Red Hat Enterprise Linux 9

스토리지 장치 관리

Red Hat Enterprise Linux 9에서 단일 노드 스토리지 배포 및 구성

Last Updated: 2022-08-15

Red Hat Enterprise Linux 9 스토리지 장치 관리

Red Hat Enterprise Linux 9에서 단일 노드 스토리지 배포 및 구성

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 스토리지 장치를 효과적으로 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.

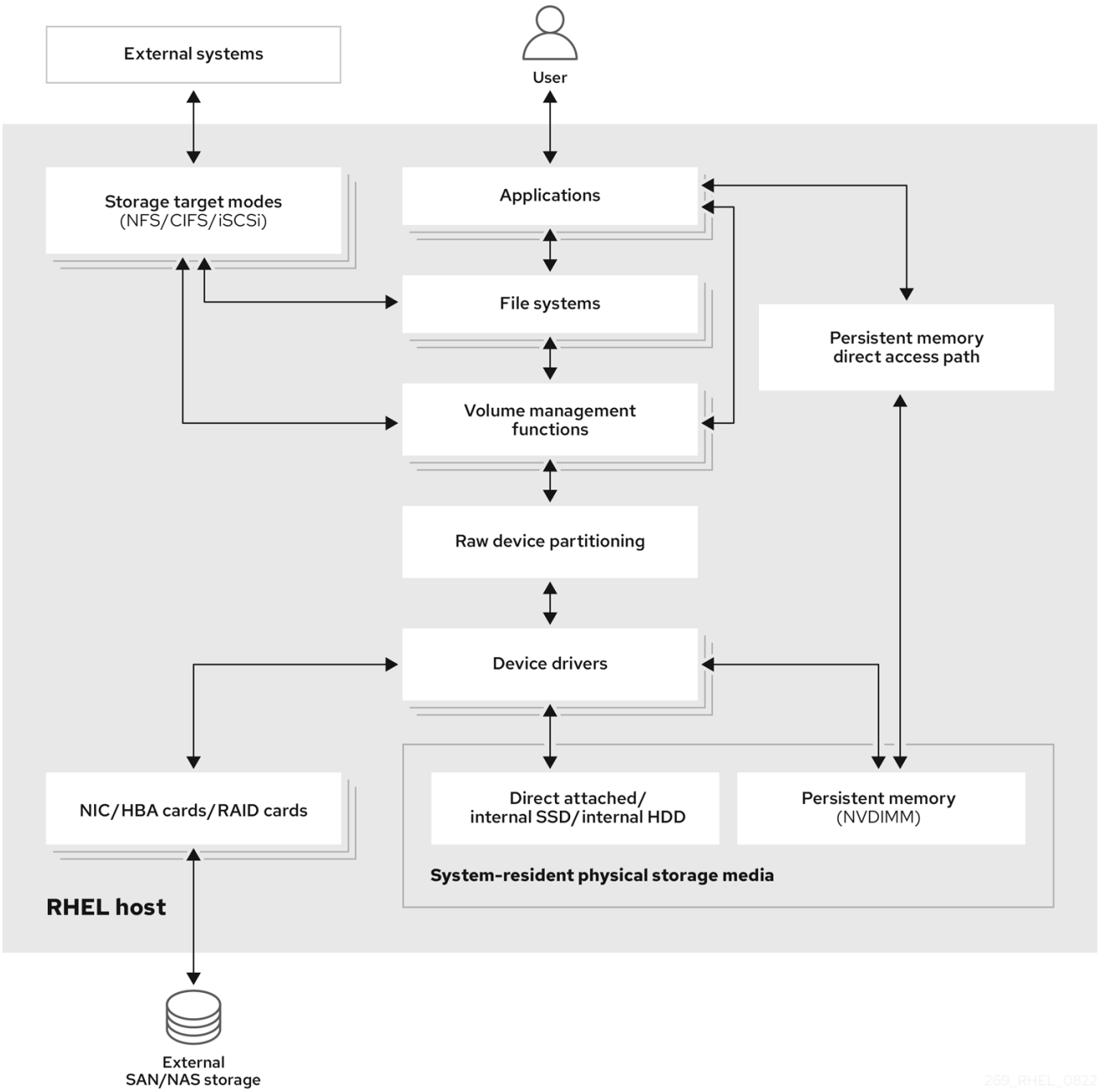
1장. 사용 가능한 스토리지 옵션 개요

Red Hat Enterprise Linux 8에는 여러 로컬, 원격 및 클러스터 기반 스토리지 옵션이 있습니다.

로컬 스토리지는 스토리지 장치가 시스템에 설치되거나 시스템에 직접 연결되어 있음을 의미합니다.

원격 스토리지를 사용하면 LAN, 인터넷 또는 파이버 채널 네트워크를 통해 장치에 액세스합니다. 다음과 같은 높은 수준의 Red Hat Enterprise Linux 스토리지 다이어그램은 다양한 스토리지 옵션을 설명합니다.

그림 1.1. 높은 수준의 Red Hat Enterprise Linux 스토리지 다이어그램



1.1. 로컬 스토리지 개요

Red Hat Enterprise Linux 9는 다양한 로컬 스토리지 옵션을 제공합니다.

기본 디스크 관리

parted 및 **fdisk** 를 사용하여 디스크 파티션을 생성, 수정, 삭제 및 볼 수 있습니다. 다음은 파티션 레이아웃 표준입니다.

마스터 부팅 레코드 (MBR)

BIOS 기반 컴퓨터에서 사용됩니다. 기본, 확장 및 논리 파티션을 생성할 수 있습니다.

GUID 파티션 테이블(GPT)

GUID(Globally Unique identifier)를 사용하며 고유 디스크 및 파티션 GUID를 제공합니다.

파티션을 암호화하려면 Linux Unified Key Setup-on-disk-format(LUKS)을 사용할 수 있습니다. 파티션을 암호화하려면 설치 중에 옵션을 선택하고 암호를 입력하라는 프롬프트가 표시됩니다. 이 암호는 암호화 키를 잠금 해제합니다.

스토리지 사용 옵션

비 Volatile Dual In-line Memory Module (NVDIMM) 관리

이는 메모리 및 스토리지의 조합입니다. 시스템에 연결된 NVDIMM 장치에서 다양한 유형의 스토리지를 활성화하고 관리할 수 있습니다.

블록 스토리지 관리

데이터는 각 블록에 고유 식별자를 갖는 블록 형태로 저장됩니다.

파일 스토리지

데이터는 로컬 시스템의 파일 수준에서 저장됩니다. 이러한 데이터는 NFS 및 SMB를 사용하여 XFS(기본값) 또는 ext4를 사용하여 로컬로 액세스할 수 있으며 네트워크를 통해 로컬로 액세스할 수 있습니다.

논리 볼륨

LVM(Logical Volume Manager)

물리적 장치에서 논리적 장치를 생성합니다. 논리 볼륨(LV)은 물리 볼륨(PV) 및 볼륨 그룹(VG)의 조합입니다. LVM을 구성하는 방법은 다음과 같습니다.

- 하드 드라이브에서 PV 생성.
- PV에서 VG 만들기.
- VG에서 LV에 대한 마운트 지점을 할당하는 LV 만들기.

VDO(가상 데이터 최적화 도구)

중복 제거, 압축 및 씬 프로비저닝을 사용하여 데이터 감소에 사용됩니다. VDO 아래의 VDO를 사용하면 다음과 같은 이점이 있습니다.

- VDO 볼륨의 확장
- 여러 장치에 걸쳐 VDO 볼륨 확장

로컬 파일 시스템

XFS

기본 RHEL 파일 시스템.

Ext4

레거시 파일 시스템.

Stratis

기술 프리뷰로 사용할 수 있습니다. Stratis는 고급 스토리지 기능을 지원하는 하이브리드 사용자 및 커널 로컬 스토리지 관리 시스템입니다.

1.2. 원격 스토리지 개요

다음은 Red Hat Enterprise Linux 8에서 사용할 수 있는 원격 스토리지 옵션입니다.

스토리지 연결 옵션

iSCSI

RHEL 9는 IdP 툴을 사용하여 iSCSI 스토리지 상호 연결을 추가, 제거, 보기 및 모니터링합니다.

파이버 채널(FC)

RHEL 9에서는 다음과 같은 기본 파이버 채널 드라이버를 제공합니다.

- **lpfc**
- **qla2xxx**
- **Zfcp**

NVMe(Non-volatile Memory Express)

호스트 소프트웨어 유틸리티가 솔리드 스테이트 드라이브와 통신할 수 있는 인터페이스입니다. 다음 유형의 패브릭 전송을 사용하여 패브릭을 통해 NVMe를 구성합니다.

- RDMA(Remote Direct Memory Access)를 사용하는 NVMe over fabric.
- 파이버 채널(FC)을 사용하는 NVMe over fabric

장치 매퍼 다중 경로(DM Multipath)

서버 노드와 스토리지 어레이 간의 여러 I/O 경로를 단일 장치로 구성할 수 있습니다. 이러한 I/O 경로는 별도의 케이블, 스위치 및 컨트롤러를 포함할 수 있는 물리적 SAN 연결입니다.

네트워크 파일 시스템

- NFS
- SMB

1.3. GFS2 파일 시스템 개요

Red Hat Global File System 2(GFS2) 파일 시스템은 공유 이름 공간을 제공하고 공통 블록 장치를 공유하는 여러 노드 간에 일관성을 관리하는 64비트 대칭 클러스터 파일 시스템입니다. GFS2 파일 시스템은 로컬 파일 시스템에 최대한 가까운 기능 세트를 제공하는 반면, 동시에 노드 간에 전체 클러스터 일관성을 적용할 수 있습니다. 이를 위해 노드는 파일 시스템 리소스에 클러스터 전체 잠금 체계를 사용합니다. 이 잠금 방식은 TCP/IP와 같은 통신 프로토콜을 사용하여 잠금 정보를 교환합니다.

어떤 경우에는 Linux 파일 시스템 API에서 GFS2의 클러스터형 특성을 완전히 투명하게 사용할 수 없습니다. 예를 들어, GFS2에서 POSIX 잠금을 사용하는 프로그램은 클러스터형 환경에서 **GETLK** 함수를 사용하지 않아야 합니다. 그러나 대부분의 경우 GFS2 파일 시스템의 기능은 로컬 파일 시스템의 기능과 동일합니다.

Red Hat Enterprise Linux 복구 스토리지 애드온은 GFS2를 제공하며 Red Hat Enterprise Linux High Availability Add-On을 통해 GFS2에 필요한 클러스터 관리를 제공합니다.

gfs2.ko 커널 모듈은 GFS2 파일 시스템을 구현하고, GFS2 클러스터 노드에 로드됩니다.

GFS2에서 최상의 성능을 얻으려면 기본 설계에서 줄인 성능 고려 사항을 고려해야 합니다. 로컬 파일 시스템과 마찬가지로 GFS2는 자주 사용하는 데이터의 로컬 캐싱으로 성능을 개선하기 위해 페이지 캐시를 사용합니다. 클러스터의 노드에서 일관성을 유지하기 위해 *glock* 상태 시스템에서 캐시 제어를 제공합니다.

추가 리소스

- GFS2 파일 시스템 구성

1.4. GLUSTER STORAGE 개요

RHGS(Red Hat Gluster Storage)는 클러스터에 배포할 수 있는 소프트웨어 정의 스토리지 플랫폼입니다. 여러 서버의 디스크 스토리지 리소스를 단일 글로벌 네임스페이스로 집계합니다. GlusterFS는 클라우드 및 하이브리드 솔루션에 적합한 오픈 소스 분산 파일 시스템입니다.

볼륨은 GlusterFS의 기반을 형성하고 다른 요구 사항을 제공합니다. 각 볼륨은 신뢰할 수 있는 스토리지 풀의 서버의 내보내기 디렉터리로 표시되는 기본 스토리지 단위인 brick 컬렉션입니다.

다음 유형의 GlusterFS 볼륨을 사용할 수 있습니다.

- **분산 GlusterFS 볼륨**은 각 파일이 하나의 brick에 저장되는 기본 볼륨이며 파일은 다른 brick 간에 공유할 수 없습니다.
- **복제 GlusterFS 볼륨** 유형은 사용자 데이터를 복제하므로 하나의 brick에 오류가 발생하면 데이터에 계속 액세스할 수 있습니다.
- **분산 복제 GlusterFS 볼륨**은 다수의 시스템에 복제본을 배포하는 하이브리드 볼륨입니다. 스토리지 확장성 및 높은 신뢰성이 중요한 환경에 적합합니다.

추가 리소스

- Red Hat gluster 스토리지 관리 가이드

1.5. CEPH STORAGE 개요

RHCS(Red Hat Ceph Storage)는 Ceph 스토리지 시스템의 가장 안정적인 버전을 Ceph 관리 플랫폼, 배포 유틸리티 및 지원 서비스와 결합하는 확장 가능한 오픈 소프트웨어 정의 스토리지 플랫폼입니다.

Red Hat Ceph Storage는 클라우드 인프라 및 웹 스케일 오브젝트 스토리지를 위해 설계되었습니다. Red Hat Ceph Storage 클러스터는 다음과 같은 유형의 노드로 구성됩니다.

Red Hat Ceph Storage Ansible 관리 노드

이 유형의 노드는 이전 버전의 Red Hat Ceph Storage에서 수행했던 기존 Ceph 관리 노드로 작동합니다. 이 유형의 노드는 다음과 같은 기능을 제공합니다.

- 중앙 집중식 스토리지 클러스터 관리
- Ceph 구성 파일 및 키

- 보안상의 이유로 인터넷에 액세스할 수 없는 노드에 Ceph를 설치하기 위한 로컬 리포지토리 (선택 사항)

노드 모니터링

각 모니터 노드는 클러스터 맵의 사본을 유지 관리하는 모니터 데몬(**ceph-mon**)을 실행합니다. 클러스터 맵에는 클러스터 토폴로지가 포함됩니다. Ceph 클러스터에 연결하는 클라이언트는 모니터에서 클러스터 맵의 현재 사본을 검색하여 클라이언트가 클러스터에 데이터를 읽고 쓸 수 있습니다.

중요

Ceph는 하나의 모니터를 사용하여 실행할 수 있지만 프로덕션 클러스터에서 고가용성을 보장하기 위해 Red Hat은 최소 3개의 모니터 노드가 있는 배포만 지원합니다. Red Hat은 스토리지 클러스터에 대해 총 5개의 Ceph Monitor를 750 OSD를 초과할 것을 권장합니다.

OSD 노드

각 OSD(오브젝트 스토리지 장치) 노드는 노드에 연결된 논리 디스크와 상호 작용하는 Ceph OSD 데몬(**ceph-osd**)을 실행합니다. Ceph는 이러한 OSD 노드에 데이터를 저장합니다.
Ceph는 매우 적은 OSD 노드로 실행할 수 있습니다. 기본값은 3개이지만, 스토리지 클러스터에서는 50개의 OSD(예: 스토리지 클러스터에서 50개의 OSD)를 모드형 확장에서 시작하는 더 나은 성능을 실현할 수 있습니다. Ceph 클러스터에는 CRUSH 맵을 생성하여 격리된 장애 도메인을 허용하는 여러 OSD 노드가 있는 것이 좋습니다.

MDS 노드

각 메타데이터 서버(MDS) 노드는 MDS 데몬(**ceph-mds**)을 실행하여 Ceph 파일 시스템(CephFS)에 저장된 파일과 관련된 메타데이터를 관리합니다. MDS 데몬은 공유 클러스터에 대한 액세스도 조정합니다.

오브젝트 게이트웨이 노드

Ceph Object Gateway 노드는 Ceph RADOS Gateway 데몬(**ceph-radosgw**)을 실행하고, **librados** 위에 구축된 개체 스토리지 인터페이스로, Ceph Storage 클러스터에 RESTful 게이트웨이를 제공합니다. Ceph Object Gateway는 다음 두 가지 인터페이스를 지원합니다.

S3

Amazon S3 RESTful API의 대규모 하위 집합과 호환되는 인터페이스가 포함된 오브젝트 스토리지 기능을 제공합니다.

Swift

OpenStack Swift API의 대규모 하위 집합과 호환되는 인터페이스가 포함된 오브젝트 스토리지 기능을 제공합니다.

추가 리소스

- Red Hat Ceph Storage

2장. 디스크 파티션

디스크를 하나 이상의 논리 영역으로 분할하려면 디스크 파티션 유틸리티를 사용합니다. 각 파티션을 별도로 관리할 수 있습니다.

2.1. 파티션 개요

하드 디스크는 파티션 테이블의 각 디스크 파티션의 위치와 크기에 대한 정보를 저장합니다. 파티션 테이블의 정보를 사용하여 운영 체제는 각 파티션을 논리 디스크로 처리합니다. 디스크 파티션의 몇 가지 장점은 다음과 같습니다.

- 물리 볼륨의 관리 감독 가능성 감소
- 충분한 백업 확인
- 효율적인 디스크 관리 제공

추가 리소스

- 직접 또는 LVM을 사용하여 LUN에서 파티셔닝을 사용하는 경우의 장점과 단점은 무엇입니까? ?

2.2. 디스크에서 파티션을 수정하기 전에 고려 사항

디스크 파티션을 생성, 제거 또는 크기 조정하기 전에 다음 측면을 고려하십시오.

장치에서 파티션 테이블의 유형에 따라 개별 파티션의 최대 수와 크기가 결정됩니다.

최대 파티션 수:

- **마스터 부팅 레코드(MBR)** 파티션 테이블을 사용하여 포맷된 장치에서 다음을 수행할 수 있습니다.
  - 최대 4개의 기본 파티션.
  - 최대 3개의 기본 파티션, 즉 하나의 확장 파티션
    - 확장 파티션 내의 여러 논리 파티션
- **GUID 파티션 테이블(GPT)** 으로 포맷된 장치의 경우 다음을 수행할 수 있습니다.
  - **parted** 유틸리티를 사용하는 경우 최대 128개의 파티션이 있습니다.
    - GPT 사양은 파티션 테이블의 예약된 크기를 늘려 파티션을 더 허용하지만 parted 유틸리티는 128 파티션에 필요한 영역을 제한합니다.

최대 파티션 크기:

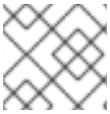
- **Master Boot Record (MBR)** 파티션 테이블로 포맷된 장치의 최대 크기는 TiB입니다.
- **GUID 파티션 테이블(GPT)** 으로 포맷된 장치의 최대 크기는 8ZiB입니다.

parted 유틸리티를 사용하여 여러 다른 접미사를 사용하여 파티션 크기를 지정할 수 있습니다.

- **MiB, GiB 또는 TiB**
  - 2의 힘으로 표시된 크기입니다.

- 파티션의 출발점은 크기에 따라 지정된 정확한 섹터에 맞춰져 있습니다.
- 마지막 포인트는 지정된 크기 - 1 섹터에 맞춰집니다.

- **MB, GB 또는 TB**
  - 10의 힘으로 표시된 크기입니다.
  - 시작점과 종료점은 지정된 유닛의 1/2에 맞춰져 있습니다. 예를 들어 MB 접미사를 사용하는 경우 500KB입니다.



참고

이 섹션에서는 IBM Z 아키텍처에 특정한 DASD 파티션 테이블에는 적용되지 않습니다.

추가 리소스

- IBM Z에서 Linux 인스턴스 구성
- DASD에 대해 알아야 할 사항

2.3. 파티션 테이블 유형 비교

장치에서 파티션을 활성화하려면 다른 유형의 파티션 테이블로 블록 장치를 포맷합니다. 다음 표에서는 블록 장치에서 만들 수 있는 다양한 유형의 파티션 테이블의 속성을 비교합니다.

