpool** 매개변수를 사용하여 씬 풀을 생성할 수도 있습니다. **-T** 옵션과 달리 **--thinpool** 매개 변수에는 생성 중인 씬 풀 논리 볼륨의 이름을 지정해야 합니다. 다음 예제에서는 **--thinpool** 매개 변수를 사용하여 **VG 001** 의 크기가 **100M** 인 볼륨 그룹 **mythinpool** 에 **mythinpool**이라는 씬 풀을 생성합니다.

```
# 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.
```

- 풀 생성 시 스트라이핑이 지원되면 **-i** 및 **-I** 옵션을 사용하여 스트라이프를 만들 수 있습니다. 다음 명령을 실행하면 볼륨 그룹의 **thinpool**이라는 **100M** 씬 풀 을 생성합니다. 다음 명령은 **64kB** 스트라이프 및 청크 크기가 **256kB**. 또한 **vg001/thinvolume** 이라는 **1T** 씬 볼륨도 생성합니다.



참고

볼륨 그룹에 사용 가능한 공간이 충분한 물리 볼륨이 두 개 있거나 씬 풀을 생성할 수 없는지 확인합니다.

```
# lvcreate -i 2 -I 64 -c 256 -L 100M -T vg001/thinpool -V 1T --name thinvolume
```

2.

thin 볼륨을 생성합니다.

```
# 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.
```

이 경우 볼륨이 포함된 풀보다 큰 볼륨의 가상 크기를 지정합니다. 다음 예와 같이 추가 매개변수를 사용하여 씬 볼륨을 생성할 수도 있습니다.

- *씬 볼륨과 씬 풀을 모두 생성하려면 **lvcreate** 명령의 **-T** 옵션을 사용하고 **size** 및 가상 크기 인수를 모두 지정합니다.*

```
# 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.
```

- *남은 사용 가능한 공간을 사용하여 씬 볼륨과 씬 풀을 만들려면 **100%FREE** 옵션을 사용합니다.*

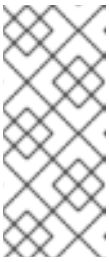
```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool -n thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of
data.
  Logical volume "thinvolume" created.
```

-

*기존 논리 볼륨을 씬 풀 볼륨으로 변환하려면 **lvconvert** 명령의 **--thinpool** 매개 변수를 사용합니다. 기존 논리 볼륨을 씬 풀 볼륨의 메타데이터 볼륨으로 변환하려면 **--thinpool** 매개변수와 함께 **--poolmetadata** 매개변수를 사용해야 합니다.*

*다음 예제에서는 **VG001** 볼륨 그룹의 기존 논리 볼륨 **lv1** 을 씬 풀 볼륨으로 변환하고, **VG 001** 의 기존 논리 볼륨 **lv2** 를 해당 씬 풀 볼륨의 메타데이터 볼륨으로 변환합니다.*

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



참고

*논리 볼륨을 씬 풀 볼륨 또는 씬 풀 메타데이터 볼륨으로 변환하는 경우 **lvconvert** 에서 장치의 내용을 보존하지 않고 콘텐츠를 덮어쓰므로 논리 볼륨의 콘텐츠가 삭제됩니다.*

- *기본적으로 **lvcreate** 명령은 다음 공식에 따라 **thin pool**의 **metadata** 논리 볼륨의 크기를 설정합니다.*

```
Pool_LV_size / Pool_LV_chunk_size * 64
```

*스냅샷 수가 많이 있거나 씬 풀에 작은 청크 크기가 있으므로 나중에 씬 풀 크기의 크기가 상당한 증가를 예상하는 경우 **lvcreate** 명령의 **--poolmetadatasize** 매개변수를 사용하여 **thin pool**의 **metadata** 볼륨의 기본값을 늘려야 할 수 있습니다. 씬 풀의 **metadata** 논리 볼륨에 지원되는 값은 **2MiB**에서 **16GiB** 사이의 범위에 있습니다.*

다음 예제에서는 씬 풀의 메타데이터 볼륨의 기본값을 늘리는 방법을 보여줍니다.