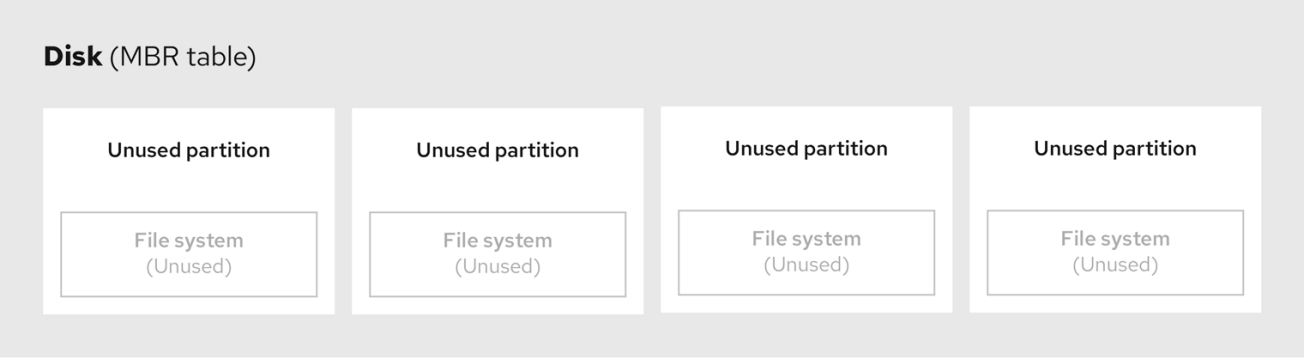
표 2.1. 파티션 테이블 유형

파티션 테이블	최대 파티션 수	최대 파티션 크기
마스터 부팅 레코드 (MBR)	4 기본 또는 3 기본 파티션 및 12 개의 논리 파티션이있는 확장 파티션	2TiB
GUID 파티션 테이블(GPT)	128	8ZiB

2.4. GPT 디스크 파티션

파티션 테이블은 디스크 시작 부분에 저장된 파일 시스템 또는 사용자 데이터 앞에 저장됩니다. 보다 명확한 예를 들어 다음 다이어그램에서 파티션 테이블이 별도의 것으로 표시됩니다.

그림 2.1. GPT 파티션 테이블이 있는 디스크



이전 다이어그램에서 볼 수 있듯이 파티션 테이블은 사용되지 않은 4개의 기본 파티션 4개의 섹션으로 나뉩니다. 기본 파티션은 하나의 논리 드라이브(또는 섹션)만 포함하는 하드 드라이브의 파티션입니다. 각 논리 드라이브에는 단일 파티션을 정의하는 데 필요한 정보가 있습니다. 즉 파티션 테이블이 4개 이상의 기본 파티션을 정의할 수 없습니다.

각 파티션 테이블 항목에는 파티션의 중요한 특성이 포함되어 있습니다.

- 파티션이 시작되고 끝나는 디스크 지점
- 파티션의 상태는 하나의 파티션만 **active**로 플래그를 지정할 수 있으므로
- 파티션 유형입니다.

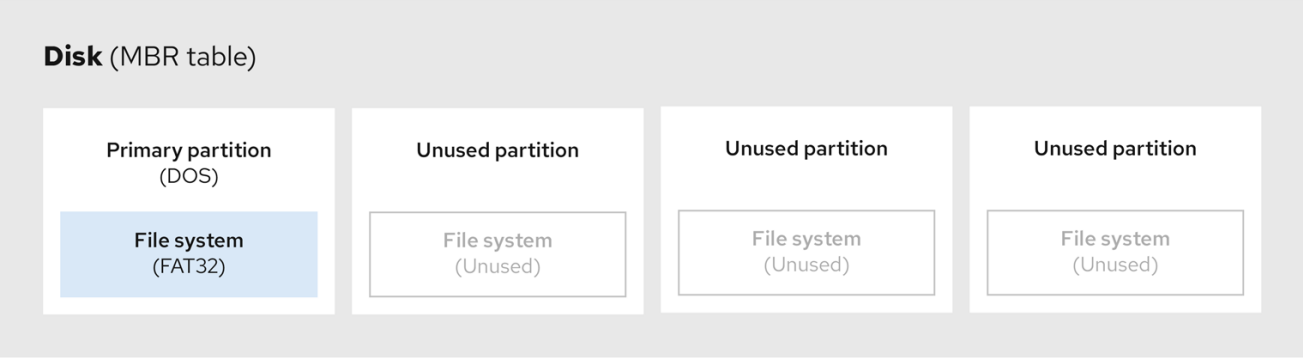
시작 및 종료 포인트는 디스크의 파티션의 크기와 위치를 정의합니다. 일부 운영 체제 부트 로더는 **활성** 플래그를 사용합니다. 즉, "활성"으로 표시된 파티션의 운영 체제가 부팅됩니다.

유형은 파티션의 예상 사용을 식별하는 번호입니다. 일부 운영 체제는 다음과 같이 파티션 유형을 사용합니다.

- 특정 파일 시스템 유형을 나타냅니다.
- 파티션을 특정 운영 체제와 관련된 플래그
- 파티션에 부팅 가능한 운영 체제가 포함되어 있음을 나타냅니다.

다음 다이어그램에서는 단일 파티션이 있는 드라이브의 예를 보여줍니다. 이 예에서 첫 번째 파티션은 idrac 파티션 **유형으로** 레이블이 지정됩니다.

그림 2.2. 단일 파티션이 있는 디스크



269_RHEL_0822

추가 리소스

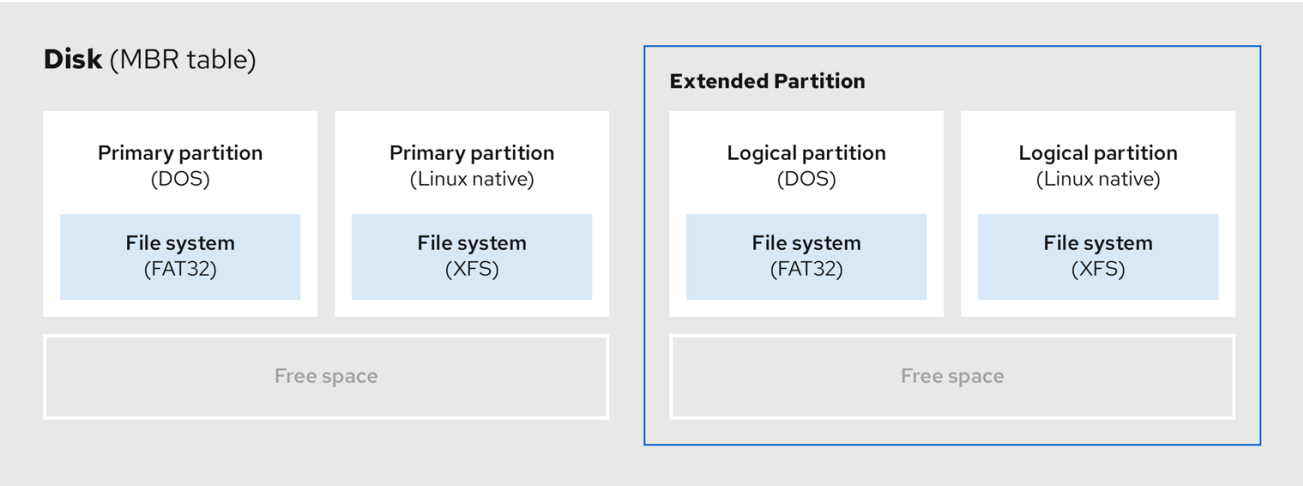
- GPT 파티션 유형

2.5. 확장된 GPT 파티션

필요한 경우 추가 파티션을 만들려면 유형을 **확장** 로 설정합니다.

확장 파티션은 디스크 드라이브와 유사합니다. 확장 파티션에 전체적으로 포함된 하나 이상의 논리 파티션을 가리키는 자체 파티션 테이블이 있습니다. 다음 다이어그램에서는 두 개의 기본 파티션이 있는 디스크 드라이브와 파티션되지 않은 여유 공간과 함께 두 개의 논리 파티션을 포함하는 확장 파티션 하나를 보여줍니다.

그림 2.3. 기본 파티션과 확장된 reserved 파티션이 둘 다 있는 디스크



269_RHEL_0822

최대 4개의 기본 파티션과 확장 파티션만 보유할 수 있지만 논리 파티션 수에는 고정된 제한이 없습니다. Linux에서 파티션에 액세스하기 위한 제한으로 단일 디스크 드라이브는 최대 15개의 논리 파티션을 허용합니다.

2.6. GPT 파티션 유형

아래 표는 가장 일반적으로 사용되는 STATUS 파티션 유형 및 16진수 숫자 중 일부를 나타내는 목록을 보여줍니다.

표 2.2. GPT 파티션 유형

iDRAC 파티션 유형	값	iDRAC 파티션 유형	값
empty	00	7.7 Netware 386	65
idrac 12비트 FAT	01	PIC/IX	75
XENIX 루트	O2	이전 MINIX	80
XENIXSecurityGroup	O3	Linux/MINUX	81
idrac 16-bit databind32M	04	Linux swap	82
확장	05	Linux 네이티브	83
DOS 16-bit >=32	06	Linux 확장	85
OS/2 HPFS	07	amoeba	93
AIX	08	Amoeba 2.7T	94
AIX 부팅 가능	09	BSD/386	a5
OS/2 Boot Manager	0a	OpenBSD	a6
Win95 FAT32	0b	NEXTSTEP	a7
Win95 FAT32 (LBA)	0c	BSDI fs	b7
Win95 FAT16 (LBA)	0e	BSDI 스왑	b8
Win95 Extended (LBA)	0f	Syrinx	c7
Venix 80286	40	CP/M	db
NetNamespace	51	idrac 액세스	e1
Prep Boot	41	DOS R/O	e3
GNU HURD	63	idrac secondary	f2
controlPlane Netware 286	64	BBT	ff

2.7. GUID 파티션 테이블

GUID 파티션 테이블(GPT)은 GUID(Globally Unique Identifier)를 기반으로 하는 파티션 구성입니다.

GPT는 GPT 파티션 테이블의 제한 사항을 다룹니다. iDRAC 파티션 테이블은 약 2.2TB와 동등한 2TiB 이상의 스토리지를 처리할 수 없습니다. 대신 GPT는 용량이 큰 하드 디스크를 지원합니다. 최대 주소 가능한 디스크 크기는 2.2ZiB이며, 기본적으로 GPT는 최대 128개의 기본 파티션을 만들 수 있습니다. 이 수는 파티션 테이블에 더 많은 공간을 할당하여 확장할 수 있습니다.

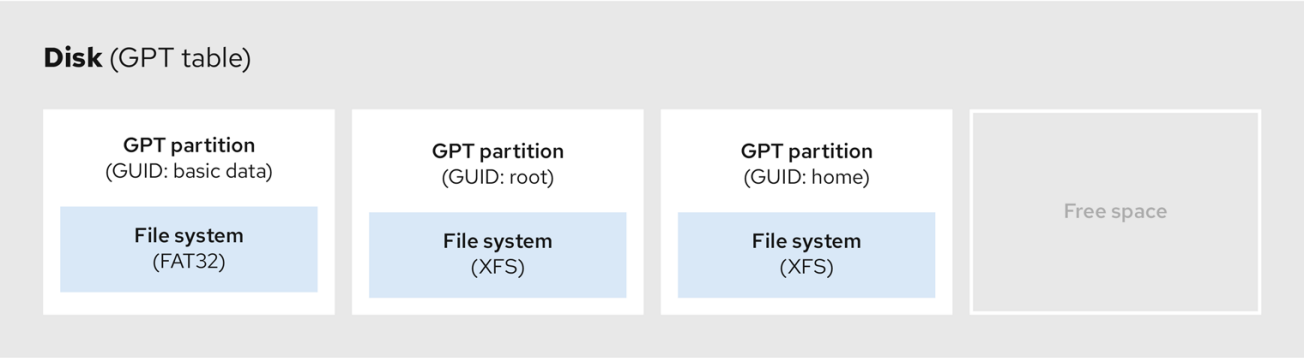
참고

GPT에는 GUID를 기반으로 하는 파티션 유형이 있습니다. 특정 파티션에는 특정 GUID가 필요합니다. 예를 들어 EFI 부트 로더의 시스템 파티션에는 GUID **C12A7328-F81F-11D2-BA4B-00A0C93EC93B** 가 필요합니다.

GPT 디스크는 다음과 같이 논리 블록 주소 지정(LBA)과 파티션 레이아웃을 사용합니다.

- GPT 디스크 이전 버전과의 호환성을 위해 GPT의 첫 번째 섹터(LBA 0)는 GPT 데이터용으로 예약되어 있으며 "Protective reserved"라고 합니다.
- 기본 GPT
  - 헤더는 장치의 두 번째 논리 블록 (LBA 1)에서 시작됩니다. 헤더에는 디스크 GUID, 기본 파티션 테이블의 위치, 보조 GPT 헤더의 위치, CRC32 체크섬, 기본 파티션 테이블이 포함되어 있습니다. 또한 테이블의 파티션 항목 수를 지정합니다.
  - 기본적으로 기본 GPT에는 128개의 파티션 항목이 포함되어 있습니다. 각 파티션에는 128바이트, 파티션 유형 GUID 및 고유한 파티션 GUID의 입력 크기가 있습니다.
- secondary GPT
  - 복구를 위해 기본 파티션 테이블이 손상된 경우 백업 테이블로 유용합니다.
  - 보조 GPT 헤더는 디스크의 마지막 논리 섹터에 있으며 기본 헤더가 손상된 경우 GPT 정보를 복구합니다.
  - 다음이 포함됩니다.
    - 디스크 GUID
    - 보조 파티션 테이블 및 기본 GPT 헤더의 위치
    - CRC32 체크섬s
    - 보조 파티션 테이블
    - 사용 가능한 파티션 항목 수

그림 2.4. GUID 파티션 테이블이 있는 디스크



중요

GPT 디스크에 부트 로더를 성공적으로 설치하려면 BIOS 부팅 파티션이 있어야 합니다. 디스크에 이미 BIOS 부팅 파티션이 포함된 경우 다시 사용할 수 있습니다. 여기에는 **Anaconda** 설치 프로그램에서 초기화한 디스크가 포함됩니다.

2.8. 파티션 유형

파티션 유형을 관리하는 방법은 여러 가지가 있습니다.

- &lt ;.> 유틸리티는 16진수 코드를 지정하여 파티션 유형의 전체 범위를 지원합니다.
- 장치 생성기 유틸리티인 **systemd-gpt-auto-generator** 는 파티션 유형을 사용하여 장치를 자동으로 식별하고 마운트합니다.
- **parted** 유틸리티는 *플래그* 와 함께 파티션 유형을 매핑합니다. **parted** 유틸리티는 LVM, 스왑 또는 RAID와 같은 파티션 유형(예: LVM) 파티션 유형만 처리합니다.
  parted 유틸리티는 다음 플래그 설정을 지원합니다.
  - **boot**
  - **root**
  - **swap**
  - 숨김
  - **RAID**
  - **lvm**
  - **lba**
  - **legacy_boot**
  - **irst**
  - **esp**
  - **Palo**

parted 3.5가 있는 Red Hat Enterprise Linux 9에서는 추가 플래그 Chrome **os_kernel 및 bls_ boot** 를 사용할 수 있습니다.

parted 유틸리티는 선택적으로 파티션을 생성하는 동안 파일 시스템 유형 인수를 허용합니다. 필수 조건 목록을 보려면 parted로 파티션 생성을 참조하십시오. 다음 값을 사용합니다.

- GPT에 파티션 플래그를 설정합니다.
- GPT에서 파티션 UUID 유형을 설정합니다. 예를 들어 **스왑**,**fat** 또는 **hfs** 파일 시스템 유형은 다른 GUID를 설정합니다. 기본값은 Linux Data GUID입니다.

인수는 파티션의 파일 시스템을 변경하지 않습니다. 지원되는 플래그와 GUID만 구분합니다.

다음 파일 시스템 유형이 지원됩니다.

- **xfs**
- **ext2**
- **ext3**
- **ext4**
- **fat16**
- **fat32**
- **hfs**
- **hfs+**
- **linux-swap**
- **ntfs**
- **reiserfs**

2.9. 파티션 이름 지정 체계

Red Hat Enterprise Linux는 **/dev/xy*N*** 형식의 파일 이름과 함께 파일 기반 이름 지정 체계를 사용합니다.

장치 및 파티션 이름은 다음 구조로 구성됩니다.

/dev/

모든 장치 파일이 들어 있는 디렉터리의 이름입니다. 하드 디스크에는 파티션이 포함되어 있으므로 가능한 모든 파티션을 나타내는 파일은 **/dev** 에 있습니다.

xx

파티션 이름의 처음 두 문자는 파티션을 포함하는 장치의 유형을 나타냅니다.

y

이 문자는 파티션을 포함하는 특정 장치를 나타냅니다. 예를 들어 첫 번째 하드 디스크인 **/dev/sda** 와 두 번째 하드 디스크의 **/dev/sdb** 입니다. 26개 이상의 드라이브(예: **/dev/sdaa** 1)가 있는 시스템에서 더 많은 문자를 사용할 수 있습니다.

N

마지막 문자는 파티션을 나타내는 숫자를 나타냅니다. 처음 4개(기본값 또는 확장) 파티션은 **1** 에서 **4**

까지 번호가 매겨집니다. 논리 파티션은 **5** 에서 시작합니다. 예를 들어 **/dev/sda3** 은 첫 번째 하드 디스크의 세 번째 기본 또는 확장 파티션이며 **/dev/sdb6** 은 두 번째 하드 디스크의 두 번째 논리 파티션입니다. 드라이브 파티션 번호 지정은 GPT 파티션 테이블에만 적용됩니다. ***N*** 이 항상 파티션을 의미하지는 않습니다.

참고

Red Hat Enterprise Linux가 *모든* 유형의 디스크 파티션을 식별하고 참조할 수 있지만 파일 시스템을 읽지 못하므로 모든 파티션 유형에서 저장된 데이터에 액세스하지 못할 수 있습니다. 그러나 대부분의 경우 다른 운영 체제에 전용 파티션의 데이터에 성공적으로 액세스할 수 있습니다.

2.10. 마운트 지점 및 디스크 파티션

Red Hat Enterprise Linux에서 각 파티션은 스토리지의 일부를 형성하며 단일 파일 및 디렉터리를 지원하는 데 필요합니다. 파티션을 마운트하면 *마운트 지점* 이라는 지정된 디렉터리에서 시작하여 해당 파티션을 스토리지를 사용할 수 있습니다.

예를 들어 파티션 **/dev/sda5** 가 /usr/ 에 마운트된 경우 **/usr/** 아래의 모든 파일 및 디렉터리가 **/dev/** sda5 에 물리적으로 상주함을 의미합니다. **/usr/share/doc/FAQ/txt/Linux-FAQ** 파일은 **/dev/sda5** 에 있지만 **/etc/gdm/custom.conf** 파일은 그렇지 않습니다.

이 예제를 계속 사용하면 **/usr/** 아래의 하나 이상의 디렉터리가 다른 파티션의 마운트 지점이 될 수도 있습니다. 예를 들어 **/usr/local/man/whatis** 는 **/dev/sda5** 가 아닌 **/dev/sda7** 에 있습니다. **/usr/local** 에 마운트된 **/dev/sda7** 파티션이 포함된 경우.

3장. 파티션 시작하기

디스크 파티셔닝을 사용하여 각 파티션에서 개별적으로 작업할 수 있는 하나 이상의 논리 영역으로 디스크를 나눕니다. 하드 디스크는 파티션 테이블의 각 디스크 파티션의 위치와 크기에 대한 정보를 저장합니다. 이 테이블을 사용하면 각 파티션이 운영 체제에 대한 논리 디스크로 표시됩니다. 그런 다음 개별 디스크를 읽고 쓸 수 있습니다.