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool --poolmetadatasize 16M -n
thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "thinvolume" created.
```

3. *생성된 **thin pool** 및 **thin** 볼륨을 확인합니다.*

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m
/dev/sda(0)
  mythinpool         vg001 twi-aotz-- 100.00m                 0.00   10.94
mythinpool_tdata(0)
```

```
  [mythinpool_tdata] vg001 Twi-ao---- 100.00m
/dev/sda(1)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume        vg001 Vwi-a-tz--   1.00g mythinpool       0.00
```

4.

*선택 사항: **lvextend** 명령을 사용하여 씬 풀의 크기를 확장합니다. 그러나 씬 풀의 크기를 줄일 수는 없습니다.*

참고

*씬 풀 및 씬 볼륨을 생성하는 동안 **-l 100%FREE** 인수를 사용하면 이 명령이 실패합니다.*

*다음 명령은 다른 **100M** 을 확장하여 크기가 **100M** 인 기존 씬 풀의 크기를 조정합니다.*

```
# lvextend -L+100M vg001/mythinpool
  Size of logical volume vg001/mythinpool_tdata changed from 100.00 MiB (25 extents)
to 200.00 MiB (50 extents).
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (200.00 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 vg001/mythinpool successfully resized
```

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m
/dev/sda(0)
  mythinpool         vg001 twi-aotz-- 200.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(1)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(27)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool       0.00
```

5.

*선택 사항: **thin** 풀 및 **thin** 볼륨의 이름을 변경하려면 다음 명령을 사용합니다.*

```
# lvrename vg001/mythinpool vg001/mythinpool1
```

```
  Renamed "mythinpool" to "mythinpool1" in volume group "vg001"

# lvrename vg001/thinvolume vg001/thinvolume1
  Renamed "thinvolume" to "thinvolume1" in volume group "vg001"
```

이름을 변경한 후 thin 풀 및 thin 볼륨을 확인합니다.

```
# lvs
  LV          VG      Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
mythinpool1 vg001   twi-a-tz 100.00m                    0.00
thinvolume1 vg001   Vwi-a-tz   1.00g mythinpool1         0.00
```

6. 선택 사항: 씬 풀을 제거하려면 다음 명령을 사용합니다.

```
# lvremove -f vg001/mythinpool1
  Logical volume "thinvolume1" successfully removed.
  Logical volume "mythinpool1" successfully removed.
```

추가 리소스

- lvcreate(8), lvrename(8), lvs(8) 및 lvconvert(8) 매뉴얼 페이지

7.3. 씬 프로비저닝된 스냅샷 볼륨

Red Hat Enterprise Linux는 씬 프로비저닝된 스냅샷 볼륨을 지원합니다. 씬 논리 볼륨의 스냅샷도 씬 논리 볼륨(LV)을 생성합니다. 씬 스냅샷 볼륨은 다른 씬 볼륨과 동일한 특성을 갖습니다. 볼륨을 독립적으로 활성화하고, 볼륨 확장, 볼륨 이름 변경, 볼륨 제거, 볼륨 스냅샷도 수행할 수 있습니다.

참고

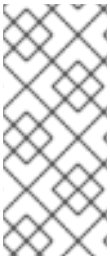
모든 LVM 스냅샷 볼륨 및 모든 씬 볼륨과 유사하게 클러스터의 노드에서 씬 스냅샷 볼륨은 지원되지 않습니다. 스냅샷 볼륨은 하나의 클러스터 노드에서만 독점적으로 활성화되어야 합니다.

기존 스냅샷은 생성된 각 스냅샷에 새 공간을 할당해야 합니다. 여기서 데이터는 원본의 변경으로 유지됩니다. 하지만 씬 프로비저닝 스냅샷은 원본과 동일한 공간을 공유합니다. thin LV의 스냅샷은 thin LV에 공통된 데이터 블록과 해당 스냅샷이 공유되기 때문에 효율적입니다. thin LV 또는 다른 씬 스냅샷에서 스냅샷을 생성할 수 있습니다. 재귀 스냅샷에 공통된 블록은 씬 풀에서도 공유됩니다.

씬 스냅샷 볼륨은 다음과 같은 이점을 제공합니다.

- 원본의 스냅샷 수를 늘리면 성능에 부정적인 영향을 미칩니다.
- 씬 스냅샷 볼륨은 새 데이터만 작성되고 각 스냅샷에 복사되지 않기 때문에 디스크 사용량을 줄일 수 있습니다.
- 기존 스냅샷 요구 사항인 원본을 사용하여 씬 스냅샷 볼륨을 동시에 활성화할 필요가 없습니다.
- 스냅샷에서 원본을 복원할 때 씬 스냅샷을 병합할 필요가 없습니다. 원본을 제거하고 대신 스냅샷을 사용할 수 있습니다. 기존 스냅샷에는 변경 사항을 저장해야 하는 별도의 볼륨이 있습니다. 즉, 원본을 재설정하기 위해 병합됩니다.
- 기존 스냅샷과 비교하면 허용되는 스냅샷 수가 훨씬 더 제한됩니다.

씬 스냅샷 볼륨 사용에 대한 많은 이점이 있지만 필요에 따라 기존 **LVM** 스냅샷 볼륨 기능이 더 적합할 수 있는 몇 가지 사용 사례가 있습니다. 모든 유형의 볼륨에서 기존 스냅샷을 사용할 수 있습니다. 그러나 **thin-snapshot**을 사용하려면 씬 프로비저닝을 사용해야 합니다.



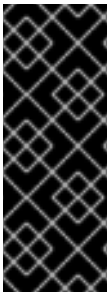
참고

씬 스냅샷 볼륨의 크기를 제한할 수 없습니다. 스냅샷은 필요한 경우 씬 풀의 모든 공간을 사용합니다. 일반적으로 사용할 스냅샷 형식을 결정할 때 사이트의 특정 요구 사항을 고려해야 합니다.

기본적으로 씬 스냅샷 볼륨은 정상적인 활성화 명령 중에 건너뜁니다.

7.4. 씬 프로비저닝된 스냅샷 볼륨 생성

씬 프로비저닝된 스냅샷 볼륨을 사용하면 동일한 데이터 볼륨에 더 많은 가상 장치를 저장할 수 있습니다.



중요

thin 스냅샷 볼륨을 생성할 때 볼륨 크기를 지정하지 마십시오. size 매개변수를 지정하면 생성되는 스냅샷은 씬 스냅샷 볼륨이 아니며 데이터를 저장하는 데 씬 풀을 사용하지 않습니다. 예를 들어 lvcreate -s vg/thinvolume -L10M 명령은 원본 볼륨이 씬 볼륨인 경우에도 씬 스냅샷을 생성하지 않습니다.

씬 스냅샷은 씬 프로비저닝된 원본 볼륨 또는 씬 프로비저닝되지 않은 원본 볼륨에 대해 생성할 수 있습니다. 다음 절차에서는 씬 프로비저닝된 스냅샷 볼륨을 생성하는 다양한 방법을 설명합니다.

사전 요구 사항

- 씬 프로비저닝된 논리 볼륨을 생성했습니다. 자세한 내용은 씬 프로비저닝된 논리 볼륨 생성을 참조하십시오.

절차

- 씬 프로비저닝된 스냅샷 볼륨을 생성합니다. 다음 명령은 씬 프로비저닝된 논리 볼륨 vg001/thinvolume 의 mysnapshot1 이라는 씬 프로비저닝된 스냅샷 볼륨을 생성합니다.