블록 장치에서 파티션을 사용하기 위한 이점과 단점에 대한 개요 는 직접 또는 LVM에 있는 LVM을 통해 LUN에서 파티셔닝을 사용하는 데 따른 장단점 을 참조하십시오.

3.1. PARTED를 사용하여 디스크에서 파티션 테이블 만들기

parted 유틸리티를 사용하여 파티션 테이블로 블록 장치를 보다 쉽게 포맷합니다.

주의

파티션 테이블을 사용하여 블록 장치를 포맷하면 장치에 저장된 모든 데이터가 삭제됩니다.

절차

1. 대화형 **parted** 쉘을 시작합니다.

```
# parted block-device
```

2. 장치에 이미 파티션 테이블이 있는지 확인합니다.

```
# (parted) print
```

장치에 이미 파티션이 포함된 경우 다음 단계에서 삭제됩니다.

3. 새 파티션 테이블을 만듭니다.

```
# (parted) mklabel table-type
```

- *table-type* 을 원하는 파티션 테이블 유형으로 바꿉니다.
  - **df용** MSDOS
  - GPT용 GPT

예 3.1. GUID 파티션 테이블(GPT) 만들기

디스크에 GPT 테이블을 만들려면 다음을 사용합니다.

```
# (parted) mklabel gpt
```

이 명령을 입력한 후 변경 사항이 적용되기 시작합니다.

4. 파티션 테이블을 보고 해당 테이블이 생성되었는지 확인합니다.

```
# (parted) print
```

5. **parted** 쉘을 종료합니다.

```
# (parted) quit
```

추가 리소스

- **parted(8)** 도움말 페이지.

3.2. PARTED로 파티션 테이블 보기

블록 장치의 파티션 테이블을 표시하여 파티션 레이아웃과 개별 파티션에 대한 세부 정보를 확인합니다. **parted** 유틸리티를 사용하여 블록 장치에서 파티션 테이블을 볼 수 있습니다.

절차

1. **parted** 유틸리티를 시작합니다. 예를 들어 다음 출력에는 **/dev/sda** 가 나열됩니다.

```
# parted /dev/sda
```

2. 파티션 테이블 보기:

```
# (parted) print

Model: ATA SAMSUNG MZNLN256 (scsi)
Disk /dev/sda: 256GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  269MB   268MB   primary   xfs          boot
 2      269MB   34.6GB  34.4GB  primary
```

```
3     34.6GB  45.4GB  10.7GB  primary
4     45.4GB  256GB   211GB   extended
5     45.4GB  256GB   211GB   logical
```

3. 선택 사항: 다음을 검사하려는 장치로 전환합니다.

```
# (parted) select block-device
```

출력 명령 출력에 대한 자세한 설명은 다음을 참조하십시오.

모델: ATASampleSUNG MZNN256 (scsi)

디스크 유형, 제조업체, 모델 번호, 인터페이스.

디스크 /dev/sda: 256GB

블록 장치 및 스토리지 용량의 파일 경로입니다.

파티션 테이블: msdos

디스크 레이블 유형입니다.

숫자

파티션 번호입니다. 예를 들어, 마이너 번호가 1인 파티션은 **/dev/sda1** 에 해당합니다.

시작 및 종료

파티션이 시작되고 끝나는 장치의 위치입니다.

유형

유효한 유형은 **metadata, free, primary, extended** 또는 **logical**입니다.

파일 시스템

파일 시스템 유형입니다. 장치의 파일 시스템 필드에 값이 없음을 나타내는 경우 해당 파일 시스템 유형이 알 수 없음을 의미합니다. **parted** 유틸리티는 암호화된 장치의 파일 시스템을 인식할 수 없습니다.

플래그

파티션에 설정된 플래그를 나열합니다. 사용 가능한 플래그는 부팅,루트,스왑,숨겨진,**raid,lvm** 또는 **ba** 입니다.

추가 리소스

- **parted(8)** 도움말 페이지.

3.3. PARTED로 파티션 생성

시스템 관리자는 **parted** 유틸리티를 사용하여 디스크에 새 파티션을 만들 수 있습니다.

참고

필요한 파티션은 스왑,/boot/, 및 / (root) 입니다.

사전 요구 사항

- 디스크의 파티션 테이블입니다.
- 생성하려는 파티션이 **2TiB**보다 크면 **GUID** 파티션 테이블(GPT) 으로 디스크를 포맷합니다.

절차

1. **parted** 유틸리티를 시작합니다.

```
# parted block-device
```

2. 사용 가능한 공간이 충분한지 확인하려면 현재 파티션 테이블을 확인합니다.

```
# (parted) print
```

   - 사용 가능한 공간이 충분하지 않은 경우 파티션의 크기를 조정합니다.
   - 파티션 테이블에서 다음을 확인합니다.
     - 새 파티션의 시작 및 끝점입니다.

- GPT에서 파티션 유형은 무엇입니까.

3. 새 파티션을 만듭니다.

```
# (parted) mkpart part-type name fs-type start end
```

- *part-type* 을 기본,논리 또는 확장으로 바꿉니다. 이는 **reserved** 파티션 테이블에만 적용됩니다.
- *이름을* 임의의 파티션 이름으로 바꿉니다. 이는 **GPT** 파티션 테이블에 필요합니다.
- ***fs-type*** 을 **xfs,ext2,ext3,ext4,fat16,fat32,hfs,hfs+, linux-swap,ntfs** 또는 **reiserfs** 로 바꿉니다. ***fs-type*** 매개변수는 선택 사항입니다. **parted** 유틸리티는 파티션에 파일 시스템을 생성하지 않습니다.
- ***start*** 및 ***end*** 를 파티션의 시작 및 종료 지점을 결정하는 크기로 바꾸고 디스크 시작부터 계산합니다. **512MiB,20GiB** 또는 **1.5TiB** 와 같은 크기 접미사를 사용할 수 있습니다. 기본 크기는 메가바이트입니다.

예 3.2. 작은 기본 파티션 만들기

1024MiB에서 **STATUS** 테이블의 기본 파티션을 **2048MiB**까지 만들려면 다음을 사용합니다.

```
# (parted) mkpart primary 1024MiB 2048MiB
```

변경 사항은 명령을 입력한 후 적용을 시작합니다.

4. 파티션 테이블을 보고 생성된 파티션이 올바른 파티션 유형, 파일 시스템 유형 및 크기가 있는 파티션 테이블에 있는지 확인합니다.

```
# (parted) print
```

5. **parted** 쉘을 종료합니다.

```
# (parted) quit
```

6. 새 장치 노드를 등록합니다.

```
# udevadm settle
```

7. 커널이 새 파티션을 인식하는지 확인합니다.

```
# cat /proc/partitions
```

추가 리소스

- **parted(8)** 도움말 페이지.
- parted를 사용하여 디스크에서 파티션 테이블 만들기
- parted로 파티션 크기 조정

3.4. <.>을 사용하여 파티션 유형 설정

<.> 유틸리티를 사용하여 파티션 유형 또는 플래그를 설정할 수 있습니다.

사전 요구 사항

- 디스크의 파티션입니다.

절차

1. 대화형 **<.&gt ;** 쉘을 시작합니다.

```
# fdisk block-device
```

2. 현재 파티션 테이블을 보고 마이너 파티션 번호를 확인합니다.

```
Command (m for help): print
```

유형 열에서 현재 파티션 유형 과 **Id** 열의 해당 유형 **ID**를 확인할 수 있습니다.

3. 파티션 유형 명령을 입력하고 마이너 번호를 사용하여 파티션을 선택합니다.

```
Command (m for help): type
Partition number (1,2,3 default 3): 2
```

4. 선택 사항**: 16**진수 코드의 목록을 확인합니다.

```
Hex code (type L to list all codes): L
```

5. 파티션 유형을 설정합니다.

```
Hex code (type L to list all codes): 8e
```

6. 변경 사항을 작성하고 **<.>** 쉘 을 종료합니다.

```
Command (m for help): write
The partition table has been altered.
Syncing disks.
```

7. 변경 사항을 확인합니다.

```
# fdisk --list block-device
```

3.5. PARTED로 파티션 크기 조정

parted 유틸리티를 사용하여 사용되지 않은 디스크 공간을 활용하도록 파티션을 확장하거나 다른 용도로 용량을 사용하도록 파티션을 줄입니다.

사전 요구 사항

- 파티션을 축소하기 전에 데이터를 백업합니다.
- 생성하려는 파티션이 **2TiB**보다 크면 **GUID** 파티션 테이블**(GPT)** 으로 디스크를 포맷합니다.

- 파티션을 축소하려면 먼저 크기가 조정된 파티션보다 크지 않도록 파일 시스템을 축소합니다.

참고

XFS는 축소를 지원하지 않습니다.

절차

1. **parted** 유틸리티를 시작합니다.

```
# parted block-device
```

2. 현재 파티션 테이블을 확인합니다.

```
# (parted) print
```

파티션 테이블에서 다음을 확인합니다.

- 파티션의 마이너 번호입니다.
- 크기 조정 후 기존 파티션 및 해당 새 종료 지점의 위치입니다.

3. 파티션의 크기를 조정합니다.

```
# (parted) resizepart 1 2GiB
```

- ***1*** 을 크기 조정 중인 파티션의 마이너 번호로 바꿉니다.
- ***2*** 를 크기가 조정된 파티션의 새 끝 지점을 결정하는 크기로 바꾸고 디스크 처음부터 계산합니다. **512MiB,20GiB** 또는 **1.5TiB** 와 같은 크기 접미사를 사용할 수 있습니다. 기본 크기는 메가바이트입니다.

4. 파티션 테이블을 보고 크기 조정이 올바른 크기의 파티션 테이블에 있는지 확인합니다.

```
# (parted) print
```

5. **parted** 쉘을 종료합니다.

```
# (parted) quit
```

6. 커널이 새 파티션을 등록하는지 확인합니다.

```
# cat /proc/partitions
```

7. 선택 사항: 파티션을 확장한 경우 파일 시스템도 확장합니다.

추가 리소스

- **parted(8)** 도움말 페이지.
- parted를 사용하여 디스크에서 파티션 테이블 만들기
- ext3 파일 시스템 크기 조정
- XFS 파일 시스템의 크기 증가

3.6. PARTED로 파티션 제거

parted 유틸리티를 사용하면 디스크 파티션을 제거하여 디스크 공간을 확보할 수 있습니다.

주의

파티션을 제거하면 파티션에 저장된 모든 데이터가 삭제됩니다.

절차

1. 대화형 **parted** 쉘을 시작합니다.

```
# parted block-device
```

2. 제거할 파티션의 마이너 수를 확인하려면 현재 파티션 테이블을 확인합니다.

```
# (parted) print
```

3. 파티션을 제거합니다.

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

- ***minor-number*** 를 제거하려는 파티션의 마이너 번호로 바꿉니다.

변경 사항은 이 명령을 입력하는 즉시 적용을 시작합니다.

4. 파티션 테이블에서 파티션을 제거했는지 확인합니다.

```
# (parted) print
```

5. **parted** 쉘을 종료합니다.

```
# (parted) quit
```

6. 커널이 파티션이 제거되었는지 확인합니다.

```
# cat /proc/partitions
```

7. **/etc/fstab** 파일이 있는 경우 파티션을 제거합니다. 제거된 파티션을 선언하고 파일에서 제거하는 행을 찾습니다.

8. 시스템이 새 **/etc/fstab** 구성을 등록하도록 다시 마운트 단위를 다시 생성합니다.

```
# systemctl daemon-reload
```

-

9. 스왑 파티션을 삭제하거나 **LVM** 조각을 제거한 경우 **/etc/default/grub** 파일의 커널 명령 줄에서 파티션에 대한 모든 참조를 제거하고 **GRUB** 설정을 다시 생성합니다.

   - **BIOS** 기반 시스템에서 다음을 수행합니다.

     ```
     # grub2-mkconfig --output=/etc/grub2.cfg
     ```

   - **UEFI** 기반 시스템에서 다음을 수행합니다.

     ```
     # grub2-mkconfig --output=/etc/grub2-efi.cfg
     ```

10. 초기 부팅 시스템에 변경 사항을 등록하려면 **initramfs** 파일 시스템을 다시 빌드합니다.

    ```
    # dracut --force --verbose
    ```

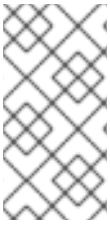
추가 리소스

- **parted(8)** 도움말 페이지

4장. 디스크를 다시 분할하기 위한 전략

디스크를 다시 분할하는 방법은 다양합니다. 여기에는 다음이 포함됩니다.

- 파티션되지 않은 여유 공간을 사용할 수 있습니다.
- 사용되지 않은 파티션을 사용할 수 있습니다.
- 활발하게 사용되는 파티션의 여유 공간을 사용할 수 있습니다.



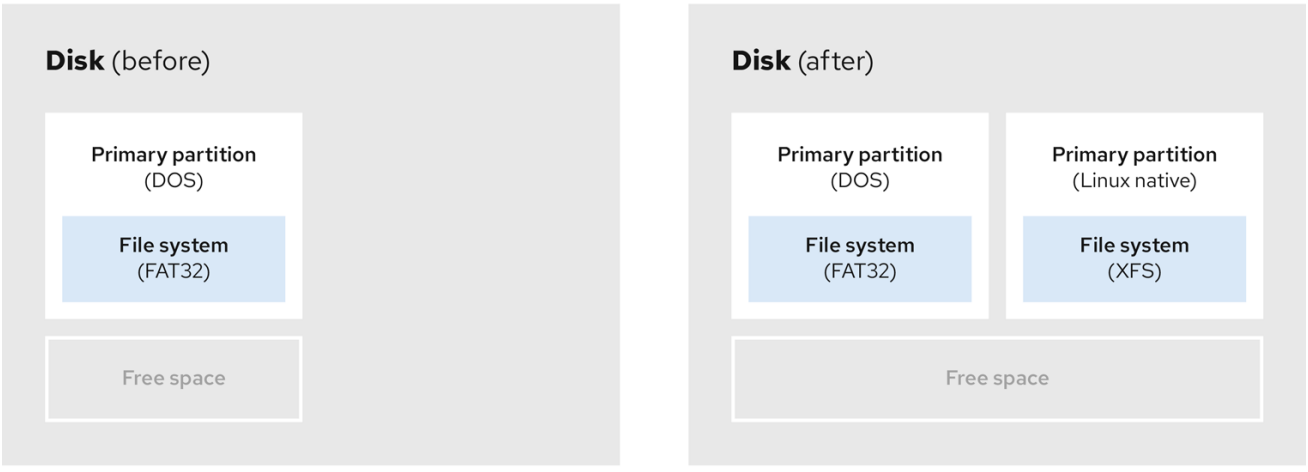
참고

다음 예제는 명확성을 위해 단순화되며 **Red Hat Enterprise Linux**를 실제로 설치할 때 정확한 파티션 레이아웃을 반영하지 않습니다.

4.1. 파티션되지 않은 여유 공간 사용

이미 정의되어 있고 전체 하드 디스크에 걸쳐 있지 않은 파티션은 정의된 파티션에 속하지 않는 할당되지 않은 공간을 남겨 둡니다. 다음 다이어그램에서는 이것이 어떻게 보이는지 보여줍니다.

그림 4.1. 파티션 여유 공간이 파티션되지 않은 디스크



첫 번째 다이어그램은 하나의 기본 파티션과 할당되지 않은 공간이 있는 정의되지 않은 파티션이 있는 디스크를 나타냅니다. 두 번째 다이어그램은 공간이 할당된 두 개의 정의된 파티션이 있는 디스크를 나타

냅니다.

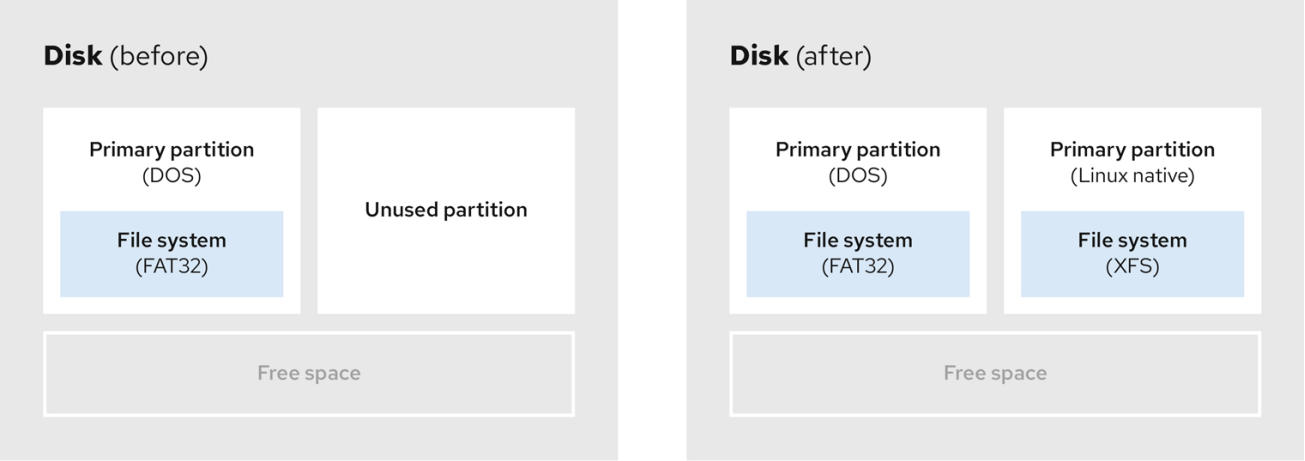
사용되지 않은 하드 디스크도 이 범주에 속합니다. 유일한 차이점은 모든 공간이 정의된 파티션의 일부가 아니라는 것입니다.

새 디스크에서 사용되지 않는 공간을 통해 필요한 파티션을 만들 수 있습니다. 대부분의 사전 설치된 운영 체제는 디스크 드라이브에서 사용 가능한 모든 공간을 차지하도록 구성됩니다.

4.2. 사용되지 않은 파티션의 공간 사용

다음 예에서 첫 번째 다이어그램은 사용되지 않은 파티션이 있는 디스크를 나타냅니다. 두 번째 다이어그램은 **Linux**에서 사용되지 않은 파티션을 찾는 것을 나타냅니다.

그림 4.2. 사용되지 않는 파티션이 있는 디스크



사용되지 않은 파티션에 할당된 공간을 사용하려면 파티션을 삭제한 다음 적절한 **Linux** 파티션을 만듭니다. 또는 설치 프로세스 중에 사용되지 않은 파티션을 삭제하고 새 파티션을 수동으로 만듭니다.

4.3. 활성 파티션의 여유 공간 사용

이 프로세스에는 이미 사용 중인 활성 파티션에 필요한 여유 공간이 포함되어 있으므로 이 프로세스를 관리하기 어려울 수 있습니다. 대부분의 경우 소프트웨어가 사전 설치된 컴퓨터 하드 디스크에는 운영 체제 및 데이터를 보유한 더 큰 파티션이 포함되어 있습니다.

주의

활성 파티션에서 운영 체제(OS)를 사용하려면 OS를 다시 설치해야 합니다. 사전 설치된 소프트웨어를 포함하는 일부 컴퓨터는 원래 OS를 재설치하기 위한 설치 미디어를 포함하지 않습니다. 원래 파티션과 OS 설치를 제거하기 전에 이것이 OS에 적용되는지 확인하십시오.

사용 가능한 여유 공간 사용을 최적화하기 위해, 안전하지 않거나 파괴되지 않은 재파운딩 방법을 사용할 수 있습니다.

4.3.1. 안전하지 않은 재파티션

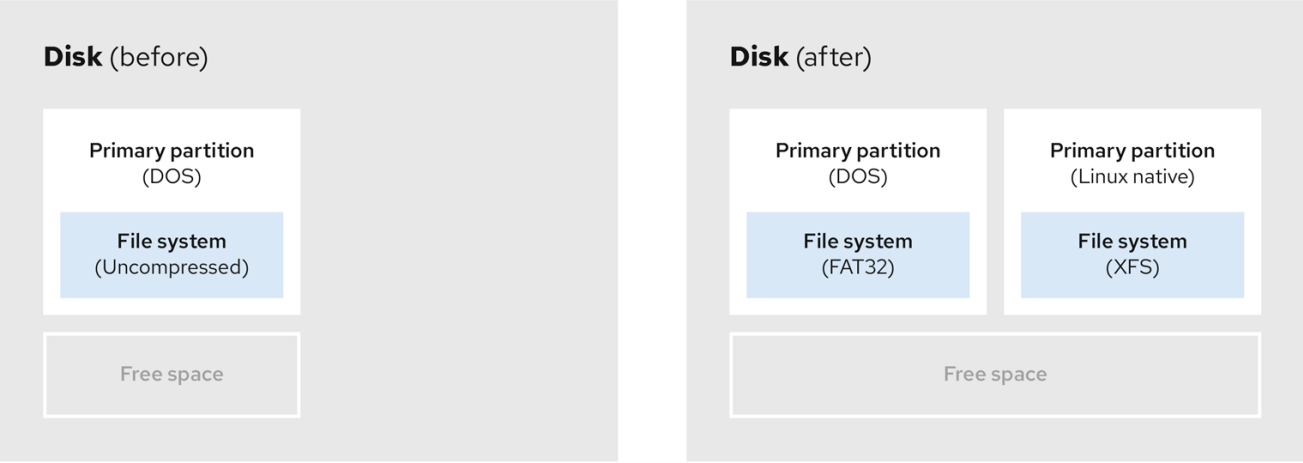
안전하지 않은 재파티션은 하드 드라이브에서 파티션을 제거하고 대신 작은 여러 파티션을 만듭니다. 이 방법을 사용하면 전체 내용이 삭제되므로 원래 파티션에서 필요한 모든 데이터를 백업하십시오.

기존 운영 체제에 대해 더 작은 파티션을 생성한 후 다음을 수행할 수 있습니다.

- 소프트웨어 재설치.
- 데이터를 복원합니다.
- Red Hat Enterprise Linux 설치를 시작하십시오.

다음 다이어그램은 안전하지 않은 repartitioning 방법 사용에 대한 단순화된 표현입니다.

그림 4.3. 디스크에서 안전하지 않은 재파티션 작업



269_RHEL_0822

주의

이 방법은 원래 파티션에 이전에 저장된 모든 데이터를 삭제합니다.

4.3.2. 거부되지 않은 repartitioning

데이터 손실 없이 파티션 재파티브 크기 조정이 지연되지 않습니다. 이 방법은 신뢰할 수 있지만 큰 드라이브에서 처리 시간이 오래 걸립니다.

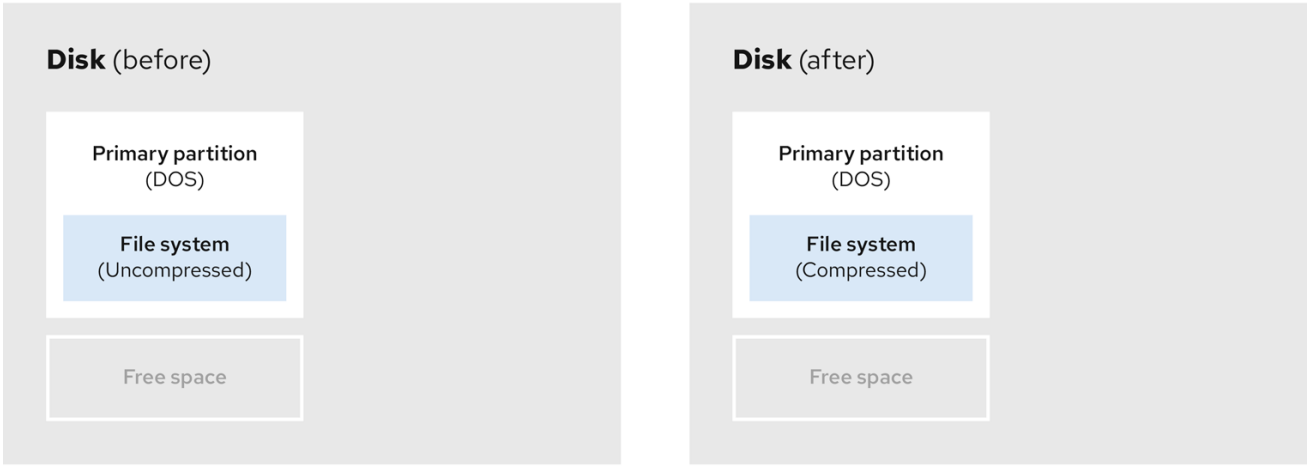
다음은 거부된 복원을 시작하는 데 도움이 될 수 있는 메서드 목록입니다.

- 기존 데이터 압축

일부 데이터의 저장 위치는 변경할 수 없습니다. 이렇게 하면 파티션이 필요한 크기로 크기를 조정하는 것을 방지할 수 있으며 궁극적으로 안전하지 않은 재파티션 프로세스가 발생할 수 있습니다. 이미 존재하는 파티션의 데이터를 압축하면 필요에 따라 파티션의 크기를 조정하는 데 도움이 될 수 있습니다. 또한 사용 가능한 공간을 최대화하는 데 도움이 될 수 있습니다.

다음 다이어그램은 이 프로세스를 간단하게 나타냅니다.

그림 4.4. 디스크의 데이터 압축



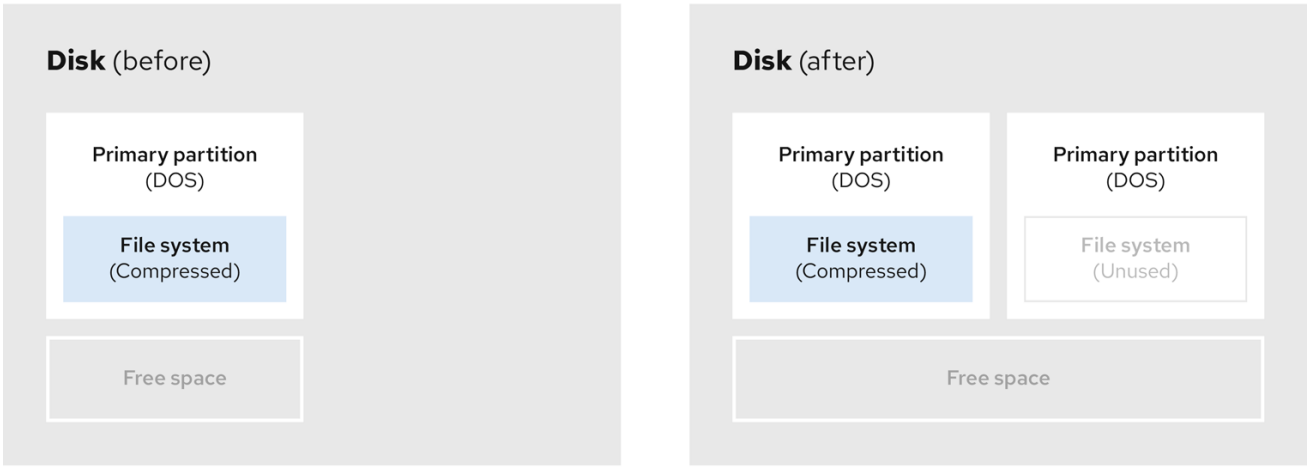
가능한 데이터 손실을 방지하려면 압축 프로세스를 계속하기 전에 백업을 만듭니다.

- 기존 파티션의 크기 조정

기존 파티션의 크기를 조정하면 더 많은 공간을 확보할 수 있습니다. 소프트웨어 크기 조정에 따라 결과가 다를 수 있습니다. 대부분의 경우 원래 파티션과 동일한 유형의 포맷되지 않은 새 파티션을 만들 수 있습니다.

크기 조정 후 수행하는 단계는 사용하는 소프트웨어에 따라 달라질 수 있습니다. 다음 예제에서 가장 좋은 방법은 새 **ClusterTask (Disk Operating System)** 파티션을 삭제하고 대신 **Linux** 파티션을 만드는 것입니다. 크기 조정 프로세스를 시작하기 전에 디스크에 가장 적합한 항목을 확인합니다.

그림 4.5. 디스크의 파티션 크기 조정

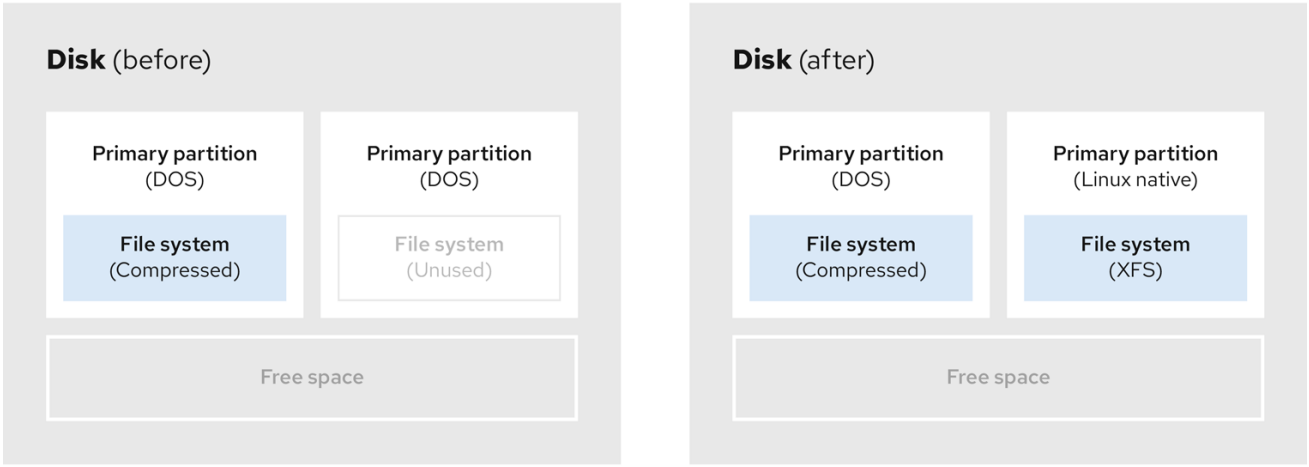


- 선택 사항: 새 파티션 생성

소프트웨어 크기 조정 중 일부는 **Linux** 기반 시스템을 지원합니다. 이러한 경우 크기 조정 후 새로 생성된 파티션을 삭제할 필요가 없습니다. 나중에 새 파티션을 만드는 것은 사용하는 소프트웨어에 따라 다릅니다.

다음 다이어그램은 새 파티션을 만들기 전과 후에 디스크 상태를 나타냅니다.

그림 4.6. 최종 파티션 구성이 있는 디스크



5장. ISCSI 대상 구성

Red Hat Enterprise Linux는 targetcli 쉘을 명령줄 인터페이스로 사용하여 다음 작업을 수행합니다.

- iSCSI 하드웨어를 활용하기 위해 iSCSI 스토리지 상호 연결을 추가, 제거, 보기 및 모니터링합니다.
- 파일, 볼륨, 로컬 SCSI 장치 또는 RAM 디스크에서 지원하는 로컬 스토리지 리소스를 원격 시스템으로 내보냅니다.

targetcli 툴에는 기본 제공 탭 완성, 자동 완성 지원 및 인라인 설명서를 포함한 트리 기반 레이아웃이 있습니다.

5.1. TARGETCLI 설치

targetcli 도구를 설치하여 iSCSI 스토리지 상호 연결을 추가, 모니터링 및 제거합니다.

절차

1. targetcli 툴을 설치합니다.

```
# dnf install targetcli
```

2. 대상 서비스를 시작합니다.

```
# systemctl start target
```

3. 부팅 시 시작할 대상을 설정합니다.

```
# systemctl enable target
```

4. 방화벽에서 포트 3260 을 열고 방화벽 구성을 다시 로드합니다.

```
# firewall-cmd --permanent --add-port=3260/tcp
Success
```

```
# firewall-cmd --reload
Success
```

검증

- **targetcli** 레이아웃을 확인합니다.

```
# targetcli
/> ls
o- /.....[...]
  o- backstores.....[...]
  | o- block.....[Storage Objects: 0]
  | o- fileio.....[Storage Objects: 0]
  | o- pscsi.....[Storage Objects: 0]
  | o- ramdisk.....[Storage Objects: 0]
  o- iscsi.....[Targets: 0]
  o- loopback.....[Targets: 0]
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.2. ISCSI 대상 생성

iSCSI 타겟을 생성하면 클라이언트의 **iSCSI** 이니시에이터가 서버의 스토리지 장치에 액세스할 수 있습니다. 타겟과 이니시에이터 모두 고유한 식별 이름을 갖습니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.

절차

1. **iSCSI** 디렉토리로 이동합니다.

```
/> iscsi/
```

참고

cd 명령은 디렉터리를 변경하고 로 이동할 경로를 나열하는 데 사용됩니다.

2. 다음 옵션 중 하나를 사용하여 **iSCSI** 대상을 생성합니다.

   a. 기본 대상 이름을 사용하여 **iSCSI** 대상 생성:

   ```
   /iscsi> create

   Created target
   iqn.2003-01.org.linux-iscsi.hostname.x8664:sn.78b473f296ff
   Created TPG1
   ```

   b. 특정 이름을 사용하여 **iSCSI** 대상 생성:

   ```
   /iscsi> create iqn.2006-04.com.example:444

   Created target iqn.2006-04.com.example:444
   Created TPG1
   Here iqn.2006-04.com.example:444 is target_iqn_name
   ```

   iqn.2006-04.com.example:444 를 특정 대상 이름으로 바꿉니다.

3. 새로 생성된 대상을 확인합니다.

```
/iscsi> ls

o- iscsi.....[1 Target]
  o- iqn.2006-04.com.example:444.....[1 TPG]
    o- tpg1.....[enabled, auth]
      o- acls.....[0 ACL]
       o- luns.....[0 LUN]
      o- portals.....[0 Portal]
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.3. ISCSI 보조 저장소

iSCSI 보조 저장소를 사용하면 내보낸 **LUN** 데이터를 로컬 시스템에 저장하는 다양한 방법을 지원할 수 있습니다. 스토리지 오브젝트를 생성하면 보조 저장소에서 사용하는 리소스를 정의합니다.

관리자는 LIO(Linux-IO)에서 지원하는 다음 보조 저장소 장치 중 하나를 선택할 수 있습니다.

FileIO 보조 저장소

로컬 파일 시스템에서 일반 파일을 디스크 이미지로 사용하는 경우 **fileio** 스토리지 오브젝트를 생성합니다. **fileio** 백 저장소 생성은 **fileio** 스토리지 오브젝트 생성을 참조하십시오.

블록 보조 저장소

로컬 블록 장치 및 논리적 장치를 사용하는 경우 블록 스토리지 오브젝트를 생성합니다. 블록 백 저장소 생성은 블록 스토리지 오브젝트 생성을 참조하십시오.

pscsi backstore

스토리지 오브젝트가 SCSI 명령의 직접 패스스루를 지원하는 경우 **pscsi** 스토리지 오브젝트를 만듭니다. **pscsi backstore**를 생성하려면 **pscsi** 스토리지 오브젝트 생성을 참조하십시오.

ramdisk 보조 저장소

임시 RAM 백업 장치를 생성하려면 **ramdisk** 스토리지 오브젝트를 생성합니다. 램디스크 백업 저장소 생성은 메모리 복사 RAM 디스크 스토리지 오브젝트 생성을 참조하십시오.

추가 리소스

- **targetcli(8)** 도움말 페이지

5.4. FILEIO 스토리지 오브젝트 생성

FileIO 스토리지 오브젝트는 **write_back** 또는 **write_ thru** 작업을 지원할 수 있습니다. **write_back** 작업을 사용하면 로컬 파일 시스템 캐시가 활성화됩니다. 이렇게 하면 성능이 향상되지만 데이터 손실 위험이 높아집니다.

write_back=false를 사용하여 **write_ thru** 작업을 위해 **write_back** 작업을 비활성화하는 것이 좋습니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.

절차

1. **backstores/** 디렉토리에서 **fileio/** 로 이동합니다.

```
/> backstores/fileio
```

2. **fileio** 스토리지 오브젝트를 생성합니다.

```
/backstores/fileio> create file1 /tmp/disk1.img 200M write_back=false

Created fileio file1 with size 209715200
```

검증

- 생성된 **fileio** 스토리지 오브젝트를 확인합니다.

```
/backstores/fileio> ls
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.5. 블록 스토리지 오브젝트 생성

블록 드라이버를 사용하면 **/sys/block/** 디렉터리에 나타나는 모든 블록 장치를 **LIO(Linux-IO)**와 함께 사용할 수 있습니다. 여기에는, **DASD, SSD, CD** 및 **DVD**와 같은 물리적 장치, 소프트웨어 또는 하드웨어 **RAID** 볼륨 또는 **LVM** 볼륨과 같은 논리 장치가 포함됩니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.

절차

1. **backstores/** 디렉터리에서 **block/** 로 이동합니다.

```
/> backstores/block/
```

2. 블록 보조 저장소를 생성합니다.

```
/backstores/block> create name=block_backend dev=/dev/sdb

Generating a wwn serial.
Created block storage object block_backend using /dev/vdb.
```

검증

- 생성된 블록 스토리지 오브젝트를 확인합니다.

```
/backstores/block> ls
```

참고

논리 볼륨에 블록 보조 저장소를 생성할 수도 있습니다.

추가 리소스

- **targetcli(8)** 도움말 페이지

5.6. PSCSI 스토리지 오브젝트 생성

보조 저장소로 **SCSI** 에뮬레이션 없이 **SCSI** 명령의 직접 패스스루를 지원하는 스토리지 오브젝트와 **SAS** 하드 드라이브와 같은 **/proc/scsi/ scsi**에 **lsscsi** 와 함께 표시되는 기본 **SCSI** 장치를 사용할 수 있습니다. 이 하위 시스템에서 **SCSI-3** 이상이 지원됩니다.

주의

pscsi 는 고급 사용자만 사용해야 합니다. **ALUA(Asymmetric Logical Unit Assignment)** 또는 영구 예약(예: **VMware ESX** 및 **vSphere**)과 같은 고급 **SCSI** 명령은 일반적으로 장치 펌웨어에 구현되지 않으며 오작동 또는 충돌을 일으킬 수 있습니다. 확실하지 않은 경우 대신 **block backstore**를 프로덕션 설정에 사용합니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.

설차

1. **backstores/** 디렉터리에서 **pscsi/** 로 이동합니다.

```
/> backstores/pscsi/
```

2. 이 예에서는 **/dev/sr0** 을 사용하여 **TYPE_ROM** 장치의 **pscsi** 백 저장소를 생성합니다.

```
/backstores/pscsi> create name=pscsi_backend dev=/dev/sr0

Generating a wwn serial.
Created pscsi storage object pscsi_backend using /dev/sr0
```

검증

- 생성된 **pscsi** 스토리지 오브젝트를 확인합니다.