```
# lvcreate -s --name mysnapshot1 vg001/thinvolume
  Logical volume "mysnapshot1" created
```

```
# lvs
  LV          VG      Attr     LSize   Pool       Origin     Data%  Move Log Copy%  Convert
  mysnapshot1 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
```

참고

씬 프로비저닝을 사용할 때는 스토리지 관리자가 스토리지 풀을 모니터링하고 용량을 늘리기 시작하는 것이 중요합니다. 씬 볼륨 크기 확장에 대한 자세한 내용은 씬 프로비저닝된 논리 볼륨 생성을 참조하십시오.

- 프로비저닝되지 않은 논리 볼륨의 씬 프로비저닝된 스냅샷을 생성할 수도 있습니다. 프로비저닝되지 않은 논리 볼륨은 씬 풀 내에 포함되어 있지 않으므로 외부 원본이라고 합니다. 외부 원본 볼륨은 다른 씬 풀에서도 씬 프로비저닝된 여러 스냅샷 볼륨에서 사용하고 공유할 수 있습니다. 씬 프로비저닝된 스냅샷이 생성될 때 외부 원본은 비활성이고 읽기 전용이어야 합니다.

다음 예제에서는 origin_volume 이라는 읽기 전용 비활성 논리 볼륨의 씬 스냅샷 볼륨을 생

성합니다. thin 스냅샷 볼륨의 이름은 mythinsnap 입니다. 그런 다음 논리 볼륨 origin_volume 은 기존 씬 풀을 사용하는 vg001 /pool 그룹의 thin snapshot 볼륨 mythinsnap 에 대한 씬 외부 원본이 됩니다. origin 볼륨은 스냅샷 볼륨과 동일한 볼륨 그룹에 있어야 합니다. origin 논리 볼륨을 지정할 때 볼륨 그룹을 지정하지 마십시오.

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

- 다음 명령을 실행하여 첫 번째 스냅샷 볼륨에 대해 씬 프로비저닝된 두 번째 스냅샷 볼륨을 생성할 수 있습니다.

```
# lvcreate -s vg001/mysnapshot1 --name mysnapshot2
Logical volume "mysnapshot2" created.
```

씬 프로비저닝된 세 번째 스냅샷 볼륨을 생성하려면 다음 명령을 사용합니다.

```
# lvcreate -s vg001/mysnapshot2 --name mysnapshot3
Logical volume "mysnapshot3" created.
```

검증

- 씬 스냅샷 논리 볼륨의 모든 상위 및 하위 항목 목록을 표시합니다.

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
 LV          Ancestors                    Descendants
 mysnapshot2 mysnapshot1,thinvolume       mysnapshot3
 mysnapshot1 thinvolume                   mysnapshot2,mysnapshot3
 mysnapshot3 mysnapshot2,mysnapshot1,thinvolume
 mythinpool
 thinvolume                               mysnapshot1,mysnapshot2,mysnapshot3
```

여기,

- thinvolume 은 VG 001 볼륨 그룹의 원본 볼륨 입니다.
- mysnapshot1 은 thinvolume의 스냅샷입니다.
- mysnapshot2 는 mysnapshot1의 스냅샷입니다.

- ***mysnapshot3** 은 **mysnapshot2**의 스냅샷입니다.*

참고

***lv_ancestors** 및 **lv_descendants** 필드에 기존 종속성이 표시됩니다. 그러나 해당 항목이 체인 중간에서 제거된 경우 종속성 체인을 중단할 수 있는 제거된 항목을 추적하지 않습니다.*

추가 리소스

- ***lvcreate(8)** 도움말 페이지*

8장. LVM 문제 해결

LVM 도구를 사용하여 LVM 볼륨 및 그룹의 다양한 문제를 해결할 수 있습니다.

8.1. LVM에서 진단 데이터 수집

LVM 명령이 예상대로 작동하지 않는 경우 다음과 같은 방법으로 진단을 수집할 수 있습니다.

절차

- 다음 방법을 사용하여 다양한 종류의 진단 데이터를 수집합니다.
  - LVM 명령에 -v 인수를 추가하여 명령 출력의 세부 정보 표시 수준을 높입니다. v를 추가하여 자세한 정보 표시를 늘릴 수 있습니다. 이러한 v의 최대 4가지는 허용됩니다(예: -vvv).
  - /etc/lvm/lvm.conf 구성 파일의 log 섹션에서 level 옵션의 값을 늘립니다. 이로 인해 LVM에서 시스템 로그에 자세한 정보를 제공합니다.
  - 문제가 논리 볼륨 활성화와 관련된 경우 활성화하는 동안 LVM을 활성화하여 메시지를 기록합니다.
    i. /etc/lvm/lvm.conf 구성 파일의 log 섹션에서 activation = 1 옵션을 설정합니다.
    ii. v vv v 옵션을 사용하여 LVM 명령을 실행합니다.
    iii. 명령 출력을 검사합니다.
    iv. 활성화 옵션을 0 으로 재설정합니다.

    옵션을 0 으로 재설정하지 않으면 메모리 부족 상황에서 시스템이 응답하지 않을 수 있습니다.

- 진단 목적으로 정보 덤프를 표시합니다.

```
# lvmdump
```

- 추가 시스템 정보를 표시합니다.

```
# lvs -v
```

```
# pvs --all
```

```
# dmsetup info --columns
```

- **/etc/lvm/backup/** 디렉토리에서 **LVM** 메타데이터의 마지막 백업을 검사하고 **/etc/lvm/archive/** 디렉토리에 보관된 버전을 검사합니다.

- 현재 구성 정보를 확인합니다.

```
# lvmconfig
```

- **/run/lvm/hints** 캐시 파일에서 물리 볼륨이 있는 장치의 레코드를 확인합니다.

추가 리소스

- **lvmdump(8)** 도움말 페이지

8.2. 실패한 LVM 장치에 대한 정보 표시

볼륨이 실패한 이유를 확인하는 데 도움이 될 수 있는 실패한 **LVM** 볼륨에 대한 정보를 표시할 수 있습니다.

절차

- **vgs** 또는 **lvs** 유틸리티를 사용하여 실패한 볼륨을 표시합니다.

예 8.1. 실패한 볼륨 그룹

이 예에서는 **myvg** 볼륨 그룹을 구성하는 장치 중 하나에 실패했습니다. 볼륨 그룹을 사용

할 수 없지만 오류가 발생한 장치에 대한 정보를 볼 수 있습니다.

```
# vgs --options +devices
 /dev/vdb1: open failed: No such device or address
 /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

VG    #PV #LV #SN Attr   VSize  VFree  Devices
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](0)
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/vdb1(0)
```

예 8.2. *실패한 논리 볼륨*

이 예에서는 볼륨 그룹의 논리 볼륨이 실패하여 장치 중 하나가 실패했습니다. 명령 출력에는 실패한 논리 볼륨이 표시됩니다.

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

  /dev/vdb1: open failed: No such device or address
  /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

  LV    VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert Devices
  mylv myvg -wi-a---p- 20.00g                                                     [unknown](0)
[unknown](5120),/dev/sdc1(0)
```

8.3. 볼륨 그룹에서 손실된 LVM 물리 볼륨 제거

물리 볼륨이 실패하면 볼륨 그룹에서 나머지 물리 볼륨을 활성화하고 볼륨 그룹에서 해당 물리 볼륨을 사용한 모든 논리 볼륨을 제거할 수 있습니다.

절차

1. 볼륨 그룹에서 나머지 물리 볼륨을 활성화합니다.

```
# vgchange --activate y --partial myvg
```

2. 제거될 논리 볼륨을 확인합니다.

```
# vgreduce --removemissing --test myvg
```

3. 볼륨 그룹에서 손실된 물리 볼륨을 사용한 논리 볼륨을 모두 제거합니다.

```
# vgreduce --removemissing --force myvg
```

4. 선택 사항: 유지하려는 논리 볼륨을 실수로 제거한 경우 **vgreduce** 작업을 되돌릴 수 있습니다.