```
/backstores/pscsi> ls
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.7. 메모리 복사 RAM 스토리지 오브젝트 생성

메모리 복사 **RAM** 디스크(**ramdisk**)는 **RAM** 디스크에 전체 **SCSI** 에뮬레이션 및 이니시에이터의 메모리 복사본을 사용하여 별도의 메모리 매핑을 제공합니다. 이는 멀티 세션에 대한 기능을 제공하며, 특히 생산 용도로 빠르고 휘발성 대량 스토리지에 유용합니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.

절차

1. **backstores/** 디렉토리에서 **ramdisk/** 로 이동합니다.

```
/> backstores/ramdisk/
```

2. **1GB RAM** 디스크 보조 저장소를 생성합니다.

```
/backstores/ramdisk> create name=rd_backend size=1GB

Generating a wwn serial.
Created rd_mcp ramdisk rd_backend with size 1GB.
```

검증

- 생성된 **ramdisk** 스토리지 오브젝트를 확인합니다.

```
/backstores/ramdisk> ls
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.8. ISCSI 포털 생성

iSCSI 포털을 만들면 타겟을 계속 활성화하는 대상에 **IP** 주소와 포트가 추가됩니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.
- **TPG**(대상 포털 그룹)와 연결된 **iSCSI** 대상입니다. 자세한 내용은 **iSCSI** 대상 생성을 참조하십시오.

절차

1. **TPG** 디렉토리로 이동합니다.

```
/iscsi> iqn.2006-04.example:444/tpg1/
```

2. 다음 옵션 중 하나를 사용하여 **iSCSI** 포털을 생성합니다.

   a. 기본 포털을 생성하면 기본 **iSCSI** 포트 **3260** 을 사용하며 타겟에서 해당 포트의 모든 **IP**

주소를 수신 대기할 수 있습니다.

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create

Using default IP port 3260
Binding to INADDR_Any (0.0.0.0)
Created network portal 0.0.0.0:3260
```

참고

iSCSI 대상이 생성되면 기본 포털도 생성됩니다. 이 포털은 기본 포트 번호인 모든 **IP** 주소를 수신 대기하도록 설정되어 있습니다. **0.0.0.0:3260.**

기본 포털을 제거하려면 다음 명령을 사용합니다.

```
/iscsi/iqn-name/tpg1/portals delete ip_address=0.0.0.0 ip_port=3260
```

b.

특정 **IP** 주소를 사용하여 포털 생성:

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create 192.168.122.137

Using default IP port 3260
Created network portal 192.168.122.137:3260
```

검증

- 새로 생성된 포털을 확인합니다.

```
/iscsi/iqn.20...mple:444/tpg1> ls

o- tpg..... [enambled, auth]
   o- acls .....[0 ACL]
   o- luns .....[0 LUN]
   o- portals .....[1 Portal]
     o- 192.168.122.137:3260.....[OK]
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.9. ISCSI LUN 생성

LUN(Logical Unit Number)은 **iSCSI** 보조 저장소에서 지원하는 물리적 장치입니다. 각 **LUN**에는 고유한 번호가 있습니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.
- **TPG(대상 포털 그룹)**와 연결된 **iSCSI** 대상입니다. 자세한 내용은 **iSCSI** 대상 생성을 참조하십시오.
- 생성된 스토리지 오브젝트. 자세한 내용은 **iSCSI Backstore** 를 참조하십시오.

절차

1. 이미 생성된 스토리지 오브젝트의 **LUN**을 만듭니다.

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/ramdisk/rd_backend
Created LUN 0.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/block/block_backend
Created LUN 1.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/fileio/file1
Created LUN 2.
```

2. 생성된 **LUN**을 확인합니다.

```
/iscsi/iqn.20...mple:444/tpg1> ls

o- tpg..... [enambled, auth]
   o- acls .....[0 ACL]
   o- luns .....[3 LUNs]
   |  o- lun0.....[ramdisk/ramdisk1]
   |  o- lun1.....[block/block1 (/dev/vdb1)]
   |  o- lun2.....[fileio/file1 (/foo.img)]
   o- portals .....[1 Portal]
      o- 192.168.122.137:3260.....[OK]
```

기본 **LUN** 이름은 **0** 부터 시작됩니다.

중요

기본적으로 LUN은 읽기-쓰기 권한으로 생성됩니다. ACL이 생성된 후 새 LUN이 추가되면 LUN은 사용 가능한 모든 ACL에 자동으로 매핑되며 보안 위험이 발생할 수 있습니다. 읽기 전용 권한으로 LUN을 만들려면 읽기 전용 iSCSI LUN 만들기 를 참조하십시오.

3. ACL 구성. 자세한 내용은 iSCSI ACL 생성을 참조하십시오.

추가 리소스

- targetcli(8) 도움말 페이지

5.10. 읽기 전용 ISCSI LUN 생성

기본적으로 LUN은 읽기-쓰기 권한으로 생성됩니다. 다음 절차에서는 읽기 전용 LUN을 만드는 방법을 설명합니다.

사전 요구 사항

- targetcli 설치 및 실행. 자세한 내용은 NFD 설치를 참조하십시오.
- TPG(대상 포털 그룹)와 연결된 iSCSI 대상입니다. 자세한 내용은 iSCSI 대상 생성을 참조하십시오.
- 생성된 스토리지 오브젝트. 자세한 내용은 iSCSI Backstore 를 참조하십시오.

절차

1. 읽기 전용 권한을 설정합니다.

```
/> set global auto_add_mapped_luns=false

Parameter auto_add_mapped_luns is now 'false'.
```

이렇게 하면 LUN을 수동으로 매핑할 수 있도록 하는 기존 ACL과 LUN의 자동 매핑이 금지됩니다.

2. ***initiator_iqn_name*** 디렉터리로 이동합니다.

```
/> iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/
```

3. **LUN**을 생성합니다.

```
/iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name> create
mapped_lun=next_sequential_LUN_number tpg_lun_or_backstore=backstore
write_protect=1
```

예제:

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example.foo:888> create mapped_lun=1
tpg_lun_or_backstore=/backstores/block/block2 write_protect=1

Created LUN 1.
Created Mapped LUN 1.
```

4. 생성된 **LUN**을 확인합니다.

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example.foo:888> ls
 o- 2006-04.com.example.foo:888 .. [Mapped LUNs: 2]
 | o- mapped_lun0 ..... [lun0 block/disk1 (rw)]
 | o- mapped_lun1 ..... [lun1 block/disk2 (ro)]
```

이제 **mapped_lun1** 행의 끝에 **(rw)**가 읽기 전용임을 나타내는 **mapped_lun0**의 **(rw)**가 있습니다.

5. **ACL** 구성. 자세한 내용은 **iSCSI ACL** 생성을 참조하십시오.

추가 리소스

- **targetcli(8)** 도움말 페이지

5.11. ISCSI ACL 생성

targetcli 에서 **ACL**(액세스 제어 목록)은 액세스 규칙을 정의하는 데 사용되며 각 이니시에이터는 **LUN**

에 대한 독점적인 액세스 권한을 갖습니다.

타겟과 이니시에이터 모두 고유한 식별 이름을 갖습니다. **ACL**을 구성하려면 이니시에이터의 고유 이름을 알아야 합니다. **iSCSI** 이니시에이터는 **/etc/iscsi/initiatorname.iscsi** 파일에 있습니다.

사전 요구 사항

- **targetcli** 설치 및 실행. 자세한 내용은 **NFD** 설치를 참조하십시오.
- **TPG(**대상 포털 그룹**)**와 연결된 **iSCSI** 대상입니다. 자세한 내용은 **iSCSI** 대상 생성을 참조하십시오.

절차

1. **acls** 디렉토리로 이동합니다.

```
/iscsi/iqn.20...mple:444/tpg1> acls/
```

2. 다음 옵션 중 하나를 사용하여 **ACL**을 만듭니다.

a. 이니시에이터의 **/etc/iscsi/initiatorname.iscsi** 파일에서 이니시에이터 이름 사용.

b. 쉽게 이해할 수 있는 이름을 사용하는 경우, **ACL**이 이니시에이터와 일치하는지 확인하는 **iSCSI** 이니시에이터 생성 섹션을 참조하십시오.

```
/iscsi/iqn.20...444/tpg1/acls> create iqn.2006-04.com.example.foo:888

Created Node ACL for iqn.2006-04.com.example.foo:888
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```

참고

앞의 예제에서 사용되는 글로벌 설정 **auto_add_mapped_luns** 는 생성된 모든 **ACL**에 **LUN**을 자동으로 매핑합니다.

대상 서버의 TPG 노드 내에 사용자 생성 ACL을 설정할 수 있습니다.

```
/iscsi/iqn.20...scsi:444/tpg1> set attribute generate_node_acls=1
```

검증

- 생성된 ACL을 확인합니다.

```
/iscsi/iqn.20...444/tpg1/acls> ls

o- acls .....[1 ACL]
  o- iqn.2006-04.com.example.foo:888 ....[3 Mapped LUNs, auth]
    o- mapped_lun0 .....[lun0 ramdisk/ramdisk1 (rw)]
    o- mapped_lun1 .....[lun1 block/block1 (rw)]
    o- mapped_lun2 .....[lun2 fileio/file1 (rw)]
```

추가 리소스

- targetcli(8) 도움말 페이지

5.12. 대상에 대한 CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL 설정

CHAP (Challenge-Handshake Authentication Protocol)를 사용하면 사용자가 암호로 대상을 보호할 수 있습니다. 이니시에이터는 대상에 연결할 수 있으려면 이 암호를 알고 있어야 합니다.

사전 요구 사항

- iSCSI ACL 생성. 자세한 내용은 iSCSI ACL 생성을 참조하십시오.

절차

1. 속성 인증 설정:

```
/iscsi/iqn.20...mple:444/tpg1> set attribute authentication=1

Parameter authentication is now '1'.
```

2. userid 및 암호 설정 :

```
/tpg1> set auth userid=redhat
Parameter userid is now 'redhat'.

/iscsi/iqn.20...689dcbb3/tpg1> set auth password=redhat_passwd
Parameter password is now 'redhat_passwd'.
```

추가 리소스

- **targetcli(8)** 도움말 페이지

5.13. TARGETCLI 도구를 사용하여 ISCSI 오브젝트 제거

다음 절차에서는 **targetcli** 툴을 사용하여 **iSCSI** 오브젝트를 제거하는 방법을 설명합니다.

절차

1. 대상에서 로그아웃합니다.

```
# iscsiadm -m node -T iqn.2006-04.example:444 -u
```

대상에 로그인하는 방법에 대한 자세한 내용은 **iSCSI** 이니시에이터 생성을 참조하십시오.

2. 모든 **ACL, LUN** 및 포털을 포함한 전체 대상을 제거합니다.

```
/> iscsi/ delete iqn.2006-04.com.example:444
```

iqn.2006-04.com.example:444 를 **target_iqn_name**으로 바꿉니다.

- **iSCSI** 보조 저장소를 제거하려면 다음을 수행합니다.

```
/> backstores/backstore-type/ delete block_backend
```

  - ***backstore-type*** 을 **fileio,block,pscsi** 또는 **ramdisk** 로 바꿉니다.
  - ***block_backend*** 를 삭제하려는 *보조 저장소* 이름으로 교체합니다.

- **ACL과 같은 iSCSI 대상의 일부를 제거하려면 다음을 수행합니다.**

```
/> /iscsi/iqn-name/tpg/acls/ delete iqn.2006-04.com.example:444
```

검증

- **변경 사항을 확인합니다.**

```
/> iscsi/ ls
```

추가 리소스

- **targetcli(8) 도움말 페이지**

6장. ISCSI 개시자 구성

iSCSI 이니시에이터는 iSCSI 대상에 연결하기 위한 세션을 형성합니다. 기본적으로 iSCSI 서비스는 지연 시작되며 iscsiadm 명령을 실행한 후에 서비스가 시작됩니다. root가 iSCSI 장치에 없거나 node.startup = 자동으로 표시된 노드가 없는 경우 iscsid 또는 iscsi 커널 모듈이 필요한 iscsiadm 명령이 실행될 때까지 iSCSI 서비스가 시작되지 않습니다.

systemctl start iscsid.service 명령을 root로 실행하여 iscsid 데몬이 실행되고 iSCSI 커널 모듈이 로드되도록 강제 적용합니다.

6.1. ISCSI 개시자 생성

이 섹션에서는 iSCSI 이니시에이터를 만드는 방법을 설명합니다.

사전 요구 사항

- 서버 시스템에 targetcli 설치 및 실행. 자세한 내용은 NFD 설치를 참조하십시오.
- 서버 시스템의 TPG(대상 포털 그룹)와 연결된 iSCSI 대상입니다. 자세한 내용은 iSCSI 대상 생성을 참조하십시오.
- iSCSI ACL 생성. 자세한 내용은 iSCSI ACL 생성을 참조하십시오.

절차

1. 클라이언트 시스템에 iscsi-initiator-utils 를 설치합니다.

```
# dnf install iscsi-initiator-utils
```

2. 이니시에이터 이름을 확인합니다.

```
# cat /etc/iscsi/initiatorname.iscsi

InitiatorName=2006-04.com.example.foo:888
```

3. ACL에 iSCSI ACL 생성에 사용자 지정 이름이 지정된 경우 그에 따라 /etc/iscsi/initiatorname.iscsi 파일을 수정합니다.

```
# vi /etc/iscsi/initiatorname.iscsi
```

4. 대상을 검색하고 표시된 대상 **IQN**을 사용하여 타겟에 로그인합니다.

```
# iscsiadm -m discovery -t st -p 10.64.24.179
   10.64.24.179:3260,1 iqn.2006-04.example:444

# iscsiadm -m node -T iqn.2006-04.example:444 -l
   Logging in to [iface: default, target: iqn.2006-04.example:444, portal: 10.64.24.179,3260]
(multiple)
   Login to [iface: default, target: iqn.2006-04.example:444, portal: 10.64.24.179,3260]
successful.
```

10.64.24.179 를 **target-ip-address**로 바꿉니다.

iSCSI ACL 생성에 설명된 대로 해당 이니시에이터 이름이 **ACL**에 추가되는 경우 동일한 대상에 연결된 수의 이니시에이터에 이 절차를 사용할 수 있습니다.

5. **iSCSI** 디스크 이름을 찾아서 이 **iSCSI** 디스크에 파일 시스템을 생성합니다.

```
# grep "Attached SCSI" /var/log/messages

# mkfs.ext4 /dev/disk_name
```

disk_name 을 **/var/log/messages** 파일에 표시된 **iSCSI** 디스크 이름으로 바꿉니다.

6. 파일 시스템을 마운트합니다.

```
# mkdir /mount/point

# mount /dev/disk_name /mount/point
```

/mount/point 를 파티션의 마운트 지점으로 바꿉니다.

7. 시스템이 부팅될 때 **/etc/fstab** 파일을 편집하여 파일 시스템을 자동으로 마운트합니다.

```
# vi /etc/fstab

/dev/disk_name /mount/point ext4 _netdev 0 0
```

disk_name 을 **iSCSI** 디스크 이름으로 바꾸고 ***/mount/point*** 를 파티션의 마운트 지점으로 바꿉니다.

추가 리소스

- **targetcli(8)** 및 **iscsiadm(8)** 도움말 페이지

6.2. 이니시에이터에 대한 CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL 설정

CHAP (Challenge-Handshake Authentication Protocol)를 사용하면 사용자가 암호로 대상을 보호할 수 있습니다. 이니시에이터는 대상에 연결할 수 있으려면 이 암호를 알고 있어야 합니다.

사전 요구 사항

- **iSCSI** 이니시에이터 생성. 자세한 내용은 **iSCSI** 이니시에이터 생성을 참조하십시오.
- 대상에 대한 **CHAP** 를 설정합니다. 자세한 내용은 대상에 대한 **Challenge-Handshake Authentication Protocol** 설정을 참조하십시오.

절차

1. **iscsid.conf** 파일에서 **CHAP** 인증을 활성화합니다.

```
# vi /etc/iscsi/iscsid.conf

node.session.auth.authmethod = CHAP
```

기본적으로 **node.session.auth.authmethod** 는 **None**으로 설정됩니다.

2. **iscsid.conf** 파일에 대상 사용자 이름 및 암호를 추가합니다.

```
node.session.auth.username = redhat
node.session.auth.password = redhat_passwd
```

3. **iscsid** 데몬을 시작합니다.

```
# systemctl start iscsid.service
```

추가 리소스

- **iscsiadm(8)** 도움말 페이지

6.3. ISCSIADM 유틸리티를 사용하여 ISCSI 세션 모니터링

다음 절차에서는 **iscsi adm** 유틸리티를 사용하여 **iscsi** 세션을 모니터링하는 방법을 설명합니다.

기본적으로 **iSCSI** 서비스는 지연 시작되며 **iscsiadm** 명령을 실행한 후에 서비스가 시작됩니다. **root**가 **iSCSI** 장치에 없거나 **node.startup =** 자동으로 표시된 노드가 없는 경우 **iscsid** 또는 **iscsi** 커널 모듈이 필요한 **iscsiadm** 명령이 실행될 때까지 **iSCSI** 서비스가 시작되지 않습니다.

systemctl start iscsid.service 명령을 **root**로 실행하여 **iscsid** 데몬이 실행되고 **iSCSI** 커널 모듈이 로드되도록 강제 적용합니다.

절차

1. 클라이언트 시스템에 **iscsi-initiator-utils** 를 설치합니다.

```
# dnf install iscsi-initiator-utils
```

2. 실행 중인 세션에 대한 정보를 찾습니다.

```
# iscsiadm -m session -P 3
```

이 명령은 세션 또는 장치 상태, 세션 **ID(sid),** 일부 협상 매개 변수 및 세션을 통해 액세스할 수 있는 **SCSI** 장치를 표시합니다.

- 예를 들어 **sid -to-node** 매핑만 표시하려면 더 짧은 출력을 보려면 다음을 실행합니다.

```
# iscsiadm -m session -P 0
    or
```

```
# iscsiadm -m session

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

이러한 명령은 실행 중인 세션 목록을 다음 형식으로 출력합니다. **driver [sid] target_ip:port,target_group_tag proper_target_target_name.**

추가 리소스

- **/usr/share/doc/iscsi-initiator-utils-version/README file**
- **iscsiadm(8)** 도움말 페이지

6.4. DM MULTIPATH가 장치 시간 초과를 덮어씁니다

recovery_tmo sysfs 옵션은 특정 **iSCSI** 장치에 대한 시간 초과를 제어합니다. 다음 옵션은 **recovery_tmo** 값을 전역적으로 덮어씁니다.

- **replacement_timeout** 구성 옵션은 모든 **iSCSI** 장치의 **recovery_tmo** 값을 전역적으로 덮어씁니다.
- **DM Multipath**에서 관리하는 모든 **iSCSI** 장치의 경우 **DM Multipath**의 **fast_io_fail_tmo** 옵션은 **recovery_tmo** 값을 전역적으로 덮어씁니다.

  DM Multipath의 **fast_io_fail_tmo** 옵션은 파이버 채널 장치의 **fast_io_fail_tmo** 옵션도 재정의합니다.

DM Multipath fast_io_fail_tmo 옵션이 **replacement_timeout** 보다 우선합니다. **DM Multipath**는 **multipathd** 서비스가 다시 로드될 때 항상 **recovery_tmo** 를 재설정하므로 **replacement_timeout** 을 사용하여 **DM Multipath**에서 관리하는 장치에서 **recovery_tmo** 를 덮어쓰는 것을 권장하지 않습니다.

7장. 파이버 채널 장치 사용

Red Hat Enterprise Linux 9는 다음과 같은 기본 파이버 채널 드라이버를 제공합니다.

- **lpfc**
- **qla2xxx**
- **zfcp**

7.1. 파이버 채널 논리 단위 크기 조정

시스템 관리자는 파이버 채널 논리 단위의 크기를 조정할 수 있습니다.

절차

1. 다중 경로 논리적 장치의 경로를 결정합니다.