```
# vgcfgrestore myvg
```

주의

씬 풀을 제거하면 **LVM**에서 작업을 되돌릴 수 없습니다.

8.4. 누락된 LVM 물리 볼륨의 메타데이터 찾기

물리 볼륨의 볼륨 그룹의 메타데이터 영역을 실수로 덮어쓰거나 삭제하는 경우 메타데이터 영역이 올바르지 않거나 특정 **UUID**가 있는 물리 볼륨을 찾을 수 없음을 나타내는 오류 메시지가 표시됩니다.

이 절차에서는 누락되거나 손상된 물리 볼륨의 보관된 최신 메타데이터를 찾습니다.

절차

1. 물리 볼륨이 포함된 볼륨 그룹의 아카이브 메타데이터 파일을 찾습니다. 아카이브된 메타데이터 파일은 **/etc/lvm/archive/volume-group-name_backup-number.vg** 경로에 있습니다.

```
# cat /etc/lvm/archive/myvg_00000-1248998876.vg
```

00000-1248998876 을 **backup-number**로 바꿉니다. 볼륨 그룹의 번호가 가장 높은 마지막으로 알려진 유효한 메타데이터 파일을 선택합니다.

2. 물리 볼륨의 **UUID**를 찾습니다. 다음 방법 중 하나를 사용합니다.

- 논리 볼륨을 나열합니다.

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

 Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
```

- 아카이브된 메타데이터 파일을 검사합니다. 볼륨 그룹 구성의 **physical_volumes** 섹션에서 **id =** 라는 값으로 **UUID**를 찾습니다.

- **partial** 옵션을 사용하여 볼륨 그룹을 비활성화 합니다.

```
# vgchange --activate n --partial myvg

 PARTIAL MODE. Incomplete logical volumes will be processed.
 WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
 WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/vdb1).
 0 logical volume(s) in volume group "myvg" now active
```

8.5. LVM 물리 볼륨에서 메타데이터 복원

이 절차에서는 새 장치로 손상되거나 교체되는 물리 볼륨에 메타데이터를 복원합니다. 물리 볼륨의 메타데이터 영역을 다시 작성하여 물리 볼륨에서 데이터를 복구할 수 있습니다.

주의

작동 중인 **LVM** 논리 볼륨에서는 이 절차를 시도하지 마십시오. 잘못된 **UUID**를 지정하면 데이터가 손실됩니다.

사전 요구 사항

- *누락된 물리 볼륨의 메타데이터를 확인했습니다. 자세한 내용은 누락된 LVM 물리 볼륨의 메타데이터 찾기를 참조하십시오.*

절차

1. *물리 볼륨에서 메타데이터를 복원합니다.*

```
# pvcreate --uuid physical-volume-uuid \
        --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \
        block-device
```

참고

*명령은 **LVM** 메타데이터 영역만 덮어쓰고 기존 데이터 영역에는 영향을 주지 않습니다.*

예 8.3. /dev/vdb1에서 물리 볼륨 복원

*다음 예제는 **/dev/vdb1** 장치를 다음 속성을 사용하여 물리 볼륨으로 레이블을 지정합니다.*

- ***FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**의 **UUID***
- ***VG_00050.vg**에 포함된 메타데이터 정보. 이는 볼륨 그룹의 가장 최근 적절한 아카이브 메타데이터입니다.*

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \
        --restorefile /etc/lvm/archive/VG_00050.vg \
        /dev/vdb1

...
Physical volume "/dev/vdb1" successfully created
```

2. *볼륨 그룹의 메타데이터를 복원합니다.*

```
# vgcfgrestore myvg

 Restored volume group myvg
```

3. 볼륨 그룹의 논리 볼륨을 표시합니다.

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

논리 볼륨은 현재 비활성 상태입니다. 예를 들면 다음과 같습니다.

```
 LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
 mylv myvg   -wi--- 300.00G                       /dev/vdb1 (0),/dev/vdb1(0)
 mylv myvg   -wi--- 300.00G                       /dev/vdb1 (34728),/dev/vdb1(0)
```

4. 논리 볼륨의 세그먼트 유형이 **RAID**인 경우 논리 볼륨을 다시 동기화합니다.

```
# lvchange --resync myvg/mylv
```

5. 논리 볼륨을 활성화합니다.

```
# lvchange --activate y myvg/mylv
```

6. 디스크상의 **LVM** 메타데이터가 최소한 지나치게 많은 공간을 차지하는 경우 이 절차에서는 물리 볼륨을 복구할 수 있습니다. 메타데이터 영역을 넘어서는 메타데이터가 오버라이드된 경우 볼륨의 데이터가 영향을 받을 수 있습니다. **fsck** 명령을 사용하여 해당 데이터를 복구할 수 있습니다.

검증 단계

- 활성 논리 볼륨을 표시합니다.

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

 LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
 mylv myvg   -wi--- 300.00G                       /dev/vdb1 (0),/dev/vdb1(0)
 mylv myvg   -wi--- 300.00G                       /dev/vdb1 (34728),/dev/vdb1(0)
```

8.6. LVM 출력에서 오류 반올림

볼륨 그룹에서 공간 사용량을 보고하는 **LVM** 명령은 보고된 번호를 **2** 진수로 반올림하여 사람이 읽을 수 있는 출력을 제공합니다. 여기에는 **vgdisplay** 및 **vgs** 유틸리티가 포함됩니다.

반올림 결과, 가용 공간의 보고된 값은 볼륨 그룹의 물리 확장 영역보다 클 수 있습니다. 논리 볼륨을 생성하려고 하면 보고된 여유 공간의 크기가 다음과 같은 오류가 발생할 수 있습니다.

```
Insufficient free extents
```

오류를 해결하려면 볼륨 그룹의 사용 가능한 물리 확장 영역 수를 검사해야 하며 이는 사용 가능한 공간의 정확한 값입니다. 그런 다음 확장 영역 수를 사용하여 논리 볼륨을 성공적으로 만들 수 있습니다.

8.7. LVM 볼륨을 만들 때 라운드 오류 방지

LVM 논리 볼륨을 만들 때 논리 볼륨의 논리 확장 영역 수를 지정하여 반올림 오류를 방지할 수 있습니다.

절차

1. 볼륨 그룹에서 사용 가능한 물리 확장 영역의 수를 찾습니다.

```
# vgdisplay myvg
```

예 8.4. 볼륨 그룹에서 사용 가능한 확장 영역

예를 들어 다음 볼륨 그룹에는 **8780** 사용 가능한 물리 확장 영역이 있습니다.

```
--- Volume group ---
 VG Name               myvg
 System ID
 Format                lvm2
 Metadata Areas        4
 Metadata Sequence No  6
 VG Access             read/write
[...]
Free  PE / Size       8780 / 34.30 GB
```

2. 논리 볼륨을 만듭니다. 볼륨 크기를 바이트가 아닌 확장 영역 단위로 입력합니다.

예 8.5. 확장 영역 수를 지정하여 논리 볼륨 만들기

```
# lvcreate --extents 8780 --name mylv myvg
```

예 8.6. 남아 있는 모든 공간을 차지하기 위한 논리 볼륨 만들기

또는 논리 볼륨을 확장하여 볼륨 그룹에서 사용 가능한 나머지 공간의 백분율을 사용할 수 있습니다. 예를 들면 다음과 같습니다.

```
# lvcreate --extents 100%FREE --name mylv myvg
```

검증 단계

- 볼륨 그룹에서 현재 사용하는 확장 영역의 수를 확인합니다.

```
# vgs --options +vg_free_count,vg_extent_count

  VG     #PV #LV #SN  Attr   VSize   VFree  Free  #Ext
  myvg   2   1   0   wz--n- 34.30G    0    0     8780
```