```
multipath -ll
```

2. 다중 경로를 사용하는 시스템에서 파이버 채널 논리 단위를 다시 스캔합니다.

```
$ echo 1 > /sys/block/sdX/device/rescan
```

추가 리소스

- **multipath(8)** 도움말 페이지

7.2. 파이버 채널을 사용하여 장치의 링크 손실 동작 확인

드라이버가 **Transport dev_loss_tmo** 콜백을 구현하는 경우, 전송 문제가 감지되면 링크를 통해 장치에 대한 액세스가 차단됩니다.

절차

- 원격 포트의 상태를 확인합니다.

```
$ cat /sys/class/fc_remote_port/rport-host:bus:remote-port/port_state
```

이 명령은 다음 출력 중 하나를 반환합니다.

  - 원격 포트가 를 통해 액세스되는 장치와 함께 차단 되면 차단됩니다.
  - 원격 포트가 정상적으로 작동하는 경우 온라인

dev_loss_tmo 초 내에 문제가 해결되지 않으면 rport 및 장치가 차단 해제됩니다. 해당 장치에 전송된 새로운 I/O와 함께 해당 장치에서 실행되는 모든 I/O가 실패합니다.

링크 손실이 dev_loss_tmo 를 초과하면 scsi_device 및 sd_N_ 장치가 제거됩니다. 일반적으로 파이버 채널 클래스는 /dev/ sdx는 /dev/ sdx는 /dev/*sdx* 와 같이 해당 장치를 그대로 둡니다 . 이는 대상 바인딩이 파이버 채널 드라이버에 의해 저장되고 대상 포트가 반환되면 SCSI 주소가 다시 생성되기 때문입니다. 그러나 이를 보장할 수는 없으며 LUN의 스토리지 상자 구성이 추가 변경되지 않은 경우에만 ***sdx*** 장치가 복원됩니다.

추가 리소스

- multipath.conf(5) 도움말 페이지
- Oracle RAC 클러스터 지식 베이스 문서를 구성하는 동안 scsi,multipath 및 애플리케이션 계층에서 튜닝하는 것이 좋습니다.

7.3. 파이버 채널 구성 파일

다음은 파이버 채널에 사용자 공간 API를 제공하는 /sys/class/ 디렉토리의 구성 파일 목록입니다.

항목은 다음 변수를 사용합니다.

H

호스트 번호

B

버스 번호

T

대상

L

논리 단위(LUN)

R

원격 포트 번호

중요

시스템에서 다중 경로 소프트웨어를 사용하는 경우 이 섹션에 설명된 값을 변경하기 전에 하드웨어 공급업체에 문의하는 것이 좋습니다.

/sys/class/fc_transport/target*H*:*B*:*T*/

port_id

24비트 포트 ID/주소

node_name

64비트 노드 이름

port_name

64비트 포트 이름

/sys/class/fc_remote_ports/rport-*H*:*B*-*R*/의 원격 포트 구성

- **port_id**
- **node_name**

- port_name

- dev_loss_tmo

  scsi 장치가 시스템에서 제거될 시기를 제어합니다. dev_loss_tmo 가 트리거되면 scsi 장치가 제거됩니다. multipath.conf 파일에서 dev_loss_tmo 를 infinity 로 설정할 수 있습니다.

  Red Hat Enterprise Linux 9에서 fast_io_fail_tmo 옵션을 설정하지 않으면 dev_loss_tmo 가 600 초로 제한됩니다. 기본적으로 Red Hat Enterprise Linux 9에서 fast_io_fail_tmo 는 multipathd 서비스가 실행 중인 경우 5 초로 설정됩니다. 그렇지 않으면 off 로 설정됩니다.

- fast_io_fail_tmo

  링크를 "bad"로 표시하기 전에 대기하는 시간(초)을 지정합니다. 링크가 잘못 표시되면 기존 실행 중인 I/O 또는 해당 경로의 새 I/O가 실패합니다.

  I/O가 차단된 큐에 있는 경우 dev_loss_tmo 가 만료되고 큐가 차단되지 않을 때까지 실패합니다.

  fast_io_fail_tmo 가 off를 제외한 값으로 설정된 경우 dev_loss_tmo 는 적용되지 않습니다. fast_io_fail_tmo 가 off로 설정된 경우 장치를 시스템에서 제거할 때까지 I/O가 실패하지 않습니다. fast_io_fail_tmo 를 숫자로 설정하면 fast_io_fail_tmo 제한 시간이 트리거되면 즉시 I/O가 실패합니다.

/sys/class/fc_host/host*H*에서의 호스트 구성 /

- port_id

- node_name

- port_name

- issue_lip

는 드라이버에 원격 포트를 다시 검색하도록 지시합니다.

7.4. DM MULTIPATH가 장치 시간 초과를 덮어씁니다

recovery_tmo sysfs 옵션은 특정 iSCSI 장치에 대한 시간 초과를 제어합니다. 다음 옵션은 recovery_tmo 값을 전역적으로 덮어씁니다.

- replacement_timeout 구성 옵션은 모든 iSCSI 장치의 recovery_tmo 값을 전역적으로 덮어씁니다.
- DM Multipath에서 관리하는 모든 iSCSI 장치의 경우 DM Multipath의 fast_io_fail_tmo 옵션은 recovery_tmo 값을 전역적으로 덮어씁니다.

  DM Multipath의 fast_io_fail_tmo 옵션은 파이버 채널 장치의 fast_io_fail_tmo 옵션도 재정의합니다.

DM Multipath fast_io_fail_tmo 옵션이 replacement_timeout 보다 우선합니다. DM Multipath는 multipathd 서비스가 다시 로드될 때 항상 recovery_tmo 를 재설정하므로 replacement_timeout 을 사용하여 DM Multipath에서 관리하는 장치에서 recovery_tmo 를 덮어쓰는 것을 권장하지 않습니다.

8장. RDMA를 사용하여 패브릭을 통한 NVME

NVMe/RDMA(NVMe over RDMA) 설정에서 NVMe 대상 및 NVMe 이니시에이터를 구성합니다.

시스템 관리자로서 다음 작업을 완료하여 NVMe/RDMA 설정을 배포합니다.

- configfs를 사용하여 NVMe/RDMA 대상 설정
- nvmetcli를 사용하여 NVMe/RDMA 대상 설정
- NVMe/RDMA 클라이언트 구성

8.1. 패브릭 장치를 통한 NVME 개요

NVMe(Non-volatile Memory Express)는 호스트 소프트웨어 유틸리티가 솔리드 스테이트 드라이브와 통신할 수 있도록 하는 인터페이스입니다.

다음 유형의 패브릭 전송을 사용하여 패브릭 장치로 NVMe를 구성합니다.

NVMe over Remote Direct Memory Access(NVMe/RDMA)

NVMe/RDMA를 구성하는 방법에 대한 자세한 내용은 RDMA를 사용하여 NVMe over fabrics를 참조하십시오.

NVMe over Fibre Channel (FC-NVMe)

FC-NVMe을 구성하는 방법에 대한 자세한 내용은 FC를 사용하여 패브릭을 참조하십시오.

FC(Fibre Channel) 및 RDMA(Remote Direct Memory Access)를 사용하는 경우 솔리드 스테이트 드라이브가 시스템의 로컬일 필요는 없으며 FC 또는 RDMA 컨트롤러를 통해 원격으로 구성할 수 있습니다.

8.2. CONFIGFS를 사용하여 NVME/RDMA 대상 설정

configfs 를 사용하여 NVMe/RDMA 대상을 구성하려면 다음 절차를 사용하십시오.

사전 요구 사항

- **nvmet** 하위 시스템에 할당할 블록 장치가 있는지 확인합니다.

절차

1. **nvmet-rdma** 하위 시스템을 생성합니다.

```
# modprobe nvmet-rdma
# mkdir /sys/kernel/config/nvmet/subsystems/testnqn
# cd /sys/kernel/config/nvmet/subsystems/testnqn
```

   testnqn 을 하위 시스템 이름으로 교체합니다.

2. 모든 호스트가 이 대상에 연결하도록 허용합니다.

```
# echo 1 > attr_allow_any_host
```

3. 네임스페이스를 구성합니다.

```
# mkdir namespaces/10
# cd namespaces/10
```

   10 을 네임 스페이스 번호로 바꿉니다.

4. **NVMe** 장치의 경로를 설정합니다.

```
# echo -n /dev/nvme0n1 > device_path
```

5. 네임스페이스를 활성화합니다.

```
# echo 1 > enable
```

6. **NVMe 포트를 사용하여 디렉터리를 생성합니다.**

```
# mkdir /sys/kernel/config/nvmet/ports/1

# cd /sys/kernel/config/nvmet/ports/1
```

7. ***mlx5_ib0* 의 IP 주소 표시 :**

```
# ip addr show mlx5_ib0

8: mlx5_ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 4092 qdisc mq state UP
group default qlen 256
   link/infiniband 00:00:06:2f:fe:80:00:00:00:00:00:00:00:e4:1d:2d:03:00:e7:0f:f6 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
   inet 172.31.0.202/24 brd 172.31.0.255 scope global noprefixroute mlx5_ib0
      valid_lft forever preferred_lft forever
   inet6 fe80::e61d:2d03:e7:ff6/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
```

8. **대상의 전송 주소를 설정합니다.**

```
# echo -n 172.31.0.202 > addr_traddr
```

9. **RDMA를 전송 유형으로 설정합니다.**

```
# echo rdma > addr_trtype

# echo 4420 > addr_trsvcid
```

10. **포트의 주소 제품군을 설정합니다.**

```
# echo ipv4 > addr_adrfam
```

11. **소프트 링크를 생성합니다.**

```
# ln -s /sys/kernel/config/nvmet/subsystems/testnqn
/sys/kernel/config/nvmet/ports/1/subsystems/testnqn
```

검증

- **NVMe 대상이 지정된 포트에서 수신 대기 중이며 연결 요청에 대한 준비가 되었는지 확인합**

니다.

```
# dmesg | grep "enabling port"
[ 1091.413648] nvmet_rdma: enabling port 1 (172.31.0.202:4420)
```

추가 리소스

- **nvme(1) man page**

8.3. NVMETCLI를 사용하여 NVME/RDMA 대상 설정

nvmetcli 유틸리티를 사용하여 **NVMe** 대상을 편집, 보기 및 시작합니다. **nvmetcli** 유틸리티는 명령줄과 대화형 쉘 옵션을 제공합니다. **nvmetcli** 에서 **NVMe/RDMA** 대상을 구성하려면 다음 절차를 사용하십시오.

사전 요구 사항

- **nvmet** 하위 시스템에 할당할 블록 장치가 있는지 확인합니다.
- 다음 **nvmetcli** 작업을 **root** 사용자로 실행합니다.

절차

1. **nvmetcli** 패키지를 설치합니다.

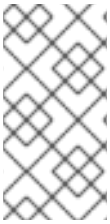
```
# dnf install nvmetcli
```

2. **rdma.json** 파일을 다운로드합니다.

```
# wget
http://git.infradead.org/users/hch/nvmetcli.git/blob_plain/0a6b088db2dc2e5de11e6f23f
1e890e4b54fee64:/rdma.json
```

3. **rdma.json** 파일을 편집하고 **traddr** 값을 **172.31.0.202** 로 변경합니다.
4. **NVMe** 대상 구성 파일을 로드하여 대상을 설정합니다.

```
# nvmetcli restore rdma.json
```



참고

NVMe 대상 구성 파일 이름이 지정되지 않은 경우 **nvmetcli** 는 **/etc/nvmet/config.json** 파일을 사용합니다.

검증

- **NVMe** 대상이 지정된 포트에서 수신 대기 중이며 연결 요청에 대한 준비가 되었는지 확인합니다.

```
# dmesg | tail -1
[ 4797.132647] nvmet_rdma: enabling port 2 (172.31.0.202:4420)
```

- 선택 사항: 현재 **NVMe** 대상을 지웁니다.

```
# nvmetcli clear
```

추가 리소스

- **nvmetcli** 및 **nvme(1)** 도움말 페이지

8.4. NVME/RDMA 클라이언트 구성

NVMe 관리 명령줄 인터페이스(**nvme-cli**) 도구를 사용하여 **NVMe/RDMA** 클라이언트를 구성하려면 다음 절차를 사용하십시오.

절차

1. **nvme-cli** 툴을 설치합니다.

```
# dnf install nvme-cli
```

2. 로드되지 않은 경우 **nvme-rdma** 모듈을 로드합니다.

```
# modprobe nvme-rdma
```

3.

NVMe 대상에서 사용 가능한 하위 시스템을 검색합니다.

```
# nvme discover -t rdma -a 172.31.0.202 -s 4420

Discovery Log Number of Records 1, Generation counter 2
=====Discovery Log Entry 0======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified, sq flow control disable supported
portid:  1
trsvcid: 4420
subnqn:  testnqn
traddr:  172.31.0.202
rdma_prtype: not specified
rdma_qptype: connected
rdma_cms:    rdma-cm
rdma_pkey: 0x0000
```

4.

검색된 하위 시스템에 연결합니다.

```
# nvme connect -t rdma -n testnqn -a 172.31.0.202 -s 4420

# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0 465.8G  0 disk
├─sda1                      8:1    0     1G  0 part /boot
└─sda2                      8:2    0 464.8G  0 part
  ├─rhel_rdma--virt--03-root 253:0    0    50G  0 lvm  /
  ├─rhel_rdma--virt--03-swap 253:1    0     4G  0 lvm  [SWAP]
  └─rhel_rdma--virt--03-home 253:2    0 410.8G  0 lvm  /home
nvme0n1

# cat /sys/class/nvme/nvme0/transport
rdma
```

testnqn **을 NVMe 하위 시스템 이름으로 교체합니다.**

172.31.0.202 **를 대상 IP 주소로 바꿉니다.**

4420 **을 포트 번호로 바꿉니다.**

검증

- **현재 연결된 NVMe 장치를 나열합니다.**

```
# nvme list
```

- **선택 사항: 타겟에서 연결을 끊습니다.**

```
# nvme disconnect -n testnqn
NQN:testnqn disconnected 1 controller(s)

# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0 465.8G  0 disk
├─sda1                      8:1    0     1G  0 part /boot
└─sda2                      8:2    0 464.8G  0 part
  ├─rhel_rdma--virt--03-root 253:0    0    50G  0 lvm  /
  ├─rhel_rdma--virt--03-swap 253:1    0     4G  0 lvm  [SWAP]
  └─rhel_rdma--virt--03-home 253:2    0 410.8G  0 lvm  /home
```

추가 리소스

- **nvme(1) man page**

- **NVMe-cli Github 리포지토리**

9장. FC를 사용하는 패브릭을 통한 NVME

특정 Broadcom Emulex 및 Marvell Qlogic Fibre 채널 어댑터와 함께 사용하는 경우 개시자 모드에서 NVMe/NVMe 전송이 완전 지원됩니다. 시스템 관리자로서 다음 섹션의 작업을 완료하여 FC-NVMe 설정을 배포합니다.

- Broadcom 어댑터에 대한 NVMe 이니시에이터 구성
- QLogic 어댑터에 대한 NVMe 이니시에이터 구성

9.1. 패브릭 장치를 통한 NVME 개요

NVMe(Non-volatile Memory Express)는 호스트 소프트웨어 유틸리티가 솔리드 스테이트 드라이브와 통신할 수 있도록 하는 인터페이스입니다.

다음 유형의 패브릭 전송을 사용하여 패브릭 장치로 NVMe를 구성합니다.

NVMe over Remote Direct Memory Access(NVMe/RDMA)

NVMe/RDMA를 구성하는 방법에 대한 자세한 내용은 RDMA를 사용하여 NVMe over fabrics를 참조하십시오.

NVMe over Fibre Channel (FC-NVMe)

FC-NVMe을 구성하는 방법에 대한 자세한 내용은 FC를 사용하여 패브릭을 참조하십시오.

FC(Fibre Channel) 및 RDMA(Remote Direct Memory Access)를 사용하는 경우 솔리드 스테이트 드라이브가 시스템의 로컬일 필요는 없으며 FC 또는 RDMA 컨트롤러를 통해 원격으로 구성할 수 있습니다.

9.2. BROADCOM 어댑터에 대한 NVME 이니시에이터 구성

NVMe 관리 명령줄 인터페이스(nvme-cli) 도구를 사용하여 Broadcom 어댑터 클라이언트에 NVMe 이니시에이터를 구성하려면 다음 절차를 사용하십시오.

절차

1. **nvme-cli 툴을 설치합니다.**

```
# dnf install nvme-cli
```

그러면 /etc/ nvme/ 디렉터리에 hostn qn 파일이 생성됩니다. hostnqn 파일은 NVMe 호스트를 식별합니다.

새 hostnqn 을 생성하려면 다음 명령을 사용합니다.

```
# nvme gen-hostnqn
```

2. **로컬 포트와 원격 포트의 WWNN 및 WWPN 식별자를 찾고 출력을 사용하여 하위 시스템 NQN을 찾습니다.**

```
# cat /sys/class/scsi_host/host*/nvme_info

NVME Initiator Enabled
XRI Dist lpfc0 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc0 WWPN x10000090fae0b5f5 WWNN x20000090fae0b5f5 DID x010f00
ONLINE
NVME RPORT       WWPN x204700a098cbcac6 WWNN x204600a098cbcac6 DID
x01050e TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 000000000e Cmpl 000000000e Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 00000000000008ea Issue 00000000000008ec OutIO 0000000000000002
    abort 00000000 noxri 00000000 nondlp 00000000 qdepth 00000000 wqerr 00000000
err 00000000
FCP CMPL: xb 00000000 Err 00000000
```

```
# nvme discover --transport fc \
              --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 \
              --host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5

Discovery Log Number of Records 2, Generation counter 49530
=====Discovery Log Entry 0======
trtype:  fc
adrfam:  fibre-channel
subtype: nvme subsystem
treq:    not specified
portid:  0
trsvcid: none
subnqn:  nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_
1
traddr:  nn-0x204600a098cbcac6:pn-0x204700a098cbcac6
```

nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 을 **traddr** 로 교체합니다.

nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 를 **host-traddr** 로 바꿉니다.

3. **nvme-cli** 를 사용하여 **NVMe** 대상에 연결합니다.

```
# nvme connect --transport fc \
        --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 \
        --host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 \
        -n nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_
1
```

nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 을 **traddr** 로 교체합니다.

nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 를 **host-traddr** 로 바꿉니다.

nqn.1992-08.com.netapp:sn.e18bfca8d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1 을 ***subnqn*** 으로 바꿉니다.

검증

- 현재 연결된 **NVMe** 장치를 나열합니다.

```
# nvme list
Node          SN                   Model                                    Namespace Usage
Format          FW Rev
----------------- -------------------- ---------------------------------------- --------- ------------------------
- ---------------- --------
/dev/nvme0n1     80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller                   1
107.37  GB / 107.37  GB      4 KiB +  0 B   FFFFFFFF
```

```
# lsblk |grep nvme
nvme0n1                 259:0    0   100G  0 disk
```

추가 리소스

- **nvme(1) man page**

- NVMe-cli Github 리포지토리

9.3. QLOGIC 어댑터에 대한 NVME 이니시에이터 구성

NVMe 관리 명령줄 인터페이스(nvme-cli) 도구를 사용하여 Qlogic 어댑터 클라이언트에 NVMe 이니시에이터를 구성하려면 다음 절차를 사용하십시오.

절차

1. **nvme-cli 툴을 설치합니다.**

```
# dnf install nvme-cli
```

그러면 /etc/ nvme/ 디렉터리에 hostn qn 파일이 생성됩니다. hostnqn 파일은 NVMe 호스트를 식별합니다.

새 hostnqn 을 생성하려면 다음 명령을 사용합니다.

```
# nvme gen-hostnqn
```

2. **qla2xxx 모듈을 다시 로드합니다.**

```
# rmmod qla2xxx
# modprobe qla2xxx
```

3. **로컬 포트와 원격 포트의 WWNN 및 WWPN 식별자를 찾습니다.**

```
# dmesg |grep traddr

[    6.139862] qla2xxx [0000:04:00.0]-ffff:0: register_localport: host-traddr=nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 on portID:10700
[    6.241762] qla2xxx [0000:04:00.0]-2102:0: qla_nvme_register_remote: traddr=nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 PortID:01050d
```

이러한 host-traddr 및 traddr 값을 사용하여 하위 시스템 NQN을 찾습니다.

```
# nvme discover --transport fc \
```

```
            --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
            --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62

Discovery Log Number of Records 2, Generation counter 49530
=====Discovery Log Entry 0======
trtype:  fc
adrfam:  fibre-channel
subtype: nvme subsystem
treq:    not specified
portid:  0
trsvcid: none
subnqn:  nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipat
h_1_subsystem_468
traddr:  nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6
```

nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 를 **traddr** 로 교체합니다.

nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 를 **host-traddr** 로 바꿉니다.

4. **nvme-cli** 툴을 사용하여 **NVMe** 대상에 연결합니다.

```
# nvme connect  --transport fc \
            --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
            --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 \
            -n nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipat
h_1_subsystem_468
```

nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 를 **traddr** 로 교체합니다.

nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 를 **host-traddr** 로 바꿉니다.

nqn.1992-08.com.netapp:sn.c9ecc9187b11e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468 를 *subnqn* 으로 바꿉니다.

검증

- 현재 연결된 **NVMe** 장치를 나열합니다.

```
# nvme list
```

```
Node            SN                   Model                                    Namespace Usage
Format         FW Rev
--------------- -------------------- ---------------------------------------- --------- --------------------------
- ---------------- --------
/dev/nvme0n1     80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller                   1
107.37  GB / 107.37  GB      4 KiB +  0 B   FFFFFFFF

# lsblk |grep nvme
nvme0n1                 259:0    0   100G  0 disk
```

추가 리소스

- nvme(1) man page
- NVMe-cli Github 리포지토리

10장. NVME 장치에서 멀티패스 활성화

파이버 채널(FC)과 같은 패브릭 전송을 통해 시스템에 연결된 멀티패스 NVMe 장치를 사용할 수 있습니다. 다중 경로 솔루션 중에서 선택할 수 있습니다.

10.1. 네이티브 NVME 멀티패스 및 DM MULTIPATH

NVMe 장치는 기본 다중 경로 기능을 지원합니다. NVMe에서 멀티패스를 구성할 때 표준 DM Multipath 프레임워크와 기본 NVMe 다중 경로 중에서 선택할 수 있습니다.

DM Multipath 및 네이티브 NVMe 다중 경로 모두 NVMe 장치의 Asymmetric Namespace Access (ANA) 멀티패스 스키마를 지원합니다. ANA는 대상과 이니시에이터 간의 최적화된 경로를 식별하고 성능을 향상시킵니다.

네이티브 NVMe 멀티패스가 활성화되면 모든 NVMe 장치에 전역적으로 적용됩니다. 이는 더 높은 성능을 제공할 수 있지만 DM Multipath가 제공하는 모든 기능을 포함하지는 않습니다. 예를 들어 네이티브 NVMe 멀티패스는 장애 조치(failover) 및 라운드 로빈 경로 선택 방법만 지원합니다.

기본적으로 Red Hat Enterprise Linux 9에서는 NVMe 멀티패스가 활성화되며 권장되는 다중 경로 솔루션입니다.

10.2. 네이티브 NVME 멀티패스 활성화

이 절차에서는 네이티브 NVMe 다중 경로 솔루션을 사용하여 연결된 NVMe 장치에서 멀티패스를 활성화합니다.

사전 요구 사항

- NVMe 장치가 시스템에 연결되어 있습니다.

  패브릭 전송을 통해 NVMe를 연결하는 방법에 대한 자세한 내용은 패브릭 장치를 통한 NVMe 개요 를 참조하십시오.

절차

1. 커널에서 네이티브 NVMe 멀티패스가 활성화되어 있는지 확인합니다.

```
# cat /sys/module/nvme_core/parameters/multipath
```

명령은 다음 중 하나를 표시합니다.

N

네이티브 **NVMe** 멀티패스가 비활성화되어 있습니다.

Y

네이티브 **NVMe** 멀티패스가 활성화되어 있습니다.

2. 네이티브 **NVMe** 멀티패스가 비활성화된 경우 다음 방법 중 하나를 사용하여 활성화합니다.

- 커널 옵션 사용:

  i. 커널 명령줄에 **nvme_core.multipath=Y** 옵션을 추가합니다.

  ```
  # grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
  ```

  ii. **64**비트 **IBM Z** 아키텍처에서 부팅 메뉴를 업데이트합니다.

  ```
  # zipl
  ```

  iii. 시스템을 재부팅합니다.

- 커널 모듈 구성 파일 사용:

  i. 다음 콘텐츠를 사용하여 **/etc/modprobe.d/nvme_core.conf** 구성 파일을 만듭니다.

  ```
  options nvme_core multipath=Y
  ```

  ii. **initramfs** 파일 시스템을 백업합니다.

```
# cp /boot/initramfs-$(uname -r).img \
    /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

iii. **initramfs** 파일 시스템을 다시 빌드합니다.

```
# dracut --force --verbose
```

iv. 시스템을 재부팅합니다.

3. 선택 사항: 실행 중인 시스템에서 **NVMe** 장치의 **I/O** 정책을 변경하여 사용 가능한 모든 경로에 **I/O**를 배포합니다.

```
# echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

4. 선택 사항: **udev** 규칙을 사용하여 **I/O** 정책을 영구적으로 설정합니다. 다음 콘텐츠를 사용하여 **/etc/udev/rules.d/71-nvme-io-policy.rules** 파일을 생성합니다.

```
ACTION=="add|change", SUBSYSTEM=="nvme-subsystem", ATTR{iopolicy}="round-robin"
```

검증

1. 시스템이 **NVMe** 장치를 인식하는지 확인합니다.

```
# nvme list

Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- ------------------------
- ---------------- --------
/dev/nvme0n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme0n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
```

2. 연결된 모든 **NVMe** 하위 시스템을 나열합니다.

```
# nvme list-subsys

nvme-subsys0 - NQN=testnqn
\
 +- nvme0 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
```

```
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme1 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
 +- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

활성 전송 유형을 확인합니다. 예를 들어 nvme0 fc 는 장치가 파이버 채널 전송을 통해 연결되어 있음을 나타내고 nvme tcp 는 장치가 TCP를 통해 연결되어 있음을 나타냅니다.

3. **커널 옵션을 편집한 경우 커널 명령줄에서 네이티브 NVMe 멀티패스가 활성화되어 있는지 확인합니다.**

```
# cat /proc/cmdline

BOOT_IMAGE=[...] nvme_core.multipath=Y
```

4. **DM Multipath가 NVMe 네임스페이스를 nvme0c3n1에서 nvme0 c3n1 로 보고하는지 확인합니다(예: nvme0n1 ~ nvme3n1:)**

```
# multipath -e -ll | grep -i nvme

uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03 [nvme]:nvme0n1 NVMe,Linux,4.18.0-2
| `- 0:0:1    nvme0c0n1 0:0     n/a   optimized live
| `- 0:1:1    nvme0c1n1 0:0     n/a   optimized live
| `- 0:2:1    nvme0c2n1 0:0     n/a   optimized live
  `- 0:3:1    nvme0c3n1 0:0     n/a   optimized live

uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22 [nvme]:nvme0n2 NVMe,Linux,4.18.0-2
| `- 0:0:1    nvme0c0n1 0:0     n/a   optimized live
| `- 0:1:1    nvme0c1n1 0:0     n/a   optimized live
| `- 0:2:1    nvme0c2n1 0:0     n/a   optimized live
  `- 0:3:1    nvme0c3n1 0:0     n/a   optimized live
```

5. ***I/O 정책을 변경한 경우 NVMe 장치의 라운드 로빈이 활성 I/O 정책인지 확인합니다.***

```
# cat /sys/class/nvme-subsystem/nvme-subsys0/iopolicy

round-robin
```

추가 리소스

- *커널 명령줄 매개변수 구성*

10.3. NVME 장치에서 DM MULTIPATH 활성화

이 절차에서는 **DM Multipath** 솔루션을 사용하여 연결된 **NVMe** 장치에서 멀티패스를 활성화합니다.

사전 요구 사항

- **NVMe** 장치가 시스템에 연결되어 있습니다.

  패브릭 전송을 통해 **NVMe**를 연결하는 방법에 대한 자세한 내용은 *패브릭 장치를 통한 NVMe 개요* 를 참조하십시오.

절차

1. 네이티브 **NVMe** 멀티패스가 비활성화되어 있는지 확인합니다.

```
# cat /sys/module/nvme_core/parameters/multipath
```

   명령은 다음 중 하나를 표시합니다.

   N

   네이티브 **NVMe** 멀티패스가 비활성화되어 있습니다.

   Y

   네이티브 **NVMe** 멀티패스가 활성화되어 있습니다.

2. 네이티브 **NVMe** 멀티패스가 활성화된 경우 비활성화합니다.

   a. 커널 명령줄에서 **nvme_core.multipath=Y** 옵션을 제거합니다.

```
# grubby --update-kernel=ALL --remove-args="nvme_core.multipath=Y"
```

   b. **64**비트 **IBM Z** 아키텍처에서 부팅 메뉴를 업데이트합니다.

```
# zipl
```

c. */etc/modprobe.d/n vme_core.conf* 파일에서 **nvme_core=Y** 줄이 있는 경우 해당 옵션을 제거합니다.

d. 시스템을 재부팅합니다.

3. **DM Multipath**가 활성화되어 있는지 확인합니다.

```
# systemctl enable --now multipathd.service
```

4. 사용 가능한 모든 경로에 **I/O**를 분산합니다. **/etc/multipath.conf** 파일에 다음 내용을 추가합니다.

```
device {
  vendor "NVME"
  product ".*"
  path_grouping_policy    group_by_prio
}
```

참고

DM Multipath가 **NVMe** 장치를 관리하는 경우 **/sys/class/nvme-subsys0/iopolicy** 구성 파일은 **I/O** 배포에 영향을 미치지 않습니다.

5. **multipathd** 서비스를 다시 로드하여 구성 변경 사항을 적용합니다.

```
# multipath -r
```

6. **initramfs** 파일 시스템을 백업합니다.

```
# cp /boot/initramfs-$(uname -r).img \
    /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

7. **initramfs** 파일 시스템을 다시 빌드합니다.

```
# dracut --force --verbose
```

검증

1. *시스템이 **NVMe** 장치를 인식하는지 확인합니다.*

```
# nvme list

Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- --------------------------
- ---------------- --------
/dev/nvme0n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme0n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme1n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme1n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme2n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme2n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme3n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme3n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
```

2. *연결된 **NVMe** 하위 시스템을 모두 나열합니다. 명령이 **nvme0n1** 에서 **nvme3n2** 를 통해 보고하고 있는지 확인합니다(예: **nvme0c0n1** 에서 **nvme0c3n 1) :***

```
# nvme list-subsys

nvme-subsys0 - NQN=testnqn
\
 +- nvme0 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme1 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
 +- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

```
# multipath -ll

mpathae (uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03) dm-36 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 0:1:1:1  nvme0n1 259:0   active ready running
  |- 1:2:1:1  nvme1n1 259:2   active ready running
  |- 2:3:1:1  nvme2n1 259:4   active ready running
  `- 3:4:1:1  nvme3n1 259:6   active ready running
```

```
mpathaf (uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22) dm-39 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 0:1:2:2  nvme0n2 259:1   active ready running
  |- 1:2:2:2  nvme1n2 259:3   active ready running
  |- 2:3:2:2  nvme2n2 259:5   active ready running
  `- 3:4:2:2  nvme3n2 259:7   active ready running
```

추가 리소스

- *커널 명령줄 매개변수 구성*
- ***DM Multipath** 구성*.

11장. 스왑 시작하기

이 섹션에서는 스왑 공간 및 추가 및 제거하는 방법에 대해 설명합니다.

11.1. 스왑 공간 개요

Linux의 스왑 공간은 실제 메모리(RAM)가 가득 찰 때 사용됩니다. 시스템에 더 많은 메모리 리소스가 필요하고 RAM이 가득 차면 메모리의 비활성 페이지가 스왑 공간으로 이동합니다. 스왑 공간은 RAM이 적은 시스템에 도움이 될 수 있지만 더 많은 RAM을 대체하는 것으로 간주해서는 안 됩니다.

스왑 공간은 실제 메모리보다 느린 액세스 시간이 있는 하드 드라이브에 있습니다. 스왑 공간은 전용 스왑 파티션(권장), 스왑 파일 또는 스왑 파티션과 스왑 파일의 조합일 수 있습니다.

지난 몇 년 동안 권장 스왑 공간은 시스템의 RAM 용량으로 선형적으로 증가했습니다. 그러나 최신 시스템에는 수백 기가 바이트의 RAM이 포함되어 있는 경우가 많습니다. 결과적으로 권장 스왑 공간은 시스템 메모리가 아닌 시스템 메모리 워크로드의 기능으로 간주됩니다.

스왑 공간 추가

다음은 스왑 공간을 추가하는 다양한 방법입니다.

- LVM2 논리 볼륨에서 스왑 확장
- 스왑용 LVM2 논리 볼륨 생성
- 스왑 파일 만들기

예를 들어 시스템의 RAM 크기를 1GB에서 2GB로 업그레이드할 수 있지만 2GB의 스왑 공간만 있습니다. 메모리 강화 작업을 수행하거나 메모리가 많은 메모리가 필요한 애플리케이션을 실행하는 경우 스왑 공간의 양을 4GB로 늘리는 것이 유용할 수 있습니다.

스왑 공간 제거

다음은 스왑 공간을 제거하는 다양한 방법입니다.

- LVM2 논리 볼륨의 스왑 감소

- *스왑용 LVM2 논리 볼륨 제거*
- *스왑 파일 제거*

예를 들어 시스템의 RAM 크기를 1GB에서 512MB로 다운그레이드했지만 여전히 2GB의 스왑 공간이 할당되어 있습니다. 2GB가 큰 디스크 공간을 낭비할 수 있기 때문에 스왑 공간의 양을 1GB로 줄이는 것이 유용할 수 있습니다.

11.2. 시스템 스왑 공간 권장

이 섹션에서는 시스템의 RAM 용량 및 시스템의 RAM에 따라 스왑 파티션의 권장 크기에 대해 설명합니다. 권장되는 스왑 파티션 크기는 설치 중에 자동으로 설정됩니다. 그러나 간격을 허용하려면 사용자 지정 파티션 영역에서 스왑 공간을 편집해야 합니다.

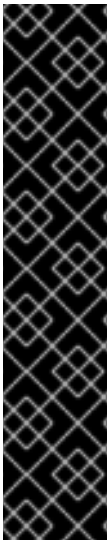
특히 1GB 이하의 메모리가 부족한 시스템에서 중요한 권장 사항은 다음과 같습니다. 이러한 시스템에 충분한 스왑 공간을 할당하지 않으면 불안정성이나 설치된 시스템을 부팅할 수 없는 문제가 발생할 수 있습니다.

표 11.1. 권장되는 스왑 공간

시스템의 RAM 크기	권장되는 스왑 공간	hibernation을 허용하는 경우 권장되는 스왑 공간
2GB	RAM의 양 2배	RAM의 양 3배
> 2GB - 8GB	RAM의 양과 동일	RAM의 양 2배
> 8 GB – 64 GB	최소 4GB	1.5 배 RAM의 양입니다.
> 64 GB	최소 4GB	간격은 권장되지 않습니다.

이 표에 나열된 각 범위 간 테두리 (예: 2GB, 8GB 또는 64GB의 시스템 RAM이 있는 시스템)에서 선택한 스왑 공간 및 수면 지원과 관련하여 재량을 행사할 수 있습니다. 시스템 리소스를 허용하는 경우 스왑 공간을 늘리면 성능이 향상될 수 있습니다.

여러 스토리지 장치에 스왑 공간을 분산하면 특히 빠른 드라이브, 컨트롤러 및 인터페이스가 있는 시스템에서 스왑 공간 성능이 향상됩니다.



중요

스왑 공간으로 할당된 파일 시스템 및 **LVM2** 볼륨은 수정할 때 사용하지 않아야 합니다. 시스템 프로세스 또는 커널이 스왑 공간을 사용하는 경우 스왑을 수정하려고 합니다. **free** 및 **cat /proc/swaps** 명령을 사용하여 스왑 크기 및 사용 위치를 확인합니다.

스왑 공간 크기를 조정하려면 시스템에서 스왑 공간을 일시적으로 제거해야 합니다. 실행 중인 애플리케이션이 추가 스왑 공간을 사용하고 메모리가 부족한 상황에서 실행되는 경우 문제가 발생할 수 있습니다. 바람직하게는 복구 모드에서 스왑 크기 조정을 수행합니다. 고급 **RHEL 9** 설치 수행 의 부팅 디버그 옵션을 참조하십시오. 파일 시스템을 마운트하라는 메시지가 표시되면 **Skip** 를 선택합니다.

11.3. LVM2 논리 볼륨에서 스왑 확장

이 절차에서는 기존 **LVM2** 논리 볼륨의 스왑 공간을 확장하는 방법을 설명합니다. **/dev/VolGroup00/LogVol01** 이 **2GB** 로 확장할 볼륨이라고 가정합니다.

사전 요구 사항

- 충분한 디스크 공간이 있습니다.

절차

1. 연결된 논리 볼륨의 스왑을 비활성화합니다.

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. **LVM2** 논리 볼륨의 크기를 **2GB** 로 조정합니다.

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. 새 스왑 공간을 포맷합니다.

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 확장된 논리 볼륨을 활성화합니다.

```
# swapon -v /dev/VolGroup00/LogVol01
```

-

검증

- 스왑 논리 볼륨이 성공적으로 확장 및 활성화되었는지 테스트하려면 다음 명령을 사용하여 활성 스왑 공간을 검사합니다.

```
$ cat /proc/swaps
$ free -h
```

11.4. 스왑을 위한 LVM2 논리 볼륨 생성

이 절차에서는 스왑을 위한 **LVM2** 논리 볼륨을 만드는 방법을 설명합니다. **/dev/VolGroup00/LogVol02** 가 추가할 스왑 볼륨이라고 가정합니다.

사전 요구 사항

- 충분한 디스크 공간이 있습니다.

절차

1. 크기가 **2GB** 인 **LVM2** 논리 볼륨 만들기:

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 새 스왑 공간을 포맷합니다.

```
# mkswap /dev/VolGroup00/LogVol02
```

3. **/etc/fstab** 파일에 다음 항목을 추가합니다.

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

4. 시스템이 새 구성을 등록하도록 다시 마운트 단위를 다시 생성합니다.

```
# systemctl daemon-reload
```

5. 논리 볼륨에서 스왑을 활성화합니다.

```
# swapon -v /dev/VolGroup00/LogVol02
```

검증

- 스왑 논리 볼륨이 성공적으로 생성 및 활성화되었는지 테스트하려면 다음 명령을 사용하여 활성 스왑 공간을 검사합니다.

```
$ cat /proc/swaps
$ free -h
```

11.5. 스왑 파일 만들기

이 절차에서는 스왑 파일을 만드는 방법을 설명합니다.

사전 요구 사항

- 충분한 디스크 공간이 있습니다.

절차

1. 새 스왑 파일의 크기를 메가바이트로 결정하고 **1024**로 곱하여 블록 수를 결정합니다. 예를 들어, **64MB** 스왑 파일의 블록 크기는 **65536**입니다.

2. 빈 파일을 생성합니다.

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

65536 을 원하는 블록 크기와 동일한 값으로 바꿉니다.

3. 명령으로 스왑 파일을 설정합니다.

```
# mkswap /swapfile
```

4. 읽을 수 없도록 스왑 파일의 보안을 변경합니다.

```
# chmod 0600 /swapfile
```

5. 부팅 시 스왑 파일을 활성화하려면 다음 항목으로 **/etc/fstab** 파일을 편집합니다.

```
/swapfile swap swap defaults 0 0
```

다음에 시스템을 부팅할 때 새 스왑 파일을 활성화합니다.

6. 시스템이 새 **/etc/fstab** 구성을 등록하도록 다시 마운트 단위를 다시 생성합니다.

```
# systemctl daemon-reload
```

7. 스왑 파일을 즉시 활성화합니다.

```
# swapon /swapfile
```

검증

- 새 스왑 파일이 성공적으로 생성 및 활성화되었는지 테스트하려면 다음 명령을 사용하여 활성 스왑 공간을 검사합니다.

```
$ cat /proc/swaps
$ free -h
```

11.6. LVM2 논리 볼륨에서 스왑 감소

이 절차에서는 **LVM2** 논리 볼륨의 스왑을 줄이는 방법을 설명합니다. **/dev/VolGroup00/LogVol01** 이 축소할 볼륨이라고 가정합니다.

절차

1. 연결된 논리 볼륨의 스왑을 비활성화합니다.

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. ***LVM2** 논리 볼륨을 **512MB**로 줄입니다.*

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. *새 스왑 공간을 포맷합니다.*

```
# mkswap /dev/VolGroup00/LogVol01
```

4. *논리 볼륨에서 스왑을 활성화합니다.*

```
# swapon -v /dev/VolGroup00/LogVol01
```

검증

- *스왑 논리 볼륨이 성공적으로 감소했는지 테스트하려면 다음 명령을 사용하여 활성 스왑 공간을 검사합니다.*

```
$ cat /proc/swaps
$ free -h
```

*11.7. 스왑을 위한 **LVM2** 논리 볼륨 제거*

*이 절차에서는 스왑을 위한 **LVM2** 논리 볼륨을 제거하는 방법을 설명합니다.*
***/dev/VolGroup00/LogVol02** 가 삭제하려는 스왑 볼륨이라고 가정합니다.*

절차

1. *연결된 논리 볼륨의 스왑을 비활성화합니다.*

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2. ***LVM2** 논리 볼륨을 제거합니다.*

```
# lvremove /dev/VolGroup00/LogVol02
```

3. ***/etc/fstab** 파일에서 다음 관련 항목을 제거하십시오.*

-

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

4. 시스템이 새 구성을 등록하도록 다시 마운트 단위를 다시 생성합니다.

```
# systemctl daemon-reload
```

검증

- 논리 볼륨이 제거되었는지 테스트하려면 다음 명령을 사용하여 활성 스왑 공간을 검사합니다.

```
$ cat /proc/swaps
$ free -h
```

11.8. 스왑 파일 제거

이 절차에서는 스왑 파일을 제거하는 방법을 설명합니다.

절차

1. 쉘 프롬프트에서 다음 명령을 실행하여 스왑 파일을 비활성화합니다. 여기서 **/swapfile** 은 스왑 파일입니다.

```
# swapoff -v /swapfile
```

2. 그에 따라 **/etc/fstab** 파일에서 해당 항목을 제거합니다.

3. 시스템이 새 구성을 등록하도록 다시 마운트 단위를 다시 생성합니다.

```
# systemctl daemon-reload
```

4. 실제 파일을 제거합니다.

```
# rm /swapfile
```

12장. 파이버 채널 OVER 이더넷 구성

IEEE T11 FC-BB-5 표준에 따라 FCoE(Fib over Ethernet)는 이더넷 네트워크를 통해 파이버 채널 프레임을 전송하는 프로토콜입니다. 일반적으로 데이터 센터에는 고유한 구성으로 서로 분리된 전용 LAN 및 SAN(Storage Area Network)이 있습니다. FCoE는 이러한 네트워크를 단일 및 통합 네트워크 구조로 결합합니다. FCoE의 이점은 예를 들어 하드웨어 및 에너지 비용을 절감할 수 있습니다.

12.1. RHEL에서 하드웨어 FCOE HBA 사용

RHEL에서는 다음 드라이버에서 지원되는 하드웨어 Fibre Channel over Ethernet (FCoE) HBA(Host Bus Adapter)를 사용할 수 있습니다.

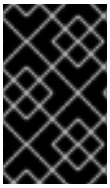
- qedf
- bnx2fc
- fnic

이러한 HBA를 사용하는 경우 HBA 설정에서 FCoE 설정을 구성합니다. 자세한 내용은 어댑터 설명서를 참조하십시오.

HBA를 구성한 후 SAN(Storage Area Network)에서 내보낸 LUN(Logical Unit Numbers)을 RHEL에서 /dev/sd* 장치로 자동으로 사용할 수 있습니다. 로컬 스토리지 장치와 유사한 장치를 사용할 수 있습니다.

12.2. 소프트웨어 FCOE 장치 설정

소프트웨어 FCoE 장치를 사용하여 FCoE(Logical Unit Numbers)에 액세스하여 FCoE 오프로드를 부분적으로 지원하는 이더넷 어댑터를 사용하여 사용합니다.



중요

RHEL은 fcoe.ko 커널 모듈이 필요한 소프트웨어 FCoE 장치를 지원하지 않습니다.

이 절차를 완료하면 SAN(Storage Area Network)에서 내보낸 LUN을 RHEL에서 /dev/sd* 장치로 자

동으로 사용할 수 있습니다. 이러한 장치를 로컬 스토리지 장치와 유사한 방식으로 사용할 수 있습니다.

사전 요구 사항

- VLAN을 지원하도록 네트워크 스위치를 구성했습니다.
- SAN은 VLAN을 사용하여 스토리지 트래픽을 일반 이더넷 트래픽과 분리합니다.
- BIOS에서 서버의 HBA를 구성했습니다.
- HBA는 네트워크에 연결되어 있으며 링크가 켜집니다. 자세한 내용은 HBA 문서를 참조하십시오.

절차

1. fcoe-utils 패키지를 설치합니다.

```
# dnf install fcoe-utils
```

2. /etc/fcoe/cfg-ethx 템플릿 파일을 /etc/fcoe/cfg-interface_name 에 복사합니다. 예를 들어 FCoE를 사용하도록 enp1s0 인터페이스를 구성하려면 다음 명령을 입력합니다.

```
# cp /etc/fcoe/cfg-ethx /etc/fcoe/cfg-enp1s0
```

3. fcoe 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now fcoe
```

4. 인터페이스 enp1s0 에서 FCoE VLAN을 검색하고, 검색된 VLAN에 대한 네트워크 장치를 생성하고, 이니시에이터를 시작합니다.

```
# fipvlan -s -c enp1s0
Created VLAN device enp1s0.200
Starting FCoE on interface enp1s0.200
Fibre Channel Forwarders Discovered
```

```
interface      | VLAN | FCF MAC
-------------------------------------------
enp1s0         | 200  | 00:53:00:a7:e7:1b
```

5. 선택 사항: 검색된 대상, LUN, LUN과 연결된 장치에 대한 세부 정보를 표시합니다.

```
# fcoeadm -t
Interface:       enp1s0.200
Roles:           FCP Target
Node Name:       0x500a0980824acd15
Port Name:       0x500a0982824acd15
Target ID:       0
MaxFrameSize:    2048 bytes
OS Device Name:  rport-11:0-1
FC-ID (Port ID): 0xba00a0
State:           Online

LUN ID  Device Name   Capacity   Block Size  Description
------  -----------  ----------  ----------  --------------------
    0  sdb          28.38 GiB      512     NETAPP LUN (rev 820a)
    ...
```

이 예에서는 SAN의 LUN 0이 /dev/sdb 장치로 호스트에 연결되었음을 보여줍니다.

검증

- 모든 활성 FCoE 인터페이스에 대한 정보를 표시합니다.

```
# fcoeadm -i
Description:     BCM57840 NetXtreme II 10 Gigabit Ethernet
Revision:        11
Manufacturer:    Broadcom Inc. and subsidiaries
Serial Number:   000AG703A9B7

Driver:          bnx2x Unknown
Number of Ports: 1

   Symbolic Name:    bnx2fc (QLogic BCM57840) v2.12.13 over enp1s0.200
   OS Device Name:   host11
   Node Name:        0x2000000af70ae935
   Port Name:        0x2001000af70ae935
   Fabric Name:      0x20c8002a6aa7e701
   Speed:            10 Gbit
   Supported Speed:  1 Gbit, 10 Gbit
   MaxFrameSize:     2048 bytes
   FC-ID (Port ID):  0xba02c0
   State:            Online
```

추가 리소스

- ***fcoeadm(8)* 도움말 페이지**
- ***/usr/share/doc/fcoe-utils/README***
- *파이버 채널 장치 사용*

13장. 테이프 장치 관리

테이프 장치는 데이터를 순차적으로 저장하고 액세스할 수 있는 자기용 테이프입니다. 데이터는 테이프 드라이브의 도움말을 사용하여 이 테이프 장치에 기록됩니다.**Data is written to this tape device with the help of a tape drive.** 테이프 장치에 데이터를 저장하기 위해 파일 시스템을 만들 필요가 없습니다. 테이프 드라이브는 **SCSI, FC, USB, SATA** 및 기타 인터페이스와 같은 다양한 인터페이스가 있는 호스트 컴퓨터에 연결할 수 있습니다.

13.1. 테이프 장치 유형

다음은 다양한 유형의 테이프 장치 목록입니다.

- **/dev/st0** 은 재개된 테이프 장치입니다.
- **/dev/nst0** 은 **non-rewinding tape device**입니다. 일별 백업에 대해 비**rewinding** 장치를 사용합니다.

테이핑 장치를 사용할 때 몇 가지 이점이 있습니다. 이는 비용 효율적이고 안정적입니다. 테이프 장치는 데이터 손상에도 탄력적이며 데이터 보존에 적합합니다.

13.2. 테이프 드라이브 관리 도구 설치

mt 명령을 사용하여 데이터를 뒤로 앞뒤로 전환합니다. **mt** 유틸리티는 자조 용 테이크 드라이브 작업을 제어하고 **st** 유틸리티는 **SCSI tape** 드라이버에 사용됩니다. 이 절차에서는 테이프 드라이브 작업을 위한 **mt-st** 패키지를 설치하는 방법을 설명합니다.

절차

- **mt-st** 패키지를 설치합니다.

```
# dnf install mt-st
```

추가 리소스

- **MT(1)** 및 **st(4)** 도움말 페이지

13.3. 테이프 장치 다시 사용하기 위한 쓰기

테이프 장치가 모든 작업 후에 다시 멈춘다. 데이터를 백업하려면 **tar** 명령을 사용할 수 있습니다. 기본적으로 테이프 장치에서 블록 크기는 **10KB(bs=10k)**입니다. **export TAPE =/dev/st0** 특성을 사용하여 **TAPE** 환경 변수를 설정할 수 있습니다. 대신 **-f** 장치 옵션을 사용하여 테이프 장치 파일을 지정합니다. 이 옵션은 두 개 이상의 테이프 장치를 사용할 때 유용합니다.

사전 요구 사항

1. **mt-st** 패키지가 설치되어 있어야 합니다. 자세한 내용은 저하 드라이브 관리 도구 설치를 참조하십시오.**For more information, see Installing tape drive management tool.**

2. 테이프 드라이브를 로드합니다.

```
# mt -f /dev/st0 load
```

절차

1. 테이프 헤드 확인:

```
# mt -f /dev/st0 status

SCSI 2 tape drive:
File number=-1, block number=-1, partition=0.
Tape block size 0 bytes. Density code 0x0 (default).
Soft error count since last status=0
General status bits on (50000):
 DR_OPEN IM_REP_EN
```

여기:

- 현재 파일 번호는 **-1**입니다.
- 블록 번호는 테이크 헤드를 정의합니다. 기본적으로 이 값은 **-1**로 설정됩니다.
- 블록 크기 **0**은 테이프 장치에 고정 블록 크기가 없음을 나타냅니다.
- **soft error count** 는 **mt status** 명령을 실행한 후 발생한 오류 수를 나타냅니다.

- *일반 상태 비트 는 테이프 장치의 통계를 설명합니다.*

- ***DR_OPEN** 은 문이 열려 있고 테이프 장치가 비어 있음을 나타냅니다. **IM_REP_EN** 은 즉각적인 보고서 모드입니다.*

2. *테이프 장치가 비어 있지 않으면 덮어씁니다.**If the tape device is not empty, overwrite it:***

```
# tar -czf /dev/st0 _/source/directory
```

*이 명령은 테이프 장치의 데이터를 **/source/directory** 의 콘텐츠로 덮어씁니다.*

3. ***/source/directory** 를 테이프 장치로 백업합니다.*

```
# tar -czf /dev/st0 _/source/directory
tar: Removing leading `/' from member names
/source/directory
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

4. *테이프 장치의 상태 보기:*

```
# mt -f /dev/st0  status
```

검증 단계

- *테이프 장치의 모든 파일 목록을 확인합니다.*

```
# tar -tzf /dev/st0
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

추가 리소스

- *MT(1), st(4) 및 tar(1) 매뉴얼 페이지*
- *기록 보호 Red Hat Knowlegebase 문서로 감지된 테이프 드라이브 미디어*
- *시스템 Red Hat Knowlegebase에서 테이크 드라이브가 검색되었는지 확인하는 방법*

13.4. NON-REWINDING TAPE DEVICES에 쓰기

unrewinding tape 장치는 특정 명령의 실행을 완료한 후 현재 상태로 유지됩니다. 예를 들어 백업 후 취소되지 않은 테이프 장치에 더 많은 데이터를 추가할 수 있습니다.For example, after a backup, you could append more data to a non-rewinding tape device. 또한 예기치 않은 재생 목록을 피하기 위해 사용할 수도 있습니다.

사전 요구 사항

1. *mt-st 패키지가 설치되어 있어야 합니다. 자세한 내용은 저하 드라이브 관리 도구 설치를 참조하십시오.For more information, see Installing tape drive management tool.*
2. *테이프 드라이브를 로드합니다.*

```
# mt -f /dev/nst0 load
```

절차

1. *non-rewinding tape device /dev/nst0:*

```
# mt -f /dev/nst0 status
```

2. *테이프의 끝 또는 앞쪽에 있는 포인터를 지정합니다.*

```
# mt -f /dev/nst0 rewind
```

3. *테이프 장치에 데이터를 추가합니다.*

```
# mt -f /dev/nst0 eod
# tar -czf /dev/nst0 /source/directory/
```

4. */source/directory/*를 테이프 장치로 백업합니다.

```
# tar -czf /dev/nst0 /source/directory/
tar: Removing leading `/' from member names
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

5. 테이프 장치의 상태 보기:

```
# mt -f /dev/nst0  status
```

검증 단계

- 테이프 장치의 모든 파일 목록을 확인합니다.

```
# tar -tzf /dev/nst0
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

추가 리소스

- **MT(1), st(4)** 및 **tar(1)** 매뉴얼 페이지
- 기록 보호 **Red Hat Knowlegebase** 문서로 감지된 테이프 드라이브 미디어
- 시스템 **Red Hat Knowlegebase**에서 테이크 드라이브가 검색되었는지 확인하는 방법

13.5. 테이프 장치에서 테이크 헤드 전환

다음 절차에 따라 테이프 장치의 테이크 헤드를 전환하십시오.

사전 요구 사항

1. **mt-st** 패키지가 설치되어 있어야 합니다. 자세한 내용은 저하 드라이브 관리 도구 설치를 참조하십시오.**For more information, see Installing tape drive management tool.**

2. 데이터는 테이프 장치에 기록됩니다. 자세한 내용은 **deleteing tape devices or writes to non-rewinding tape devices** 를 참조하십시오.

절차

- 테이프 포인터의 현재 위치를 보려면 다음을 수행합니다.

```
# mt -f /dev/nst0 tell
```

- 테이프 헤드를 전환 하는 동안 테이프 장치에 데이터를 추가 하려면 다음을 수행 합니다.

```
# mt -f /dev/nst0 eod
```

- 이전 레코드로 이동하려면 다음을 수행합니다.

```
# mt -f /dev/nst0 bsfm 1
```

- 이전 레코드로 이동하려면 다음을 수행합니다.

```
# mt -f /dev/nst0 fsf 1
```

추가 리소스

- **MT(1)** 매뉴얼 페이지

13.6. 테이프 장치에서 데이터 복원

테이프 장치에서 데이터를 복원하려면 **tar** 명령을 사용합니다.

사전 요구 사항

1. **mt-st** 패키지가 설치되어 있어야 합니다. 자세한 내용은 저하 드라이브 관리 도구 설치를 참조하십시오.**For more information, see Installing tape drive management tool.**

2. 데이터는 테이프 장치에 기록됩니다. 자세한 내용은 **deleteing tape devices or writes to non-rewinding tape devices** 를 참조하십시오.

절차

- 테이프 장치를 다시 여는 경우 **/dev/st0:**
  - **/source/directory** 복원:

```
# tar -xzf /dev/st0 /source/directory/
```

- **non-rewinding tape devices /dev/nst0:**
  - 테이프 장치를 다시 닫습니다.

```
# mt -f /dev/nst0 rewind
```

  - **etc** 디렉토리를 복원하십시오.

```
# tar -xzf /dev/nst0 /source/directory/
```

추가 리소스

- **MT(1)** 및 **tar(1)** 도움말 페이지

13.7. 테이프 장치에서 데이터 삭제

테이프 장치에서 데이터를 지우려면 삭제 옵션을 사용하십시오.

사전 요구 사항

1. **mt-st** 패키지가 설치되어 있어야 합니다. 자세한 내용은 저하 드라이브 관리 도구 설치를 참조하십시오.**For more information, see Installing tape drive management tool.**

2. *데이터는 테이프 장치에 기록됩니다. 자세한 내용은 **deleteing** tape devices or writes to non-rewinding tape devices 를 참조하십시오.*

절차

1. *테이프 장치에서 데이터 삭제:*

```
# mt -f /dev/st0 erase
```

2. *테이프 장치를 언로드합니다.*

```
mt -f /dev/st0 offline
```

추가 리소스

- ***MT(1)** 매뉴얼 페이지*

13.8. 보존 명령

*다음은 일반적인 **mt** 명령입니다.*

표 13.1. MT 명령

명령	설명
mt -f /dev/*st0* status	테이프 장치의 상태를 표시합니다.
mt -f /dev/*st0* erase	전체 테이프를 지웁니다.
mt -f /dev/*nst0* rewind	테이프 장치를 다시 닫습니다.
mt -f /dev/*nst0* fsf *n*	테이프 헤드를 forward 레코드로 전환합니다. 여기서 *n* 은 선택적 파일 수입니다. 파일 수를 지정하면 tape head는 *n* 레코드를 건너뜁니다.
mt -f /dev/*nst0* bsfm *n*	테이프 헤드를 이전 레코드로 전환합니다.
mt -f /dev/*nst0* eod	테이프 헤드를 데이터 끝으로 전환합니다.